UNIVERSIDADE DA CORUÑA

FACULTADE DE INFORMÁTICA

DEPARTAMENTO DE COMPUTACIÓN

PROYECTO DE INGENIERIA TÉCNICA EN INFORMÁTICA DE GESTIÓN

EXPLOTACIÓN REMOTA DEL SISTEMA GALENA

Proyecto dirigido por
D. Manuel Vilares Ferro
y presentado por
Luis Docampo Gutiérrez

Septiembre de 1997

A mis padres y mi hermana.
A mis dos abuelas, por su constante apoyo en mis estudios.
A todos mis amigos.

Índice

ÍNDICE **1**

INTRODUCCIÓN **2**

INTERNET Y SU MUNDO **5**

¿QUÉ ES INTERNET? 5
¿CÓMO ES INTERNET? 6
SISTEMA DE NOMBRES POR DOMINIOS O DNS 10

SERVICIOS QUE OFRECE INTERNET **12**

CORREO 12
EL WEB 12
USENET 13
GOPHER 14
FTP ANÓNIMO 14
TELNET (CONEXIÓN REMOTA) 15

NAVEGADORES DEL WEB **16**

INTRODUCCIÓN AL SISTEMA GALENA **21**

ANÁLISIS LÉXICO EN GALENA **23**

INTRODUCCIÓN 23
LAS REGLAS LÉXICAS 27
UNA APROXIMACIÓN INTUITIVA 28
LA INTERFAZ DE USUARIO CON RESPECTO AL ANALIZADOR LÉXICO 29

ANÁLISIS E IMPLEMENTACIÓN **34**

INTRODUCCIÓN 34
UTILIDAD DE LA APLICACIÓN 37
INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN JAVA 40
ANÁLISIS PRELIMINAR DE LA APLICACIÓN 48
MANUAL DE USUARIO DE LA INTERFAZ LÉXICA DE GALENA 50
Modo Consultar-Modificar-Borrar *52*
Modo Introducir *57*
Modo Introducir Verbo Irregular *64*
MANUAL TÉCNICO DE LA APLICACIÓN 67

CONCLUSIONES **89**

BIBLIOGRAFÍA **92**

APENDICE I **93**

Introducción

Los últimos años han constituido una auténtica revolución en el mundo de la informática, principalmente desde la aparición de **Internet**. Ésta ha dado lugar a que millones de personas que nunca habían utilizado el ordenador, se hayan lanzado a una utilización repentina y masiva, que fundamentalmente se basa en navegar electrónicamente a través de bases de datos, buscando información de todo tipo, material bibliográfico, imágenes, etc..

De hecho, se perfila un cambio en la forma de concebir el trabajo con ordenador. En su concepción tradicional, el usuario disponía de un ordenador y de un medio de almacenamiento. En la actualidad, se piensa en que una buena parte del **software** y de los datos no es necesario que sean almacenados por el usuario, sino que simplemente este los cargará de algún servidor de la red cuando lo crea necesario. Es más, las últimas tendencias se encaminan a que el usuario, ni siquiera posea en su máquina las aplicaciones que necesita ejecutar, sino que éstas residan en otro lugar. Cuando le sean necesarias, accederá a ellas a través de la red y las ejecutará localmente sin necesidad de que sean compiladas para su sistema operativo ni de que le ocupen espacio cuando no las utilice. En el caso de los **applets**[1] de Java, además de que poseen estas características y ventajas, el usuario podrá en todo momento disponer de las versiones más actualizadas de sus aplicaciones al ser actualizadas de una forma totalmente transparente para el usuario y sin que este necesite realizar algún trabajo adicional.

[1] Pequeños (o no tan pequeños) programas que se integran en las páginas Web, aportando interactividad y nuevas posibilidades a las mismas, y que se ejecutan en la máquina en la cual está el navegador que carga la página, utilizando lrecursos que incorpora el propio navegador

Pues bien, con este proyecto se ha pretendido aprovechar estas nuevas tecnologías, con sus ventajas e inconvenientes, para que las herramientas construidas en el entorno GALENA puedan ser accesibles a través de **Internet**.

Con él, se consigue que varias herramientas desarrolladas originalmente para plataformas **Unix** y que únicamente estaban disponibles mediante una conexión con el servidor en el cual residen, puedan ahora ser ejecutadas en cualquier plataforma y en cualquier maquina, que posea acceso a **Internet,** es decir, estamos ante el concepto de **sistema distribuido**.

El término sistema distribuido se utiliza corrientemente para designar conjuntos informáticos constituidos de unidades de tratamiento o almacenamiento interconectadas por un sistema de comunicación. En estas unidades se ejecutan programas que cooperan a la realización de una tarea común o comparten recursos comunes.

Esta idea de crear una aplicación distribuida ha surgido basándose en las necesidades de las personas que actualmente están utilizando el entorno GALENA, el cual ha sido desarrollado conjuntamente por equipos de las tres universidades gallegas. Esta circunstancia conlleva la necesidad de acceder fácilmente a la herramienta sin necesidad de realizar conexiones con alguna máquina concreta y ejecutar las aplicaciones en ella o de poseerlas compiladas específicamente para una máquina. En base a ello se ha decidido que la mejor forma de acceder las herramientas es a través de una red común para todos, **Internet.**

Como consecuencia de esta decisión, se optó por empezar a aplicar dicha óptica a las herramientas de la Interfaz Léxica de GALENA, obteniéndose como resultado la aplicación desarrollada en este proyecto.

De forma que ahora la interfaz puede ser ejecutada desde cualquiera de los ámbitos en los cuales se está utilizando.

De forma adicional, se pretende que esta aplicación también pueda ser utilizada por personas ajenas al grupo de desarrollo del entorno GALENA, de forma que cualquiera que esté interesada en ello pueda consultar la base de datos, interactuar con ella observando los resultados que se obtienen al hallar formas derivadas de raíces y lemas y también que pueda añadir a la base de datos nuevos términos que enriquezcan su contenido.

Internet y su mundo

¿Qué Es Internet?

Internet es el nombre de una red de alcance mundial que consta de personas, información y computadoras. Estos recursos son tan amplios que van mas allá de lo que podíamos imaginar hace tan sólo unos pocos años.

El origen de **Internet** puede situarse en un proyecto, llamado **Arpanet**, patrocinado por la Agencia de Proyectos de Investigación Avanzada del Departamento de Defensa de los Estados Unidos. El Departamento de Defensa estaba interesado en construir una red que fuese autosuficiente en condiciones adversas y capaz de transportar información militar y gubernamental en caso de un conflicto nuclear.

El proyecto comenzó en 1968 y evolucionó pronto en una meta más general de técnicas de desarrollo para construir una red a gran escala. **Arpanet** siguió existiendo durante unos años y fue retirada de forma progresiva una vez que se declaró terminado el proyecto.

En principio, la meta de los investigadores de **Arpanet** era desarrollar una gran red para conectar computadoras a gran distancia. No obstante, hacia la mitad de los años 70, se hizo evidente que una sola red no podría satisfacer las necesidades de todos los usuarios potenciales. Los investigadores vieron que sería mucho más útil desarrollar una tecnología que pudiese conectar distintos tipos de redes en un solo sistema de mayor entidad. Esto llevó al concepto de **Internet**.

Hoy en día **Internet** no es realmente una única red de computadoras. En realidad es un conjunto de miles de redes distribuidas por todo el planeta. Sería un error, sin embargo, considerar la **Internet** únicamente como una red de computadoras o

un grupo de redes de computadoras conectadas entre sí, en realidad las redes de computadoras son simplemente el medio que transporta la información e **Internet** es la información misma y la gente que participa.

¿Cómo es Internet?

Para utilizar **Internet** correctamente es necesario entender algunos conceptos fundamentales.

El termino **red** se refiere a dos o mas computadoras conectadas entre sí. Hay un gran número de razones para unir las computadoras en redes, pero las dos más importantes, son :

- El desarrollo de arquitecturas distribuidas.
- La compartición de recursos.

Gracias a las redes, los usuarios pueden comunicarse y establecer contacto de muy diversas formas, quizás la más conocida es el correo electrónico.

También gracias al concepto de red es posible compartir recursos con el consiguiente ahorro económico.

Una **red de área local**, o **LAN**, es una red en la que las computadoras se conectan directamente, normalmente con algún tipo de cable.

Cuando se conectan unas **LAN** con otras, se forma lo que se llama **red de área extensa** o **WAN** . La mayoría de las redes de área extensa se conectan utilizando algún tipo de línea telefónica, aunque también existen otras tecnologías disponibles.

Las **LAN** cuando se conectan se hace mediante unos dispositivos especiales denominados **encaminadores** o **routers** si utilizamos el termino inglés.

La misión de un encaminador es proporcionar un enlace de una red a otra. Se utilizan estos dispositivos para conectar varias **LAN**, así como para conectar distintas **WAN** y formar así **WAN** aún mayores.

Otros dos conceptos fundamentales para entender como es **Internet** son los de **cliente** y **servidor**. Un servidor es un programa que proporciona uno o mas recursos y un cliente es otro programa que se utiliza para acceder a dichos recursos.

Pongamos un ejemplo, para entenderlo mejor:

Un servidor podría ser, por ejemplo, un gestor de bases de datos, que posee un determinado número de bases de datos y está instalado en una máquina determinada.

Los recursos que ofrece pues este servidor son los datos almacenados en sus bases de datos. Por el otro lado el programa cliente podría ser un **browser**[2] que mediante un formulario de una pagina **Web** interroga a una base de datos y devuelve sus resultados.

Tanto clientes como servidores, en **Internet**, residen en lo que se denominan **hosts**, que no son otra cosa que simplemente cada uno de los ordenadores conectados a la red, es decir cada uno de los nodos que la conforman. El servidor se ejecuta en un host servidor y el cliente en un host cliente, aunque esto no quiere decir que los papeles no se puedan invertir entre ambas máquinas.

Finalmente lo último que hay que mencionar para comprender como es **Internet** es lo que podemos decir que constituye la parte fundamental y que hace funcionar la red, el conjunto de protocolos (serie de reglas que describen como hacer unas tareas) **TCP/IP**. Estos protocolos juegan el papel de catalizadores que hacen que todas las redes de las que está compuesta **Internet** puedan estar unidas.

TCP/IP es el nombre común de una colección de más de cien protocolos que nos permiten conectar computadoras y redes. El nombre proviene de los dos protocolos

[2] Aplicación gráfica que permite visualizar documentos de hipertexto.

más importantes: **TCP** (Transmision Control Protocol, Protocolo de control de transmisión) e **IP** (Internet Protocol, Protocolo Internet).

Aunque explicar el funcionamiento de **TCP/IP** sería extenso y complicado, explicaré brevemente su funcionamiento.

Dentro de **Internet**, la información no se transmite como una cadena continua de carácteres de host a host, sino que los datos se transmiten en pequeños trozos de información denominados **paquetes**. Un ejemplo aclarará las cosas:

Supongamos que enviamos un mensaje de correo electrónico muy extenso a un amigo al otro lado del país. **TCP** dividirá este mensaje en paquetes, marcará cada paquete con un número de secuencia y le añadirá además la dirección del destinatario e información para el control de errores. Finalmente estos paquetes son enviados por la red, donde la misión de **IP** es transportarlos hasta el host remoto[3].

En el otro extremo, **TCP** recibe los paquetes y comprueba si hay errores en la transmisión, si es así, pide que el paquete en cuestión le sea reenviado. Una vez que se hayan recibido todos los paquetes de forma correcta, **TCP** utilizará los números de secuencia incluidos en cada paquete para reconstruir el mensaje original y enviárselo a la aplicación a la que corresponda.

En otras palabras, la misión de **IP** es transportar los datos en bruto[4] de un lugar a otro, conociendo las direcciones correspondientes. La misión de **TCP** es manejar el flujo de datos y asegurarse de que éstos son correctos.

Dividir los datos en paquetes ofrece algunas ventajas importantes. Primero, permite que **Internet** utilice las mismas líneas de comunicación con distintos usuarios al mismo tiempo. Puesto que los paquetes no tienen que viajar juntos, una línea de

[3] Por ello a IP también se le conoce como el protocolo de transporte de TCP/IP.

[4] Los paquetes.

comunicación puede transportar todo tipo de paquetes de un lugar a otro. En su camino, los paquetes se envían de host a host hasta que alcanzan su destino final[5]. Esto significa que **Internet** tiene una gran flexibilidad. Si una computadora está fuera de servicio, las computadoras que controlan el flujo de datos pueden encontrar normalmente una ruta alternativa. De hecho, es posible que dentro de una misma transferencia de datos, varios paquetes sigan distintas rutas hacia el mismo destino. Esto significa también que, cuando las condiciones cambian, la red puede utilizar la mejor vía disponible en ese momento.

Otra ventaja de utilizar paquetes es que cuando algo va mal, sólo se tiene que retransmitir un paquete o un pequeño grupo de ellos y no el mensaje entero, aumentando así la velocidad de la red al evitarse la saturación de mensajes repetidos.
En resumen toda esta flexibilidad redunda en que se consigue una gran fiabilidad, de una forma u otra, **TCP/IP** asegura la entrega de los datos de forma correcta. Por tanto se puede estar seguro de que los datos que se envían por **Internet** llegarán a su destino de forma correcta.

Vista esta breve explicación de como es **Internet**, por que está compuesta y como transmite los datos de un lado a otro, en la siguiente figura podemos observar como todos los elementos descritos antes, están relacionados y dispuestos en la red.

[5] La ruta la eligen unas computadoras especiales llamadas encaminadores.

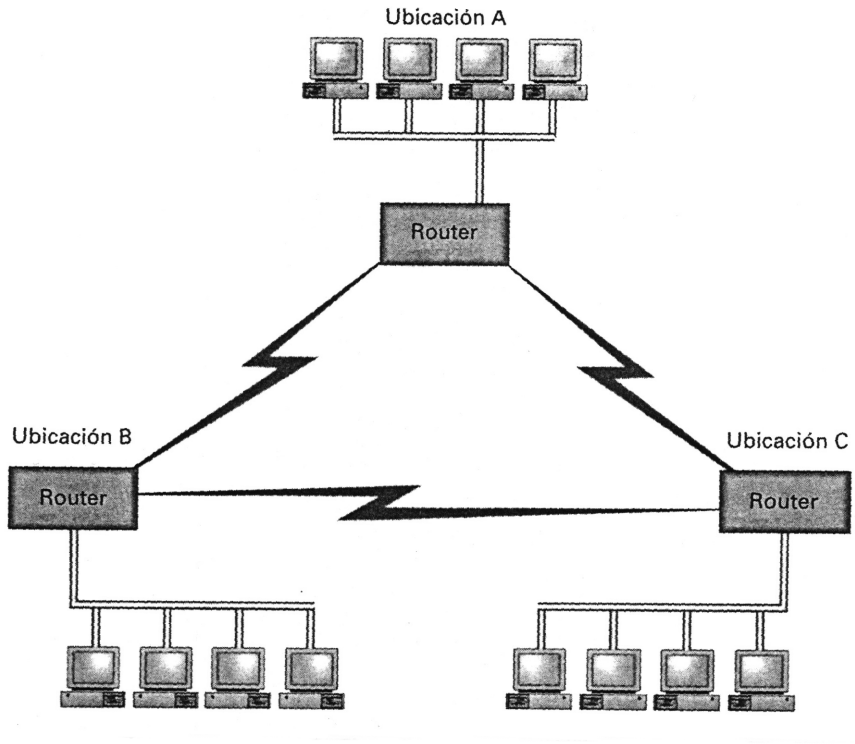


Figura 1. Configuración de Internet.

Sistema de nombres por dominios o DNS

Para que un ordenador sea identificable en Internet debe poseer un identificador o número IP. Un número IP identifica a un ordenador en **Internet** de forma análoga a como un número de teléfono (con todos los prefijos de país, ciudad...) identifica dentro de la red telefónica a un teléfono determinado.

Un número IP consta de cuatro números (del 0 al 255) separados por puntos. Por ejemplo el número IP 193.144.51.75 identifica a una maquina del Departamento de Computación de la Universidad de A Coruña.

Todo ordenador que está conectado a **Internet**, identificado por un número IP único, puede tener uno o más nombres. Por ejemplo la máquina 193.144.51.75 se denomina obelix.dc.fi.udc.es. Se trata de una serie de nombres separados por puntos, y que se encuentran ordenados jerárquicamente por dominios de derecha a izquierda. Así obelix.dc.fi.udc.es es una máquina situada en España (.es), en la Universidad de A Coruña (.udc), en la Facultad de Informática (.fi) y de nombre obelix .

Servicios que ofrece Internet

Una vez cubiertos los fundamentos básicos de **Internet**, veamos qué nos puede ofrecer. En este apartado nos centraremos en aquellos servicios de los que un usuario puede disponer si se conecta.

Correo

Un usuario puede enviar y recibir mensajes de cualquier otro usuario que también esté conectado. Más aún, puede enviar mensajes a otros sistemas de correo, como pueden ser CompuServe o MCI Mail, que tienen conexiones con **Internet**.

Sin embargo, correo electrónico no sólo significa mensajes personales. Cualquier cosa que se pueda almacenar en un archivo de texto se puede enviar por correo electrónico: programas informáticos, anuncios, etc.

El Web

Es este servicio, de entre todos los que ofrece **Internet**, el que más nos interesa a la hora de desarrollar el proyecto, pues la ejecución de las herramientas desarrolladas en él deberán ser cargadas y ejecutadas a través del **Web**.

El **Web** es un sistema distribuido de servidores que ofrece todo tipo de información a los usuarios de **Internet**. La información puede aparecer en formato normal de texto, así como en imágenes, sonidos u otro tipo de datos como pueden ser programas

ejecutables. Este es el caso de los **applets** de **Java** que en este proyecto son los elementos fundamentales del desarrollo de las herramientas implementadas.

Para acceder a esta información se utiliza un programa cliente llamado **explorador**.

Hay dos motivos por los que el **Web** es tan popular. En primer lugar, es fácil de usar. En segundo lugar, es también fácil crear una información **Web** propia y compartirla con el resto de usuarios de **Internet**.

En el **Web** la información se almacena en lo que se denominan páginas, que realmente en la base son ficheros con texto etiquetado de una forma especial. Cada página puede contener no sólo información, sino también vínculos a otras páginas, pudiéndose seguir el vínculo de una página a otra.

Un texto que contiene vínculos a otros datos se dice que es **hipertexto**. El hipertexto permite la presencia de palabras o términos que posean más información, similares a notas a pie de página en un libro. Estas palabras están marcadas según lo permita la interfaz de usuario: subrayadas, en campo inverso, etc. La activación de cualquiera de estas palabras supone la visualización de un nuevo documento hipertexto, ubicado en cualquier lugar del mundo. De este nuevo hipertexto es posible llamar y visualizar nuevos documentos de hipertexto, retornar al anterior, etc.

Usenet

Usenet es un sistema de grupos de discusión en el que se distribuyen artículos individuales por todo el mundo. Usenet tiene literalmente miles de grupos de discusión diferentes.

Su nombre viene de la contracción **User's Network** –red de usuarios--, aunque Usenet no es realmente una red, sino un sistema de grupos de discusión.

Originalmente, Usenet se configuró para ofrecer un servicio de tablón de anuncios electrónico al que se pudieran enviar noticias.

Para participar en Usenet, se utiliza un programa cliente llamado lector de noticias.

Gopher

El sistema **Gopher** se parece al **Web** en que se utiliza un programa cliente para conectar con servidores en todo el mundo, de uno en uno. La diferencia entre **gopherespacio** y **Web** está en el modo en que se organiza la información.

En el **gopherespacio**, la información se presenta como una serie de menús simples. Cada servidor gopher tiene un menú principal y una serie de submenús. Para utilizar un gopher todo lo que se tiene que hacer es seleccionar el elemento de menú deseado y el programa cliente lo buscará. Los elementos del menú pueden apuntar a distintos tipos de recursos: archivos, imágenes, otros recursos de Internet, etc. Los elementos de un menú pueden apuntar a otros menús.

Ftp anónimo

El servicio **ftp** permite copiar archivos de una computadora a otra. Las siglas ftp corresponden a **f**ile **t**ranfer **p**rotocol –protocolo de transferencia de ficheros—uno de los muchos protocolos que forman parte de la familia **TCP/IP**.

Por razones de seguridad, no se pueden copiar archivos entres dos computadoras a menos que se haya establecido una conexión con el sistema remoto.

Para hacer la conexión previamente habrá que identificarse con un **login**[6] y un **password**[7].

Este servicio es cómodo para aquellas personas que tienen cuentas en más de una computadora. Sin embargo, muchas ubicaciones **Internet** tienen archivos de interés publico que ponen a disposición de todo el mundo en la red. Para acceder a estos archivos se utiliza el sistema llamado ftp anónimo. Este sistema permite que cualquier persona utilice una cuenta de invitado para conectarse con la computadora. Una vez establecida la conexión, se podrán copiar aquellos ficheros situados en un área publica especial.

