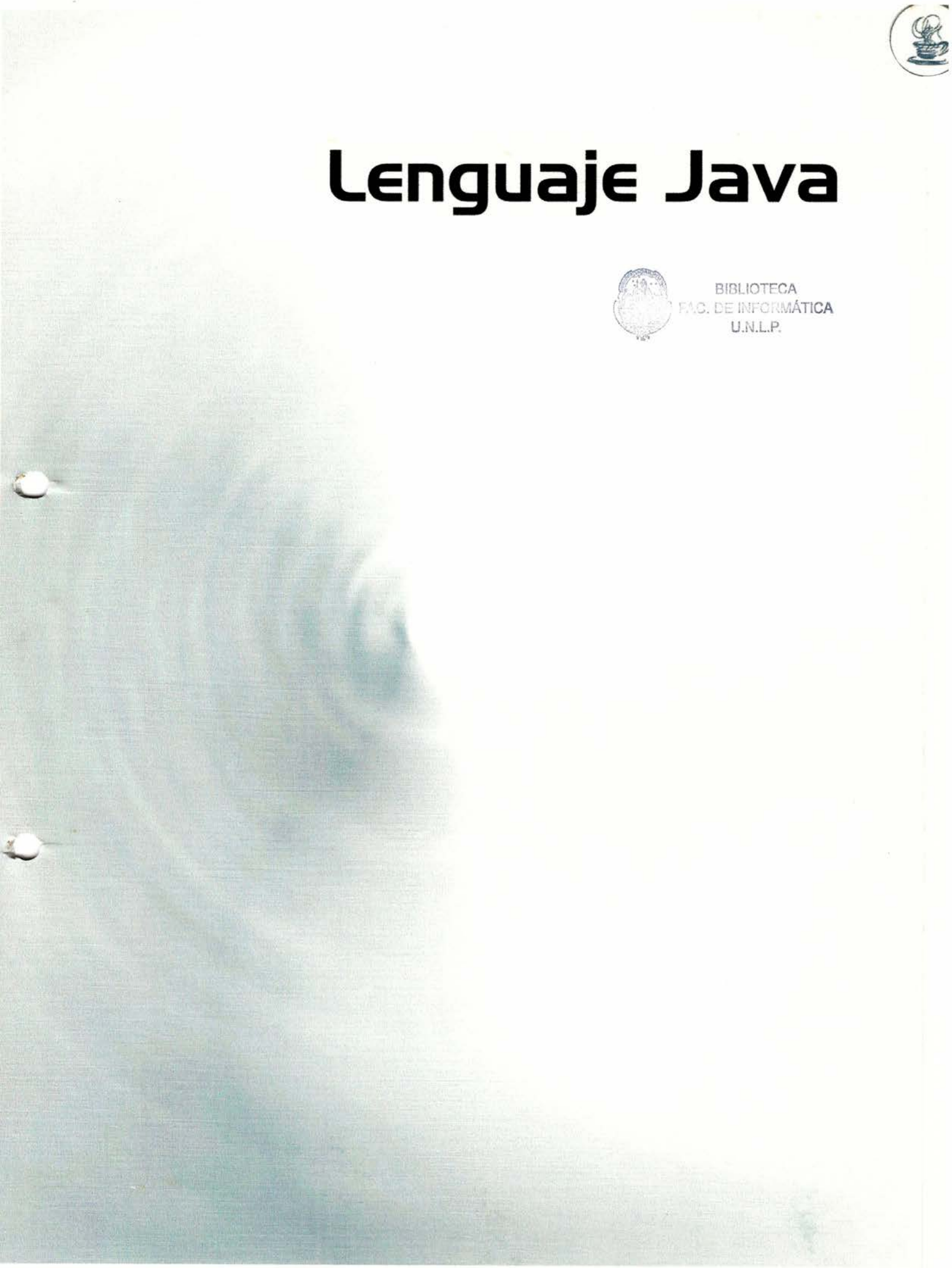
Apliades.

Aplicación a Distancia para Estudiantes

Antonia Maraviglia - Mariángeles Pucci



Lenguaje Java

UN BREVE RESUMEN DE SU CORTA HISTORIA...

Java se remonta a 1991, cuando un grupo de ingenieros de Sun, buscaron un lenguaje que pudiera ser usado para artefactos de consumo inteligentes. Estos artefactos tenían la característica de no poseer mucha memoria, mucha energía, y cada uno de los fabricantes elegían una CPU distinta. Por esto, dicho lenguaje debía ser pequeño, con una generación de código ajustada y que no dependiera de un solo tipo de CPU.

El proyecto se conoció con el nombre de Green. Las características requeridas hicieron resurgir el modelo que un lenguaje llamado UCSD Pascal probó en los tempranos años de las PC's. Este diseñaba un lenguaje portable que generaba código intermedio que se ejecutaba en cualquier máquina que tuviera un interpretador adecuado. El código intermedio generado con este modelo era siempre pequeño, así como también su interpretador; esto resolvía el problema principal.

Sin embargo, la gente de Sun, trabajaba en un ambiente Unix, por lo tanto basaron su lenguaje en C + +, en lugar de Pascal. En particular orientaron su lenguaje a la filosofía de objetos y no a la de procedimientos.

El lenguaje Java fue originalmente bautizado como "Oak" (roble), por James Gosling. Su inspiración para dicho nombre provino de un gran árbol de roble que veía desde la ventana de su oficina en Sun Microsystems. Más tarde, el equipo de desarrollo de Java descubrió que Oak era el nombre de otro lenguaje, motivo por el cual lo rebautizaron como Java (nombre de un tipo de café).

En 1992, distribuyeron el proyecto, pero no encontraron a nadie que lo produjera. Así pasaron todo el año 1993 y parte de 1994, buscando alguien que quisiera vender su tecnología. Mientras tanto la World Wide Web iba creciendo cada vez más, y así los desarrolladores del proyecto se dieron cuenta que podían manifestar toda la potencialidad de su proyecto, construyendo un browser amigable, que fuera independiente de la plataforma, seguro, confiable, características que hasta ese momento no eran importantes en el mundo de las estaciones de trabajo. Aparece entonces el browser denominado HotJava.

Transcurría el año 1995, cuando la construcción del Browser HotJava se llevó a cabo. Para su creación se utilizó el lenguaje Java, aprovechando para demostrar las capacidades del lenguaje. Los creadores además tenían en mente lo que hoy conocemos como *Applets*, por lo tanto, hicieron que el browser fuera capaz de interpretar los *bytecodes* intermedios.

El verdadero auge de Java comenzó hacia fines de 1995, cuando Netscape permitió visualizar las aplicaciones para Internet construidas con Java.

CARACTERÍSTICAS DE JAVA

♣ Java es simple;

Java es similar a C++, pero resulta mucho más simple. Todas aquellas características propias de los lenguajes de alto nivel que no son absolutamente necesarias, han sido dejadas de lado. Por ejemplo, Java no tiene sobrecarga de operadores, archivos de encabezado, preprocesadores, aritmética indicada, uniones, estructuras, arreglos multidimensionales, templates, o conversión de tipo implícita. Además, los programadores ya no deben preocuparse acerca de la administración de la memoria, pues Java incorpora un programa denominado "Garbage Collector", que hace un scanning de la memoria y libera automáticamente cualquier pieza de memoria que ya no está siendo utilizada.

♣ Es Orientado a Objetos;

Java es un lenguaje de programación orientado a objetos. La mayoría de las cosas dentro de Java son objetos, con la excepción de tipos simples parecidos a números y operadores booleanos.

Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo, y tal como sucede en todo lenguaje orientado a objetos, el código de Java está organizado en clases. Cada clase define el comportamiento de un objeto. Una clase puede heredar conductas desde otra clase diferente. En la raíz de la jerarquía de clases, siempre está la clase Objeto.

Soporta una jerarquía de clases con una sola herencia. Esto significa que cada clase solamente puede heredar desde una clase a la vez. Algunos lenguajes también admiten herencia múltiple, pero ésto puede ser confuso y hacer que el lenguaje resulte innecesariamente complicado. Es difícil de imaginar, por ejemplo, qué puede llegar a hacer un objeto que hereda conductas desde dos clases diferentes.

Java también soporta interfaces, las cuales son clases abstractas. Esto permite que los programadores puedan definir métodos para interfaces sin tener que preocuparse inmediatamente en como se implementarán dichos métodos. Una clase puede implementar múltiples interfaces. Esto representa contar con muchas de las ventajas de las verdaderas

herencias múltiples, pero sin muchos de sus problemas. Un objeto puede implementar varias interfaces.

Java incorpora funcionalidades inexistentes en C++, como por ejemplo, la resolución dinámica de métodos. En C++ se suele trabajar con librerías dinámicas (DLL's) que obligan a recompilar la aplicación cuando se retocan las funciones que se encuentran en su interior. Este inconveniente Java lo resuelve mediante una interfaz específica llamada RTTI (RunTime Type Identification), que define la interacción entre objetos excluyendo variables de instancia o implementación de métodos. Las clases en Java tienen una representación en el runtime que permite a los programadores interrogar por el tipo de clase y enlazar dinámicamente la clase con el resultado de la búsqueda.

- Es Distribuido;

Java se ha construido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para acceder e interactuar con protocolos como *http* y *ftp*.

Las aplicaciones Java pueden abrir y acceder a los objetos a través de Internet (vía las URL's), tan fácil como se puede acceder a archivos locales.

Java proporciona las librerías y herramientas para que los programas puedan ser distribuidos.

- Es interpretado y compilado;

Al ejecutarse un programa Java, este es en primer término compilado a código en bytes. Estos *bytecodes* son similares a instrucciones de máquina lo cual hace que los programas Java lleguen a ser muy eficientes. Sin embargo, los *bytecodes* no son específicos en ninguna máquina en particular, por lo cual pueden ser ejecutados en varias y diferentes computadoras sin tener que ser recompilados.

Los programas fuente Java son compilados en archivos de clases, los cuales contienen la representación del programa en *bytecodes*. En un archivo de clases Java, todas las referencias a métodos y variables de instancia, son realizadas por nombre, y son resueltas cuando el código es ejecutado. Esto hace que el código sea más general y menos susceptible a los cambios, y todo sin perder nada de su eficacia.

♣ Es Robusto;

Los programas Java no pueden hacer que una computadora se cuelgue. El sistema Java controla cuidadosamente cada acceso a memoria y se asegura que el mismo sea legal, evitando así que cause algún problema.

Pero hasta los programas Java pueden tener sus bugs. Si sucede algo imprevisto, los programas no se pinchan, pero lanzan un código de excepción. Los programas han de identificar dichas excepciones y manejarlas.

Los programas tradicionales pueden acceder a cualquier parte y a toda la memoria de la computadora. Un programa puede (en forma no intencional), cambiar cualquier valor en la memoria, lo que puede causar problemas. Los programas Java, sólo pueden acceder a aquellas partes de la memoria a las que se les está permitido hacerlo, de modo que un programa Java no puede cambiar un valor, si no está autorizado a hacerlo.

♣ Es Seguro;

Los poderosos mecanismos de seguridad de Java actúan sobre cuatro niveles diferentes de la arquitectura de sistemas:

En primer lugar, tenemos que el lenguaje Java en sí mismo fue diseñado para ser seguro, y es su compilador el que garantiza que el código fuente no pueda violar las normas de seguridad. Con respecto a esto, sabemos que los programas Java no cuentan con punteros, y que, además, es un lenguaje fuertemente tipado, de modo que resulta posible verificar estos programas antes de su ejecución.

En segundo término, tenemos que todos los bytecodes ejecutados por el motor runtime de Java son verificados para resguardar que también ellos obedezcan a dichas reglas. Esta capa protege contra la posibilidad de que un compilador alterado pueda producir código que viole las reglas de seguridad. Un programa Java que ha sido verificado, garantiza que no ha de romper ninguna de las restricciones del lenguaje y, además, que puede ser ejecutado con toda seguridad. La verificación bytecode de Java es utilizada por los Web browsers para asegurarse que las *Applets* no contengan virus.

El tercer tema es el relativo al Alimentador de Clases, el cual garantiza que las clases no violen el espacio de nombres o restricciones de acceso en el momento en que son cargadas dentro del sistema.

Finalmente, la seguridad especifica para APIs evita que las *Applets* realicen acciones destructivas. Esta capa depende de la seguridad e integridad provista por las otras tres primeras capas.

♣ Es Multihilado;

Un programa Java puede tener más de un thread o hilo de ejecución. Un Thread es un flujo de control secuencial dentro de un programa. Por ejemplo, se puede hacer un trabajo de computación extenso sobre un thread, mientras que otros interactúan con el usuario, así los usuarios no tienen que detener su trabajo para esperar que Java complete operaciones prolongadas.

La programación en un entorno multihilado es siempre dificultosa, debido a que son muchas las cosas que pueden ocurrir al mismo tiempo. Java, sin embargo, provee dispositivos destinados a la sincronización que resultan muy sencillos para su uso, lo que hace que la programación sea mucho más amigable. Para dicha sincronización cuenta con un extenso conjunto de primitivas, basadas en el paradigma de monitores.

Los hilos de Java están generalmente mapeados dentro de los hilos del sistema operativo, siempre y cuando el sistema operativo subyacente soporte esta clase de actividad. De esta forma, se puede decir que las aplicaciones escritas con Java son lo que se denomina "MP-Hot", lo cual significa que se benefician si son ejecutadas en una máquina multiprocesamiento.

La programación multihilado permite mejorar la interactividad y la performance del Sistema.

♣ Es Portable;

El lenguaje Java es el mismo en cualquier computadora. Por ejemplo, los tipos simples no varían: un integer es siempre de 32 bits y un longint es siempre de 64 bits.

Sin embargo, esto mismo no resulta cierto en otros lenguajes de programación, tales como C o C++, debido a que estos lenguajes están tan vagamente definidos, cada compilador y entorno de desarrollo es ligeramente diferente, lo que convierte la portabilidad en una ardua tarea. A diferencia de ésto, portar programas Java es muy fácil, ya que no necesitan ser recompilados.

- Es Independiente de la Plataforma;

El compilador genera un formato de archivo neutral a la arquitectura, el código compilado es ejecutable en cualquier procesador dado la presencia del sistema runtime de Java. El compilador Java realiza esta conversión, generando instrucciones de *bytecode*, los cuales no dependen de ninguna arquitectura de computadora en particular. Estos *bytecodes* están diseñados para ser fácilmente interpretados y traducidos a código nativo de máquina.

- Es Extensible;

Es posible hacer una interfaz del programa Java a las librerías de software existentes que estén escritas en otro lenguaje. Debido a que las estructuras de datos de Java son muy parecidas a las estructuras y tipos de C, esto resulta sumamente simple.

Un programa Java puede declarar que determinados métodos le resulten nativos. Estos métodos nativos son luego mapeados en funciones definidas en librerías de software que son dinámicamente encadenadas dentro de una máquina virtual.