Telnet (conexión remota)

En algunos sistemas informáticos es necesario introducir un nombre de usuario y una contraseña para poder iniciar una sesión de trabajo. A este proceso se le conoce como iniciar una sesión. Uno de los motivos originales por los que se creó **Internet** fue para permitir que la gente trabajase en computadoras remotas. El servicio que permite iniciar una sesión y utilizar una computadora remota se llama **telnet**.

6 Palabra que sirve para identificar a un usuario del servicio de ftp.

7 Palabra que sirve para que un determinado usuario de ftp pueda entrar en el servicio. Es una clave que posee el usuario y que normalmente sólo él conoce. Con esta palabra se impide el acceso a personas ajenas.

Navegadores del Web

Como se ha mencionado en otro apartado, este proyecto está orientado a utilizar fundamentalmente el servicio que proporciona **Internet** denominado **Web**. Como también ya se dijo, para poder utilizar este servicio, los usuarios deben tener instalado un programa cliente denominado **browser**, mediante el cual se puede acceder a las paginas del **Web**. Las características especiales de los **browsers** quizás fuesen fundamentales para que el **Web** obtuviese el éxito que actualmente posee.

Pues bien, en este apartado se pretende tratar algo acerca de estos programas, que interesan en particular a la hora de hablar del presente proyecto, porque con ellos cualquier persona podrá ejecutar las aplicaciones desarrolladas.

Si en algo se caracteriza el **Web** es en su facilidad de uso, lo que redunda en un atractivo para el usuario. A esta facilidad de uso contribuyen los **browsers**, principalmente porque son herramientas gráficas que cualquier persona puede manejar sin apenas conocimientos previos ni grandes aprendizajes.

El primer navegador **Web** que se desarrolló recibió el nombre de **Mosaic**. Fue escrito, entre otros, por Marc Andressen, un estudiante becado en el centro de supercomputación de la Universidad de Illinois. Esta universidad, viendo su trabajo, decidió proporcionar el navegador y toda la información técnica necesaria a todas las personas interesadas. Este fue el comienzo de la explosión definitiva de **Internet**

Otro **browser** también conocido es **Lynx**. Este navegador es característico de los entornos **Unix** en modo comando, y por lo tanto como tal, es un navegador que sólo permite visualizar texto. Además dado que se ejecuta normalmente en entornos en modo comando, no posee una interfaz gráfica, siendo su uso más dificultoso.

Netscape, sin duda alguna es el navegador más conocido y utilizado de entre todos. Este navegador, es una herramienta gráfica que además de poseer un visualizador de paginas **Web**, también posee las herramientas necesarias para poder utilizar prácticamente todos los servicios que ofrece **Internet** (Correo, ftp, etc).

En lo que a este proyecto se refiere, hay que mencionar que **Netscape** sirve para poder arrancar las herramientas desarrolladas en él, pues soporta **Java** a partir de la versión 2.0 (en el momento de hacerse esta memoria la última versión es la 4.01).[8]

Sin embargo, hay que hacer notar que este navegador, no soporta enteramente todas las especificaciones de Java, sobre todo a partir de la versión 1.1.0 del **JDK**(Java Developer Kit). Otro gran problema que presenta con respecto a **Java** es que las restricciones de seguridad se las toma muy en serio, sin que estas puedan ser modificadas por el usuario, impidiendo realizar ciertas operaciones, que en ocasiones son de interés, como por ejemplo acceder a un servidor que no sea aquel desde el que se carga el **applet**.

Esto implica que en el desarrollo de este proyecto no se hallan podido utilizar algunas características nuevas de **Java** presentes en la version 1.1.3 del JDK, que podrían haber sido de gran utilidad.

También ha sido necesario cambiar el gestor de bases de datos y la base de datos que se utiliza en el proyecto. El nuevo destino tenía que ser la máquina en el que se encuentra el servidor HTTP al que se accede para cargar la página desde la cual se arranca la interfaz construída. La razón de esto es que una de las restricciones impuestas por este navegador es no permitir conexiones con otros servidores que no sean aquel desde donde se ha cargado el **applet** que intenta realizar la conexión.

[8] A partir de la versión 2.02 puede considerarse que es lo suficientemente estable soportando **Java**.

En nuestro caso se necesitaba hacer una conexion con el gestor de bases de datos situado en otro servidor, por lo que estábamos en esa situación.

Un aspecto positivo es que la implementación de **Java** de **Netscape** lleva algún tiempo en el mercado, lo que ha contribuido a que se hayan ido solucionando poco a poco pequeños problemas que otros navegadores poseen. Otra ventaja es su disponibilidad para multitud de sistemas operativos, característica que lo convierte en una herramienta habitual, siendo obligado para el programador de **applets** el contar con él.

Figura 2. El navegador Netscape.

Otro navegador que es obligatorio mencionar es el de la empresa **Microsoft,** la cual ha comenzado a dar los pasos para introducirse también en el mercado de navegadores, monopolizado por la compañía **Netscape**, y que está llevando a cabo una campaña muy agresiva de promoción de su producto, denominado **Internet Explorer**. En este navegador, el soporte de **Java** únicamente es posible a partir de la versión 3.0 (en el momento de escribir esta memoria la última versión es la 4.0). Debe tenerse en cuenta que la implementación de **Java** en este producto es muy reciente, por lo que se pueden encontrar algunos errores. Por lo demás, las prestaciones que ofrece y sus características son muy similares a las de **Netscape**.

Figura 3. El navegador Internet Explorer.

Por último mencionaré el producto que mejor soporta **Java**, y que se recomienda para poder ver las utilidades desarrolladas en este trabajo. Este producto es el navegador **Sun Hotjava**. Fue el primer navegador que soportó Java y sigue siendo el único que lo soporta enteramente. Además su principal baza es que se puede configurar totalmente, de modo que permite relajar las restricciones de seguridad impuestas sobre los **applets**, según el nivel de seguridad deseado. Así aunque en ningún caso se autoriza el acceso al sistema de ficheros local, si que se permite especificar el nivel de seguridad en las comunicaciones que establezcan los **applets** con otros ordenadores.

Figura 4. El navegador HotJava.

Introducción al sistema Galena

El entorno GALENA[9] es un proyecto conjunto de las tres universidades gallegas: Universidad de Santiago de Compostela, de Vigo y de A Coruña, y es el resultado de la inquietud y colaboración de un equipo de informáticos y otro de lingüistas. Este trabajo se inscribe tanto en el terreno de la lingüística computacional como en el de la lingüística de corpus. Dicho en pocas palabras, el objetivo es desarrollar herramientas de análisis lingüístico automático que permitan un tratamiento eficaz de la interpretación de lenguajes naturales en su forma escrita.

Una vez logrado el objetivo de construir estas herramientas, las puertas que se abren son muchas y muy variadas:

- traducción automática.
- sintetización de voz.
- análisis de las construcciones lingüísticas.
- sistemas de aprendizaje.
- interfaces de lenguaje natural.
- ...

GALENA tiene como idioma objetivo el Español, pero paralelamente se está desarrollando el sistema para adecuarlo a las muchas particularidades que presenta el Gallego. De esta forma los avances son, generalmente, más rápidos ya que la filosofía

[9] Generador de Analizadores de LEnguaje NAtural.

de desarrollo va paralela. La única diferencia, que no es poca, es la debida a las diferencias que existen entre una lengua y la otra.

El objetivo es construir programas informáticos que sean capaces de etiquetar gramaticalmente cada forma (**tagging**), vincularla al lema que le corresponde y, en el paso siguiente, realizar el análisis sintáctico de las cláusulas en los elementos funcionales que las constituyen. Posteriormente, en un futuro muy cercano, habrá que abordar la parte siguiente: el análisis semántico, para permitir así que el ordenador pueda asociar acciones efectivas al texto interpretado.

Las herramientas resultantes, de interés ya en sí mismas en tanto que son instrumentos de análisis lingüístico automatizado, permitirán abordar todas las posibilidades que se enumeraron con anterioridad en esta misma página.

Análisis léxico en Galena

Introducción

El análisis léxico constituye el primer paso en el procesamiento informático de las lenguas naturales, entendido como el etiquetado de clases de palabras, así como la detección y posible eliminación de ambigüedades en la determinación de dichas clases. En el estudio de los lenguajes de programación dicha fase suele tener un tratamiento práctico totalmente independiente de las fases de análisis sintáctico y semántico. Sin embargo, en el caso de las lenguas naturales esta parte del proceso general de análisis tiene especial importancia en el tratamiento subsiguiente. Es por ello necesario asegurar un tratamiento eficiente del mismo.

En este contexto, el trabajo que a continuación se describe pretende dar un tratamiento práctico al problema de la etiquetación de lenguas naturales [Graña, Alonso y Valderruten, 1994], tanto en lo que se refiere a la velocidad de tratamiento como a su dominio de aplicación. Para ello, se ha partido de la base representada por la experiencia acumulada durante años en el tratamiento de este mismo problema, a nivel de los lenguajes formales de programación. En concreto, la técnica propuesta representa un sustancial distanciamiento de las aplicadas tradicionalmente en la etiquetación de textos, fundamentadas en el uso de bases de datos, para orientarse hacia métodos basados en el uso de autómatas finitos de diseño específico, que incorporan mecanismos operacionales para el tratamiento del conocimiento lingüístico disponible acerca de la estructura morfológica de las palabras.

El entorno aplicativo más estudiado en relación al tratamiento del análisis léxico lo constituyen sin duda los lenguajes de programación. El conocimiento obtenido en este área podría resultar de gran provecho en el tratamiento de otro tipo de lenguajes, tales como los utilizados para la comunicación humana, si tenemos en cuenta las diferencias que separan a unos y otros:

- En un lenguaje de programación, cada palabra se corresponde con un único componente léxico. En el caso de las lenguas naturales, una palabra, por sí misma y aislada (sin contexto), puede tener múltiples acepciones que se corresponden con diferentes categorías gramaticales.
- Una palabra puede contener más de un componente léxico[10] e incluso puede suceder que un único componente léxico englobe a más de una palabra[11].
- Las lenguas naturales poseen un complejo entramado de reglas morfológicas que controlan la construcción de las variantes válidas de cada palabra, aspecto éste que no aparece en los lenguajes de programación.

En consecuencia, un analizador léxico para lenguas naturales debe proveer mecanismos no deterministas que permitan obtener todas las posibles interpretaciones de una palabra, así como mecanismos operacionales que le permitan rentabilizar las posibilidades que ofrece el análisis morfológico. Más exactamente, el analizador debe proveer todos los posibles reconocimientos para una palabra dada y no debe pasar a reconocer un nuevo componente léxico hasta que no se hayan agotado todas las posibilidades para el precedente. Sólo de esta manera se puede asegurar que el posterior

[10] Tal es el caso de los verbos con pronombres enclíticos: **cógelo**.

[11] Es el caso de las locuciones prepositivas: **de cara a**, ...

análisis sintáctico y semántico sea capaz de tratar correctamente el texto, seleccionando de entre todas las opciones aquellas que mejor concuerden con la estructura general del mismo.

Un caso particular interesante es el de la lengua española, especialmente rica en lo que a estructuras morfológicas se refiere, con decenas de miles de **lemas** que se corresponden con las entradas en el diccionario. Cada lema representa lo que podríamos llamar la forma canónica de un conjunto de palabras. Así, el lema de un sustantivo lo constituye la forma masculina singular. Las demás formas, femenino y plurales en masculino y singular, pueden ser derivadas a partir de la forma canónica utilizando las reglas morfológicas correspondientes al tipo de sustantivo del que se trate. En el caso de los verbos, la necesidad de utilización de estas reglas se hace más acusada. En los verbos regulares, nos basta con conocer la forma del infinitivo para poder conjugar todos los posibles tiempos verbales. Sin embargo, los verbos irregulares siguen sus propias reglas léxicas. En este caso, es incluso probable que sea preciso utilizar más de una raíz para su conjugación. Un ejemplo típico es el del verbo **pensar,** para cuya conjugación es necesario utilizar las raíces **pens y piens.**

Una forma de solucionar los problemas planteados consiste en utilizar una inmensa base de datos en la cual estén almacenadas todas las formas de todas las palabras. Cada vez que se tenga que reconocer una palabra, se accede a esta base de datos y se recuperan todas las posibles alternativas. Este enfoque presenta serios inconvenientes:

- El tamaño desproporcionado que deberá tener la base de datos para ser capaz de obtener buenos resultados con textos grandes. Ello, o bien obliga a que

residan en almacenamiento secundario gran parte de los recursos del sistema, o bien limita notablemente sus prestaciones.

- Cada vez que se encuentra una nueva palabra se deben introducir todas las formas derivadas de su lema, si queremos que la base de datos sea consistente. Por ejemplo, para cada verbo sería necesario introducir todas las formas verbales relativas a su conjugación. Se podría evitar la entrada manual de las formas mediante un generador de formas derivadas, pero aún así se tiene el problema del crecimiento exponencial del tamaño de la base de datos con respecto al número de lemas, lo que degrada el rendimiento ya que introduce retardos en el tiempo de acceso, siempre considerando que se posee una buena estructura de índices.

Para evitar esos inconvenientes hemos considerado que la mejor solución es utilizar un generador de analizadores léxicos en el que se puedan incorporar las reglas morfológicas y el comportamiento no determinista, adoptando el concepto de autómata como formalismo operacional. El resultado es un programa ejecutable que se encarga de realizar las labores de etiquetación con un buen rendimiento temporal y un consumo de memoria mínimo. Como punto de partida para el diseño de esta nueva herramienta se ha optado por utilizar FLEX, un generador de analizadores léxicos estándar en el sistema operativo **Unix** [Kaare, 1990].

Pese a todo lo anterior en el sistema actual existe una base de datos de lemas que nos facilita mucho las tareas de consulta, inserción y depuración de los lemas que se introducen en el sistema. Sin embargo, no se cae en los problemas mencionados con anterioridad puesto que a partir de la base de datos de lemas se generan los autómatas

en código FLEX a partir de una serie de programas. De esta manera se consigue mantener una interfaz agradable de cara al usuario a la vez que una eficiencia muy alta.

Las Reglas Léxicas

Las palabras de una lengua como la española se pueden dividir en dos partes: el **lexema** y los **afijos**. El lexema es la parte invariante de la palabra. En ciertos casos la palabra está formada únicamente por el lexema. Tal es el caso de las preposiciones (a, de, para, ...). Sin embargo, normalmente las palabras se forman mediante la combinación de una serie de sufijos y/o prefijos (afijos en general) junto con el lexema, como ocurre, por ejemplo, en el caso de los sustantivos, los adjetivos y los verbos. Actualmente los prefijos no son tratados de una manera especial sino que son introducidos como lemas aparte. Por el contrario, como se podrá observar a lo largo de esta exposición, sí se ha hecho un estudio exhaustivo para el tratamiento de los sufijos.

Cuando de un lexema se puede obtener un conjunto de palabras que derivan de él, se toma una de esas palabras para representar el conjunto: es lo que se denomina el lema de ese conjunto de palabras. En el caso de los sustantivos y los adjetivos el lema lo constituye la forma masculina singular, y en el de los verbos el infinitivo.

En ciertos casos, como en el de los verbos irregulares, las cosas no son tan simples, puesto que no todas las formas de la conjugación tienen la misma raíz. Consideremos por ejemplo el verbo **pensar**. Parte de la conjugación utiliza como raíz **pens** (pensé, pensaré), pero otras utilizan **piens** (pienso, piensas) como raíz. Sin embargo, todas las formas de la conjugación tienen un único lema: **pensar**.

Por lo tanto, a la hora de reconocer las palabras válidas del Español se deberá partir del reconocimiento de los lexemas para posteriormente ir aplicando las reglas

léxicas que indican cómo se formarán las palabras pertenecientes al conjunto de un determinado lema.

Mención y tratamiento especial, que aquí no se discutirá, merecen los verbos compuestos, los pronombres enclíticos y los elementos léxicos de varias palabras[12].

Una aproximación intuitiva

Para aclarar el significado de las reglas léxicas utilizaremos como ejemplo los sustantivos. En Español se han detectado, por parte de los expertos, hasta veinte formas distintas de formar el género de una palabra, catalogadas como G1, G2, ..., G20. Hay grupos muy comunes, como por ejemplo el G1, que añade una "**o**" en el caso del masculino y una "**a**" en el del femenino. Otros son más inusuales, como el grupo G8, que forma el masculino añadiendo la terminación "**que**" y el femenino mediante la terminación "**ca**". Un ejemplo de este grupo lo constituyen las palabras **cacique** y **cacica**.

Para formar el número se han detectado diez formas, catalogadas como N1, N2, ..., N10. El grupo más común es el N1, que añade una "**s**" para formar el plural y ningún carácter para el singular. Un grupo bastante inusual es el N4, que utiliza la terminación "**c**" para el singular y la "**ques**" para el plural. Es el caso de las palabras **frac** y **fraques**.

La mayoría de los sustantivos forman tanto el género como el número, por lo que es necesario combinar los dos tipos de reglas léxicas. En cualquier caso, siempre se forma primero el género y después el número. Además, no son válidas todas las

12 En concreto las locuciones prepositivas.

combinaciones de grupos de formación género-número. Por ejemplo, los lexemas que utilizan el grupo G1 para formar el género siempre utilizan el grupo N1 para el número.

En el caso de los verbos, en Español, como es fácilmente previsible, existen muchas más reglas léxicas que determinan cómo se construye cada una de las conjugaciones y la treintena de grupos de verbos irregulares.

Todo este conocimiento debe ser incorporado en la etapa de análisis léxico para determinar los componentes que se han de reconocer. Puesto que las reglas léxicas son fijas, para ampliar posteriormente el conjunto de palabras que es capaz de reconocer el analizador tan sólo es necesario introducir el lexema y establecer los grupos con los cuales se forman las palabras válidas que comparten dicho lexema. Estas modificaciones se ven también facilitadas por el hecho de utilizar un generador de analizadores, en lugar de programar directamente en un lenguaje de propósito general. Actualmente estas modificaciones carecen de dificultad ya que, como se ha dicho con anterioridad, se dispone de una base de datos que facilita este tipo de operaciones. Esta base de datos mantiene la mínima información para, a partir de ella, poder generar los ficheros que posteriormente serán compilados para generar el analizador léxico [Vilares, Graña y Pan, 1996].

La interfaz de usuario con respecto al analizador léxico

El mantenimiento de la información léxica almacenada en la base de datos puede realizarse a través de una interfaz gráfica especialmente diseñada para tal efecto, que trabaja en entornos **Unix** y que en este proyecto se ha reescrito para que pueda ser utilizada a través de **Internet** desde cualquier máquina que posea acceso y un navegador gráfico. Las posibles operaciones a realizar por el usuario se corresponden con cada uno de los tres modos de la interfaz y que en una sección posterior serán explicadas en detalle:

• **Modo de consulta, modificación y borrado**

Todos los campos, con todos sus valores, se ponen a disposición del usuario a través de unos menús, de forma que se puede construir fácilmente una transacción y realizar operaciones de **browser** sobre las raíces ya introducidas. La figura 5 muestra la pantalla asociada con este aspecto.

• **Modo de introducción**

Este modo es válido para la introducción de nuevas raíces en todas las categorías, salvo en los verbos irregulares, los cuales, debido a su complejidad, disponen de un modo propio. Se incluye un pequeño generador de formas, mediante el cual el usuario puede verificar que el lema introducido se asigna al grupo de flexión correcto. La figura 6 presenta la pantalla asociada al modo de introducción.

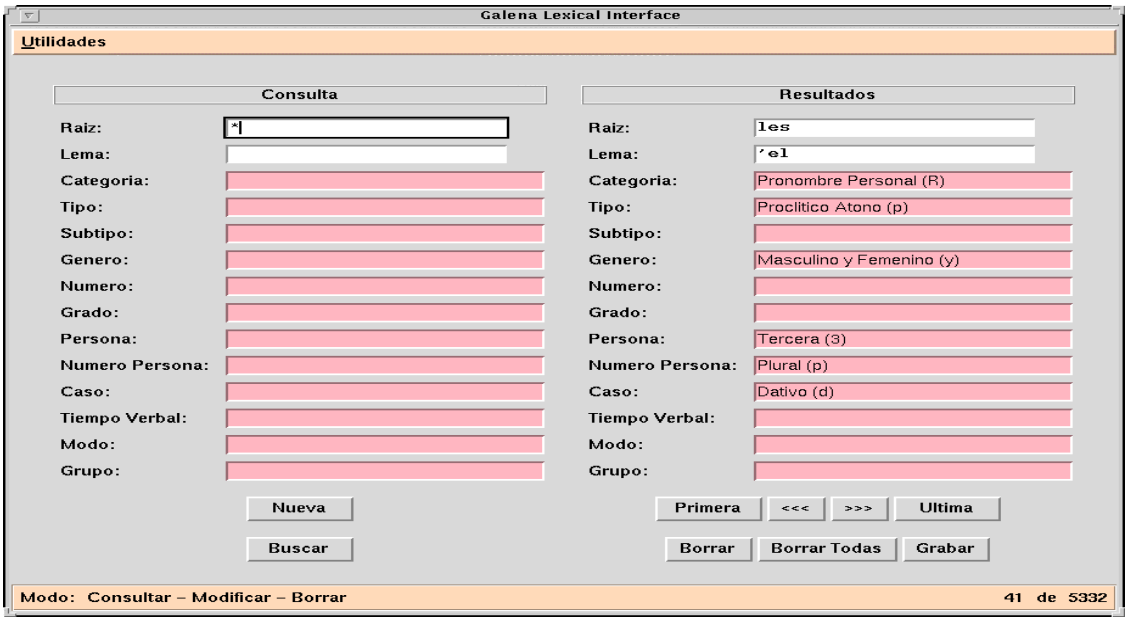


Figura 5 . Modo de consulta, modificación y borrado en la versión **Unix**.

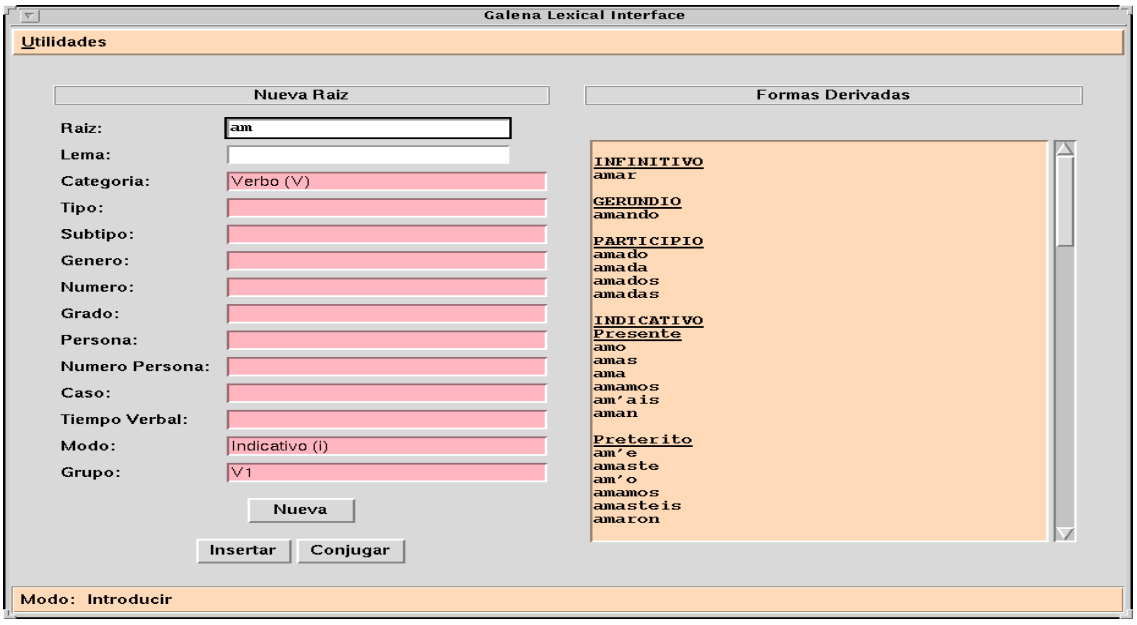


Figura 6. Modo de introducción en la versión **Unix.**

• **Modo de introducción de verbos irregulares**

Debido al hecho de que cada verbo irregular involucra a más de una raíz, se proporciona al usuario un modelo de cada grupo de irregularidad verbal, con un juego de raíces por defecto. La otra problemática principal se centra en el hecho de que los pronombres enclíticos pueden introducir cambios de acentuación en las raíces. Sin embargo, la interfaz es capaz de generar automáticamente las raíces extra que deben ser introducidas para estos casos. El generador de formas muestra algunas de las formas con enclíticos, tal como puede verse en la figura 7.

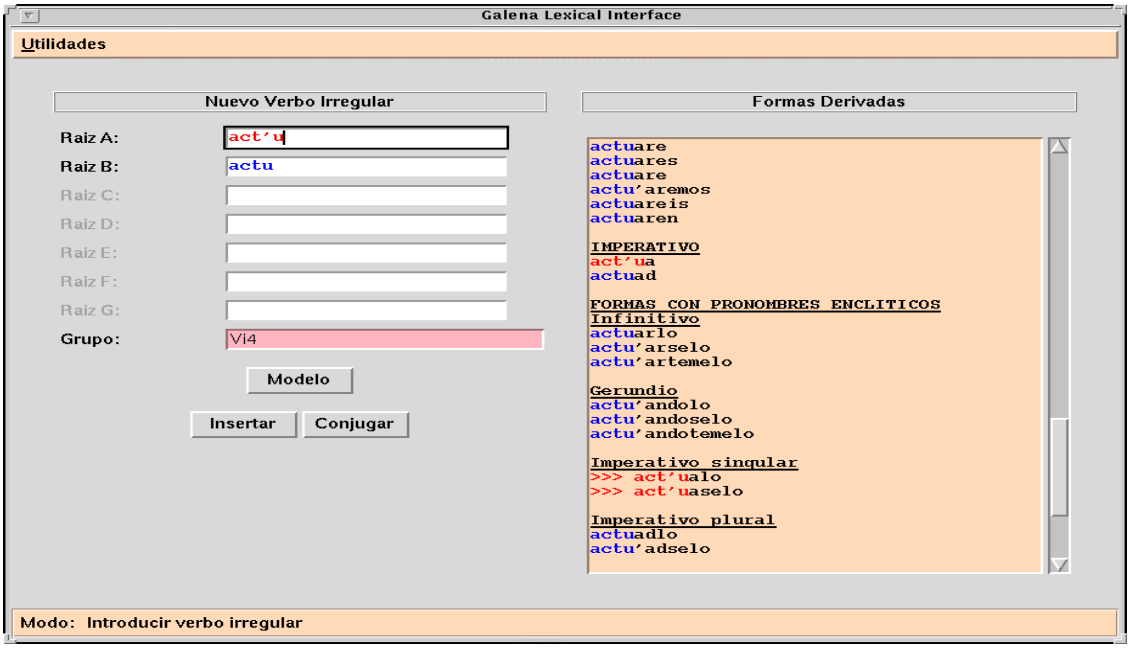


Figura 7. Modo de introducción de verbos irregulares en la versión **Unix**.

Posteriormente, a partir de toda esta información léxica, el proceso de compilación automáticamente construye el etiquetador, que en nuestro caso supone construir el código FLEX del autómata finito que actuará como etiquetador léxico.

Análisis e implementación

Introducción

Antes de pasar a realizar un análisis más detallado de la aplicación desarrollada en este proyecto, se debería hablar de aquellas herramientas que han sido utilizadas para llevarlo a cabo.

En un principio, se pensó que la solución sería utilizar como herramienta principal **HTML**, junto con **CGIs** desarrollados en un lenguaje como **PERL** o **C** para acceder a la base de datos que la aplicación debería utilizar, y construir las páginas que mostrarían los resultados, dado que lo que se pretendía era crear una interfaz que pudiese ser visualizada por un **browser** de páginas **Web**. Sin embargo, finalmente concluimos que estas herramientas no proporcionaban toda la potencia y la funcionalidad que se necesitaba.

Por tanto, viendo que esta posibilidad no serviría, se tomaron como modelo algunas aplicaciones con objetivos similares a los que en este proyecto se perseguían, esto es, construir una interfaz de usuario que pueda ser visualizada en un **browser**.

Después de realizar una pequeña búsqueda en **Internet**, se encontraron algunas aplicaciones desarrolladas con este fin. Algunas incluso relacionadas con el procesamiento de los lenguajes naturales, y que podían ser ejecutadas y visualizadas por un navegador. Además, estas poseían todas las funcionalidades de una interfaz de usuario (menús desplegables, áreas de texto, campos editables, botones, etc.) y podían realizar operaciones con datos procedentes de bases de datos a las que accedían, es decir, hacían algo semejante a lo que en este proyecto se pensaba construir. Estas aplicaciones estaban desarrolladas en el lenguaje de programación más común y el más idóneo para desarrollar herramientas que puedan trabajar en un **browser**, **Java**.

Dado que lo que ofrecían aquellas aplicaciones seguía la línea de lo que se quería desarrollar en este proyecto, se decidió profundizar más en el estudio del lenguaje y ver las posibilidades que ofrecía.

Al final se llegó a la conclusión de que si realmente se quería desarrollar una interfaz para el analizador léxico de GALENA, que pudiese trabajar en **Internet**, y que se semejase mucho al que ya había sido desarrollado para trabajar en entornos **Unix**, se debería utilizar **Java** para programar la aplicación.

Así pues, la aplicación desarrollada en este proyecto está programada íntegramente en **Java**, lenguaje del que a continuación se realizará una descripción de las características que se han aprovechado para crear la interfaz.

Como gestor de bases de datos se ha utilizado el **MSQL**, un gestor de libre distribución, instalado en una maquina bajo una plataforma **Unix** y de probada eficacia.

Para poder trabajar con este gestor de base de datos hubo que traerse de **Internet** e incorporar a la aplicación un paquete de clases de **Java** desarrolladas por Darryl Collins y que proporcionan todas las facilidades necesarias para realizar operaciones con bases de datos gestionadas por un servidor **MSQL** (consultas, modificaciones, borrados de tuplas, etc.)

Finalmente hay que señalar que para que la aplicación pueda ser ejecutada por un usuario, este debe cargar en su navegador una página **Web** desde la cual podrá arrancarla. Ésta es la página de presentación del sistema GALENA y desde la que se puede lanzar la interfaz de usuario desarrollada, creada con el lenguaje de etiquetas **HTML**.

Después de haber mencionado las herramientas a utilizar para el desarrollo de este proyecto, lo más conveniente es explicar en detalle los objetivos de la aplicación. Como ya previamente hemos mencionado, se trata de crear una interfaz para el

analizador léxico de GALENA, semejante a la que ya se había desarrollado para trabajar con **Unix,** que pueda ser ejecutada por cualquier persona que posea acceso a **Internet** y un **browser**, sin importar bajo que sistema operativo esté trabajando su máquina. Esta interfaz permite realizar todas las operaciones necesarias sobre la base de datos mantenida en el sistema y que se utiliza para generar los ficheros que serán compilados para construir el generador léxico. Dichas operaciones abarcan tanto la introducción de nuevas tuplas en la base de datos, es decir nuevas raíces y lemas de palabras con todas sus características léxicas (género, número, categoría, etc.), como la consulta y modificación de las existentes, así como la conjugación de verbos regulares e irregulares y la presentación de todas las formas derivadas de una palabra que esté presente en la base de datos (nombres, preposiciones, adjetivos, etc.).

También existe la posibilidad de eliminar aquellas raíces y lemas que se deseen borrar.

Utilidad de la aplicación

La pregunta que siempre surge a la hora de describir algo es para qué sirve, en qué nos puede ayudar.

El objetivo de la interfaz del analizador léxico de GALENA desarrollado, es que la base de datos mantenida en el sistema pueda serlo fácilmente por cualquier usuario, gracias a las ventajas que supone el utilizar componentes visuales para el manejo de las aplicaciones, como son los botones, menús desplegables, cuadros de diálogo, etc.

Pues bien, gracias a esta interfaz desarrollada se podrán realizar diversas operaciones sobre la base de datos, como anteriormente se mencionó, pero desde cualquier máquina que posea acceso a **Internet**, sin necesidad por tanto de que el usuario posea una versión de la interfaz en su máquina.

Esto supone una gran ventaja, puesto que se evita el tener que adaptar y compilar el código de la aplicación para cada plataforma en la que se vaya a utilizar. Además, de esta forma el usuario puede despreocuparse de instalar el programa y de mantenerlo en caso de que se creen nuevas versiones. Esto último se puede realizar en cualquier momento de forma transparente, pues el usuario lo único que tiene que hacer es ir a la pagina **Web** desde donde se puede arrancar la interfaz y arrancarla. Cualquier cambio que se haya introducido será cargado automáticamente.

Ahora que se han explicado las ventajas que supone el haber desarrollado una interfaz que se pueda ejecutar con un navegador **Web**, desde cualquier máquina, nos podemos preguntar acerca de su utilidad.

Esta interfaz permite que un usuario pueda realizar fundamentalmente tres tipos de operaciones con lemas de la lengua española:

- Podrá consultar lemas y raíces ya introducidas, pudiéndose realizar estas consultas de múltiples formas (por categorías de palabras, preguntando por una determinada raíz o lema, buscar lemas que cumplan unas ciertas condiciones léxicas (un cierto numero, persona, grado, etc.), etc.). Una vez que haya hecho una consulta podrá modificar las propiedades de los lemas y raíces resultantes de la búsqueda, e incluso los propios lemas y raíces, guardando los posibles cambios. También podrá eliminar uno o más lemas y/o raíces resultantes de la búsqueda, de forma que ya no figuren en la base de datos posteriormente.
- Podrá introducir nuevos lemas y raíces en la base de datos, etiquetándolas según corresponda dentro de una categoría, y asignándoles unas propiedades léxicas determinadas (género, número, etc.) según corresponda. Así mismo, también podrá observar que formas se pueden derivar de un lema y una raíz determinadas según el grupo en el que se le incluya, pudiéndose de esta forma observar si en base a las formas derivadas el correspondiente lema y raíz están bien categorizados o simplemente si se quiere saber cuales son las posibles formas derivadas de una raíz y un lema dados.
- Finalmente, puesto que en Español existen verbos que no siguen unas reglas léxicas fijas, los verbos irregulares, se podrá trabajar de forma independiente con estos. Esto es, se podrán introducir nuevos verbos irregulares, que aun no estén presentes en la base de datos, así como todas sus raíces irregulares, categorizándolos en un grupo de verbos irregulares de entre los que existen. Además se podrá conjugar cualquier verbo irregular categorizado dentro de un grupo de verbos irregulares, viendo cuales son los resultados de la conjugación, según el grupo en el que se incluya.

Como se puede ver, gracias a las operaciones de generación de formas derivadas de una raíz y un lema de la lengua española a partir de unas reglas, y que esta interfaz puede generar y visualizar, no es necesario como ya se dijo en la introducción del sistema GALENA guardar en la base de datos todas estas formas, sino simplemente la raíz y el lema, además de unas ciertas propiedades léxicas que cumplen. De esta forma, podemos ir actualizando y añadiendo términos a la base de datos de una manera sencilla.

Si además consideramos la ventaja de que este trabajo se puede realizar desde cualquier máquina, independientemente de su situación física, esto redunda en una mejora sustancial del confort de utilización de la aplicación resultante.

Introducción al lenguaje de programación JAVA

Ya que todo el trabajo ha sido desarrollado casi íntegramente utilizando como herramienta básica el lenguaje de programación **Java**, se debe de hacer una pequeña introducción de cómo es y detallar alguna de las características utilizadas aquí.

Desde hace ya algún tiempo, **Java** está en boca de todo el mundo, y se suele presentar como una nueva y revolucionaria tecnología destinada a cambiar nuestra concepción de **Internet** y el uso que de esta red podamos hacer en el futuro.

A pesar de su importancia y de la expectación que despierta, no hay que olvidar de qué se está hablando: ante todo es un lenguaje de programación, aunque presenta características especiales que lo hacen idóneo para su uso en **Internet**.

Java es un lenguaje moderno, por lo que no es extraño que incorpore un gran número de características cuya utilidad ha sido contrastada en otros lenguajes como pueden ser, por mencionar unas pocas, modularidad, orientación a objetos, o eficiencia. Sin embargo, la característica fundamental que lo diferencia de cualquier otro lenguaje de programación es su universalidad: un programa escrito en este lenguaje puede ejecutarse en cualquier tipo de arquitectura **hardware** y sobre cualquier sistema operativo para el que se haya creado una maquina virtual **Java**.

Los orígenes de **Java** se remontan al año 1990, cuando un equipo de la compañía **Sun Microsystems** investigaba en el diseño y la elaboración de **software** para pequeños dispositivos electrónicos de consumo. Este equipo se embarcó en el desarrollo de un nuevo lenguaje de programación que permitiese, en la medida de lo

posible, desarrollar programas destinados a funcionar en cualquier tipo de plataforma, sin necesidad de cambios. La primera versión del lenguaje se denominó **Oak**.

A comienzos de 1993, el mundo asistía a una auténtica revolución: la aparición de nuevas herramientas gráficas que facilitaban en gran medida la comunicación y la navegación por **Internet**.

En esos momentos, el equipo de desarrollo, que a la sazón había desarrollado **Java** y lo había empleado con éxito para la elaboración de algunos prototipos, consideró que un lenguaje independiente de la plataforma podría ser muy interesante en **Internet**.

Dado que la red consta de ordenadores de muy distintos tipos conectados entre si, se plantearon como objetivo la utilización de **Java** como el lenguaje en el que escribir aplicaciones que pudiesen ejecutarse en un entorno como es **Internet**. Como resultado de su trabajo, desarrollaron un nuevo navegador escrito totalmente en **Java**, que se denominó **HotJava**. Este navegador permitía una novedad muy interesante, la integración de pequeñas aplicaciones o **applets** en el interior de la páginas **Web**.

La integración de **applets** en las paginas **HTML**, permitía incorporar aplicaciones a las páginas, con lo que el usuario puede interactuar con las mismas como si de cualquier aplicación se tratase.

Sin embargo, la integración de aplicaciones en paginas **Web**, accesibles pública y universalmente, plantea algunas cuestiones respecto a la seguridad y las intenciones de las mismas. ¿Qué impide a un **applet** el borrar datos importantes del disco del usuario, o acceder a información de tipo privado o confidencial?. No obstante esto se solucionó, pues el mecanismo de funcionamiento de **Java** permite también asegurar la integridad de las aplicaciones escritas en este lenguaje, y detectar cualquier intento de actuación maliciosa. Así cuando el sistema compila, las referencias no se hacen a direcciones de

memoria, sino que se especifican las variables y objetos por su nombre. Además como este lenguaje no posee punteros, es imposible que un programa pueda realizar operaciones que conduzcan al acceso a una determinada posición de memoria del ordenador.

Por último, el intérprete o máquina virtual que reside en el navegador se negará a ejecutar cualquier código que pretenda acceder a los recursos físicos del ordenador, como la unidad de disco o la red local. Se permite la comunicación del **applet** a través de la red, pero únicamente con el ordenador que lo alberga. Estas reglas a veces son demasiado rígidas, con lo que los navegadores deberían poder dejar elegir al usuario el nivel de seguridad deseado. Por el momento sólo el navegador **HotJava** permite esta relajación de las restricciones.

La compilación en **Java** también presenta una particularidad. A diferencia de lo que sucede con los compiladores tradicionales, el resultado de este proceso no es un conjunto de instrucciones de máquina comprensibles por un cierto microprocesador, sino una representación binaria genérica. Con este fin, se diseñó una máquina ficticia, con un juego de instrucciones asociado. Esta recibe el nombre de Máquina Virtual y no existe ningún microprocesador real que utilice el juego de instrucciones diseñado.

La función del compilador es, por tanto, la generación de código para la máquina virtual consiguiéndose así muchos de los objetivos que siempre se han deseado:

- Generar un código binario compacto y eficiente.
- Aislar al código ejecutable de cualquier máquina particular, etc.

Sin embargo, el generar un código binario que trabaje sobre una máquina virtual no representa una solución definitiva, pues en realidad las aplicaciones tienen que trabajar con microprocesadores reales.

Por ello la máquina virtual se implementa como un módulo **software** que funciona como intérprete de la representación binaria intermedia. Sin embargo, este no procesa las líneas de código sino los **bytcodes** generados en el proceso de compilación. Por supuesto, su interpretación nunca puede ser tan rápida como la ejecución directa de programas nativos del ordenador, por lo que **Java** incurre una pequeña penalización con respecto a las prestaciones que presentan los lenguajes compilados.