LA MÁQUINA VIRTUAL DE JAVA

El elemento más importante de Java es lo que se denomina Virtual Machine. Está modelada en una pequeña y eficiente CPU; lleva código compilado al nivel de byte de Java y lo ejecuta como si fuese lenguaje de máquina. El nivel menor de Virtual Machine convierte a este código de pseudo máquina en llamadas reales del hardware (trabajando con cualquier sistema operativo que resida sobre la máquina en la que está funcionando).

La Virtual Machine es una Unidad Aritmético-Lógica (ALU) con variables locales y globales. Las variables locales son utilizadas para el almacenamiento provisorio y llamadas a subrutinas, mientras que las variables globales se utilizan para saber, por ejemplo, en qué lugar de la memoria se está ejecutando el código de máquina de programa a nivel de byte de Java, cuál es el objeto actual, en qué lugar de la memoria están almacenados los datos más relevantes, y otra clase de información global.

Para implementar correctamente la Máquina Virtual de Java se necesita leer el formato de archivo Java.class y ejecutar las operaciones que se especifican en él. Los detalles de implementación que no forman parte de la especificación de la Máquina Virtual de Java limitan innecesariamente la creatividad de los implementadores. Por ejemplo, el esquema de memoria de las áreas de ejecución de datos, el algoritmo de Garbage Collection usado y cualquier optimización de *bytecodes* (por ejemplo, la traducción en código de máquina) son dejados a criterio del programador.

ALGO MÁS ACERCA DE LA SEGURIDAD DE JAVA...

Como ya mencionamos anteriormente los mecanismos de seguridad de Java actúan a cuatro niveles diferentes. A continuación veremos cómo funciona cada una de las capas.

1. El lenguaje y el compilador

El lenguaje Java y su compilador forman, en cuanto a seguridad, la primera línea de defensa. Java fue diseñado para ser un lenguaje seguro.

La mayor parte de los lenguajes tipo C cuentan con facilidades para controlar el acceso a objetos, pero también tienen forma de generar acceso a los mismos. Esto introduce dos problemas que pueden resultar fatales para cualquier sistema realizado con estos lenguajes. Uno de ellos es que ningún objeto puede autoprotegerse de modificaciones realizadas desde su exterior, duplicación o falsificación. El otro radica en que un lenguaje con punteros muy poderosos tiene más posibilidades de tener serios bugs que comprometan la seguridad. Estos bugs en los punteros, en los que un pointer "prófugo" se ocupa de modificar la memoria de los objetos de alguien, fueron los responsables de muchos inconvenientes en Internet.

Java aventa todas estas amenazas al eliminar por completo de su lenguaje los punteros. De todos modos quedan en él algunas referencias a objetos, pero que son controladas para que resulten seguras.

Por lo tanto la definición del lenguaje y el compilador, crean una poderosa barrera para la seguridad en Java.

2. Verificación de los Bytecodes

¿Qué sucede si el compilador de Java se reescribe para malos propósitos? Para evitar que los bytecodes violen alguno de los requerimientos de seguridad, antes de correrlos, el runtime los somete a una serie de rigurosos tests, que permiten verificar que los bytecodes no alteren punteros, violen restricciones de acceso, accedan a objetos diferentes a ellos mismos, hagan llamadas a métodos con valores o tipos inapropiados, o desborden las pilas. Esto lo realiza utilizando información extra sobre tipos en un archivo

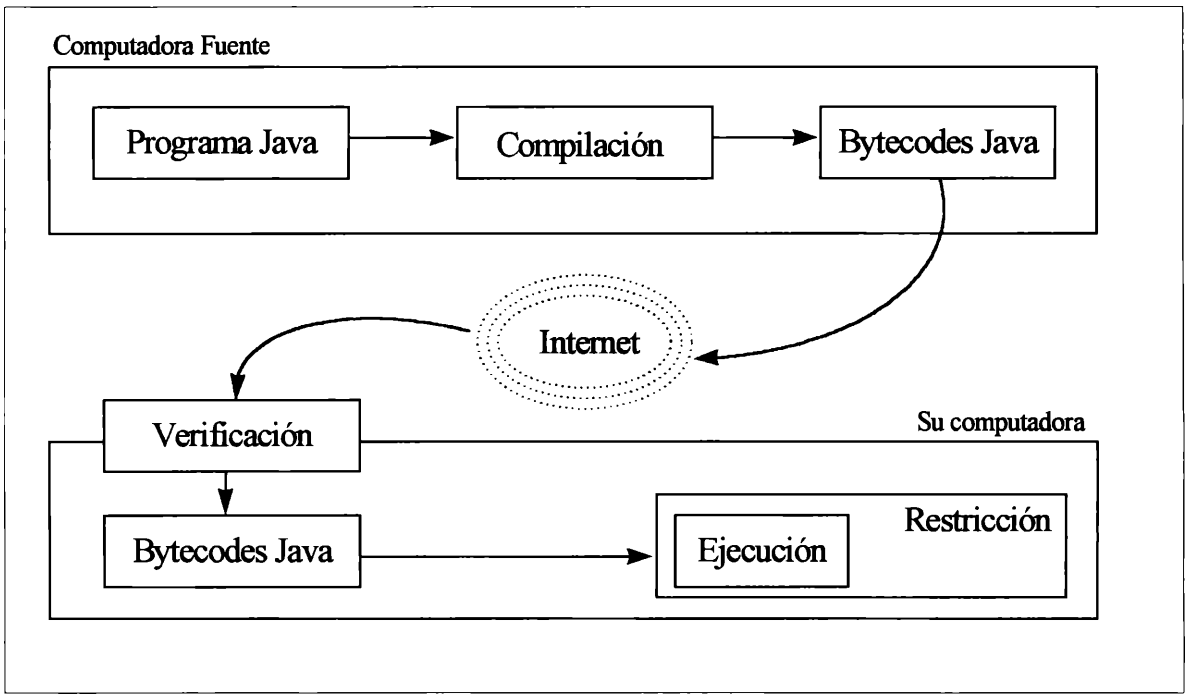
.class, y la tarea la lleva a cabo una parte del runtime, llamada *Verifier*. De esta forma actúa como si fuera una especie de portero del entorno de runtime, dejando pasar solamente a los bytecodes que cumplen los requisitos. Una vez que los bytecodes han pasado por el *Verifier*, se puede garantizar que no se violará ninguno de los requerimientos antes mencionados.

3. Cargador de Clases

El *Verifier* es lo que puede llamarse el último recurso, mientras que el *Class Loader* es un recurso de primera instancia.

Cuando se produce la carga de una nueva clase en el sistema, esta debe proceder de uno entre varios reinos. Actualmente existen tres posibles reinados: la computadora local, la red local y la red global (Internet). Cada uno de estos tres reinos es tratado en forma diferente por el cargador de clases. El cargador de clases nunca admite que una clase que provenga de un reino menos protegido reemplace a una clase que pertenece a uno más protegido. Además, las clases de un reinado no pueden invocar a los métodos de clases que están en otros ámbitos, a menos que dichas clases, hayan declarado explícitamente como públicos a dichos métodos.

En resumen, si se considera el ciclo de vida completo del método Java, éste comienza en la forma de código fuente en una determinada computadora, y luego puede viajar en la forma de un archivo de clases hasta cualquier sistema de archivos o red en cualquier parte del mundo. Cuando se ejecuta un *Applet* en un browser sensible a Java los bytecodes de los métodos son extraídos desde su archivo de clases y luego cuidadosamente escrutados por el *Verifier*. Una vez que son declarados seguros, el intérprete ya puede ejecutarlos.



Ciclo del método Java

MALENTENDIDOS ACERCA DEL LENGUAJE JAVA

1. Java es una extensión de HTML.

   Java es un lenguaje de programación, mientras que HTML es un lenguaje de descripción de páginas. No tienen nada en común, excepto que se necesitan extensiones HTML donde colocar los Applets Java, dentro de una página Web.

2. Tiene un ambiente de programación amigable.

   Java no posee un ambiente para programar. Para la construcción de programas en lenguaje Java pueden utilizarse productos como Visual Age para Java, JBuilder, entre otros.

3. Todos los programas Java corren dentro de una página Web.

   Todos los programas Java corren dentro de un browser, pero es posible y útil escribir programas stand alone, que corran independientemente de un browser. Estos programas, generalmente llamados aplicaciones, son completamente portables. Sólo toman el código y lo corren en otra máquina.

4. Java elimina la necesidad del scripting CGI

   No se elimina el uso de CGI. Con la tecnología de hoy en día CGI continua existiendo, pero no es el único camino de comunicación entre un Applet y el server,

5. Java se convertirá en un lenguaje de programación universal para todas las plataformas.

   Aún, las aplicaciones Java no se ven tan bien como las aplicaciones Windows desarrolladas por VB o MFC. En cualquier caso, las herramientas gráficas soportadas por Java son muy primitivas para hacer que el trabajo de diseño sea placentero.

6. Java es Interpretado, entonces es demasiado lento en una plataforma específica para aplicaciones serias

Varios programas gastan mucho de su tiempo en interacciones de interfaz de usuario. Es cierto que con la versión actual de Java, no haremos trabajos que requieran intensivamente la CPU a menos que tengamos que hacerlo. Sin embargo, es muy sencillo hacer un compilador que convierta los bytecodes de Java en código nativo.

7. Java revolucionará la computación Cliente-Servidor

   Esto es posible, pero a pesar de que en un entorno de desarrollo la comunicación funciona correctamente, se necesita utilizar un plug_in para el browser, pues aún éstos no soportan la comunicación vía RMI o Socket para aplicaciones Cliente-Servidor.

8. Con Java reemplazaré mi computadora por una Ampliación de Internet de $500

   Mucha gente está apostando a que ésto sucederá. Creemos que es absurdo pensar que la gente dejará de lado una computadora poderosa por una máquina limitada sin almacenamiento local. Podemos aceptar dicha Ampliación como una adición a la computadora y potenciar así a las aplicaciones Java.

LAS GARANTÍAS DE JAVA

Java es un lenguaje con un rico conjunto de garantías que, cuando se consideran como un todo, proveen un verdadero y avanzado sistema operativo para los programas. La mayoría de estas garantías están también disponibles en otros lenguajes, pero, hasta el momento ningún otro lenguaje, ni sistema operativo las ofrece a todas juntas.

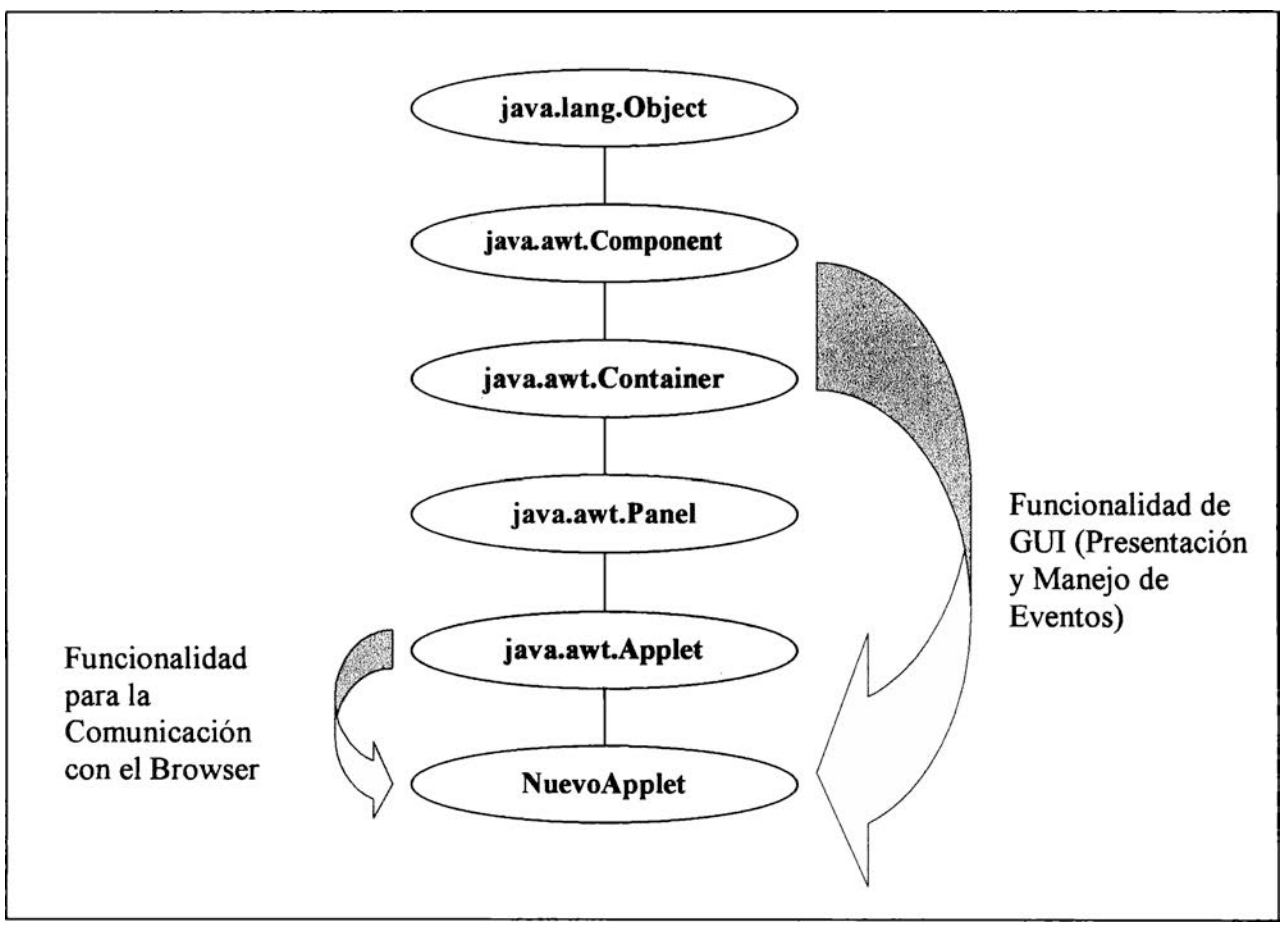
1. El código de Java es portable y correrá sin ninguna alteración en la mayoría de los sistemas operativos.
2. Java provee verdadero multihilado para los programadores, conjuntamente con métodos de sincronización.
3. Java ofrece clases de interfaces de usuarios independientes de la plataforma.
4. Java viene con capacidad de trabajo en red incorporada.
5. Todos los objetos Java saben como imprimirse.
6. El compilador Java impone el manejo de todos los errores y excepciones.
7. La memoria se limpia automáticamente.
8. Los programas escritos en forma de Applets pueden ser distribuidos a través de la red automáticamente.
9. Las Applets de Java funcionarán sin problemas en un Web Browser cliente sencillo sobre cualquier plataforma.