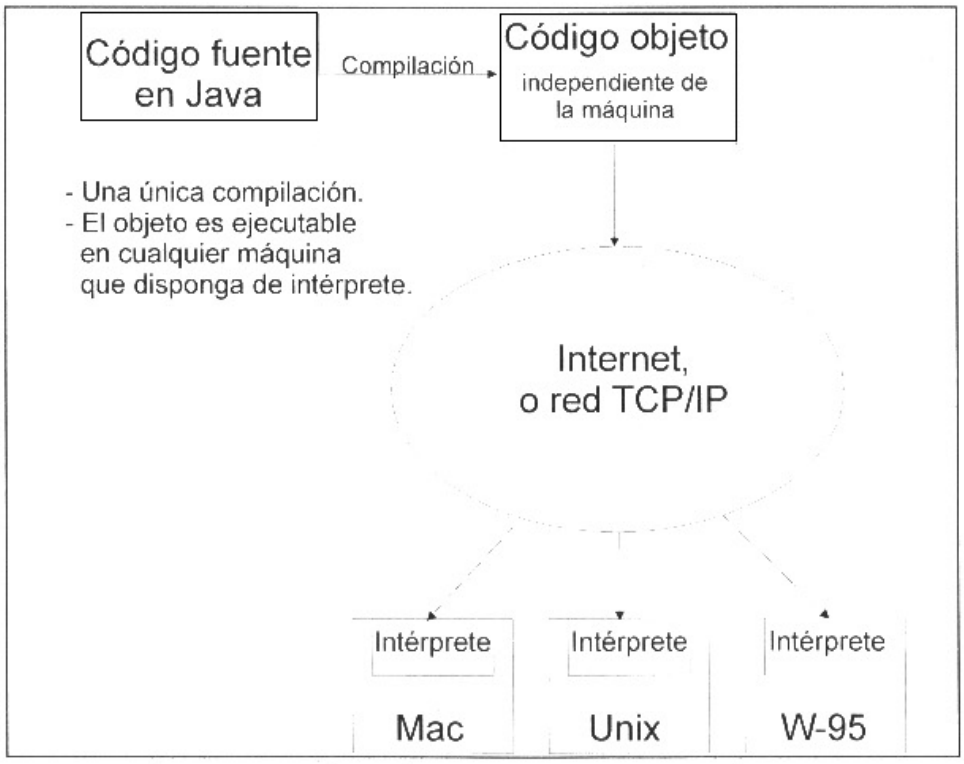


Figura 8. Proceso de compilación **Java**.

Como antes comentamos, **Java** es orientado a objetos. La orientación a objetos es uno de los paradigmas más importantes y significativos de los últimos años. En los lenguajes de programación orientados a objetos como **Java** o **C++**, es posible definir que tipo de objetos van a estar involucrados en nuestro problema, definiendo sus características particulares.

El uso de las técnicas de programación orientadas a objetos permite abordar la resolución del problema de una manera metódica y ordenada que facilita su división y permite de este modo la cooperación de varias personas. Cada objeto puede codificarse de forma independiente, pues sólo se necesita implementar la funcionalidad que de él se ha escrito. Otra ventaja de la orientación a objetos es la reutilización de código.

A continuación pasamos a mostrar la jerarquía de clases propias de este lenguaje, que se han utilizado en este proyecto, explicando brevemente para que se utilizan. Esto hará entender mejor como se estructuran las clases y como consecuencia la comprensión del modelo de la aplicación desarrollada.

- class **java.lang.object**
  - ◊ class **java.lang.boolean**
  - ◊ class **java.awt.BorderLayout (implements java.awt.LayoutManager)**
  - ◊ class **java.awt.Color**
  - ◊ class **java.awt.Component (implements java.awt.image.ImageObserver)**
    - * class **java.awt.Button**
    - * class **java.awt.Checkbox**
    - * class **java.awt.Container**
      - class **java.awt.Panel**
        ⇒ class **java.applet.Applet**
      - class **java.awt.Window**
        ⇒ class **java.awt.Dialog**
        ⇒ class **java.awt.Frame**
        (implements java.awt.MenuContainer)
    - * class **java.awt.Label**
    - * class **java.awt.TextComponent**
      - class **java.awt.TextArea**
      - class **java.awt.TextField**
  - ◊ interface **java.awt.peer.ComponentPeer**
  - ◊ class **java.awt.Event**
  - ◊ class **java.io.File**
  - ◊ class **java.awt.FlowLayout (implements java.awt.LayoutManager)**
  - ◊ class **java.awt.Font**
  - ◊ interface **java.awt.peer.FramePeer (extends java.awt.peer.WindowPeer)**
  - ◊ class **java.awt.GridBagConstraints (implements java.lang.Cloneable)**
  - ◊ class **java.awt.GridBagLayout (implements java.awt.LayoutManager)**
  - ◊ interface **java.awt.peer.LabelPeer**
    (extends java.awt.peer.ComponentPeer)
  - ◊ interface **java.awt.LayoutManager**

- ◊ interface **java.awt.peer.MenuBarPeer (extends java.awt.peer.MenuComponentPeer)**
- ◊ class **java.awt.MenuComponent**
  - ∗ class **java.awt.MenuBar (implements java.awt.MenuContainer)**
  - ∗ class **java.awt.MenuItem**
    - class **java.awt.Menu implements java.awt.MenuContainer)**
- ◊ interface **java.awt.peer.MenuComponentPeer**
- ◊ interface **java.awt.MenuContainer**
- ◊ interface **java.awt.peer.MenuItemPeer (extends java.awt.peer.MenuComponentPeer)**
- ◊ interface **java.awt.peer.MenuPeer extends java.awt.peer.MenuItemPeer)**
- ◊ class **java.lang.Number**
  - ∗ class **java.lang.Integer**
- ◊ interface **java.awt.peer.PanelPeer (extends java.awt.peer.ContainerPeer)**
- ◊ class **java.lang.String**
- ◊ interface **java.awt.peer.TextAreaPeer (extends java.awt.peer.TextComponentPeer)**
- ◊ interface **java.awt.peer.TextComponentPeer (extends java.awt.peer.ComponentPeer)**
- ◊ interface **java.awt.peer.TextFieldPeer (extends java.awt.peer.TextComponentPeer)**
- ◊ class **java.lang.Throwable**
  - ∗ class **java.lang.Exception**
- ◊ class **java.util.Vector (implements java.lang.Cloneable)**
- ◊ interface **java.awt.peer.WindowPeer (extends java.awt.peer.ContainerPeer)**

Todas las clases tienen como padre a la clase **Object**, heredando todas sus propiedades y métodos. Cualquier clase que se cree y no necesite extender ninguna clase es extendida automáticamente de la clase **Object**.

Las clases que están denominadas como interface, son semejantes a las clases propiamente dichas, pero únicamente definen un conjunto de métodos abstractos sin implementación. Es decir, métodos que deberán ser implementados por las clases que hereden de la interface. Una clase puede implementar cualquier número de interfaces, no contienen código y carecen de constructor. Su función es servir como superclase de un conjunto de clases que poseen los mismos métodos, pero con una implementación distinta.

Un paquete es un conjunto de clases relacionadas. En general esta relación se define en base al cometido específico de las clases. Los paquetes se nombran mediante una sucesión de palabras separadas por puntos. Cada nombre de paquete se corresponde con un directorio del mismo nombre en el sistema de ficheros en donde residen las clases.

Asi, por ejemplo las clases que pertenecen al paquete **java.awt** (Abstract Window Toolkit) son un conjunto de clases que representan todos los elementos relacionados con las ventanas y los interfaces[13]. Como ejemplo, la clase **Button** que pertenece a este paquete, permite representar botones en una ventana, conteniendo además todos los métodos para trabajar con el botón[14].

Otro conjunto de clases importantes en este paquete son las que implementan la interface **LayoutManager**, pues permiten disponer la colocación de los componentes en las ventanas de determinadas formas, como puede ser en forma de rejilla, colocados en relación con los bordes de la ventana, etc.

Los componentes de **AWT** están asociados a elementos gráficos del sistema operativo, de modo que se emplean recursos (botones, ventanas, etc.) proporcionados por el mismo, para conseguir que la aplicación escrita en **Java** tenga un aspecto similar al de otras aplicaciones que corren en la misma máquina. El elemento del sistema operativo asociado a un componente **AWT** se denomina **peer**. Para que se puedan relacionar los elementos gráficos del sistema operativo con los elementos gráficos de **Java** existen un grupo de clases[15] que permiten controlar estos elementos desde el propio lenguaje, estas clases son las pertenecientes al paquete **java.awt.peer**, como por ejemplo la clase que representa el par (**peer)** del componente botón, **ButtonPeer**. Otro conjunto de clases muy utilizadas en la implementación de este proyecto son las relacionadas con los menús.

13 Ventanas, diálogos, botones, áreas de texto, etc.

14 Asignarle una etiqueta, asignarle tamaño, etc.

15 Una por cada elemento gráfico que puede se puede implementar en el lenguaje.

Así la clase **MenuBar** permite crea una barra de menú en una ventana y la clase **MenuItem** permite crear elementos en un menú.

La clase **Integer**, que extiende la clase **Number**, proporciona la representación y métodos para tratar con enteros. Esta clase es distinta del tipo primitivo **int** que también sirve para representar números enteros, pero que sin embargo no posee métodos para trabajar con ellos, excepto las operaciones matemáticas de los enteros.

La clase **String** es una de las más utilizadas en el desarrollo del proyecto pues proporciona todos los métodos para trabajar con cadenas de caracteres de tamaño fijo.

La clase **Vector** es una clase que permite construir conjuntos que pueden contener cualquier tipo de objeto, incluso objetos de distinto tipo dentro de un mismo conjunto y que puede crecer o disminuir de tamaño en tiempo de ejecución. Esta clase también proporciona todos los métodos para trabajar con estos conjuntos[16].

La clase **Exception** posee una serie de subclases que se utilizan para indicar la aparición de un error al realizar una operación. El programa puede capturar la excepción y tratar de recuperarse. Si una excepción no se captura en algún punto, el interprete de **Java** terminará la ejecución del programa. El programador puede definir nuevas excepciones. Esta clase proporciona los métodos para tratar las excepciones, es decir para reconocer de que se trata y así actuar en consecuencia.

16 Recuperar elementos, borrarlos, etc.

Análisis preliminar de la aplicación

Se intentará a continuación describir los pasos previos a la implementación. En primer lugar, es importante destacar el hecho de que nos enfrentamos a un tema de muy reciente actualidad. Esto supone una serie de inconvenientes. Por un lado la información disponible es, a priori, mucho menor que la existente sobre cualquier otra materia más asentada. De hecho la bibliografía existente es de muy reciente publicación.

Para comenzar el análisis de la aplicación se planteó en primer lugar que es lo que se tenía y que se debería hacer a partir del material disponible.

En realidad dado que lo que se pretendía era hacer una reingeniería de una aplicación ya construida, adaptándola para que funcionase en **Internet**. En principio se recuperaría el código de la aplicación nativa (desarrollado en **Tcl-Tk**), transformándolo a código **Java**.

Sin embargo esto no es tan sencillo como una simple traducción, pues la naturaleza de los lenguajes es muy distinta, **Java** es orientado a objetos y **Tcl-Tk** no. Además **Tcl** posee muchas funcionalidades para tratar cadenas de caracteres, muy importantes en esta aplicación y que en **Java** deben ser implementadas como métodos.

Otro contratiempo encontrado a la hora de intentar traducir la aplicación de un lenguaje al otro es que **Java** no posee todos los componentes gráficos que posee **Tk**, como por ejemplo menús desplegables contenidos en un elemento cualquiera de la ventana. Sólo se pueden crear menús en la barra de menú de una ventana.

Debido a esto, y dado que esta interfaz usa ampliamente este tipo de menú, hubo que buscar una solución alternativa que le diese a la nueva interfaz un aspecto muy similar a la anterior. La solución fue crear para cada menú desplegable que existe una pequeña ventana con una barra de menú que se abre cada vez que se quiere desplegar el menú.

Finalmente también hubo que solucionar el problema de como acceder a la base de datos a la que necesitaba acceder la aplicación. Para ello se tomó de **Internet** un paquete de clases que permiten realizar todas las operaciones necesarias con una base de datos perteneciente a un servidor **Msql**, que es el servidor que gestiona la base de datos que se necesita.

Por todo lo demás, el trabajo debía consistir en adaptar todas las estructuras de datos de **Tcl** a las correspondientes clases de **Java**, como pueden ser listas de elementos, **arrays**, etc. y seguir la lógica del programa ya desarrollado. Así, se ha procurado que muchos de los métodos implementados siguiesen un algoritmo muy parecido al original, e incluso que sus nombres fuesen iguales a los de los procedimientos de **Tcl**.

Manual de usuario de la interfaz léxica de GALENA

Una vez que hemos visto como ha sido diseñada e implementada la aplicación es el momento de ver su funcionamiento a fondo. Para ello nos iremos ayudando de ejemplos que muestren resultados y que hagan comprender mejor como trabaja la aplicación. También se incluirán varias imágenes que mostrarán el aspecto de la aplicación cuando se están realizando las tareas que puede hacer.

Lo primero que se debe decir cuando se intenta explicar a alguien como funciona un determinado programa es precisamente como arrancarlo.

Pues bien, dado que esta aplicación ha sido diseñada para trabajar en **Internet**, lo primero que se debe hacer es arrancar un **browser** que tenga soporte para **Java**, como puede ser **HotJava** , **Netscape** o **Explorer**[17]. Una vez hecho esto se deberá ir a la siguiente dirección URL, http://obelix.dc.fi.udc.es, con lo que se cargara una pagina **Web** de presentación del Sistema GALENA.

En esta página se deberá pulsar sobre el enlace con la siguiente leyenda “interactive presentation” o bien “presentación interactiva” en la versión en castellano. Con esto se arrancará automáticamente la interfaz gráfica.

La interfaz léxica de GALENA está formada por tres ventanas, que aparecerán según la opción del menú que elijamos. Por defecto cuando se arranca la aplicación aparecerá la ventana correspondiente a la opción de **Consultar-Modificar-Borrar**, como podemos observar en la figura 9.

[17] Se supone que se tiene acceso a Internet.

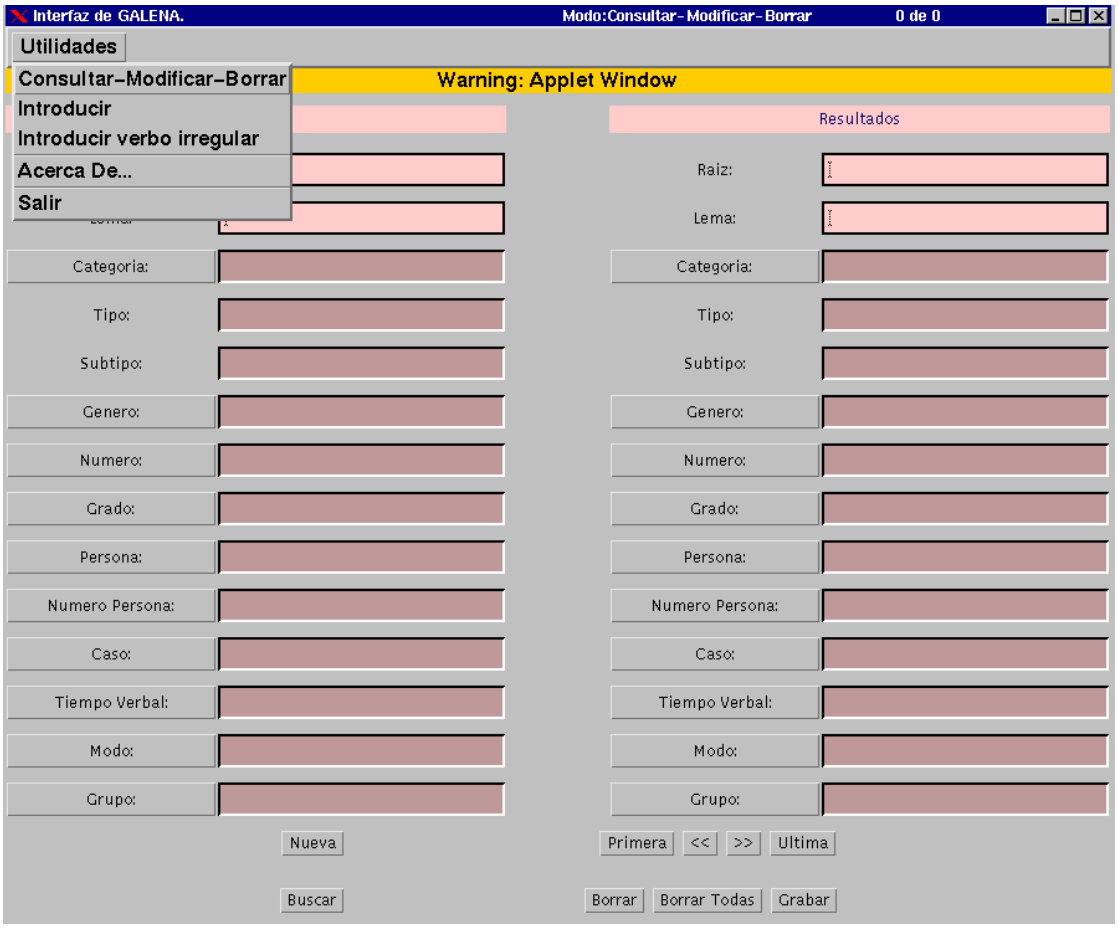


Figura 9. Menú principal de la interfaz.

Como se observa, el menú principal contiene, además de las tres opciones que permiten escoger los tres modos en los que puede trabajar la interfaz, una opción de **Acerca De**, en la cual como se puede ver en la figura 10 nos mostrará información acerca del sistema GALENA.

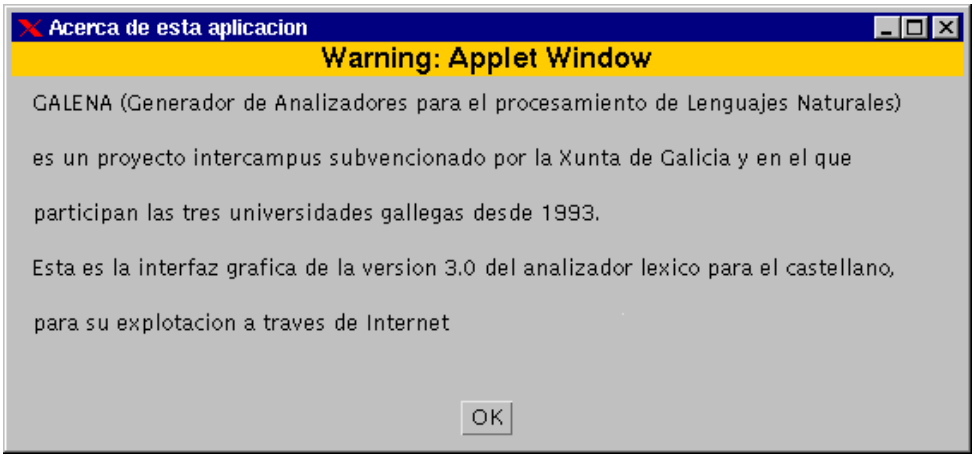


Figura 10. Dialogo Acerca De.

La ultima opción nos permitirá salir de la interfaz.

Finalmente, señalar que este menú es igual en cada una de las tres ventanas, de forma que todas sus opciones están accesibles en cualquiera de los tres modos.

Modo Consultar-Modificar-Borrar

Se empieza por explicar cada uno de los elementos de la ventana del modo **Consultar-Modificar-Borrar**, así como su utilidad.

Como se puede observar en la figura 11, en la ventana existen claramente diferenciadas dos zonas denominadas **Consultas** y **Resultados**, cada una de las cuales está compuesta por una columna de etiquetas con nombres (Raíz, Lema, Categoría, etc.) y otra de campos de texto.

Finalmente en la parte inferior de la ventana existe un conjunto de botones cuya funcionalidad se detallará a continuación. Algunas de estas etiquetas se puede observar que son botones y otras simplemente etiquetas.

Pulsando sobre las etiquetas representadas por botones conseguiremos que se abra una pequeña ventana con un menú en el que podremos elegir una de sus opciones. Al elegir una opción en el correspondiente campo de texto situado al lado se mostrará un texto que indica la opción elegida.

En el caso de que se esté eligiendo una opción del menú **Categoría**, desplegado al pulsar el botón etiquetado con el mismo nombre, dependiendo de esta, se rellenarán los campos de texto asociados con las etiquetas **Categoría**, **Tipo** y **Subtipo**, a diferencia de los demás menús que sólo rellenan el campo asociado con su etiqueta.

Todos estos menús poseen al final una opción etiquetada como "**Sin Valor**", esta opción hará que el correspondiente campo de texto quede en blanco y que por tanto no intervenga a la hora de realizar las operaciones con la base de datos.

De entre todos los campos de texto que aparecen, sólo los que se corresponden con las etiquetas **Raíz** y **Lema** pueden ser rellenados directamente con el teclado, los demás se rellenan de forma automática al seleccionar una opción en su correspondiente menú. Estos dos campos permiten que el usuario introduzca una raíz y/o un lema que desea consultar, en el caso de los campos pertenecientes al lado de consultas, o modificarlos si son los del lado de las respuestas. También hay que decir que los menús que se despliegan al pulsar las etiquetas del lado de **Respuesta**, sirven únicamente para modificar las propiedades de una raíz y un lema que previamente haya sido consultado de la base de datos.

En la figura 11 se muestra como ejemplo el menú desplegado al pulsar el botón "**Categoría**" así como campos rellenados con opciones seleccionadas en sus correspondientes menús.