APPLETS JAVA

Un Applet es una pequeña aplicación accesible en un servidor Internet, que se transporta por la red, se instala automáticamente y se ejecuta en el lugar como parte de una página Web.

JERARQUÍA DE HERENCIA

Las applets se crean como una subclase de la Clase Applet.



COMPONENTES DE UN APPLET

El lenguaje Java implementa un modelo de programación orientado a objetos. Los objetos sirven de bloques centrales de construcción de los programas Java. También tiene, al igual que otros lenguajes de programación orientados a objetos, variables de instancia y métodos.

La estructura de un Applet es la siguiente:

```
/ * Sección de importaciones */
Public class NombreDelNuevoApplet extends Applet {
/* Declaración de las variables de estado y métodos */
/* Declaración de métodos para la interacción con los objetos */
Public void MétodoUno (parámetros) {
/* Código Java que desempeña la tarea */
                                                    }
                                                }
```

MÉTODOS DE LA CLASE APPLET

Para escribir Applets Java hay que utilizar una serie de métodos. Son los que se utilizan para iniciar y detener la ejecución del applet, para pintar y actualizar la pantalla y para capturar la información que se pasa al applet desde el archivo html a través del tag APPLET.

Invocados automáticamente por el Browser

- **init()**

Esta función se invoca al crearse el applet. Se invoca sólo una vez, cuando el applet es cargado en el sistema. Las clases derivadas deben definir este método para cambiar el tamaño durante su inicialización, y cualquier otra inicialización de los datos que solamente deba realizarse una vez.

Debería realizarse, por ejemplo, la carga de imágenes y sonido, el resize del applet para que tenga su tamaño correcto y la asignación de valores a las variables globales.

Ejemplo:

```
public void init() {
  if( width < 200  ||  height < 200 )
    resize( 200,200 );
  valor_global1 = 0;
  valor_global2 = 100;
  // carga de imágenes en memoria sin mostrarlas
  // carga de música de fondo en memoria sin reproducirla    }
```

- **start()**

Llamada para activar el applet. Esta función es invocada luego del init y cada vez que la página que contiene el applet es visitada. La Clase Applet no hace nada en este método, pero las clases derivadas deberán sobrecargarlo para comenzar una animación, sonido, etc.

```
public void start() {
  estaDetenido = false;
  // comenzar la reproducción de la música
  musicClip.play();
  }
```

- **stop()**

Llamada para detener el applet. Se invoca cuando el applet desaparece de la pantalla, es decir cuando se abandona la página html que lo contiene. La Clase Applet no hace nada en este método. Las clases derivadas deberán sobrecargarlo para detener la animación, el sonido, etc.

```
public void stop() {
  estaDetenido = true;
  if( /* ¿se está reproduciendo música? */ )
    musicClip.stop();
  }
```

- **destroy()**

Este método es invocado cuando el applet no se va a usar. Se liberan los recursos del sistema que tiene alocados. La Clase Applet no hace nada en este método. Las clases derivadas deberían cargarlo para hacer una limpieza final. Por ejemplo, los Applet

Multithread deberán usar destroy() para "matar" cualquier thread del applet que quedase activo.

Otros métodos

- **resize (int width, int height)**

El método init() debería llamar a esta función para establecer el tamaño del applet. Cambiar el tamaño en otro sitio que no sea init() produce un reformateo de todo el documento y no se recomienda.

En el browser Netscape, el tamaño del applet es el que se indica con el tag APPLET del HTML, prevaleciendo éste al que se indique desde el código Java del applet.

- **width**

Variable entera, su valor es el ancho definido en el parámetro WIDTH del tag APPLET del HTML.

- **height**

Variable entera, su valor es la altura definida en el parámetro HEIGHT del tag APPLET en el HTML. Por defecto, al igual que el ancho, es la altura del ícono. Tanto width como height están siempre disponibles para que se pueda chequear el tamaño del applet.

- **paint(Graphics g)**

Se llama cada vez que se necesita refrescar el área de dibujo del applet. La Clase Applet simplemente dibuja una caja con sombreado de tres dimensiones en el área.

Para repintar toda la pantalla cuando llega un evento paint, se pide el rectángulo sobre el que se va a aplicar paint() y si es más pequeño que el tamaño real del applet se invoca a repaint(), que como va a hacer un update(), se actualizará toda la pantalla.

- **update(Graphics g)**

Esta es la función que se llama cuando se necesita actualizar la pantalla. La clase Applet simplemente limpia el área y llama al método paint().

Podemos, por ejemplo, utilizar update() para modificar selectivamente partes del área gráfica sin tener que pintar el área completa:

```
public void update( Graphics g ) {
  if( estaActualizado )
    {
    g.clear(); // garantiza la pantalla limpia
    repaint(); // podemos usar el método padre: super.update()
    }
  else
    // Información adicional
    g.drawString( "Otra información",25,50 );
    }
```

- **repaint()**

A esta función se la debería llamar cuando el applet necesite ser repintado.

Al llamar a repaint(), sin parámetros, internamente se llama a update() que borrará el rectángulo sobre el que se redibujará y luego se llama a paint(). Como a repaint() se le pueden pasar parámetros, se puede modificar el rectángulo a repintar.

- **getParameter(String attr)**

Este método carga los valores pasados al applet vía el tag APPLET del HTML. El argumento String es el nombre del parámetro que se quiere obtener. Devuelve el valor que se le haya asignado al parámetro; en caso de que no se le haya asignado ninguno, devolverá null.

Para usar getParameter() se define una cadena genérica. Una vez que se ha capturado el parámetro, se utilizan métodos de cadena o de números para convertir el valor obtenido al tipo adecuado.

- **getDocumentBase()**

Indica la ruta http, o el directorio del disco, de donde se ha recogido la página HTML que contiene el applet.

- **getCodeBase()**

  Indica la ruta http, o el directorio del disco, de donde se ha cargado el código bytecode que forma el applet.

- **print(Graphics g)**

  Imprime el mapa de bits del dibujo.

ATRIBUTOS DE LOS APPLETS

Dado que los applets están destinados a ejecutarse en navegadores Web, deben estar contenidos en páginas HTML. El esquema de tags de HTML hace fácil la adición de una marca que permite la ejecución de programas Java en ellos.

La sintaxis de las etiquetas <APPLET> y <PARAM> es la siguiente:

```
<APPLET CODE= WIDTH= HEIGTH= [CODEBASE=] [ALT=]
[NAME=] [ALIGN=] [VSPACE=] [HSPACE=]>
<PARAM NAME= VALUE= >
</APPLET>
```

En cuanto a los atributos que acompañan a la etiqueta <APPLET>, algunos son obligatorios y otros son opcionales. Todos los atributos, siguiendo la sintaxis de html, se especifican de la forma: atributo=valor.

Atributos Obligatorios

- **CODE**

  Es el nombre de la clase principal, la clase ejecutable, que tiene la extensión .class. No se permite un URL absoluto, aunque sí puede ser relativo al atributo opcional CODEBASE.

- **WIDTH**

  Indica el ancho inicial en pixels que el navegador debe reservar para el applet.

- **HEIGHT**

  Indica la altura inicial en pixels. Un applet que disponga de una geometría fija no se verá redimensionado por estos atributos. Por este motivo, si los atributos

definen una zona menor que la que el applet utiliza, únicamente se verá parte del mismo, como si se visualizará a través de una ventana, eso sí, sin ningún tipo de desplazamiento.

Atributos Opcionales

- **CODEBASE**

  Se emplea para utilizar el URL base del applet. En caso de no especificarse, se utilizará el mismo que tiene el documento.

- **ALT**

  Muestra un texto alternativo en caso de que el navegador no implemente una Máquina Virtual Java.

- **NAME**

  Otorga un nombre a esta instancia del applet en la página. Un applet puede ser cargado varias veces en la misma página tomando un alias distinto en cada momento.

- **ALIGN**

  Se emplea para alinear el applet. Puede tomar los siguientes valores: LEFT, RIGHT, TOP, TEXTTOP, MIDDLE, ABSMIDDLE, BASELINE, BOTTOM y ABSBOTTOM.

- **VSPACE**

  Indica el espacio vertical, en pixels, entre el applet y el texto. Sólo funciona cuando se ha indicado ALIGN = LEFT o RIGHT.

- **HSPACE**

  Funciona igual que el anterior pero indicando espaciamiento horizontal. Sólo funciona cuando se ha indicado ALIGN = LEFT o RIGHT.

PASAJE DE PARÁMETROS A APPLETS

El espacio que queda entre las marcas de apertura y cierre de la definición de un applet, se utiliza para definir los parámetros que recibe el mismo. Para indicarlos se utiliza la marca PARAM en la página HTML y el método getParameter() de la clase java.applet.Applet para leerlos en el código interno del applet.

Los atributos que acompañan a la marca PARAM son los siguientes:

- **NAME:** Nombre del parámetro que se desea pasar al applet.
- **VALUE:** Valor que se desea transmitir en el parámetro que se ha indicado antes.
- **Texto HTML:** Es el texto HTML que será interpretado por los navegadores que no entienden la marca APPLET en sustitución del applet mismo.

Los parámetros no se limitan a uno solo. Se puede pasar al applet cualquier número de parámetros y siempre hay que indicar un nombre y un valor para cada uno de ellos.

El método getParameter() es fácil de entender. El único argumento que necesita es el nombre del parámetro cuyo valor queremos recuperar. Todos los parámetros se pasan como Strings, en caso de necesitar pasarle al applet un valor entero, se ha de pasar como String, recuperarlo como tal y luego convertirlo al tipo deseado. Tanto el argumento de NAME como el de VALUE deben ir colocados entre comillas dobles.

VISUALIZACIÓN DE APPLETS

Para ver un applet en funcionamiento debemos utilizar un navegador World Wide Web como HotJava, Microsoft Explorer o Netscape.

Sun nos ofrece un visualizador de applets denominado Appletviewer. Este representa el mínimo espacio de navegación, incluyendo un área gráfica, donde se ejecutará el applet.

La llamada a appletviewer es de la forma: appletviewer [-debug] urls...

Tanto el navegador World Wide Web, como el appletviewer toman como parámetro de ejecución, o bien el nombre de un archivo html conteniendo el tag <APPLET>, o bien un URL hacia un archivo HTML que contenga esa marca.

La única opción válida que admite la llamada a appletviewer es -debug, que arranca el applet en el depurador de Java. Para poder ver el código fuente en el depurador, se tiene que compilar el archivo .java con la opción -g.

CICLO DE VIDA DE UN APPLET

Cuando se carga un applet en un browser, comienza su ciclo de vida, que pasaría por las siguientes fases:

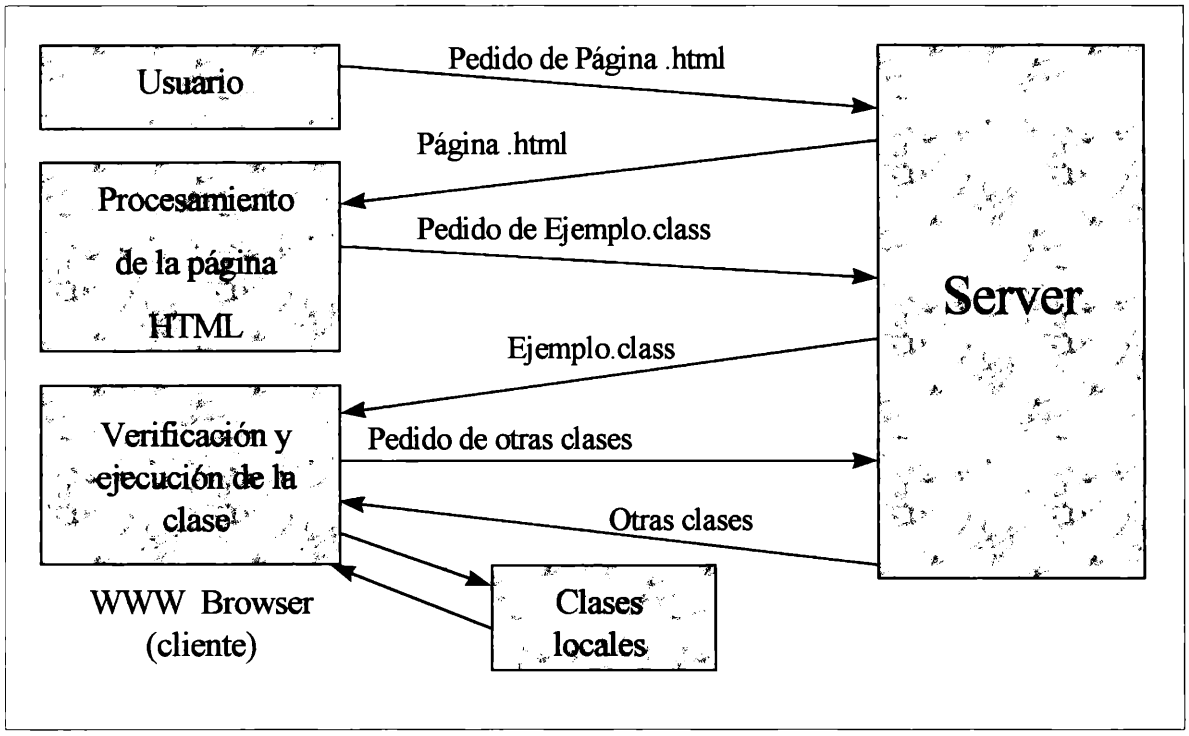
1. Se crea una instancia de la clase que controla el applet.
2. El applet se inicializa a si mismo.
3. Comienza la ejecutación del applet.
4. El applet empieza a recibir llamadas. Primero recibe una llamada init, seguida de un mensaje de start y paint. Estas llamadas pueden ser recibidas asincrónicamente.

Cuando se abandona la página, el applet detiene la ejecución. Cuando se regresa a la página que contiene el applet, se reanuda la ejecución del mismo.

Si se utiliza la opción del navegador de Reload, el applet es descargado y vuelto a cargar. El applet libera todos los recursos capturados, detiene su ejecución y ejecuta su finalizador para realizar un proceso de limpieza final de sus trazas. Después de esto, el applet se descarga de la memoria y vuelve a cargarse volviendo a comenzar su inicialización.

Finalmente, cuando se concluye la ejecución del navegador o de la aplicación que está mostrando el applet, se detiene la ejecución del mismo y se libera toda la memoria y los recursos ocupados por el applet.

UN APPLET EN FUNCIONAMIENTO



EXCEPCIONES

Las excepciones en Java están destinadas, al igual que en el resto de los lenguajes que las soportan, a la detección y corrección de errores. Si hay un error, la aplicación no debería morirse sino que se debería lanzar una excepción que debe ser capturada para resolver la situación de error. Utilizadas en forma adecuada, las excepciones aumentan la robustez de las aplicaciones, y por lo tanto la robustez del lenguaje.

GENERAR EXCEPCIONES EN JAVA

Cuando se produce un error se debería generar, o lanzar, una excepción. Para que un método en Java pueda lanzar excepciones, hay que indicarlo expresamente.

```
void MetodoAsesino() throws NullPointerException,CaidaException
```

Se pueden definir excepciones extendiendo la clase Exception.

También pueden producirse excepciones en forma implícita cuando se realiza alguna acción ilegal o no válida.

Por lo tanto, las excepciones, pueden originarse de dos modos: el programa hace algo ilegal, o el programa explícitamente genera una excepción ejecutando la sentencia throw.

La sentencia throw tiene la siguiente forma:

```
throw ObjetoException;
```

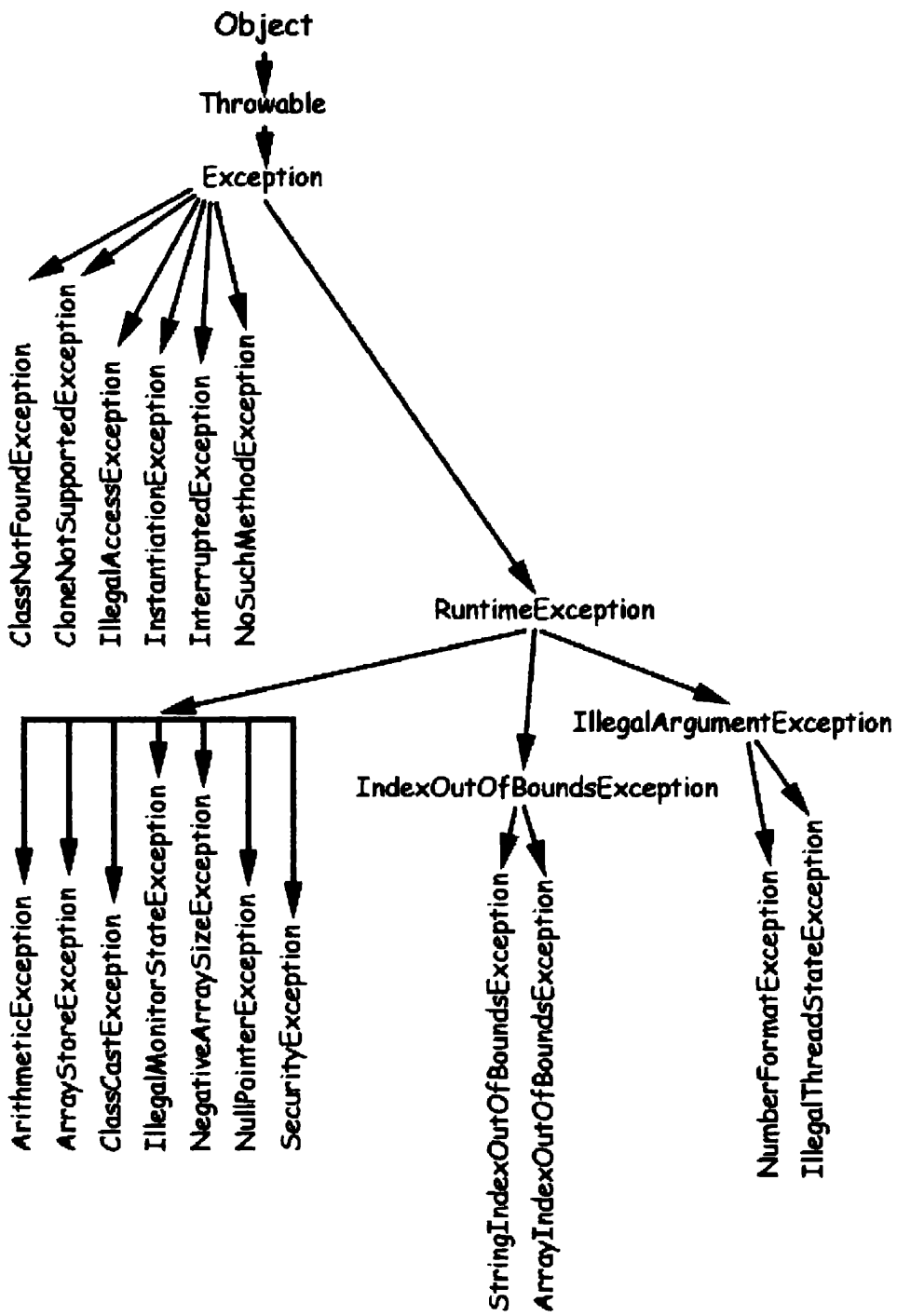
El objeto ObjetoException es un objeto de una clase que extiende la clase Exception.

El siguiente código de ejemplo origina una excepción de división por cero:

```
class melon {
  public static void main( String[] a ) {
    int i=0, j=0, k;
    k = i/j;   // Origina un error de division-por-zero
    }
  }
```

EXCEPCIONES PREDEFINIDAS

Las excepciones predefinidas se conocen como Excepciones Runtime. Su jerarquía de clases es la que se muestra el siguiente gráfico:



Las siguientes son las excepciones predefinidas más frecuentes que se pueden encontrar:

- **ArithmeticException**

  Una excepción aritmética típica es el resultado de una división por cero:

  ```
  int i = 12/ 0;
  ```

- **NullPointerException**

  Se produce cuando se intenta acceder a una variable o a un método antes de haberlo definido.

- **IncompatibleClassChangeException**

  El intento de cambiar una clase afectada por referencias en otros objetos, cuando esos objetos todavía no han sido recompilados.

- **ClassCastException**

  Se produce cuando se intenta convertir un objeto a otra clase que no es válida.

  ```
  y = (Prueba)x;     // donde x no es de tipo Prueba
  ```

- **NegativeArraySizeException**

  Puede ocurrir si hay un error aritmético al intentar cambiar el tamaño de un array.

- **OutOfMemoryException**

  Se produce si se intenta crear un objeto con el operador new y no hay memoria suficiente. No debería producirse, pues el Garbage Collector se encarga de liberarla.

- **NoClassDefFoundException**

  Se produce si se realiza una referencia a una clase que el sistema no encuentra.

- **ArrayIndexOutOfBoundsException**

  Se genera al intentar acceder a un elemento de un array que excede los límites definidos inicialmente para el mismo.

- **UnsatisfiedLinkException**

  Si se hace el intento de acceder a un método nativo que no existe. Aquí no existe un método a.kk y se llama a a.kk(), cuando debería llamar a A.kk().

```
class A {
  native void kk();
  }
```

- **InternalException**

  Este error se reserva para eventos que no deberían ocurrir. Por definición, el usuario nunca debería ver este error y esta excepción no debería lanzarse.

CREAR EXCEPCIONES PROPIAS

Se pueden lanzar excepciones propias, extendiendo la clase System.exception. Por ejemplo, si considerando un programa cliente/servidor, el código cliente se intenta conectar al servidor, y durante 5 segundos espera una respuesta. Si el servidor no responde, el servidor lanzaría la excepción de time-out:

```
class ServerTimeOutException extends Exception {}
public void conectame( String nombreServidor ) throws Exception {
   int exito;
   int puerto = 80;
   exito = open( nombreServidor,puerto );
   if( exito == -1 )
      throw ServerTimeOutException;
   }
```

Si se quieren capturar las excepciones propias, se deberá utilizar la sentencia try:

```
public void encuentraServidor() {

  try {
     conectame( servidorDefecto );
     catch( ServerTimeOutException e ) {
        g.drawString(
           "Time-out del Servidor, intentando alternativa",
           5,5 );
        conectame( servidorAlterno );
        }

  }
```

Cualquier método que lance una excepción también debe capturarla, o declararla como parte de la interfaz del método.

CAPTURAR EXCEPCIONES

Las excepciones lanzadas por un método que pueda hacerlo deben ser manejadas en un bloque try/catch o try/finally.

```
int valor;
try {
   for( x=0,valor = 100; x < 100; x ++ )
      valor /= x;
   }
catch( ArithmeticException e ) {
   System.out.println( "Matemáticas locas!" );
   }
catch( Exception e ) {
   System.out.println( "Se ha producido un error" );
   }
```

- **try**

Es el bloque de código donde se prevee que se genere una excepción. Este bloque try tiene que ir seguido por una cláusula catch o una cláusula finally

- **catch**

Es el código que se ejecuta cuando se produce la excepción. En este bloque no se debe colocar código que genere excepciones. Se pueden colocar sentencias catch sucesivas, cada una controlando una excepción diferente. No debería intentarse capturar todas las excepciones con una sola cláusula:

```
catch( Excepción e ) { ...
```

Esto representaría un uso demasiado general, podrían llegar muchas más excepciones de las esperadas. En este caso es mejor dejar que la excepción se propague hacia arriba y dar un mensaje de error al usuario.

Se pueden controlar grupos de excepciones, es decir, que se pueden controlar, a través del argumento, excepciones semejantes.

Por ejemplo:

```
class Limites extends Exception {}
class demasiadoCalor extends Limites {}
class demasiadoFrio extends Limites {}
```

```
class demasiadoRapido extends Limites {}
class demasiadoCansado extends Limites {}

  try {
    if( temp > 40 )
      throw( new demasiadoCalor() );
    if( dormir < 8 )
      throw( new demasiado Cansado() );
  } catch( Limites lim ) {
    if( lim instanceof demasiadoCalor )
      {
      System.out.println( "Capturada excesivo calor!" );
      return;
      }
    if( lim instanceof demasiadoCansado )
      {
      System.out.println( "Capturada excesivo cansancio!" );
      return;
      }
  } finally
    System.out.println( "En la clausula finally" );
```

La cláusula catch comprueba los argumentos en el mismo orden en que aparecen en el programa. Si hay alguno que coincide, se ejecuta el bloque. El operador instanceof se utiliza para identificar exactamente cual ha sido la identidad de la excepción.

- **finally**

  Es el bloque de código que se ejecuta siempre, haya o no excepción.

Este bloque finally puede ser útil cuando no hay ninguna excepción. Es un trozo de código que se ejecuta independientemente de lo que se haga en el bloque try.

Cuando se trata una excepción, se plantea el problema de qué acciones seguir. En la mayoría de los casos, bastará con presentar un mensaje de error al usuario y que él decida si quiere o no continuar con la ejecución del programa.

PROPAGACIÓN DE EXCEPCIONES

La cláusula catch comprueba los argumentos en el mismo orden en que aparecen en el programa. Si hay alguno que coincide, se ejecuta el bloque y sigue el flujo de control por el bloque finally (si lo hay) y concluye el control de la excepción.

Si ninguna de las cláusulas catch coincide con la excepción que se ha producido, entonces se ejecuta el código de la cláusula finally (en caso de que la haya).

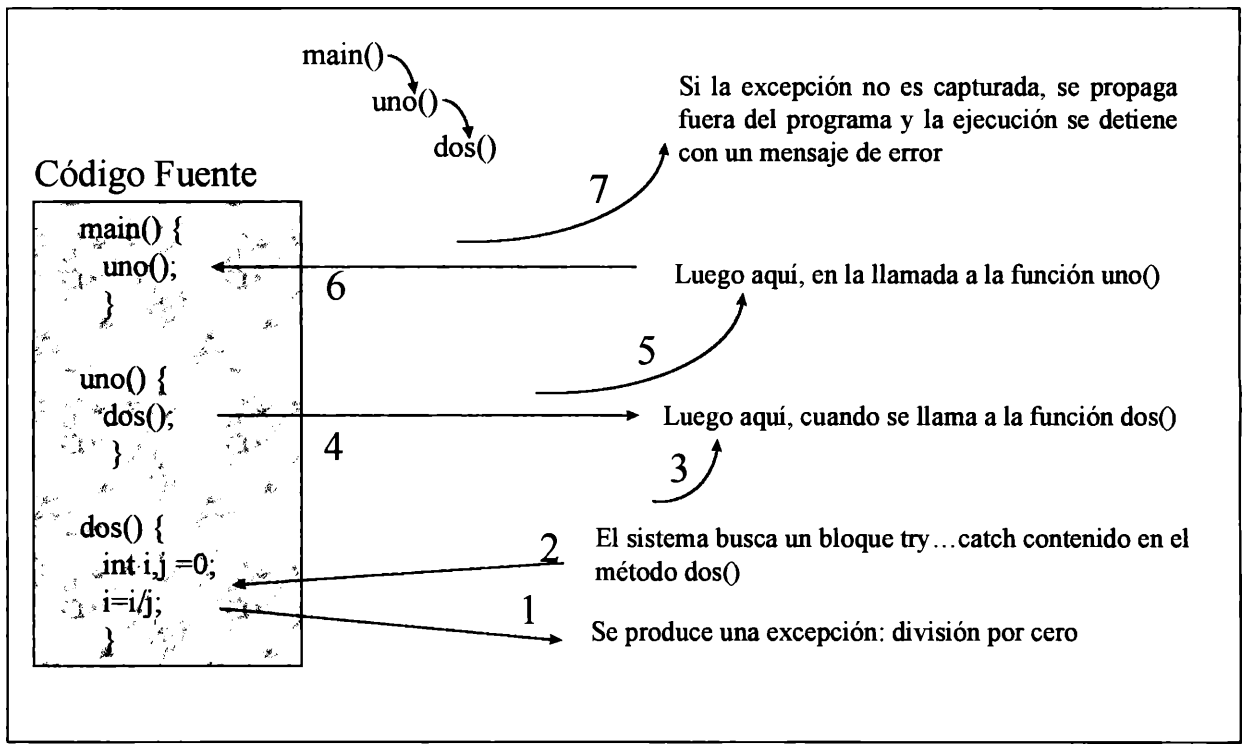
El flujo de control abandona este método y retorna al método que lo llamó. Si la llamada estaba dentro del ámbito de una sentencia try, entonces se vuelve a intentar el control de la excepción.

Si una excepción no es tratada en la rutina en donde se produce, el sistema Java busca un bloque try..catch más allá de la llamada, pero dentro del método que lo invocó. Si la excepción se propaga hasta lo alto de la pila de llamadas sin encontrar un controlador específico para la excepción, entonces la ejecución se detendrá dando un mensaje. Es decir, se puede suponer que Java proporciona un bloque catch por defecto, que imprime un mensaje de error y sale.

No hay ninguna sobrecarga en el sistema por incorporar sentencias try al código. La sobrecarga se produce cuando se genera la excepción.

Por lo tanto un método debe capturar las excepciones que genera, o en todo caso, declararlas como parte de su llamada. Esto debe ser así para que cualquiera que escriba una llamada a ese método prevea la llegada de una excepción en lugar del valor de retorno normal. Esto permite al programador que invoca a ese método, elegir entre controlar la excepción o propagarla hacia arriba en la pila de llamadas.

En el siguiente diagrama se muestra cómo se propaga la excepción que se genera en el código, a través de la pila de llamadas durante la ejecución del mismo.



Cuando se crea una nueva excepción, derivada de una clase Exception ya existente, se puede cambiar el mensaje que lleva asociado.