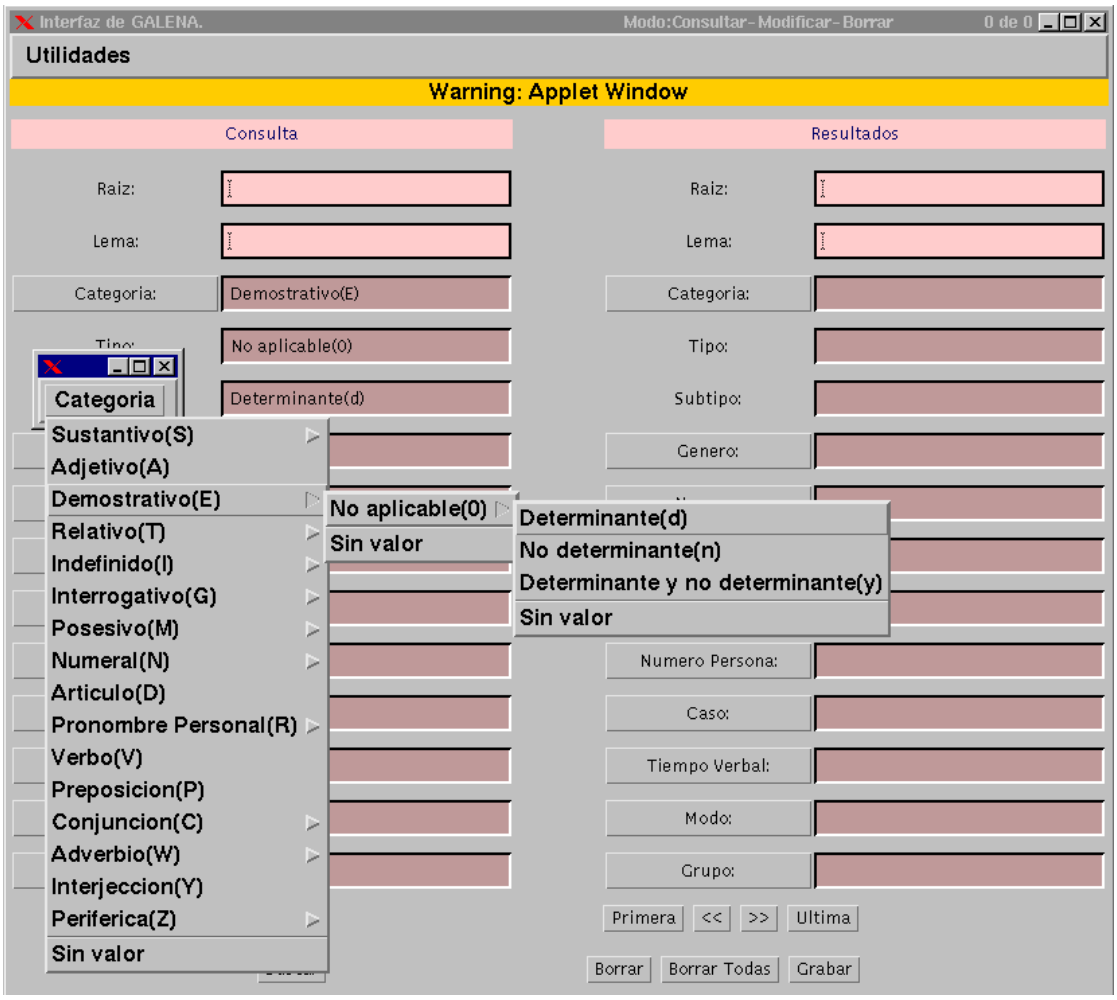


Figura 11. Menu Categoria.

Una vez explicado cómo se pueden rellenar los campos para realizar una consulta o modificar los campos generados en una respuesta, se mostrará la utilidad de los botones situados en la parte inferior de la ventana.

En primer lugar está el botón "**Nueva**", cuya única finalidad es limpiar todos los campos de texto del lado de consultas, para así poder rellenarlos para una nueva consulta.

El siguiente botón está etiquetado como "**Buscar**" y su finalidad es lanzar una consulta a la base de datos para encontrar la o las raíces y lemas que cumplan las especificaciones pedidas en los campos de texto del lado de consultas. Es decir, si por ejemplo se rellena únicamente el campo de texto asociado con la categoría,

seleccionando la opción "**Verbo(v)**" de su menú desplegable y luego pulsamos el botón "**Buscar**", le estaremos pidiendo a la interfaz que busque todos los verbos existentes. Cuando el gestor de base de datos devuelva el resultado de la consulta, la interfaz se encargará de mostrar en los campos de texto del lado de respuestas las raíces y los lemas encontrados, así como todas las propiedades que posean cada par en sus correspondientes campos (categoría, tipo, subtipo, género, número, etc.), ordenados alfabéticamente por el lema, como podemos ver en la figura 12.

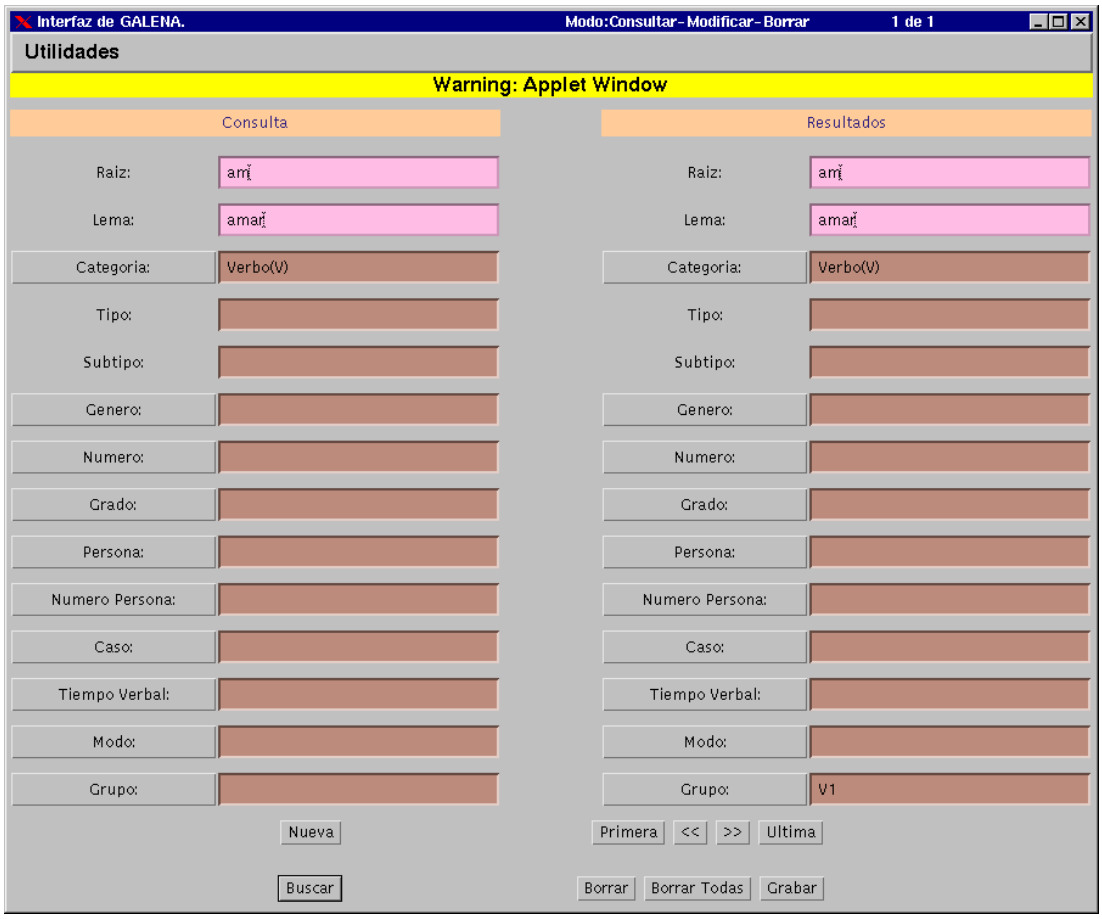


Figura 12. Resultados de una Búsqueda.

Una vez realizada una consulta es cuando tienen utilidad los botones del fondo de la ventana situados debajo de la zona de resultados.

* El botón etiquetado como "<<", hará que en la zona de resultados se muestre el par raíz, lema con sus propiedades, que está inmediatamente antes alfabéticamente del

que se muestra actualmente, en caso de que el mostrado no sea el primero. Si así fuera, pulsarlo no tiene efecto.

* El botón etiquetado como ">>", hará que el la zona de resultados se muestre el par raíz, lema con sus propiedades, que esta inmediatamente después alfabéticamente del que se muestra actualmente, en caso de que el mostrado no sea el último. Si así fuera, pulsarlo no tiene efecto.
* El botón etiquetado como "**Primera**", hará que se muestre el primer par, en orden alfabético, raíz, lema con sus propiedades, resultante de la consulta.
* El botón etiquetado como "**Última**", hará que se muestre el último par, en orden alfabético, raíz, lema con sus propiedades, resultante de la consulta.
* El botón etiquetado como "**Borrar**", permite borrar de la base de datos la tupla que contiene el par raíz, lema mostrado actualmente, previa confirmación del borrado. No se permitirá el borrado cuando la raíz sea especial, ciertas raíces que son peculiares y no se pueden introducir ni borrar con esta interfaz. Una raíz se puede ver que es especial pues aparece el texto "**esp**" en la barra de título de la ventana, cuando se muestra en pantalla.
* El botón etiquetado como "**Borrar todas**" hace lo mismo que el anterior, salvo que en este caso se borran todas las raíces (no especiales) resultantes de una consulta.
* Finalmente está el botón etiquetado como "**Grabar**", este botón permite grabar los cambios que se hagan sobre la raíz, el lema y sus correspondientes propiedades mostrados actualmente en la zona de resultados, previa confirmación de la actualización de los cambios.

  Igual que en el caso del borrado, no se pueden modificar las raíces especiales.

Modo Introducir

El usuario puede entrar en este modo, visualizándose así la correspondiente ventana, si elige la segunda opción de la barra de menú de la ventana que en ese momento esté abierta.

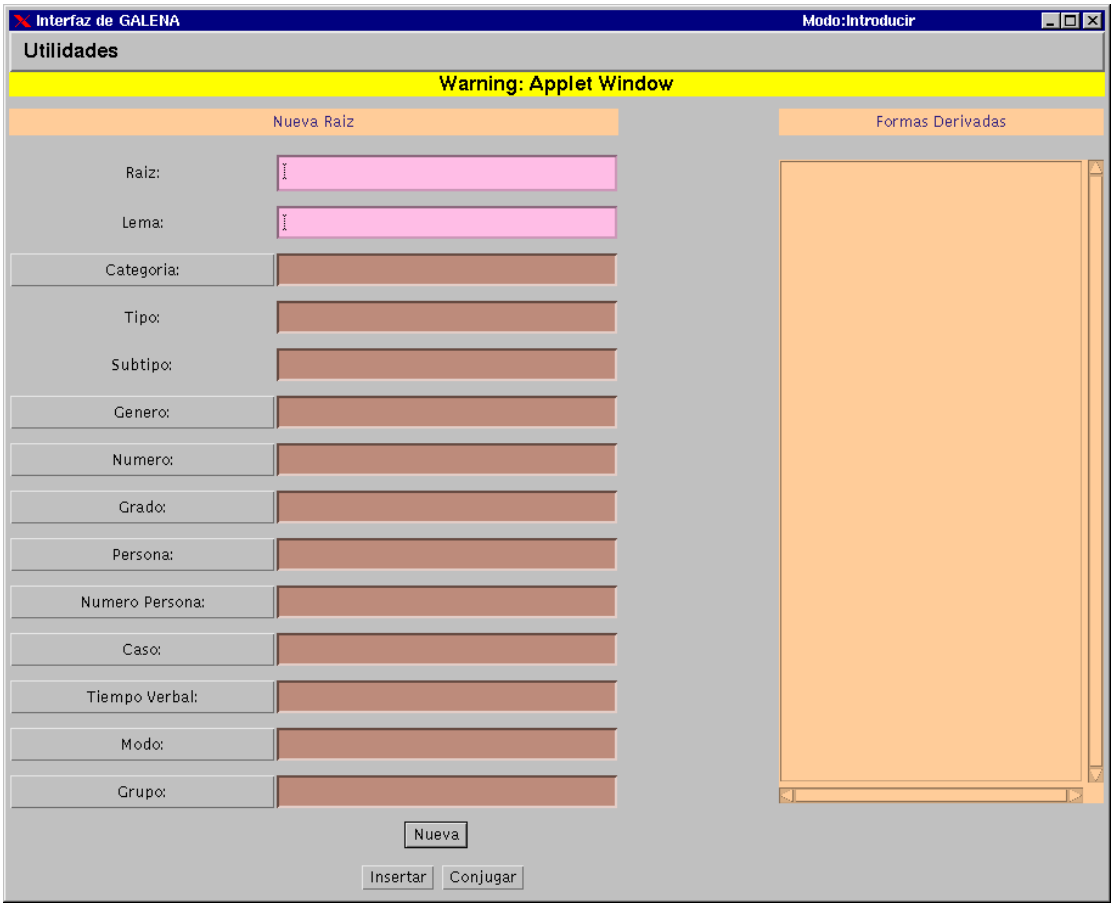


Figura 13. Ventana del modo Introducir.

Como se puede observar en la Figura 13, en la ventana existen claramente diferenciadas dos zonas denominadas **Nueva Raíz** y **Formas Derivadas**. En la primera de ellas se puede observar que es igual a cualquiera de las zonas en las que está dividida la ventana del modo **Consultar-Modificar-Borrar**. Aparecen pues en esta zona una columna formada por botones y etiquetas, y otra por campos de texto.

La segunda zona en la que se divide esta ventana, es simplemente un área de texto, cuya finalidad será mostrar las formas derivadas que se pueden construir dados una raíz y un lema con sus propiedades.

Vistas ya las zonas en las que se divide la ventana, veremos ahora cual es su funcionalidad, y como se puede explotar ésta.

La ventana sirve simplemente para introducir en la base de datos nuevas raíces y lemas con sus correspondientes propiedades de categoría, género, número, etc., salvo cuando se trate de raíces de verbos irregulares, las cuales son introducidas gracias a la ventana destinada exclusivamente a ello, y que en el siguiente apartado explicaremos.
También podremos utilizar esta estructura para generar y visualizar las formas derivadas de una raíz y un lema determinados.

Para poder introducir nuevas raíces y lemas en la base de datos, deberemos en primer lugar, escribir en los correspondientes campos la raíz y el lema deseados[18], sin que ninguno de ellos pueda quedar en blanco.
A continuación deberemos obligatoriamente categorizar la nueva raíz, escogiendo una categoría determinada dentro del menú que se despliega al pulsar el botón etiquetado como **"Categoría"** en la zona denominada **Nueva Raíz** , como se puede ver en la figura 13.
Una vez completados estos tres pasos obligatorios, se puede asignar propiedades a la nueva raíz y al nuevo lema como ejemplo número, género, tiempo verbal, modo, etc., desplegando los correspondientes menús, y en los cuales se pueden elegir éstas, como se puede observar en la figura 14 que muestra un ejemplo.
Cuando hallamos completado todos estos pasos anteriores ya estaremos en condiciones de grabar la nueva raíz. Ello se puede hacer pulsando el botón situado en la parte

[18] Campos de la zona denominada **Nueva Raíz** asociados a las etiquetas **Raíz** y **Lema**.

inferior de la ventana, etiquetado con el nombre **"Insertar"**, el cual arrancará el proceso que permite guardar correctamente en la base de datos la nueva raíz y el lema con sus correspondientes propiedades. Antes de crear esta nueva tupla en la base de datos, el proceso se encargará de comprobar que como mínimo se han completado los campos raíz, lema y categoría, así como de verificar que si se ha completado el campo correspondiente a la propiedad de **Grupo**, el lema que se introduce es correcto léxicamente y cumple las propiedades que le encuadran dentro de ese grupo.
Si falla alguna de estas comprobaciones se emitirá el correspondiente mensaje de error y se cancelará el proceso de creación de la nueva tupla.

Finalmente queda por explicar como generar las formas derivadas de una raíz y un lema determinados. Para ello, en primer lugar deberemos introducir en los correspondientes campos de texto la raíz y el lema deseados, así como sus propiedades, teniendo en cuenta que el proceso de generación atiende únicamente a las propiedades, y a la raíz y el lema que se le han introducido en los correspondientes campos, y no a que las propiedades sean correctas para esa raíz y ese lema. De forma que si estas propiedades no son correctas las formas derivadas serán totalmente distintas de las que les corresponderían. Así por ejemplo vemos en la figura 14 que se han asignado unas propiedades erróneas al lema "**Casa**", obteniéndose unas formas derivadas que no son las que le corresponden.

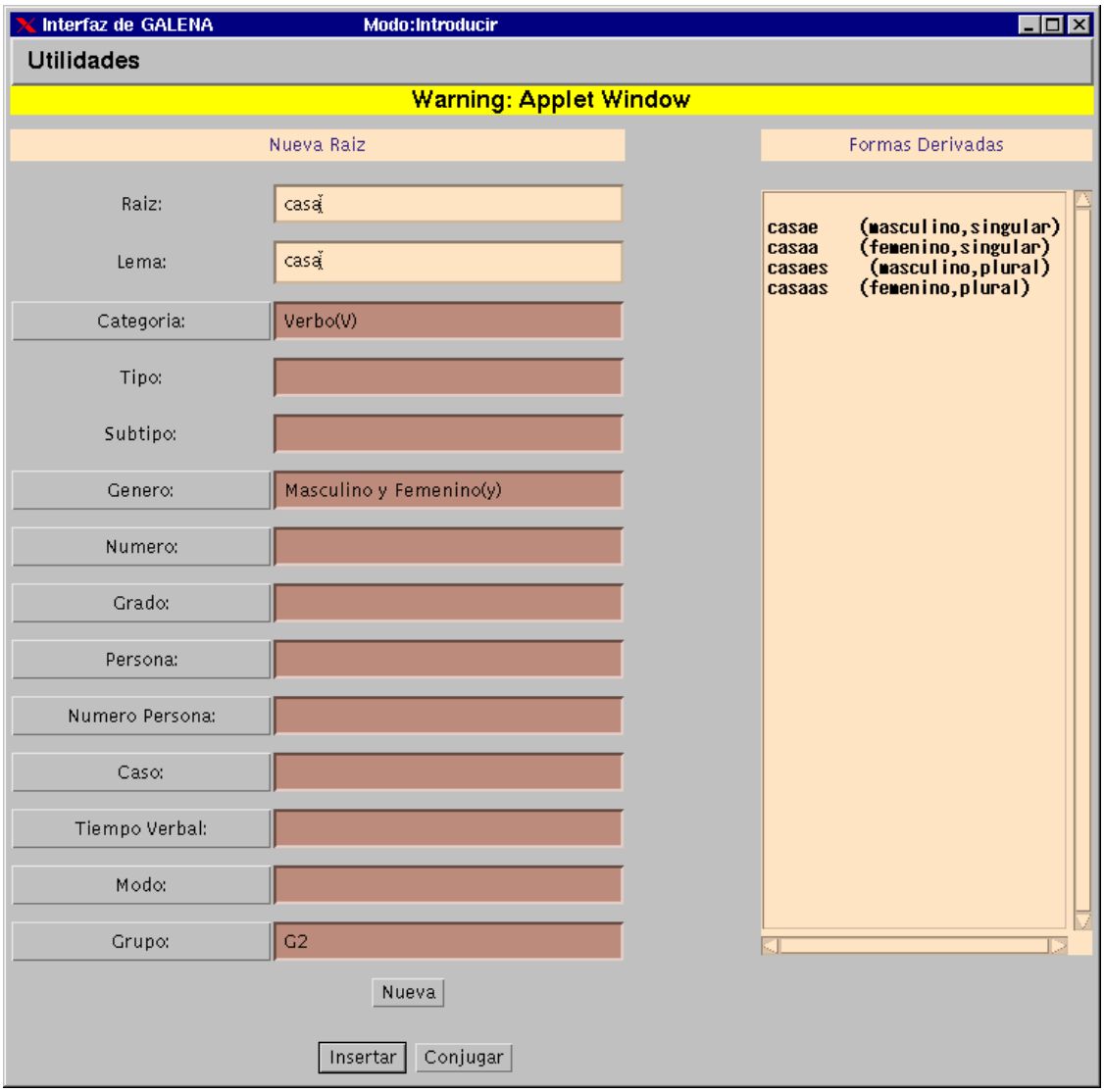


Figura 14. Propiedades erróneas de una raíz y un lema

Si se completa el campo asociado a la etiqueta de "**Grupo**", la generación de las formas derivadas se hará conforme corresponde a una raíz y un lema pertenecientes al grupo que se halla escogido. Así, si por ejemplo se escoge como raíz "**part**", como lema "**partir**" y como grupo el "**V3**", al generar las formas derivadas lo que se estará haciendo es considerar que la raíz y el lema corresponden a un verbo regular de la tercera conjugación, y por tanto lo que hará el proceso de generación será conjugar este verbo según las formas correspondientes a esta conjugación, como podemos observar en la figura 15.