Con todo el manejo de excepciones se puede concluir que proporciona un método más seguro para el control de errores, además representa una excelente herramienta para organizar en sitios concretos todo el manejo de los mismos.

La degradación que se produce en la ejecución de programas con manejo de excepciones está compensada por las ventajas que representa en cuanto a seguridad de funcionamiento de esos mismos programas.

THREADS

FLUJO EN PROGRAMAS

Programas de flujo único

Un programa de flujo único o mono-hilvanado utiliza un único flujo para controlar su ejecución. Muchos de los applets y aplicaciones son de flujo único sin especificarlo explícitamente.

Programas de flujo múltiple

Java posibilita la creación y control de threads explícitamente. La simplicidad para crear, configurar y ejecutar threads, permite que se puedan implementar poderosos applets.

Mientras que los programas de flujo único pueden realizar su tarea ejecutando las subtareas secuencialmente, un programa multithreaded permite que cada thread comience y termine tan pronto como sea posible. Este comportamiento presenta una mejor respuesta a la entrada en tiempo real.

CREACIÓN Y CONTROL DE THREADS

Creación de un Thread

Hay dos formas de obtener threads en Java. Una es implementando la interfaz Runnable y la otra es extender la clase Thread.

La implementación de la interfaz Runnable es la forma habitual de crear threads. La interfaz define el trabajo y las clases que implementan la interfaz realizan ese trabajo. Los diferentes grupos de clases que implementen la interfaz tendrán que seguir las mismas reglas de funcionamiento.

Extendiendo la clase Thread, se pueden heredar los métodos y variables de la clase padre. En este caso, solamente se puede extender o derivar una vez de la clase padre. Pero esta limitación de Java puede ser superada a través de la implementación de Runnable:

```
public class MiThread implements Runnable {
  Thread t;
  public void run() {
    // Ejecución del thread una vez creado
    }
  }
```

En este caso necesitamos crear una instancia de Thread para que el sistema pueda ejecutar el proceso como un thread. Además, el método abstracto run() que está definido en la interfaz Runnable tiene que ser implementado.

Arranque de un Thread

Las aplicaciones ejecutan main() tras arrancar. Esta es la razón por la cual main() es el lugar natural para crear y arrancar otros threads.

La línea de código:

```
t1 = new TestTh( "Thread 1",(int)(Math.random()*2000) );
```

crea un nuevo thread. Los dos argumentos pasados representan el nombre del thread y el tiempo que queremos que espere antes de imprimir el mensaje.

Al tener control directo sobre los threads, tenemos que arrancarlos explícitamente.

Manipulación de un Thread

Luego de la creación del thread t1, este es controlado a través del método run().

```
t1.start()
```

Una vez dentro de run(), podemos comenzar las sentencias de ejecución como en otros programas. El método run() sirve como rutina main() para los threads; cuando run() termina, también lo hace el thread. Todo lo que se espera que el thread realice, debe estar dentro de run(), por eso cuando se dice que un método es Runnable, se debe escribir un método run().

El método sleep() simplemente le dice al thread que duerma durante los milisegundos especificados. Se debería utilizar sleep() cuando se pretende retrasar la ejecución del thread. El método sleep() no consume recursos del sistema mientras el thread duerme. De esta forma otros threads pueden seguir funcionando.

```
t1.sleep( retardo );
```

Suspensión de un Thread

Puede resultar útil suspender la ejecución de un thread sin marcar un límite de tiempo. Para este tipo de control se puede utilizar el método suspend().

```
t1.suspend();
```

Este método no detiene la ejecución permanentemente. El thread es suspendido y para volver a activarlo se invoca al método resume():

```
t1.resume();
```

Parada de un Thread

El último elemento de control que se necesita sobre threads es el método stop(). Se utiliza para terminar la ejecución de un thread:

```
t1.stop();
```

Esta llamada no destruye el thread, sino que detiene su ejecución. La ejecución ya no se puede reanudar con t1.start(). Cuando se liberan las variables que se usan en el thread, el objeto thread (creado con new) queda marcado para ser eliminado por el Garbage Collector.

Los programas más complejos necesitan un control sobre cada uno de los threads que lanzan, el método stop() puede utilizarse para esto.

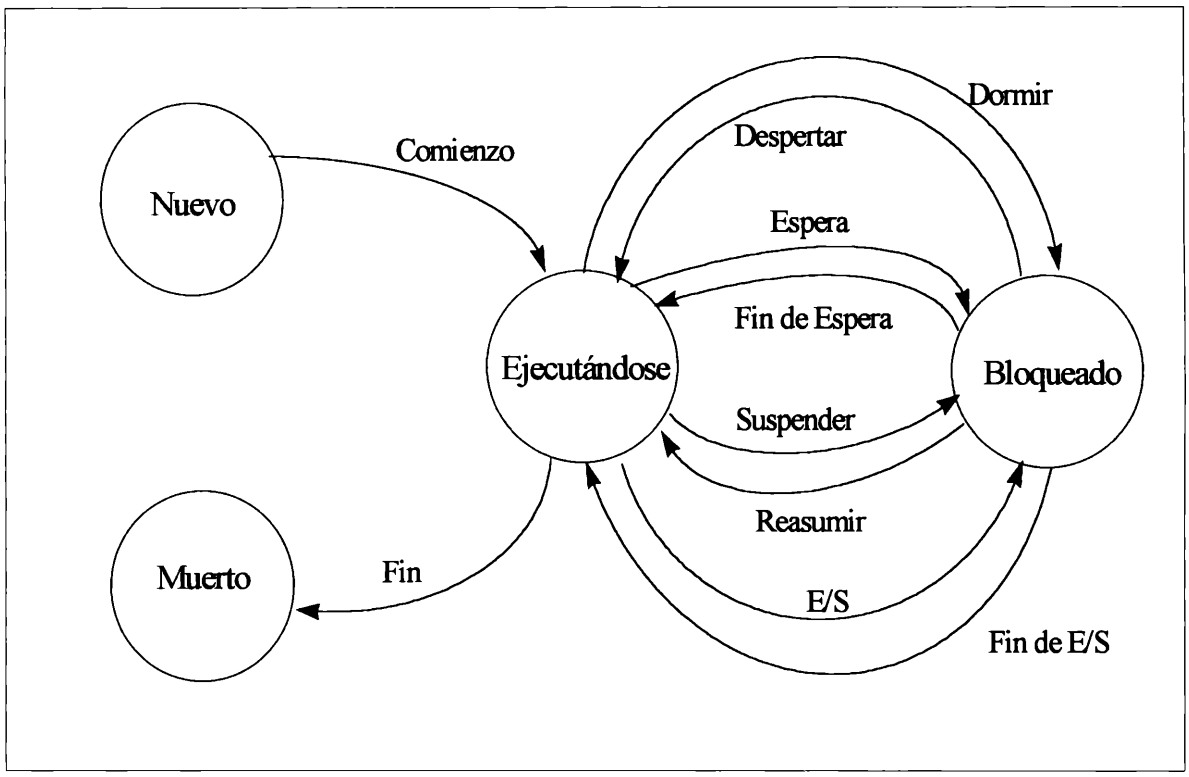
Para comprobar si un thread está vivo o no, exite el método isAlive(). Se considera vivo a un thread que ha comenzado y no ha finalizado.

```
t1.isAlive();
```

Este método devolverá true en caso de que el thread t1 esté vivo, es decir, ya se haya llamado a su método run() y no haya sido parado con un stop() ni haya terminado el método run() en su ejecución.

ESTADOS DE UN THREAD

Durante el ciclo de vida de un thread, éste se puede encontrar en diferentes estados. En el siguiente gráfico se muestran estos estados y los métodos que provocan el paso de un estado a otro.



Nuevo Thread

La siguiente sentencia crea un nuevo thread pero no lo arranca, lo deja en el estado de "Nuevo Thread":

```
Thread MiThread = new MiClaseThread();
```

Cuando un thread está en este estado, es simplemente un objeto Thread vacío, es decir, el sistema no ha destinado ningún recurso para él. Desde este estado solamente puede arrancarse llamando al método start(), o detenerse definitivamente, llamando al método stop(); la llamada a cualquier otro método carece de sentido y lo único que provocará será la generación de una excepción de tipo IllegalThreadStateException.

Ejecutable

En el siguiente código, la llamada al método start() creará los recursos del sistema necesarios para que el thread pueda ejecutarse, lo incorpora a la lista de procesos disponibles para ejecución del sistema y llama al método run() del thread.

```
Thread MiThread = new MiClaseThread();
MiThread.start();
```

En este momento nos encontramos en el estado "Ejecutable" del diagrama. Y este estado es Ejecutable y no En Ejecución, porque cuando el thread está aquí no está corriendo. Muchos ordenadores tienen solamente un procesador lo que hace imposible que todos los threads estén corriendo al mismo tiempo. Java implementa un tipo de scheduling o lista de procesos, que permite que el procesador sea compartido entre todos los procesos o threads que se encuentran en la lista. Sin embargo, para nuestros propósitos, y en la mayoría de los casos, se puede considerar que este estado es realmente un estado "En Ejecución", porque la impresión que produce ante nosotros es que todos los procesos se ejecutan al mismo tiempo.

Cuando el thread se encuentra en este estado, todas las instrucciones de código que se encuentren dentro del bloque declarado para el método run(), se ejecutarán secuencialmente.

Bloqueado

El thread entra en estado "Bloqueado" cuando se invoca al método suspend(), cuando se llama al método sleep(), cuando el thread está bloqueado en un proceso de entrada/salida o cuando el thread utiliza su método wait() para esperar a que se cumpla una determinada condición.

Para cada una de los cuatro modos de entrada en estado Bloqueado, hay una forma específica de volver a estado Ejecutable. Cada forma de recuperar ese estado es exclusiva; por ejemplo, si el thread ha sido puesto a dormir, una vez transcurridos los milisegundos que se especifiquen, él solo se despierta y vuelve a estar en estado Ejecutable. Llamar al método resume() mientras esté el thread durmiendo no serviría para nada.

Los métodos de recuperación del estado Ejecutable, en función de la forma en que han entrado en estado Bloqueado, son los siguientes:

- Si un thread está dormido, pasado el lapso de tiempo.
- Si un thread está suspendido, luego de una llamada al método resume().
- Si un thread está bloqueado en una entrada/salida, una vez que el comando E/S concluya su ejecución.
- Si un thread está esperando por una condición, cada vez que la variable que controla esa condición varíe debe llamarse a notify() o notifyAll().

Muerto

Un thread se puede morir de dos formas: por causas naturales o porque lo maten (con stop()). Un thread muere normalmente cuando concluye de forma habitual su método run().

El método stop() envía un objeto ThreadDeath al thread que quiere detener. El thread morirá en el momento en que reciba la excepción ThreadDeath.

Los applets utilizarán el método stop() para matar a todos sus threads cuando el navegador con soporte Java en el que se están ejecutando le indica al applet que se detengan, por ejemplo, cuando se minimiza la ventana del navegador o cuando se cambia de página.

El método isAlive()

La interfaz de programación de la clase Thread incluye el método isAlive(), que devuelve true si el thread ha sido arrancado (con start()) y no ha sido detenido (con stop()). Por ello, si el método isAlive() devuelve false, se tiene "Nuevo Thread" o un thread "Muerto". Si devuelve true, el thread se encuentra en estado "Ejecutable" o "Bloqueado".

SCHEDULING

Java tiene un Scheduler, una lista de procesos, que monitoriza todos los threads que se están ejecutando en todos los programas y decide cuales deben ejecutarse y cuales deben encontrarse preparados para su ejecución. Hay dos características de los threads que el scheduler identifica en este proceso de decisión. Una, la más importante, es la prioridad del thread; la otra, es el indicador de demonio. La regla básica del Scheduler es que si solamente hay threads demonio ejecutándose, la Máquina Virtual Java (JVM) concluirá. Los nuevos threads heredan la prioridad y el indicador de demonio de los threads que los han creado. El Scheduler determina qué threads deberán ejecutarse comprobando la prioridad de todos los threads, aquellos con prioridad más alta dispondrán del procesador antes de los que tienen prioridad más baja.

El Scheduler puede seguir dos patrones, preemptivo y no-preemptivo. Los schedulers preemptivos proporcionan un segmento de tiempo a todos los threads que están corriendo en el sistema. El Scheduler decide cual será el siguiente thread a ejecutarse y llama a resume() para darle vida durante un período fijo de tiempo. Cuando el thread ha estado en ejecución ese período de tiempo, se llama a suspend() y el siguiente thread en la lista de procesos será relanzado (resume()). Los schedulers no-preemptivos deciden que thread debe correr y lo ejecutan hasta que concluye. El thread tiene control total sobre el sistema mientras esté en ejecución. El método yield() es la forma en que un thread fuerza al Scheduler a comenzar la ejecución de otro thread que esté esperando. Dependiendo del sistema en que esté corriendo Java, el Scheduler será preemptivo o no-preemptivo.

AWT - ABSTRACT WINDOW TOOLKIT

INTRODUCCIÓN AL AWT

AWT es una biblioteca de clases Java para el desarrollo de Interfaces de Usuario Gráficas (GUI). El entorno que ofrece es demasiado simple, no se han tenido en cuenta las ideas de entornos gráficos novedosos, sino que se ha ahondado en estructuras orientadas a eventos, sin soporte para la construcción gráfica. Quizá la presión de tener que lanzar algo al mercado haya tenido mucho que ver en la pobreza de AWT.

La estructura básica del AWT se basa en Componentes y Contenedores. Estos últimos pueden contener Componentes y son Componentes a su vez, de forma que los eventos pueden tratarse tanto en Contenedores como en Componentes. Es responsabilidad del programador el ensamble de todas las piezas, y la seguridad del tratamiento de los eventos.

INTERFACES DE USUARIO

Las interfaces de usuario pueden adoptar muchas formas, que van desde la simple línea de comandos hasta las interfaces gráficas que proporcionan las aplicaciones más modernas.

La interfaz de usuario es el aspecto más importante de cualquier aplicación. Una aplicación sin una interfaz fácil, impide que los usuarios saquen el máximo rendimiento del programa. Java proporciona los elementos básicos para construir interfaces de usuario amigables a través del AWT.

En el nivel más bajo, el sistema operativo transmite información desde el mouse y el teclado como dispositivos de entrada al programa. El AWT fue diseñado de forma tal que el programador pueda abstraerse de detalles tales como controlar el movimiento del mouse o leer el teclado. El AWT constituye una librería de clases orientadas a objetos para cubrir estos recursos y servicios de bajo nivel.

Debido a que el lenguaje de programación Java es independiente de la plataforma en que se ejecutan sus aplicaciones, el AWT también es independiente de la plataforma en que se ejecute. Los elementos de interfaz proporcionados por el AWT están

implementados utilizando toolkits nativos de las plataformas, preservando una apariencia semejante a todas las aplicaciones que se ejecutan en dicha plataforma.

Peers

Son clases que manejan las implementaciones específicas de cada plataforma. Permiten que cada componente conserve el mismo look and feel del ambiente en que se ejecuta la aplicación (Windows, MAC, etc.).

Cada componente AWT tiene una componente peer nativa equivalente.Por ejemplo, un objeto Button tiene asociado un objeto ButtonPeer que se instancia en cada plataforma. Las componentes AWT delegan a los peers gran parte de la funcionalidad asociada a su apariencia.

ESTRUCTURA DEL AWT

La estructura de la versión actual del AWT podemos resumirla en los puntos que se detallan a continuación:

- Los Contenedores contienen Componentes, que son los controles básicos.
- Las Componetes son clases que definen objetos de GUI's (botones, labels, etc).
- A través de los Layouts se posicionan los objetos dentro de los Contenedores.
- Provee un nivel alto de abstracción respecto al entorno de ventanas en que se ejecute la aplicación.
- La arquitectura de la aplicación es dependiente del entorno de ventanas. No tiene un tamaño fijo.
- La clase Graphic permite definir métodos para realizar operaciones gráficas sobre componentes.

COMPONENTES Y CONTENEDORES

Una interfaz gráfica está construida basándose en elementos gráficos básicos, los Componentes. Típicos ejemplos de estos Componentes son los botones, las barras de desplazamiento, etiquetas, listas, campos de texto, etc. Los Componentes permiten al usuario interactuar con la aplicación. En el AWT, todos los Componentes de la interfaz de usuario son instancias de la clase Component o de una de sus subclases.

Los Componentes no se encuentran aislados, sino agrupados dentro de Contenedores. Los Contenedores contienen y organizan la situación de los Componentes; además, los Contenedores son en sí mismos Componentes y como tales pueden ser situados dentro de otros Contenedores. También contienen el código necesario para el control de eventos, cambiar la forma del cursor o modificar el ícono de la aplicación. En el AWT, todos los Contenedores son instancias de la clase Container o una de sus subclases.

COMPONENTES

Component es una clase abstracta que tiene asociada una posición, un tamaño, un peer, un formato de letra, color, y además responden a eventos.

Los Componentes deben circunscribirse dentro del Contenedor que los contiene. Esto hace que el anidamiento de Componentes dentro de Contenedores creen árboles de elementos, comenzando con un Contenedor en la raiz del árbol y expandiéndolo en sus ramas.

Los Objetos derivados de la clase Component son los que aparecen a continuación:

- Button
- Canvas
- Checkbox
- Choice
- Container
  - Panel
  - Window
    - Dialog
    - Frame
- Label
- List
- Scrollbar
- TextComponent
  - TextArea
  - TextField

JFC - JAVA FOUNDATION CLASSES

LA LIBRERÍA JFC

Contiene las componentes Swing y las APIs que se mencionan a continuación:

♣ Java 2D: Conjunto de Clases de Gráficos e Imágenes 2D. Además ofrece diferentes estilos de dibujos, mecanismos para definir formas complejas, etc.

♣ Drag and Drop: Posibilita la transferencia de información entre Java y Aplicaciones nativas y dentro de los programas JAVA.

♣ Accessibility: Estas clases son compatibles con Hardware y Software diseñados para usuarios con necesidades especiales, tales como vista limitada o incapacidad para operar con el mouse.

Accessibility es una característica muy importante de los Swing, ya que es el único conjunto de componentes de lenguaje Java que puede ser utilizado para escribir páginas Web que sean accesibles para gente discapacitada.

Además, también pueden ser útiles para gente que no posea discapacidades, ya que por ejemplo permite crear programas para "touch screen".

INTRODUCCIÓN A LAS CLASES SWING

El conjunto de componentes Swing es una herramienta de Interface Gráfica de Usuario (GUI) que simplifica el desarrollo de elementos visuales tales como menúes, barras, diálogos, etc.

Con las Swing se pueden construir GUIs que posean un determinado look & feel, el cual es definido por el programador, e independientemente de la plataforma en la cual se ejecute el programa, la apariencia definida no cambia.

Además las componentes Swing, se definen como "lightweight", ya que no se apoyan en el código de interface del sistema operativo subyacente. Consecuentemente pueden ser implementadas usando menos código del que requieren las componentes "heavyweight".

LAS TRES PARTES DE LAS SWING

Las Swing están implementadas utilizando tres elementos principales que son parte de las JFC.

- Una nueva rama de componentes que desciende de JComponent. Ésta clase desciende de la clase contenedora de AWT y es la raíz de casi todos los componentes Swing. Esta jerarquía permite a los componentes contener otros componentes.
- Un conjunto de clases de soporte que no crean componentes visibles, pero proveen servicios vitales a las APIs Swing y a las Aplicaciones que se desarrollen con éstas.
- Un nuevo conjunto de Interfaces Swing relacionadas y que están implementadas por las componentes Swing y las clases de soporte.

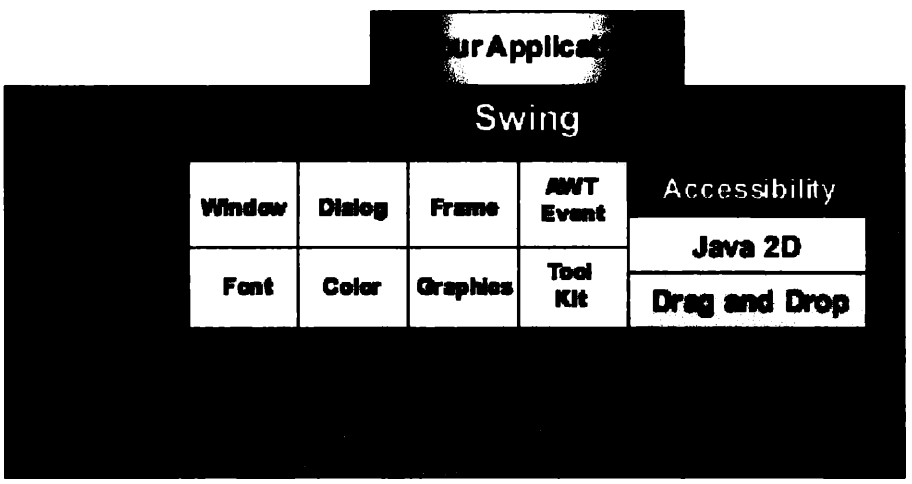
PAQUETES INCLUIDOS EN LAS SWING

- Accessibility: Permiten construir programas para gente con discapacidades físicas.
- Swing: Implementa casi todas las clases componentes utilizadas en swing.
- Basic: Este paquete define el look & feel predeterminado de los componentes swing. Extendiendo este paquete se puede redefinir el look & feel.
- BeanInfo: Define la clase SwingBeanInfo que es la Superclase de todas las clases BeanInfo swing.
- Border: Permite dibujar distintos tipos de bordes alrededor de los componentes.
- Event: Define las clases específicas de eventos.
- Multi: Provee clases que permiten a los componentes tener Interfaces con soporte de herencia múltiple.
- Look & Feel: Contiene clases que las swing utilizan para permitir que sus componentes luzcan de diferentes maneras.

- Table: A través de este paquete se provee el manejo de tablas.
- Text: Contiene clases e interfaces para el manejo de componentes con texto.
- RTF: Implementa un editor de texto para el manejo de archivos RTF (Rich Text Format).
- Tree: A través de este paquete se provee el manejo de árboles.
- Undo: Permite deshacer acciones realizadas en componentes de edición de texto.

MODELO DE ARQUITECTURA SWING

Desde el punto vista de la arquitectura, el conjunto de componentes Swing, extiende, pero no reemplaza a las componentes AWT.



El diagrama muestra como las Swing se sitúan sobre un número de APIs que implementan las diferentes partes de las Java Foundation Classes, incluyendo Java 2D y Drag & Drop. A pesar de que éstas últimas forman parte de la JFC, no se incluyen en las Swing. Esto se debe a que varias de las tareas que utilizan requieren uso de código nativo de la máquina y se sabe que las Swing nunca lo utilizan. Por esto, en el diagrama, las Java 2D y Drag & Drop no aparecen dentro del rectángulo titulado Swing.

SWING Y AWT

Tanto AWT como Swing son componentes lightweight, pero las clases Swing tienen más objetos, más propiedades y funcionalidad que las componentes AWT. Por esto en la mayoría de los casos se utilizan componentes Swing.

Por defecto las componentes Swing toman el Look & Feel de la plataforma donde corren, cuando se crea un programa y se corre en entorno Windows, éste tiene la apariencia y el comportamiento de un programa escrito para Windows. Cuando se ejecuta el mismo programa en entorno Unix, éste se comporta como un programa escrito para Unix, y así sucede con el resto de los Sistemas Operativos. Esto puede salvarse usando el Look & Feel de las componentes Swing, ya que a diferencia de las AWT, la aplicación puede presentar siempre la misma apariencia sobre los diferentes entornos. Esto es posible gracias a que los Swing no utilizan ningún código de las plataformas sobre las cuales corren. Estas presentan un conjunto de objetos con el mismo comportamiento y la misma apariencia, sin tener en cuenta el Sistema Operativo subyacente.

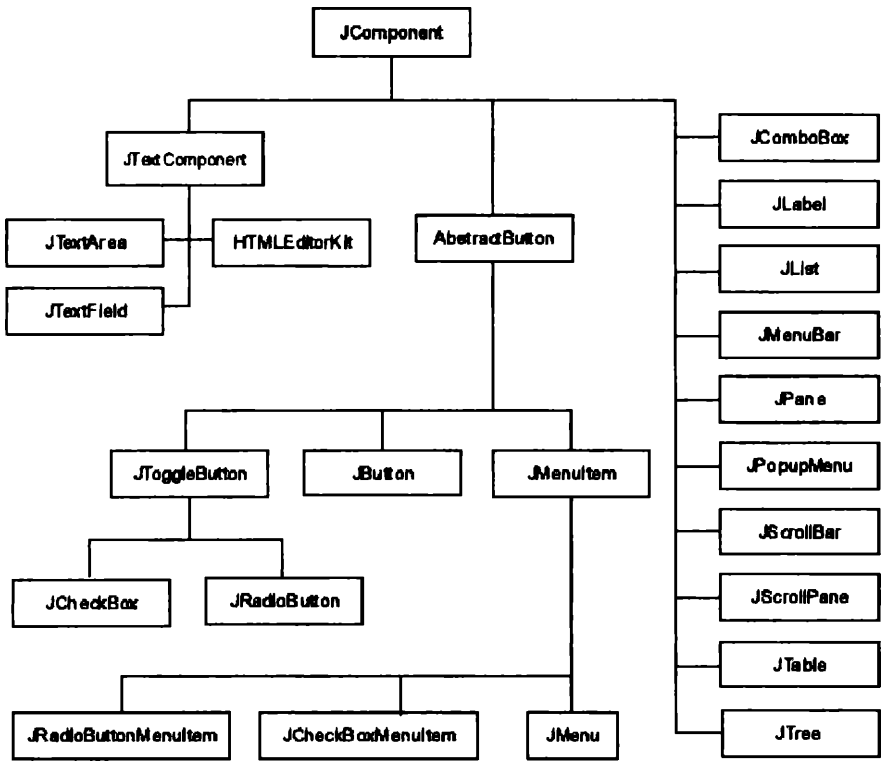
Por lo tanto, utilizando componentes Swing los desarrolladores pueden crear su propio conjunto de objetos con la apariencia que deseen.

Jerarquía de herencias

La jerarquía de las componentes Swing es similar a la de las componentes AWT, pero si se examinan con detenimiento se verán algunas diferencias.

La cantidad de componentes que posee el conjunto de herramientas de las clases Swing duplica a las de AWT. La primera es una librería extensiva de más de 250 clases y más de 75 interfaces para crear componentes lightweight 100% puro Java, cuya jerarquía de componentes ha sido simplificada.

El siguiente diagrama muestra sólo una pequeña porción de la jerarquía de las Swing, pero también muestra algunas diferencias importantes en el funcionamiento de la jerarquía de ambos paquetes.



Clases UI y clases no UI

En este diagrama se vislumbran dos características de las Swing: todas las clases de componentes Swing comienzan con "J" y todas descienden de la clase JComponent. Sin embargo no se debe concluir que todas las clases Swing son J clases, ni que todas descienden de Jcomponent.

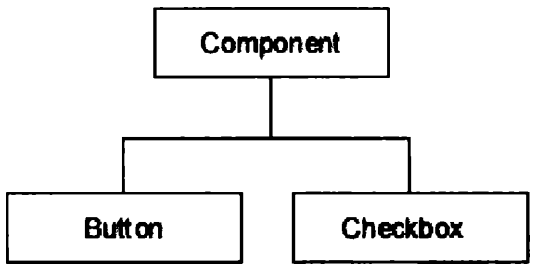
Las componentes Swing tienen dos tipos de clases: las clases UI y las clases no UI, y sólo las clases UI son J clases y descienden de Jcomponent.

La diferencia entre ambos tipos de clases es que el primero crea componentes visibles, tales como botones y menúes, y el otro, provee servicios para las del primer tipo. Las clases de eventos y de modelos corresponden al segundo grupo de clases.

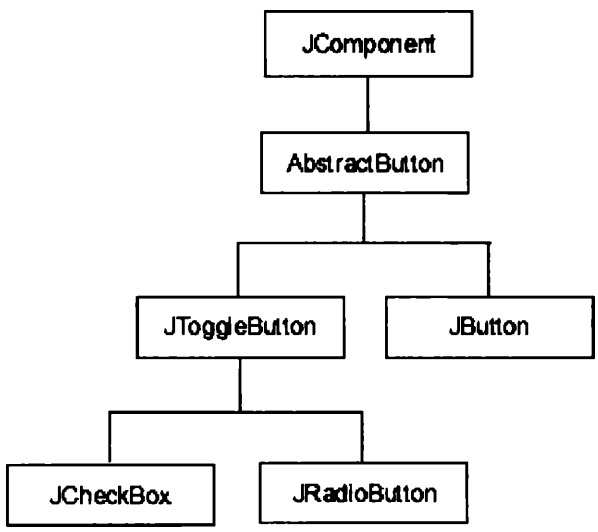
Cambios en la Jerarquía de Button

Se pueden observar diferencias en cómo funciona la jerarquía de Botones en ambos paquetes.

En AWT hay sólo dos clases de botones: Button y CheckBox, y ambas descienden de la clase Component.



En las Swing hay dos clases nuevas de botones y la jerarquía ha cambiado como muestra la figura:



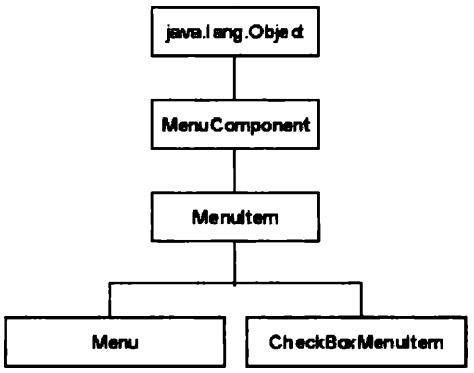
Las Swing poseen las clases radio button y check box, mientras que las AWT poseen la clase CheckBox que permite programar radio button y check box.