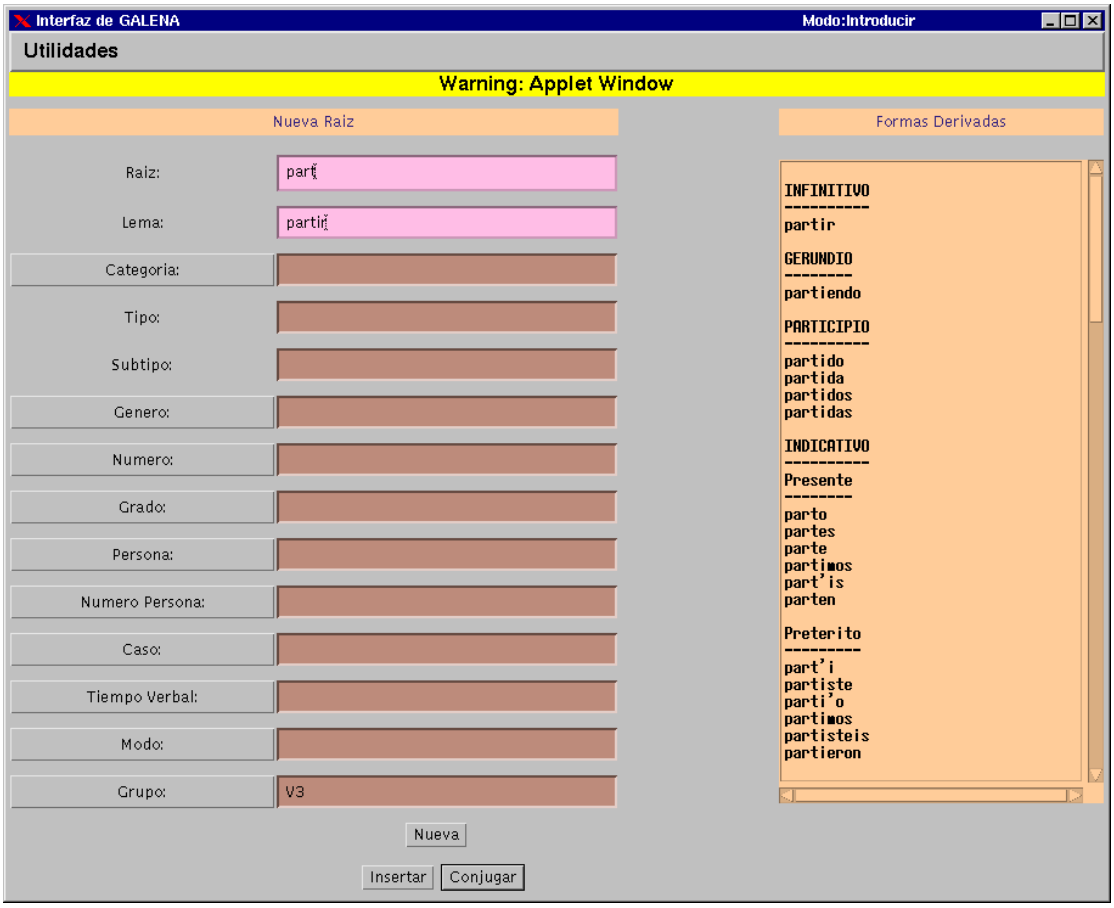


Figura 15. Resultado de conjugar una raíz y un lema.

Si en este caso en el campo **Grupo** pusiéramos que el verbo es del grupo **V2**, se conjugaría como verbo regular de la segunda conjugación, y por tanto los resultados no serían correctos. Si no se completa el campo de texto correspondiente a la propiedad de grupo, las formas derivadas serán el propio lema.

Esta ventana también ofrece la posibilidad de que en la ventana perteneciente al modo **Consultar-Modificar-Borrar**, se realice una consulta para comprobar que existe en la base de datos una raíz y un lema determinados con sus correspondientes propiedades, completando para ello los correspondientes campos de texto conforme se explicó en el apartado correspondiente a este modo.

Una vez hecha esta consulta se podrá escoger en la barra de menú el modo **Introducir** y automáticamente los campos de texto de la zona denominada **Nueva Raíz** estarán completados con los mismos datos que tenían los campos de la ventana del modo **Consultar-Modificar-Borrar** y que se utilizaron para hacer la consulta. Esto supone dos ventajas:

- Por un lado, en caso de que al realizar la consulta encontráramos que la raíz y el lema consultados, con sus correspondientes propiedades, ya existían en la base de datos, al pasar a la ventana del modo **Introducir**, ya tendremos los datos completos para realizar el proceso de generación de las formas derivadas.
- Por otro lado, si esta raíz y lema no existían en la base de datos, tendremos la oportunidad de insertarlos en ella sin tener que volver a completar todos los datos en la nueva ventana. Además podremos igualmente generar las formas derivadas correspondientes.

En cuanto a las funcionalidades que ofrece esta ventana, decir que si pulsamos el botón etiquetado como **Nueva**, lo que haremos será poner en blanco todos los campos de texto de la zona denominada **Nueva Raíz**. Si después de hacer esto, escogemos en la barra de menú la opción del menú **Utilidades, Consultar-Modificar-Borrar**, los campos de texto de la zona denominada **Consultas** de la ventana correspondiente a este modo también estarán en blanco. Sucederá lo mismo si pulsáramos el botón con igual nombre en la ventana **Consultar-Modificar-Borrar** y luego pasáramos a la ventana **Introducir**.

Finalmente indicar que al mostrar las formas derivadas en el área de texto correspondiente, situada en el lado derecho de la ventana, éstas aparecerán unas debajo de otras en orden, y en el caso de los verbos las formas conjugadas aparecerán separadas en los distintos modos existentes (indicativo, subjuntivo, imperativo, etc.) y a

su vez dentro de estos modos separadas en los distintos tiempos verbales (presente, futuro, etc.), como se puede ver en la figura 15.

Así mismo, cuando el generador de las formas derivadas modifique la raíz de la forma derivada, como cuando por ejemplo las reglas de acentuación impliquen que la raíz deba ser acentuada, esta forma derivada se marcará con el símbolo “>>”, como se observa en la figura 16.

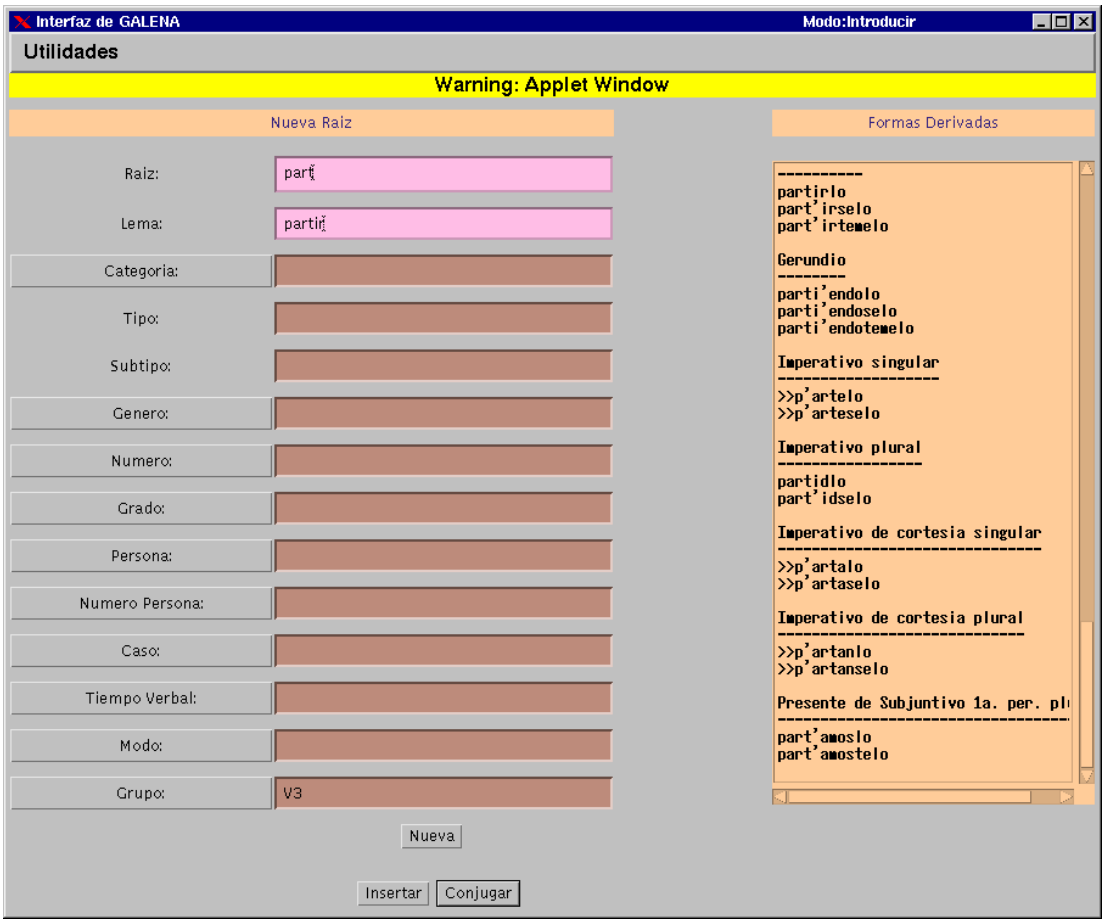


Figura 16. Formas derivadas con la raíz modificada.

Señalar por tanto que el generador de las formas derivadas, acentuará de forma automática estas formas cuando las reglas de acentuación así lo requieran.

Modo Introducir Verbo Irregular

Este es el tercero de los modos que posee la interfaz y fue creado exclusivamente para tratar con los verbos irregulares, dada la naturaleza especial de estos.

En general el aspecto es similar a los otros dos modos, especialmente al del modo **Introducir**, pues su funcionalidad es la misma, sólo que para verbos irregulares.

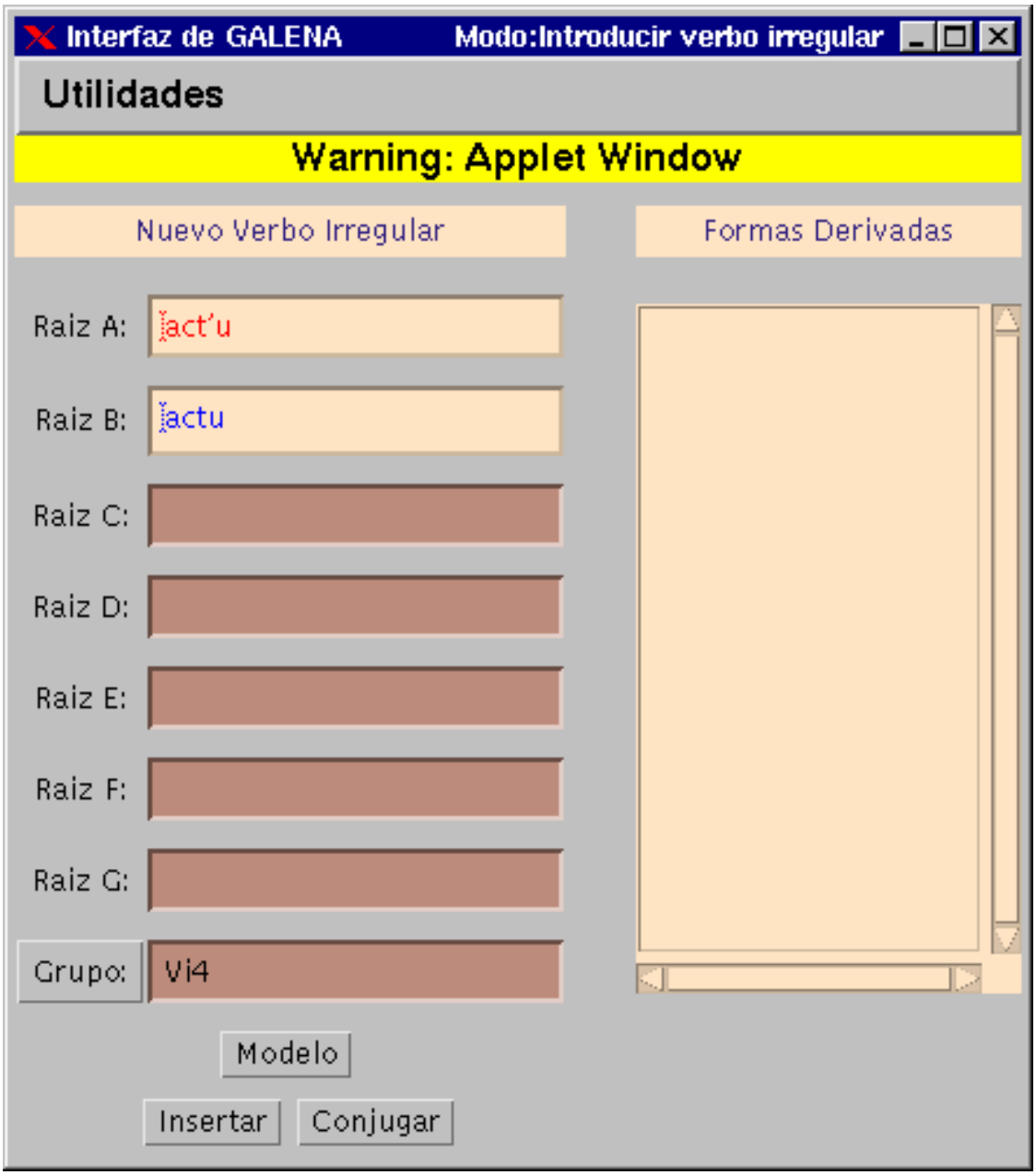


Figura 17. Ventana del modo **Introducir Verbo Irregular.**

Así pues, la ventana se divide en dos zonas, una zona denominada **Nuevo Verbo Irregular** en la cual se encuentran una serie de campos de texto asociados con etiquetas, y otra zona que contiene únicamente un área de texto y que se denomina **Formas Derivadas**, cuya finalidad es la misma que la correspondiente área de texto en la ventana del modo "**Introducir**".

Para introducir un verbo irregular con todas sus raíces en la base de datos, en primer lugar se debe escoger el grupo al que pertenece de entre los que aparecen en el menú que se despliega al pulsar el botón etiquetado como "**Grupo**". Según el grupo que se escoja, se habilitará un numero determinado de campos de texto para introducir las raíces que pertenecen al nuevo verbo en ellos. Cuando hallamos introducido todas las raíces correspondientes a ese verbo, y que deben ser tantas como campos de texto estén habilitados, para guardar el nuevo verbo con sus raíces en la base de datos se debe pulsar el botón de la parte inferior derecha etiquetado como "**Insertar**". Entonces aparecerá un diálogo que nos pregunte si realmente queremos guardar el nuevo verbo, si aceptamos éste será guardado.

Finalmente hay que mencionar cual es la utilidad de los botones etiquetados como "**Modelo**" y "**Conjugar**". El primero de ellos provoca que al pulsarlo aparezcan en los campos de texto correspondientes a las raíces de los verbos, todas las raíces del verbo que sirve de modelo del grupo que actualmente está seleccionado. El segundo de ellos permite conjugar el verbo que actualmente aparece en los campos de texto, mostrándose los resultados en el área de texto de la zona denominada **Formas Derivadas**. En la figura 18 se puede ver como en los campos de texto aparecen las raíces del verbo que sirve como modelo del grupo **Vi18** y en el área de texto aparece este verbo conjugado, consecuencia de haber escogido primero el grupo **Vi18** del menú

que se despliega al pulsar el botón "**Grupo**", de pulsar después el botón "**Modelo**" y finalmente el botón "**Conjugar**".

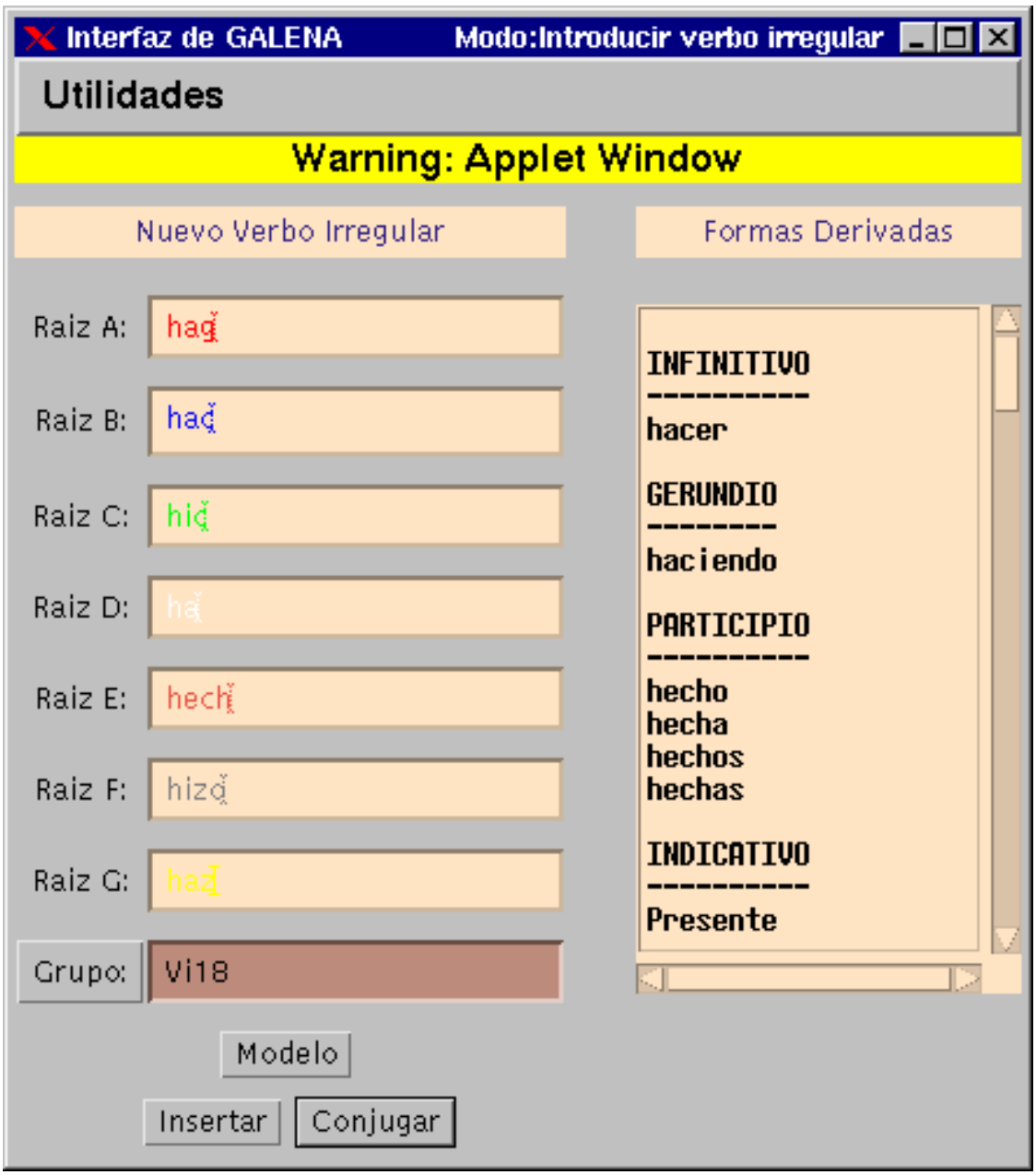
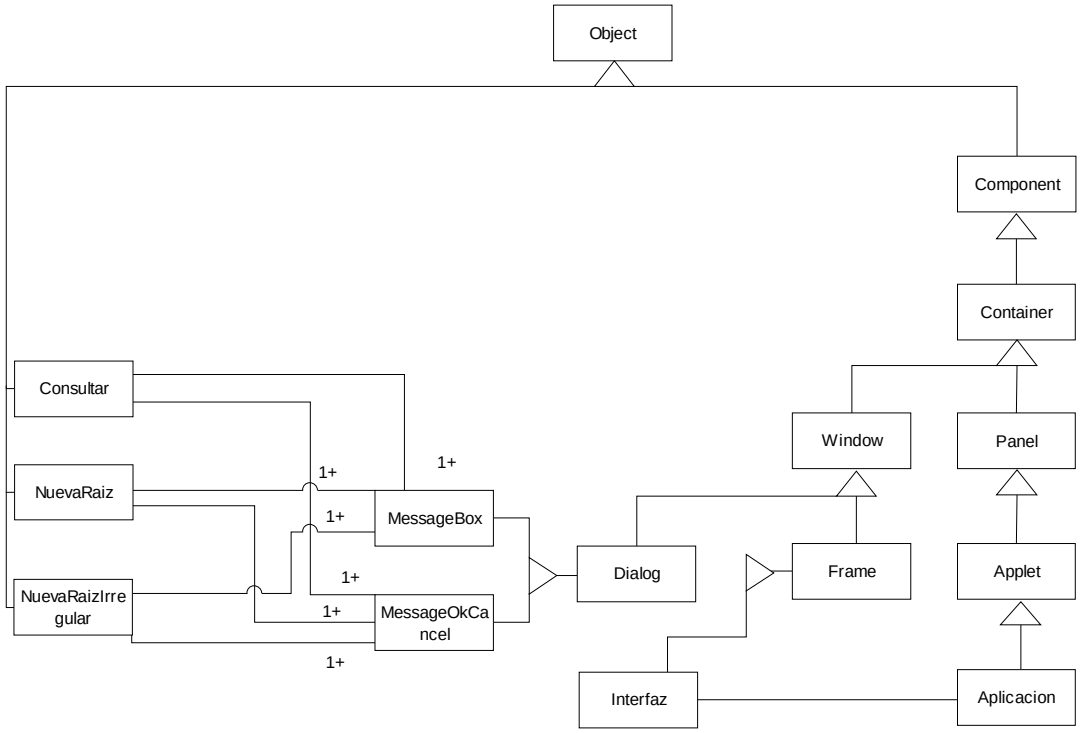


Figura 18. Formas derivadas de un verbo irregular.