En las Swing, estas clases descienden de una nueva clase: JToggleButton, la cual tiene tres estados: presionado, habilitado y seleccionado. En las Swing, todas estas clases descienden de la clase abstracta AbstractButton.

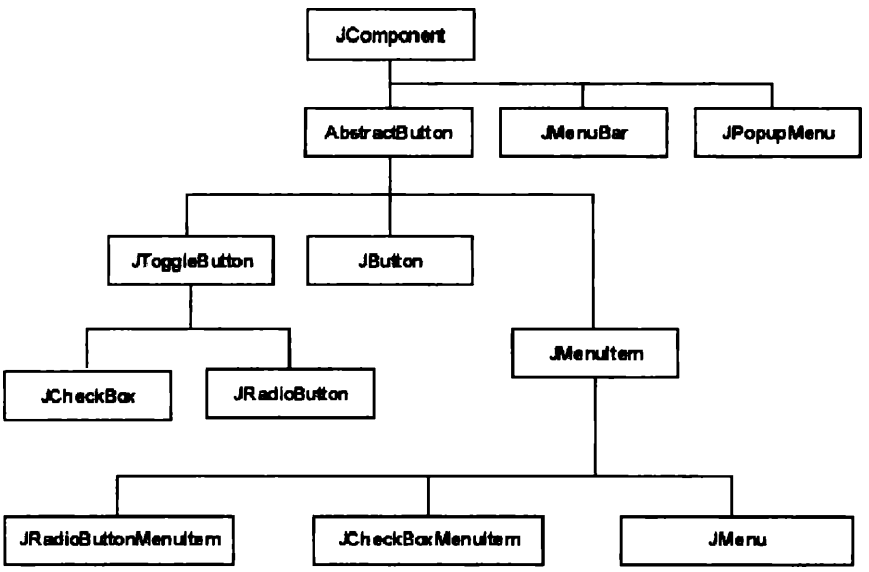
Cambios en la jerarquía Menú

Hay otras diferencias entre la jerarquía de las Swing y la de la AWT.

En las AWT, la clase Menú no desciende de la clase Component, pero sí lo hace de la clase Object. La jerarquía tiene la siguiente forma:



En las Swing, todas las clases menú descienden de la Jcomponent en un árbol más complejo que también involucra a la clase botón.



Debido a que los árboles de la clase botón y de la clase menú están estrechamente ligados, es más fácil coordinar los eventos del botón y de los menúes.

De hecho, las Swing tienen una nueva interface llamada Action Interface, que puede escuchar eventos de botón y de menú al mismo tiempo.

MEJORAR EL DISEÑO DE INTERFACES

La interfaz de usuario es el aspecto más importante de una aplicación. Un diseño pobre de interfaz es un grave problema para que el usuario pueda obtener el mejor resultado de la aplicación. Para ser efectivos, no debemos limitarnos a colocar una serie de botones, etiquetas y barras de desplazamiento sobre la pantalla.

Para crear una aplicación, se deberán tener en cuenta los puntos que se detallan a continuación:

Interface

- Crear el Marco de la aplicación (Frame, Panel).
- Inicializar Fuentes, Colores, Layouts y demás recursos.
- Crear menúes y Barras de Menúes.
- Crear los controles, diálogos, ventanas, etc.

Codificación

- Incorporar controladores de eventos
- Incorporar funcionalidad (threads, red, etc.)
- Incorporar un controlador de errores (excepciones)

CONTROL DE EVENTOS

Anteriormente, un programa que quería saber lo que estaba haciendo el usuario, debía recoger por sí mismo la información necesaria. Esto significaba que una vez inicializado, el programa entraba en un gran bucle en el que continuamente se bloqueaba para comprobar que el usuario estuviese haciendo algo interesante y tomar las acciones oportunas. Esta técnica se conoce como polling.

El polling funciona, pero se vuelve demasiado difícil de manejar con las aplicaciones modernas por dos razones fundamentales: primero, el uso de polling tiende a colocar todo el código de control de eventos en una única localización (dentro de un gran bucle); segundo, las interacciones dentro del gran bucle tienden a ser muy complejas. Además, el polling necesita que el programa esté ejecutando el bucle, consumiendo tiempo de CPU, mientras está esperando a que el usuario realice una operación, ésto supone un gran insumo de recursos.

El AWT y las JFC resuelven estos problemas abrazando un paradigma diferente, en el que están basados todos los sistemas modernos de ventanas: la orientación a eventos. Todas las acciones que pueda realizar el usuario caen dentro del conjunto de eventos. Un evento describe, con suficiente detalle, una acción particular del usuario. En lugar de que el programa activamente recoja todos los eventos generados por el usuario, el sistema Java avisa al programa cuando se produce un evento interesante.

LA CLASE EVENT

Un contenedor en un entorno gráfico se convierte en un receptor de eventos de todo tipo, relacionados con el movimiento del mouse, pulsaciones de teclas, creación/movimiento/destrucción de partes gráficas y, por último, los referidos a acciones del usuario respecto de componentes (elección de un menú, pulsación de un botón, etc.).

La clase Event es el jugador principal en el juego de los eventos. Intenta capturar las características fundamentales de todos los eventos que genera el usuario. Los datos miembro de la clase Event son los que se indican a continuación:

id - El tipo de evento que se ha producido

target - Componente sobre el que se ha producido el evento

x, y - Las coordenadas en donde se ha producido el evento relativas al Componente que actualmente está procesando ese evento. El origen se toma en la esquina superior izquierda del Componente

key - Para eventos de teclado, es la tecla que se ha pulsado. Su valor será el valor Unicode del carácter que representa la tecla. Otros valores que puede tomas son los de las teclas especiales como INICIO, FIN, F1, F2, etc.

when - Instante en que se ha producido el evento

modifiers - La combinación aritmética del estado en que se encuentran las teclas modificadoras Mays, Alt, Ctrl.

clickCount - El número de clicks de mouse consecutivos. Sólo tiene importancia en los eventos MOUSE_DOWN

arg - Es un argumento dependiente del evento. Para objetos Button, este objeto arg es un objeto String que contiene la etiqueta de texto del botón

evt - El siguiente evento en una lista encadenada de eventos

Una instancia de la clase Event será creada por el sistema Java cada vez que se genere un evento. Es posible, sin embargo, que un programa cree y envíe eventos a los Componentes a través de su método postEvent().

TIPOS DE EVENTOS

Los eventos se catalogan por su naturaleza, que se indicará en el miembro id de su estructura. Los grandes grupos de eventos son:

Eventos de Ventana

Son los que se generan en respuesta a los cambios de una ventana un frame o un diálogo.

- WINDOW_DESTROY
- WINDOW_EXPOSE
- WINDOW_ICONIFY
- WINDOW_DEICONIFY
- WINDOW_MOVED

Eventos de Teclado

Son generados en respuesta a cuando el usuario pulsa y suelta una tecla mientras un Componente tiene el foco de entrada.

- KEY_PRESS
- KEY_RELEASE
- KEY_ACTION
- KEY_ACTION_RELEASE

Eventos de Mouse

Son los eventos generados por acciones sobre el mouse dentro de los límites de un Componente.

- MOUSE_DOWN
- MOUSE_UP
- MOUSE_MOVE
- MOUSE_ENTER
- MOUSE_EXIT

- MOUSE_DRAG

Eventos de Barras

Son los eventos generados como respuesta a la manipulación de barras de desplazamiento (scrollbars).

- SCROLL_LINE_UP
- SCROLL_LINE_DOWN
- SCROLL_PAGE_UP
- SCROLL_PAGE_DOWN
- SCROLL_ABSOLUTE

Eventos de Lista

Son los eventos generados al seleccionar elementos de una lista.

- LIST_SELECT
- LIST_DESELECT

Eventos Varios

Son los eventos generados en función de diversas acciones.

- ACTION_EVENT
- LOAD_FILE
- SAVE_FILE
- GOT_FOCUS
- LOST_FOCUS

El Applet EventosPrnt.java está diseñado para observar los eventos que se producen sobre él. Cada vez que se genera un evento, el applet responde imprimiendo el evento que ha capturado en la línea de comandos desde donde se ha lanzado el applet.

MÉTODOS DE CONTROL DE EVENTOS

El método handleEvent() es un lugar para que el programador pueda insertar código para controlar los eventos. A veces, sin embargo, un Componente solamente estará interesado en eventos de un cierto tipo (por ejemplo, eventos del mouse). En estos casos, el programador puede colocar el código en un método de ayuda, en lugar de colocarlo en el método handleEvent().

La implementación del método handleEvent() proporcionada por la clase Component invoca a cada método de ayuda. Por esta razón, es importante que las implementaciones redefinidas del método handleEvent() en clases derivadas, siempre finalicen con la sentencia:

```
return( super.handleEvent( evt ) );
```

El siguiente trozo de código ilustra esta regla.

```
public boolean handleEvent( Event evt ) {
  if( evt.target instanceof MiBoton )
    {
    // Hace algo...
    return true;
    }
  return( super.handleEvent( evt ) );
  }
```

No seguir esta regla tan simple hará que no se invoquen adecuadamente los métodos de ayuda.

ACTION_EVENT

Algunos de los eventos que más frecuentemente tendremos que controlar son los siguientes:

- ACTION_EVENT
- MOUSE_DOWN
- KEY_PRESS
- WINDOW_DESTROY

Como ejemplo del manejo de eventos vamos a ver este evento que se provoca al pulsar un botón, seleccionar un menú, etc. Para su control podemos manejarlo en el método handleEvent() o en el método action().

Los dos métodos anteriores pertenecen a la clase Component por lo que todas las clases derivadas de ésta contendrán estos dos métodos y se pueden sobrecargar para que se ajuste su funcionamiento a lo que requiere nuestra aplicación.

En el siguiente ejemplo, se controla este evento a través del método handleEvent(), que es el método general de manejo de eventos:

```
public boolean handleEvent( Event evt ) {
  switch( evt.id ) {
    case Event.ACTION_EVENT:
      // evt.arg contiene la etiqueta del botón pulsado
      // o el item del menú que se ha seleccionado
      if( ( "Pulsado "+n+" veces" ).equals( evt.arg ) )
        return( true );
    default:
        return( false );
    }
  }
```

Pero en este caso, al producirse este evento, llama al método action(), que sería:

```
public boolean action( Event evt,Object arg ) {
  if( ( "Pulsado "+n+" veces" ).equals( arg ) )
    return( true );
  return( false );
  }
```

TRABAJO DE GRADO

Aplicación a distancia para estudiantes

Antonia Maraviglia – Mariángeles Pucci

Director: Lic. Javier Díaz

Facultad de Informática

Universidad Nacional de La Plata

"Si piensas que estás vencido lo estás,
si piensas que no te atreves no lo harás,
si dices me gustaría ganar, pero no puedo, no vencerás,
porque el hombre que gana no es el más fuerte,
ni el más poderoso,
sino aquel que cree poder hacerlo."

"Creer posible algo es hacerlo cierto"

Hebbel

Dedicado a:

Nuestros Padres
Martín
Fabian
María Luján
Javier Diaz
Laura Fava

Y a todos aquellos que lo creyeron posible.

INDICE

TRABAJO DE GRADO - APLIADES

Especificación

TRABAJO DE GRADO

DIRECTOR

Lic. Javier Díaz

TEMA

Front-end basado en el Lenguaje Java para posibilitar el tomado de cursos a distancia.

OBJETIVOS

Realizar una aplicación que permita a los alumnos de los distintos cursos de la Facultad de Informática de la Universidad Nacional de La Plata, el acceso a distancia a los servicios que brinda la misma.

PROPUESTA

El director de proyecto, propone realizar una aplicación kerberizada, cuyo objetivo era integrar distintas tareas que se llevan a cabo en la Facultad, con una Interface homogénea y susceptible de la definición de diferentes perfiles de usuarios, utilizando el lenguaje Java como herramienta de desarrollo.

Los servicios que utilizarían los diferentes tipos de usuarios, y el modelo de seguridad Kerberos, serían llevados a cabo por grupos de alumnos que se encontraban realizando el Trabajo de Grado. Se nos propuso la tarea de implementar sólo los servicios que utilizarían los alumnos de la Facultad.

Fundamentalmente nos interesó la propuesta por dos motivos, el primero, nuestra preferencia por el desarrollo de aplicaciones, al trabajo de investigación teórica; el segundo, el desafío de utilizar una herramienta nueva, casi desconocida, que prometía facilidad de uso, acceso a las aplicaciones desde cualquier punto geográfico, seguridad frente a virus e independencia de la plataforma de hardware y del sistema operativo.

ESPECIFICACIÓN

El Requerimiento General es diseñar e implementar una aplicación que permita automatizar las funciones que realizan los alumnos, egresados, profesores, empleados administrativos y personas ajenas, que concurran a la Facultad de Informática de la Universidad Nacional de La Plata, a realizar alguna tarea.

Un usuario puede tener más de un perfil/categoría asignados. Por ejemplo puede ser profesor de una materia (categoría profesor), y cursar una carrera de postgrado (categoría alumno).

Nuestra labor es diseñar e implementar el perfil alumno.

Al comenzar las especificaciones correspondientes, notamos que los grupos que debían implementar el resto de los perfiles abandonaron la tarea asignada. Por este motivo, la base de datos a utilizar, el modelo de acceso a los datos, la definición de la estructura de la base de datos, los servicios del perfil asignado y la apariencia de la aplicación fue definida por nosotras.

Los servicios que brinda la aplicación pueden clasificarse en *Servicios Generales*, que son aquellos comunes a todos los perfiles de usuario; y los *Servicios Específicos*, que son propios a cada categoría de usuario.

Por ejemplo la evaluación de los trabajos entregados por los alumnos, es un servicio específico de la categoría profesor; la inscripción a exámenes finales es un servicio que pueden realizar tanto los alumnos como los empleados administrativos; mientras que la consulta de las monografías aprobadas, puede ser accedida por cualquier persona, es decir es un servicio general.

Al *ingresar a la aplicación* el usuario debe conectarse a la misma, proveyéndole identificación de usuario, clave y perfil de acceso. Antes de intentar la conexión, debe estar registrado; si así no fuera, podrá registrarse a través de la aplicación, o concurrir a la oficina de alumnos.

Si se *registra* desde la aplicación se le otorgará automáticamente la categoría de *Usuario General*, siendo la asignación de otro perfil tarea exclusiva de los empleados

administrativos. De esta manera se evita la registración malintencionada o de perfiles erróneos.

Los datos obligatorios para la registración en forma remota son: nombre, apellido, tipo y número de documento, nombre de usuario y clave de acceso. Además existen otros datos, no obligatorios, relacionados con el domicilio particular, teléfono y casilla de correo electrónico.