Manual técnico de la aplicación

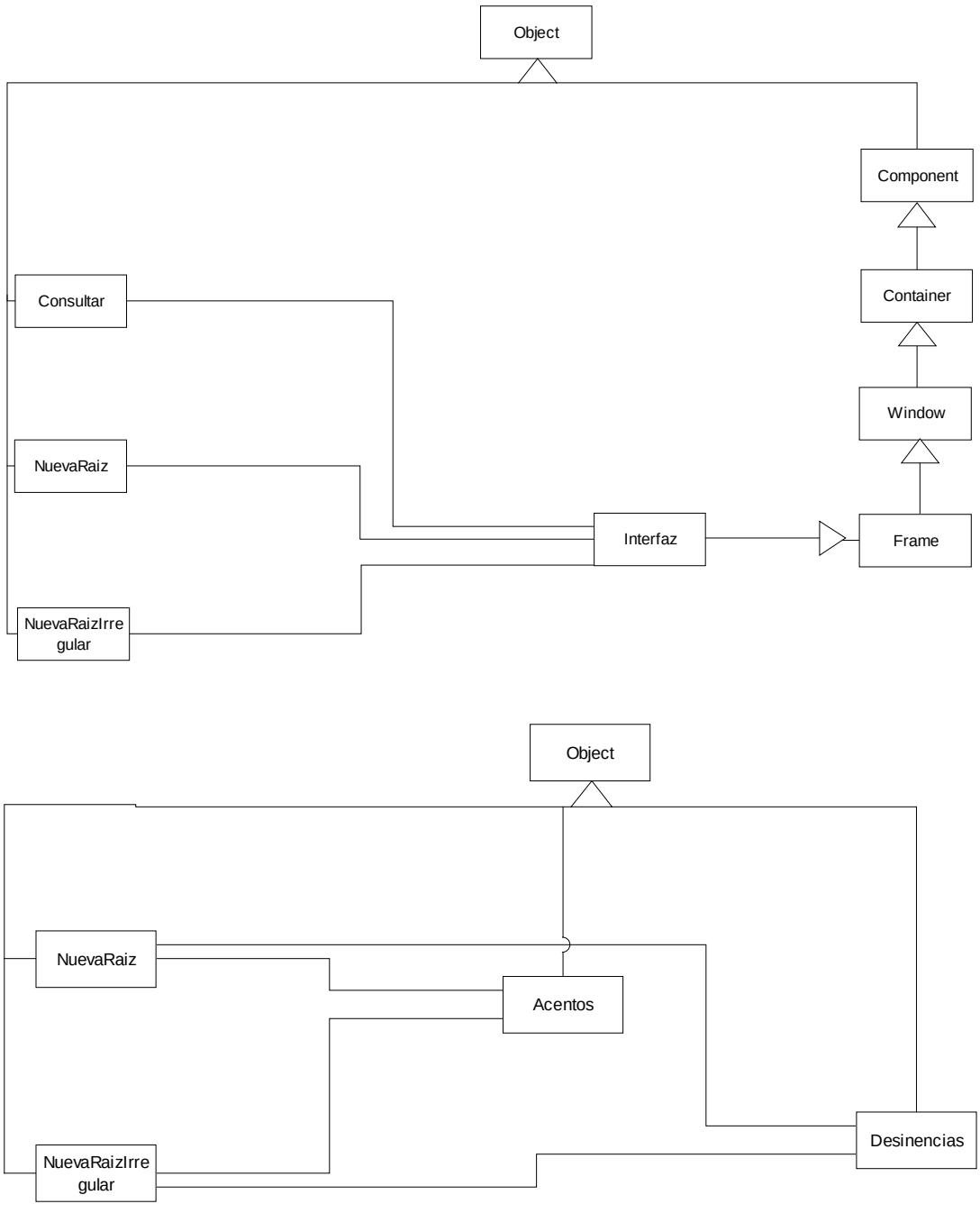
Una vez que hemos visto como se puede arrancar la aplicación desarrollada en este proyecto, y que se vio como se puede utilizar correctamente, pasamos a describir como es la arquitectura de dicha aplicación.

Dado que esta aplicación se ha desarrollado íntegramente en **Java**, y este es un lenguaje de programación orientado a objetos, en primer lugar se incluye el modelo estático de objetos, utilizando la metodología **OMT**[19]. Este modelo permite determinar que clases existen en la aplicación y cuales son las relaciones existentes entre éstas.



Modelo de objetos de la aplicación.

19 Object Modeling Technique. Ver apendice I.



Modelo de objetos de la aplicación.

En cuanto a la jerarquía de clases de la aplicación desarrollada, es la siguiente:

- Object
    - Component
        - Container
            - Panel
                - Applet
                    - Aplicación
            - Window
                - Frame
                    - Interfaz
                    - MenuCaso
                    - MenuCategoria
                    - MenuGenero
                    - MenuGrado
                    - MenuGrupo
                    - MenuGrupoIrr
                    - MenuModo
                    - MenuNumero
                    - MenuNumeroPersona
                    - MenuPersona
                    - MenuTiempoVerbal
                    - AboutDialog
                - Dialog
                    - MessageBox
                    - MessageOkCancel
    - Acentos
    - Desinencias
    - Consultar
    - NuevaRaiz
    - NuevaRaizIrregular

A continuación se muestran todas las clases creadas específicamente en esta aplicación, así como los métodos asociados, separados en los que son públicos y los que son privados, comentando aquellos que requieran una mención especial.

En primer lugar empezaremos por la clase principal de la aplicación, desde la cual se arranca toda la interfaz. Esta es la clase **Aplicación**.

Métodos privados:

Métodos públicos:

- **init**. Arranca la interfaz, creando un objeto de la clase **Interfaz**
- **main**. Crea el **applet** y llama al método **init** para inicializar la interfaz.

Clase **Interfaz**

Métodos privados:

- **inicializaCamposTexto**. Este método crea y rellena los arrays que contienen los campos de texto existentes en las ventanas de la interfaz. Estos arrays permiten trabajar fácilmente con los campos de texto, pues para cada conjunto de campos que existen en las ventanas se crea un array, de forma que cada campo se puede identificar de forma concreta, pudiendo rellenarse o leer lo que contiene en cualquier momento.
- **ResetFields**. Este método provoca que cuando sea llamado, el conjunto de campos de texto que se le pasa como parámetro, se pongan en blanco.
- **creaSeparador.** Crea un espacio en blanco que separa las zonas **Consultar** y **Resultados**.
- **creaBarraMenu.** Este método crea la barra de menú existente en todas las ventanas de la interfaz.

- **creaTituloConsModBorr.** Este método crea los títulos de las zonas en las que se divide la ventana del modo **Consultar-Modificar-Borrar**.
- **creaLineaConsModBorr.** Cada vez que se llama a este método se crea en la ventana del modo **Consultar-Modificar-Borrar** una línea formada por una etiqueta o un botón y un campo de texto para la zona de consultas y otra etiqueta o botón, con su correspondiente campo de texto para la zona de resultados.

  En cada llamada, la línea se creará en una coordenada Y de la ventana, según la que se le pase como parámetro. Además, a las etiquetas o botones les colocará el título pasado como parámetro.
- **creaPanelConsModBorr**. Desde este método se llama a los métodos para crear el título de las zonas en las que se divide la ventana y las líneas por las que está formado. Al final añade el conjunto de botones ("**Nueva**", "**Buscar**", etc.) situados en el fondo de esta ventana, la dimensiona y finalmente la visualiza.
- **creaTituloIntroducir.** Método que crea el título superior de las zonas en las que se divide la ventana del modo **Introducir**. Crea además el área de texto que ocupa la zona titulada "**Formas derivadas**" y la inicializa para que no muestre al principio ningún texto.
- **creaLineaIntroducir.** Este método, tiene la misma función que el denominado **CreaLineaConsModBorr**, excepto que para la ventana del modo **Introducir** cada línea se compone únicamente de una etiqueta o un botón y un campo de texto.
- **creaPanelIntroducir.** Desde este método se llama a los métodos para crear el título de las zonas en las que se divide la ventana del modo **Introducir** y las líneas por las que está formado. Al final añade el conjunto de botones ("**Nueva**", "**Buscar**", etc.) situados en el fondo de esta ventana, la dimensiona y finalmente la visualiza.

- **creaTituloIntroVbIrr.** Método que crea el título superior de las zonas en las que se divide la ventana del modo **Introducir Verbo Irregular**. Crea además el área de texto que ocupa la zona titulada "**Formas derivadas**" y la inicializa para que no muestre al principio ningún texto.
- **creaLineaIntroVbIrr.** Cada vez que se llama este método se crea una línea formada por una etiqueta y un campo de texto, en la coordenada Y que se indique como parámetro. Cada campo de texto servirá para poder introducir una raíz del verbo irregular que se desee grabar en la base de datos.

  Con este método también se creará igualmente el botón etiquetado como "**Grupo**", así como el campo de texto asociado.
- **creaPanelIntroVbIrr.** Su función es llamar a los dos métodos anteriores para crear las dos zonas en las que se divide la ventana, así como añadir los botones de la parte inferior y visualizar esta ventana.

Métodos públicos:

- **Interface.** Este es el método constructor de la clase Interface que permite instanciarla. Además, llama al método **creaPanelConsModBorr**, provocando que se cree la ventana del modo **Consultar-Modificar-Borrar** y se visualice. La razón por la que hace esto es porque esta clase sólo se instancia una vez, cuando se arranca la aplicación, y cuando se arranca aparece por defecto esta ventana.
- **handleEvent**. Este es el método que permite controlar los eventos originados en la ventana que en cada momento está abierta, realizando las acciones oportunas según de que evento se trate. Así, por ejemplo, cuando se selecciona una opción de la barra de menú, se genera un evento que es capturado por este método, provocando que en ese momento se ejecute la opción elegida, como puede ser salir de la aplicación si se escoge la opción **Salir**.

- **mouseDown**. Este método es llamado por el anterior cuando el evento que se ha generado en la ventana es el de haber pulsado el botón izquierdo del ratón. Este reconoce sobre que elemento de la ventana se ha pulsado el botón y ejecuta la acción correspondiente asociada con el elemento sobre el que se pulsa.

  Por ejemplo, cuando se pulsa el botón izquierdo del ratón sobre un botón que permite elegir una propiedad de una raíz y un lema en la ventana del modo **Consultar-Modificar-Borrar**, la acción que se desencadena es la de mostrar el menú con las distintas posibilidades existentes para esa propiedad.

Clase **Desinencias**:

Métodos privados:

- **creaFlags.**
- **creaTerminaciones.** Este método crea un array que contiene una entrada con todas las terminaciones que son necesarias añadir a una raíz para construir sus formas derivadas, para cada uno de los grupos de raíces de palabras de la lengua española contemplados en esta aplicación.

  Este array tiene las entradas ordenadas por el nombre del grupo, según un orden preestablecido. De esta forma, dado el nombre del grupo, se pueden obtener las terminaciones correspondientes necesarias para construir las formas derivadas de una raíz y un lema dados.

  Dentro de cada grupo de terminaciones, éstas también están ordenadas según un orden preestablecido, de forma que al unirlas con la raíz las formas derivadas aparecerán también en un orden. Por ejemplo, para los verbos en primer lugar está la terminación de infinitivo, luego la del gerundio, etc.

En el caso de los verbos irregulares estas terminaciones llevan además asociada información que permite saber con que raíz irregular se debe unir la correspondiente terminación.

En todos los casos, cuando una terminación, al ser añadida a la raíz implique la acentuación de ésta, esto irá señalado junto con la correspondiente terminación.

Métodos públicos:

- **Desinencias.** Este es el método constructor de la clase. Cuando se llama crea una nueva instancia y además llama a los métodos **creaFlags** y **creaTerminaciones** para crear las estructuras que permiten recuperar los flags y las terminaciones.
- **flags0**
- **flags1**
- **modelo.** Con este método se pueden recuperar las raíces del verbo irregular que sirve como modelo del grupo de verbos irregulares cuyo nombre se le pasa como parámetro.
- **terminaciones.** Con este método se recuperan el conjunto de terminaciones, además de la información adicional que adjuntan, necesarias para poder generar las formas derivadas de una raíz, y que se corresponden con las terminaciones del nombre de grupo que se le pasa como parámetro.

Clase **Acentos**:

Métodos privados

- **creaImperativeGroups**
- **esVerboAgudoOMonosilabo.** Reconoce si dado el nombre de un grupo de raíces, una raíz de un verbo que pertenece a ese grupo es aguda o monosílaba.
- **esVocalDebil.** Reconoce si una vocal es débil.

- **esVocalFuerte.** Reconoce si una vocal es fuerte.
- **esVocal.** Reconoce si un carácter es una vocal.
- **haySoloUnaVocal.** Reconoce si la cadena que se le pasa como parámetro contiene tan solo una vocal o más de una. En el primer caso devolverá la posición de esa vocal dentro de la cadena.
- **encuentraPosicionParaAcento.** Cuando se llama a este método, pasándole como parámetro una raíz de un verbo que debe ser acentuada cuando se le añade una terminación y el grupo al que pertenece, éste devuelve en que posición de la raíz se debe colocar el acento.
- **ponerAcento.** Realmente este método lo único que hace es devolver la raíz de un verbo que debe ser acentuada, con la tilde colocada en el lugar que le corresponde, después de llamar al método anterior para saber donde ponerla. A este método cuando se le llama se le deben de pasar como parámetros la raíz que se desea acentuar y el grupo al que pertenece la raiz.

Métodos públicos

- **escribirRaizAcentuada.** Este método es el que realmente puede ser llamado desde otra clase para colacar la tilde en la raíz que se le pasa como parámetro en el lugar que le corresponde y devolver la nueva raíz acentuada. Para ello simplemente llama al método anterior.

Clase **MenuCaso**:

Métodos privados

Métodos públicos

- **MenuCaso.** Este es el método constructor de la clase, llamado cuando se quiere instanciar ésta. Su función es crear una pequeña ventana que posee una barra con un

menú, cuyo contenido son las distintas opciones que se pueden escoger para la propiedad de **Caso**. El método recibe como parámetro el conjunto de campos de texto de la zona de la ventana desde la que se ha desplegado el menú. De esta forma cuando se seleccione una de las opciones, en su correspondiente campo se visualizará el texto que identifique que se ha seleccionado.

- **handleEvent.** Llamado de forma automática cuando se genera algún evento en la ventana que contiene el menú. Los eventos que aquí se reconocen son los originados cuando se selecciona una opción, en cuyo caso la acción que se desencadena es escribir en el campo de texto asociado con la propiedad **Caso** un texto que identifica lo que se ha seleccionado y finalmente cerrar la ventana del menú.

Una vez que se han explicado los dos métodos que posee la clase **MenuCaso**, no es necesario explicar cada uno de los dos que poseen cada clase que representa los menús que se despliegan para cada una de las propiedades existentes para las raíces y los lemas, y cuyo nombre empieza por la palabra **"Menu"**. La función de éstos es la misma que la explicada para los dos anteriores, excepto que el texto que representa la selección hecha será escrito en cada caso en el correspondiente campo.

Clase **MenuCategoria**:

Métodos privados

Métodos públicos

- **MenuCategoria.** Igual que lo explicado para **MenuCaso.**
- **handleEvent.** Igual que lo explicado para **MenuCaso**.

Clase **MenuGenero**:

Métodos privados

Métodos públicos

- **MenuGenero.** Igual que lo explicado para **MenuCaso**.
- **handleEvent.** Igual que lo explicado para **MenuCaso**.

Clase **MenuGrado**:

Métodos privados

Métodos públicos

- **MenuGrado.** Igual que lo explicado para **MenuCaso**.
- **handleEvent.** Igual que lo explicado para **MenuCaso**.

Clase **MenuGrupo**:

Métodos privados

Métodos públicos

- **MenuGrupo.** Igual que lo explicado para **MenuCaso**.
- **handleEvent.** Igual que lo explicado para **MenuCaso**.

Clase **MenuGrupoIrr**:

Métodos privados

Métodos públicos

- **MenuGrupoIrr.** Igual que lo explicado para **MenuCaso**, sólo que cuando construye este menú las opciones **Numero**, **Genero**, **Verbo Regular** son desabilitadas.
- **handleEvent.** Igual que lo explicado para **MenuCaso**.

Clase **MenuModo**:

Métodos privados

Métodos públicos

- **MenuModo.** Igual que lo explicado para **MenuCaso**.
- **handleEvent.** Igual que lo explicado para **MenuCaso**.

Clase **MenuNumero**:

Métodos privados

Métodos públicos

- **MenuNumero.** Igual que lo explicado para **MenuCaso**.
- **handleEvent.** Igual que lo explicado para **MenuCaso**.

Clase **MenuNumeroPersona**:

Métodos privados

Métodos públicos

- **MenuNumeroPersona.** Igual que lo explicado para **MenuCaso**.
- **handleEvent.** Igual que lo explicado para **MenuCaso**.

Clase **MenuPersona:**

Métodos privados

Métodos públicos

- **MenuPersona.** Igual que lo explicado para **MenuCaso**.
- **handleEvent.** Igual que lo explicado para **MenuCaso**.

Clase **MenuTiempoVerbal**:

Métodos privados

Métodos públicos

- **MenuTiempoVerbal.** Igual que lo explicado para **MenuCaso**.
- **handleEvent.** Igual que lo explicado para **MenuCaso**.

Clase **AboutDialog**:

Métodos privados

Métodos públicos

- **AboutDialog.** Método constructor de la clase, que crea una ventana con información acerca de la aplicación. Se invoca cuando se selecciona la opción **Acerca De** del menú **Opciones** de la ventana que se encuentre abierta.
- **handleEvent.** Cuando se pulsa el botón **Ok** situado en la parte inferior de esta ventana, este método reconoce el evento y como acción asociada cierra la ventana.

Clase **MessageBox**:

Métodos privados

Métodos públicos

- **MessageBox.** Es es constructor de la clase. Cuando se llama crea un nuevo objeto que representa una ventana de diálogo. En ella se muestra el mensaje que se le pasa como parámetro al método en una cadena de caracteres. La ventana posee un botón en su parte inferior, cuya misión es cerrarla cuando el usuario ya ha leído el mensaje y lo pulsa.
- **handleEvent.** Detecta si se ha pulsado el botón **Ok** situado en la parte inferior, en cuyo caso cierra el diálogo.

Clase **MessageOkCancel**:

Métodos privados

Métodos públicos

- **MessageOkCancel.** Es el constructor de esta clase. Cuando se llama crea una ventana de diálogo en la cual se puede mostrar un mensaje pasado como parámetro. Este diálogo posee dos botones en la parte inferior, uno denominado **Ok** y otro **Cancel**, que sirven para aceptar o rechazar el mensaje mostrado. Cuando el mensaje es rechazado simplemente se cierra el diálogo. Si se acepta llamará a otro método especifico de una clase. Tanto el nombre del método como la instancia de la clase lo posee son pasados como parámetros.
- **handleEvent.** Cuando se pulsa uno de los botones que posee la ventana, este método se encarga de ejecutar la acción correspondiente.

Clase **Consultar**:

Métodos privados

- **abrirBaseDatos.** Abre una conexión con la base de datos del entorno GALENA.
- **contadores.** Resetea los contadores que se mantienen de número de tuplas devueltas en una consulta y la posición de la tupla que actualmente se está mostrando en la zona de **Resultados** de la ventana del modo **Consultar-Modificar-Borrar**, dentro del conjunto leído.
- **resetFields.** Pone en blanco el contenido de los campos de texto de la zona de **Resultados**.
- **stringToField.** Dada una cadena de caracteres que se le pasa como parámetro al método, éste la modifica de forma cuando existan varios espacios en blanco juntos los reduce a uno solo, si existen al principio o al final de la cadena los elimina y los caracteres "*****", "**?**","´" los sustituye por "**%**", "**_**","\´" respectivamente. La razón de esto está en la forma en que el gestor de base de datos entiende los caracteres especiales, por eso cuando se lanza una consulta que los contiene se deben sustituir por los correctos para ella.
- **expandirCategoria**
- **expandirGenero**
- **expandirNumero**
- **expandirGrado**
- **expandirPersona**
- **expandirNumPersona**
- **expandirCaso**
- **expandirTiempoVerbal**
- **expandirModo**

Todos los métodos anteriores, cuyo nombre empieza por la palabra **expandir** tienen la siguiente función: las propiedades de cada raíz y de cada lema guardados en la base de datos están representados, para cada posibilidad existente por uno o dos caracteres. Sin embargo cuando se hace una consulta y se quieren mostrar los resultados en los correspondientes campos de texto, las propiedades deben ser cadenas de caracteres que la identifiquen correctamente.

Por ejemplo si el código que representa la propiedad categoría es "**V**" en el campo de texto asociado a ésta se debe escribir "**Verbo(V)**".

Por tanto cada uno de los métodos anteriores sirve para transformar los distintos códigos de una propiedad en cadenas de caracteres que permitan identificarlas fácilmente.

- **showCurrentTupla.** Muestra en la zona de resultados de la ventana **Consultar-Modificar-Borrar** la raíz, el lema y las propiedades de éstos de una tupla de entre las resultantes de una consulta, y cuya posición en el conjunto se le pasa como parámetro.
- **rellenar.** Al método se le pasan como parámetros una cadena de caracteres, una longitud y un carácter. Si la longitud de la cadena es menor que la especificada como parámetro, se rellena con el carácter dado hasta dicha longitud. Se utiliza para que cuando se crean en la base de datos nuevas tuplas, los campos de la raíz y el lema tengan todos la longitud definida para ellos, por eso si no la tienen se rellena hasta alcanzarla.

Métodos públicos

- **nueva.** Prepara las estructuras de datos necesarias para realizar una nueva consulta.
- **lanzarQuery.** En este método primero se prepara la consulta **SQL**, en función de los datos que deseemos consultar. Para ello se toma el contenido de cada uno de los

campos de texto de la zona de consultas de la ventana del modo **Consultar-Modificar-Borrar** y en función de este se va construyendo la consulta. Al final se interroga a la base de datos y ésta devuelve la respuesta. Las tuplas devueltas son guardadas en una lista para ser tratadas posteriormente.

- **seekFirst.** Busca cual es la primera de las respuestas de entre las devueltas en una consulta y muestra los datos en los campos de texto de la zona de resultados.
- **seekPrevious.** Busca en la lista de respuestas devueltas por una consulta, la anterior de la que actualmente se están mostrando sus datos en la zona de resultados.
- **seekNext.** Lo mismo que el método anterior, sólo que en este caso la tupla buscada es la siguiente a la actualmente mostrada.
- **seekLast.** Busca la última de las tuplas resultado de una consulta y muestra sus datos.
- **borrarTodas.** Este método es el encargado de comprobar si entre todas las raíces que se van a borrar alguna de ellas es especial, impidiendo el borrado en este caso. Si ninguna de las raíces es especial, crea una instancia de la clase **MessageOkCancel**, la cual abrirá una ventana de diálogo pidiendo la confirmación del borrado. En caso de que el usuario acepte, el diálogo creado se encargará de llamar al método **siBorrarTodas**, el cual efectuará el proceso de borrado.
- **siBorrarTodas.** Este método es llamado por una instancia de la clase **MessageOkCancel**, cuando se confirma el borrado. Lo primero que hará será cerrar la ventana de diálogo y a continuación ejecutara una petición al gestor de la base de datos para que borre todas las tuplas que aún no hallan sido borradas individualmente y que son el resultado de la última consulta efectuada.
- **borrar**
- **siBorrar**

Los dos métodos anteriores realizan la misma función que los métodos **borrarTodas** y **siBorrarTodas**, con la única diferencia de que en este caso solo se borrará la tupla cuyos datos son actualmente mostrados en la zona de resultados de la ventana del modo **Consultar-Modificar-Borrar**.

- **grabarRaiz.** Este método es el encargado de comprobar si la raíz que se ha modificado es especial, impidiendo que se efectúen los cambios en caso afirmativo. También comprobará si al haber modificado alguno de los campos de texto se ha dejado en blanco la raíz, lema y categoría (o alguno de ellos), en cuyo caso se genera un mensaje de error y se cancela el proceso. Si todo es correcto, crea una instancia de la clase **MessageOkCancel**, la cual abrirá una ventana de diálogo pidiendo la confirmación de la modificación de los datos. En caso de que el usuario acepte, el diálogo creado se encargará de llamar al método **siGrabar**, el cual efectuará el proceso de modificación.
- **siGrabar.** Este método también es llamado por una instancia de la clase **MessageOkCancel** cuando el usuario confirma que quiere modificar los datos actualmente mostrados. En primer lugar, a partir de estos datos, modifica la tupla original en la base de datos. Finalmente también modifica la tupla en la lista tuplas creada cuando se realizó la consulta, con el fin de que si en un futuro se pasa a mostrar los datos de otra de la lista y posteriormente se vuelven a mostrar los de la modificada, los nuevos aparezcan reflejados y no los antiguos.

Clase **NuevaRaiz**:

Métodos privados

- **abrirBaseDatos.** Abre una conexión con la base de datos del entorno GALENA.
- **stringToField.** Es el mismo método que el que existe en la clase consultar.
- **rellenar.** Al método se le pasan como parámetros una cadena de caracteres, una longitud y un carácter. Si la longitud de la cadena es menor que la especificada como parámetro, se rellena con el carácter dado hasta dicha longitud. Se utiliza para que cuando se crean en la base de datos nuevas tuplas, los campos de la raíz y el lema tengan todos la longitud definida para ellos, por eso si no la tienen se rellena hasta alcanzarla.

Métodos públicos

- **conjugarRaiz.** Toma las palabras contenidas en los campos de texto pertenecientes a la raíz y al lema dentro de la ventana del modo **Introducir** y llamando al método **stringToField**, las convierte al formato apropiado[20]. También toma del campo perteneciente a la propiedad de **Grupo** el valor que posee y con todos estos datos halla las formas derivadas correspondientes, mostrándolas finalmente en el área de texto de la zona denominada **Resultados** de esta ventana. Para hallar las formas derivadas, crea una instancia de la clase **Desinencias** y llamando al método **Terminaciones** de la forma explicada en su momento, recupera las terminaciones que debe añadir a la raíz para construirlas.
- **insertarRaiz.** Su función es crear una instancia de la clase **MessageOkCancel**, la cual crea un cuadro de diálogo en el que se le pregunta al usuario si desea insertar la nueva raíz con su correspondiente lema y propiedades. Previamente ya habrá

[20] Sin blancos duplicados, caracteres especiales acordes con los que admite el gestor de base de datos, etc.

comprobado que los campos de texto de la raíz, el lema y la categoría están cubiertos. En caso contrario, generará un mensaje de error y cancelará el proceso.

- **siInsertar.** Es llamado siempre por la instancia de la clase **MessageOkCancel** creada en el método anterior, en caso de que el usuario en el cuadro de diálogo pulse el botón **Ok.** Su función es grabar una nueva tupla en la base de datos a partir de los datos obtenidos de los campos de texto de la zona denominada **Nueva Raiz** de la ventana del modo **Introducir**.

  Dado que en la base de datos las propiedades de la raíz son guardadas con códigos específicos, el método deberá construir éstos a partir de los datos presentes en los correspondientes campos de texto. Además deberá generar una clave para la nueva tupla, cosa que hará encadenando el contenido de varios de los campos.

Clase **NuevaRaizIrregular**:

Métodos privados

- **abrirBaseDatos.** Es el mismo método que el de las clases **Consultar** e **Introducir**.
- **stringToField.** Es el mismo método que el de las clases **Consultar** e **Introducir**.
- **rellenar.** Al método se le pasan como parámetros una cadena de caracteres, una longitud y un carácter. Si la longitud de la cadena es menor que la especificada como parámetro, se rellena con el carácter dado hasta dicha longitud. Se utiliza para que cuando se crean en la base de datos nuevas tuplas, los campos de la raíz y el lema tengan todos la longitud definida para ellos, por eso si no la tienen se rellena hasta alcanzarla.
- **numeroRaiz.** Cuando se trata con verbos irregulares, y se quieren conjugar, las terminaciones que se deben añadir tienen asociadas un identificativo que dice cual de las raíces irregulares del verbo (A,B.C,…) le debe ser antepuesta. Este método, a partir de esta información asociada con la terminación, devuelve el número dentro

de un orden establecido que corresponde a la raíz que le debe ser antepuesta. De esta forma, posteriormente cuando se quiera construir la forma derivada se puede recuperar fácilmente del array que contiene todas las raíces pertenecientes al verbo en cuestión.

Métodos públicos

- **habilitaCampos.** Cuando se selecciona un grupo de verbos irregulares, este tiene definido cuantas raíces irregulares poseen, para que en caso de querer insertar un nuevo verbo que se encuadra dentro se sepa cuantas raíces se deben introducir. El método, conociendo dicha información, habilita para escritura tantos campos de texto en la zona denominada **Nueva Raiz** de la ventana del modo **Introducir Verbo Irregular** como raíces irregulares posean los verbos de este grupo.
- **modeloVerboIrregular.** Cuando se selecciona un grupo de verbos irregulares del menú desplegado al pulsar el botón **Grupo** y a continuación se pulsa el botón **Modelo**, este método se encarga de mostrar en los correspondientes campos de texto de la zona denominada **Nueva Raiz** de la ventana del modo **Introducir Verbo Irregular** las raíces del verbo que sirven como modelo de este grupo. Para ello crea una instancia de la clase **Desinencias**, la cual posee una estructura con todas las raíces de todos los verbos que sirven de modelos para los grupos de verbos irregulares. Llamando al método **modelo** de esta clase, de la forma que se explicó en su momento, obtiene las raíces deseadas.
- **conjugarRaizIrregular.** Realiza la misma función que el método **conjugarRaiz** de la clase **NuevaRaiz**, sólo que en este caso, a la hora de conjugar el verbo debe tener en cuenta cual de las raíces irregulares que posee debe añadir a cada terminación. Para ello llama al método **numeroRaiz**, cuya función se explicó antes, de forma que así obtiene fácilmente la que corresponde en cada caso.

- **insertarRaiz.** Idem que **insertarRaiz** de la clase **Introducir**.
- **siInsertarIrr.** Idem que **insertarRaiz** de la clase **Introducir.**

Estos dos últimos métodos realizan la misma función que sus homónimos en la clase **Nueva Raiz**, teniendo en cuenta que en este caso lo que se introducen en la base de datos son verbos irregulares con todas sus raíces.

Conclusiones

Para terminar con esta memoria vamos a hacer un pequeño resumen de lo que ha sido el desarrollo del proyecto, las ventajas que ha supuesto su realización y las posibilidades futuras de mejora y ampliación que se pueden considerar.

Como ya se ha dicho previamente, se ha desarrollado una interfaz de usuario que permita el mantenimiento y actualización de la base de datos del entorno GALENA y que puede ser explotada remotamente, arrancándose desde una página **Web** en un navegador gráfico.

Además de esta funcionalidad, desde la interfaz se pueden generar las formas derivadas de una raíz y un lema, teniendo en cuenta sus propiedades. La raíz y el lema bien se encuentran introducidas en la base de datos o bien el usuario las introduce en los campos correspondientes, junto con sus propiedades.

El desarrollo de la nueva interfaz a supuesto la obtención de las siguientes ventajas con respecto a la que ya existía para el entorno **Unix.**

- Primero que debido a que puede ser ejecutada a través de **Internet** ahora ya no es necesario establecer una conexión con el servidor en el cual estaba instalada la versión para **Unix** y ejecutarla desde allí.

- La segunda ventaja que ha supuesto es que con esta nueva implementación, las posibles actualizaciones de versiones pueden ser realizadas con total transparencia para el usuario, pues esto sólo supone cambiar la o las correspondientes clases modificadas de forma que cuando un usuario las cargue en su navegador para ejecutar la aplicación ya arrancará con la nueva versión.

- Además al haber utilizado el lenguaje **Java** para la implementación se aprovechan las ventajas que posee el código generado por este compilador, su portabilidad. Así pues,

en caso de que se desee cambiar la localización de las clases compiladas, para instalarlas y que sean accesibles desde un servidor distinto del actual no será necesario recompilar todo el código, sino que simplemente sólo hará falta copiar las clases compiladas al nuevo destino.

Los inconvenientes que se han encontrado a la hora de abordar este trabajo han sido principalmente por causa del lenguaje **Java**, que debido a su relativamente corta vida todavía no es totalmente soportado por los navegadores de **Web**, sobre todo en cuanto a las restricciones de seguridad que estos imponen y que en la mayor parte de ellos no pueden ser relajadas. Una de las restricciones que los navegadores imponen es la de que un **applet** no puede establecer una conexión con otra máquina que no sea aquella desde la cual han sido cargados, impidiendo por ejemplo acceder a una base de datos situada en otro servidor. Esto se podría solucionar permitiendo que un usuario configure su navegador para que permita que los **applets** puedan acceder a una lista de direcciones que este desea y que esperamos que todos los navegadores incluyan pronto[21].

También se han encontrado pequeños contratiempos con las herramientas que posee este lenguaje para la creación de interfaces de usuario, pues para crear ciertos elementos de las ventanas que pose la interfaz desarrollada ha sido necesario utilizar métodos distintos de los que en principio se deseaban. Sin embargo, esto no ha supuesto un grave inconveniente. Además, seguramente que en breve todas los elementos que se deseaban incluir en la interfaz ya están integrados en el lenguaje y son soportados por los navegadores, por lo que en versiones futuras del producto se podrán incluir.

[21] Alguno ya lo hace.

Finalmente, señalar que en un futuro se pretende actualizar la interfaz desarrollada para ir incorporando nuevas funcionalidades y mejorando su aspecto, apoyándose también en que el lenguaje **Java** vaya incorporando nuevos elementos gráficos[22] y que prácticamente todos los navegadores posean una implementación del lenguaje que los soporte, además de que permitan la relajación de las restricciones de seguridad cuando así se desee.

También se espera que de la misma forma que esta interfaz fue reescrita para que pueda ser explotada remotamente, se pueda también realizar el mismo trabajo con los siguientes elementos que también posee el entorno GALENA:

- El supresor de ambigüedades léxicas. [Andrade et al., 1997]
- El analizador léxico. [Graña, Alonso y Valderruten, 1997]
- El analizador sintáctico. [Vilares y Alonso, 1997]

[22] Como por ejemplo menús contenidos en cualquier elemento de la pantalla.

Bibliografía

[Ritchey, 1997] Ritchey T. Programando con Java. Prentice Hall , 1997.

[Andrade et al., 1997] Andrade G. J., Valderruten V. A., Álvarez L., Mª.C. y Sotelo D., S. Un Supresor de Ambigüedades Léxicas mediante Métodos Estadísticos. Revista de la SEPLN, Marzo de 1997.

[Arnold & Gosling, 1997] Arnold K., Gosling J. El lenguaje de programación Java. Addison-Wesley/Domo, 19997

[Gulbransen & Rawlings, 1997] Gulbransen D., Rawlings K. Cree sus applets para web con Java. Prentice-Hall Hispanoamericana, 1996

[Hahn, 1997] Hahn H. Internet.Manual de referencia. Segunda Edición. McGraw-Hill/Interamericana de España S.A., 1997.

[Welch, 1997] Welch B. B. Practical programming in Tcl and Tk. **Falta editorial**

[Mishhofl, 1997] Mishholf C. H. A fondo: Inteligencia Artificial. Ediciones Anaya Multimedia, 1985.

[Graña, Alonso y Valderruten, 1997] Graña G. J., Alonso P. M. A., Valderruten V. A. Análisis Léxico no determinista: Etiquetacion Eficiente del Lenguaje Natural. Technical Report 16, Computer Science Departament, University of Corunna, Spain, 1994.

[Vilares y Alonso, 1997] Vilares F. M., Alonso P. M. A. Exploring Incremental Chart Parsing. Logic Programming Newsletter, vol 8, no. 4, (1995) 9-10.

[Rumbaugh y otros, 1996] Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W. Modelado y diseño arientados a objetos. Metodología OMT. Prentice Hall, 1996

Apendice I

En este apéndice se da una breve introducción a la metodología **OMT**, explicando los elementos gráficos que se han utilizado en esta memoria, para una mejor comprensión del modelo incluido en ella.

OMT es una metodología orientada a objetos, que se extiende desde el análisis hasta la implementación pasando por el diseño. J. Rumbaugh y otros, la han desarrollado para el desarrollo de software en función de la orientación a objetos.

Los objetos del dominio de la aplicación y del dominio de la computadora se pueden modelar, diseñar e implementar utilizando los mismos conceptos y la misma notación orientada a objetos.

La notación gráfica y los conceptos se utilizan en la **OMT**, tanto en el dominio de la aplicación como en el del computador. Utilizando los mismos conceptos y notaciones, consigue que la información aportada en un paso cualquiera del proceso de desarrollo no se pierda ni tenga que ser traducida para su consideración en el siguiente.

A continuación se muestra una pequeña parte de la notación que posee la metodología, y que se ha utilizado en esta memoria.

Los diagramas de objetos proveen una notación gráfica formal para modelizar objetos, clases y relaciones entre unos y otras. Hay dos tipos de diagramas de objetos:

- Los diagramas de clases
- Los diagramas de instancias.

Un diagrama de clases es un esquema que describe muchas posibles instancias de datos.

El símbolo utilizado para referenciar una clase en un diagrama de clase es el siguiente:

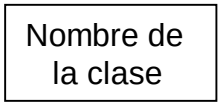


Figura 1. Notación para una clase.

Un link es una conexión física o conceptual entre instancias de objetos.

Una asociación describe un conjunto de links con idéntica estructura y semántica. Las asociaciones son bidireccionales, se leen e interpretan de izquierda a derecha y se representan con una línea que une dos clases. La denominación se nota encima, y puede omitirse si en si misma es explicativa.

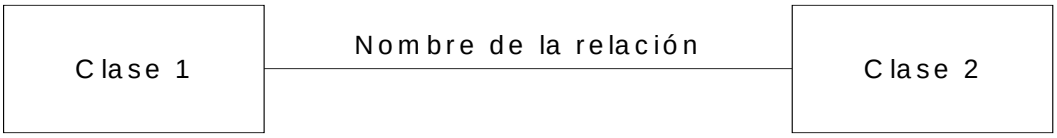


Figura 2. Notación para las asociaciones.

La multiplicidad explícita la relación entre instancias de dos clases. En los diagramas de objeto la multiplicidad se indica utilizando la siguiente notación:

Números en los extremos de la línea que representa la asociación:

- 1 = Exactamente uno.
- 1+ = Uno o mas.
- 3-5 Tres a cinco inclusive.
- Punto vacío = Cero o uno. Opcional.
- Punto lleno de negro = Cero o mas.
- Nada = Uno o mas.

La generalización es la relación entre una clase y una o más versiones refinadas de ellas. La clase que esta siendo refinada se denomina superclase y cada versión refinada es una subclase. La generalización también se denomina relación “es una”, porque cada instancia de una subclase, es también una instancia de la superclase. Cada subclase no solo hereda todas las características de sus antecesoras, sino que también tiene sus operaciones y atributos específicos. La notación de la generalización es un triángulo que conecta una superclase con sus subclases.

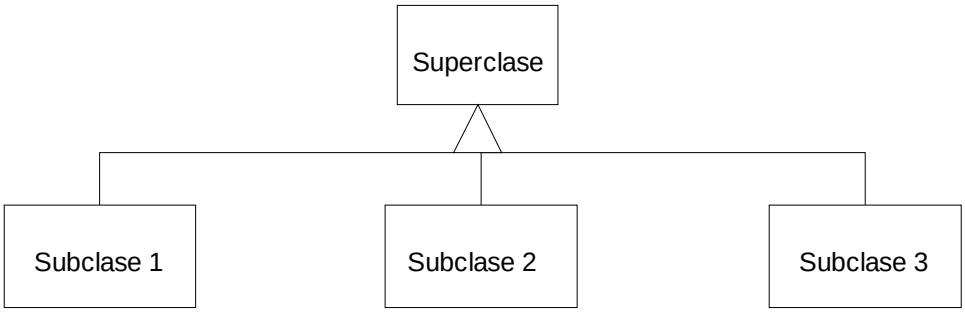


Figura 3. Notación para la generalización.