Al ingresar a la aplicación, previa verificación y autenticación de datos de acceso, el usuario dispone de un conjunto de servicios específicos, los cuales dependen de la categoría con la cual establece la conexión.

Un alumno puede realizar diferentes consultas acerca de los cursos, las materias, las fechas de examen y las notas obtenidas. Además inscribirse en los cursos que se dictan en la facultad, en las materias que desea cursar y en los exámenes finales que pretende rendir. También puede obtener trabajos para completar y luego enviarlos resueltos. Incluso dispone de servicios generales que le permiten cambiar su clave de acceso, consultar monografías realizadas por otras personas y utilizar una casilla de correo electrónico.

Consulta de Cursos: La Consulta de Cursos brinda información referente a un Curso. El término curso hace referencia a carreras de grado, carreras de postgrado, cursos, seminarios y cualquier otra actividad que pueda llevarse a cabo en el ámbito educativo.

Seleccionando el área y el curso deseados, el alumno se informa acerca del título obtenido al finalizar el curso, la duración en meses o años del mismo, el costo económico del éste, en caso de no ser gratuito, el plan de estudios vigente, los requisitos que debe cumplir para poder inscribirse, las incumbencias que correspondan de acuerdo al título obtenido y otros datos particulares del curso.

Consulta de Materias: A través de ésta consulta se obtiene información referente a una materia.

Al seleccionar el curso y la materia deseada se obtiene la duración, el cupo total de la materia en caso de ser limitado, el horario en que se dictan las clases teóricas y las clases prácticas, la bibliografía recomendada por la cátedra y los requisitos que debe cumplir para poder inscribirse en la materia.

Además se pueden *Traer Trabajos* de la Materia o *Depositar Trabajos* y acceder al *Plan de Estudios* de la misma.

Un trabajo puede ser de alguno de los siguientes tipos: trabajo práctico, monografía, guía especial, examen final, examen parcial.

Traer Trabajos: Mediante este Servicio se obtienen los trabajos que deposita la cátedra para ser resueltos. Sólo pueden acceder los alumnos que se encuentran cursando la materia elegida.

Se debe seleccionar el tipo de trabajo, el trabajo en sí y el lugar en el cual desea guardarlo. No se visualiza el trabajo completo, sino el nombre, el tipo y las especificaciones del mismo.

Depositar Trabajos: Desde esta opción se pueden enviar trabajos resueltos a la cátedra. Sólo pueden acceder los alumnos que se encuentran cursando la materia elegida.

Se debe indicar el trabajo que se desea enviar, y en caso de considerarlo necesario, se pueden colocar observaciones, antes de confirmar el envío.

Plan de Estudios: Mediante esta opción el alumno accede al plan de estudios de la materia seleccionada.

Consulta de Notas: Mediante este servicio un alumno puede obtener toda la información referente a las notas de las materias que cursó o se encuentra cursando. Al seleccionar el curso y la materia de las notas a consultar, obtiene todas las calificaciones, la fecha de evaluación y el trabajo calificado de la materia elegida.

Además, desde esta opción, se puede solicitar un certificado analítico.

Consulta de Fechas de Finales: Mediante este servicio se brinda información referente a todas las fechas de exámenes finales. El alumno indica el curso y la materia, y obtiene la fecha del examen, el horario y el lugar donde debe presentarse para rendir el mismo.

Inscripción a Cursos: A través de este servicio el alumno puede realizar la inscripción a carreras o cursos en los cuales no se encuentra inscripto. Luego de elegir el área y el curso o carrera en la cual desea inscribirse, se obtendrán los datos referentes a la misma. El alumno debe confirmar la inscripción.

***Inscripción a Cursadas*:** A través de este servicio el alumno puede realizar la inscripción a las materias para las cuales se encuentra habilitado. Es decir que debe tener todas las cursadas de las correlativas precedentes aprobadas.

Luego de elegir el curso, la materia a cursar, el año de la cursada y el turno, se obtendrán los datos referentes a los horarios de teoría y práctica en que se dicta la materia en el turno elegido. El alumno debe confirmar la inscripción.

Inscripción a Finales: A través de este servicio se realiza la inscripción a los exámenes finales. El alumno sólo puede inscribirse en aquellas materias cuya cursada haya aprobado.

Se debe elegir el curso, la materia a rendir y la fecha del examen final, para obtener los datos referentes al lugar y hora del examen y confirmar la inscripción.

Consulta de Monografías: Esta consulta muestra información referente a todas las monografías de la facultad que se encuentran publicadas.

Luego de elegir el curso y la materia a los cuales pertenece la monografía a consultar, además de la monografía deseada, el alumno podrá guardarla en su máquina, indicando previamente el destino donde la desea depositar.

Cambio de Clave: La clave que permite el acceso a la aplicación puede modificarse a través de este servicio. El alumno debe ingresar el nombre de usuario, la clave de acceso anterior y la clave con la que accederá en las próximas sesiones; y posteriormente confirmar el cambio deseado.

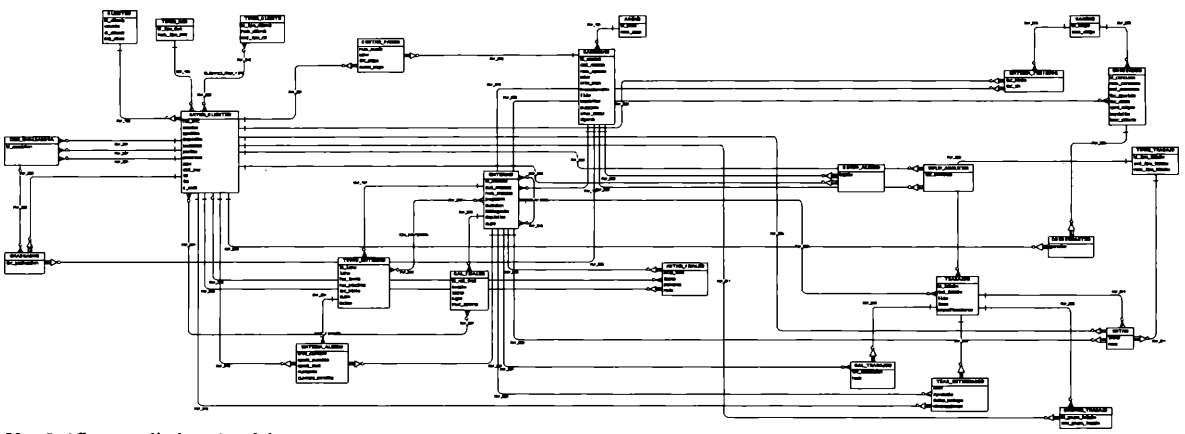
Correo Electrónico: A través de este servicio el alumno dispone de su cuenta de correo electrónico.

Modelo de Datos

MODELO DE DATOS

MODELO CONCEPTUAL

A – Gráfico del Modelo Conceptual



Ver Gráfico ampliado más adelante

B – Lista de Entidades

- ACTAS_FINALES
- AREAS
- CAL_FINALES
- CAL_TRABAJOS
- CARGOS
- CARRERAS
- CLIENTES
- COM_EVALUADORA
- CONCURSANTES
- CONCURSOS
- CUOTAS_PAGAS
- CURSO_ALUMNO
- DATOS_CLIENTES
- GRADUADOS
- GRUPOS_TRABAJO
- MATERIA_ALUMNO
- MATERIA_PROFESOR
- MATERIAS

- NOTAS
- SOLIC_ANALITICO
- TIPOS_CLIENTE
- TIPOS_DOC
- TIPOS_TRABAJO
- TRAB_ENTREGADOS
- TRABAJOS
- TURNO_MATERIAS

C – Lista de Atributos de las Entidades

- **ACTAS_FINALES**

Nombre	Tipo
acta_final	VARCHAR2(50)
id_materia	NUMBER(10)
id_cliente	NUMBER(10)
id_profesor	NUMBER(10)
fecha	DATE
presente	VARCHAR2(1)
nota	NUMBER(10)

- **AREAS**

Nombre	Tipo
id_area	NUMBER(10)
nom_area	VARCHAR2(40)

- **CAL_FINALES**

Nombre	Tipo
id_cal_final	NUMBER(10)
id_materia	NUMBER(10)
horario	VARCHAR2(100)
fecha	DATE
lugar	VARCHAR2(100)
insc_abierta	VARCHAR2(1)

- **CAL_MATERIAS**

Nombre	Tipo
id_materia	NUMBER(10)
id_turno	NUMBER(10)

- **CAL_TRABAJOS**

Nombre	Tipo
id_materia	NUMBER(10)
id_trabajo	NUMBER(10)
fec_expiracion	DATE
hora	VARCHAR2(5)

- **CARGOS**

Nombre	Tipo
id_cargo	NUMBER(10)
nom_cargo	VARCHAR2(40)

- **CARRERAS**

Nombre	Tipo
id_carrera	NUMBER(10)
id_area	NUMBER(10)
cod_carrera	VARCHAR2(20)
nom_carrera	VARCHAR2(80)
valor	NUMBER(15,2)
anio_plan	NUMBER(4)
incumbencias	VARCHAR2(200)
titulo	VARCHAR2(40)
requisitos	VARCHAR2(200)
duracion	VARCHAR2(20)
otros_datos	VARCHAR2(2000)
vigente	VARCHAR2(1)

- **CLIENTES**

Nombre	Tipo
id_cliente	NUMBER(10)
usuario	VARCHAR2(50)
cl_cliente	VARCHAR2(8)
exp_clave	DATE

- **CLIENTES_POR_TIPO**

Nombre	Tipo
id_cliente	NUMBER(10)
id_tipo_cliente	NUMBER(10)

- **COM_EVALUADORA**

Nombre	Tipo
id_comision	NUMBER(10)
id_jurado1	NUMBER(10)
id_jurado2	NUMBER(10)
id_jurado3	NUMBER(10)

- **CONCURSANTES**

Nombre	Tipo
id_concurso	NUMBER(10)
id_cliente	NUMBER(10)
ganador	VARCHAR2(1)

- **CONCURSOS**

Nombre	Tipo
id_concurso	NUMBER(10)
id_materia	NUMBER(10)
id_cargo	NUMBER(10)
nom_concurso	VARCHAR2(100)
cod_concurso	VARCHAR2(20)
fec_apertura	DATE
fec_cierre	DATE
cant_cargos	NUMBER(2)
requisitos	VARCHAR2(2000)
insc_abierta	VARCHAR2(1)

- **CORRELATIVAS**

Nombre	Tipo
id_materia	NUMBER(10)
id_correlativa	NUMBER(10)

- **CUOTAS_PAGAS**

Nombre	Tipo
id_carrera	NUMBER(10)
id_cliente	NUMBER(10)
num_cuota	NUMBER(2)
valor	NUMBER(15,2)
fec_pago	DATE
curso_pago	VARCHAR2(1)

- **CURSO_ALUMNO**

Nombre	Tipo
id_cliente	NUMBER(10)
id_carrera	NUMBER(10)
legajo	VARCHAR2(8)

- **DATOS_CLIENTES**

Nombre	Tipo
id_cliente	NUMBER(10)
id_tipo_doc	VARCHAR2(10)
nro_doc	Numeric(11)
nombre	VARCHAR2(40)
apellido	VARCHAR2(40)
domicilio	VARCHAR2(100)
localidad	VARCHAR2(40)
partido	VARCHAR2(40)
provincia	VARCHAR2(40)
pais	VARCHAR2(40)
cod_pos	VARCHAR2(10)
tel	VARCHAR2(40)
fax	VARCHAR2(40)
e_mail	VARCHAR(100)

- **GRADUADOS**

Nombre	Tipo
id_carrera	NUMBER(10)
id_cliente	NUMBER(10)
fec_graduacion	DATE
id_com_evaluadora	NUMBER(10)

- **GRUPOS_TRABAJO**

Nombre	Tipo
id_grupo_trabajo	NUMBER(10)
id_cliente	NUMBER(10)
id_trabajo	NUMBER(10)
cod_grupo_trabajo	VARCHAR2(10)

- **INSC_FINALES**

Nombre	Tipo
id_cliente	NUMBER(10)
id_cal_final	NUMBER(10)

- **MATERIA_ALUMNO**

Nombre	Tipo
id_cliente	NUMBER(10)
id_materia	NUMBER(10)
id_turno	NUMBER(10)
anio_cursada	NUMBER(4)
aprob_cursada	VARCHAR2(1)
aprob_final	VARCHAR2(1)
cursando	VARCHAR2(1)
cursada_vencida	VARCHAR2(1)

- **MATERIA_PROFESOR**

Nombre	Tipo
id_cliente	NUMBER(10)
id_materia	NUMBER(10)
id_cargo	NUMBER(10)
fec_inicio	DATE
fec_fin	DATE

- **MATERIAS**

Nombre	Tipo
id_carrera	NUMBER(10)
id_materia	NUMBER(10)
cod_materia	VARCHAR2(20)
nom_materia	VARCHAR2(80)
programa	VARCHAR2(2000)
duracion	VARCHAR2(20)
bibliografia	VARCHAR2(1000)
requisitos	VARCHAR2(200)
cupo	NUMBER(6)

- **NOTAS**

Nombre	Tipo
id_cliente	NUMBER(10)
id_materia	NUMBER(10)
id_trabajo	NUMBER(10)
id_tipo_trabajo	NUMBER(10)
fecha	DATE
nota	NUMBER(10,2)

- **SOLIC_ANALITICO**

Nombre	Tipo
id_carrera	NUMBER(10)
id_cliente	NUMBER(10)
fec_solicitud	DATE

- **TIPOS_CLIENTE**

Nombre	Tipo
id_tipo_cliente	NUMBER(10)
nom_cliente	VARCHAR2(40)
cod_tipo_cli	VARCHAR2(20)

- **TIPOS_DOC**

Nombre	Tipo
id_tipo_doc	VARCHAR2(10)
nom_tipo_doc	VARCHAR2(20)

- **TIPOS_TRABAJO**

Nombre	Tipo
id_tipo_trabajo	NUMBER(10)
cod_tipo_trabajo	VARCHAR2(20)
nom_tipo_trabajo	VARCHAR2(40)

- **TRAB_ENTREGADOS**

Nombre	Tipo
id_materia	NUMBER(10)
id_cliente	NUMBER(10)
id_trabajo	NUMBER(10)
calif	NUMBER(15,2)
Aprobado	VARCHAR2(1)
fecha_entrega	DATE
observaciones	VARCHAR2(2000)

- **TRABAJOS**

Nombre	Tipo
id_materia	NUMBER(10)
id_trabajo	NUMBER(10)
id_tipo_trabajo	NUMBER(10)
cod_trabajo	VARCHAR2(20)
titulo	VARCHAR2(100)
tema	VARCHAR2(2000)
especificaciones	VARCHAR2(2000)

- **TURNO_MATERIAS**

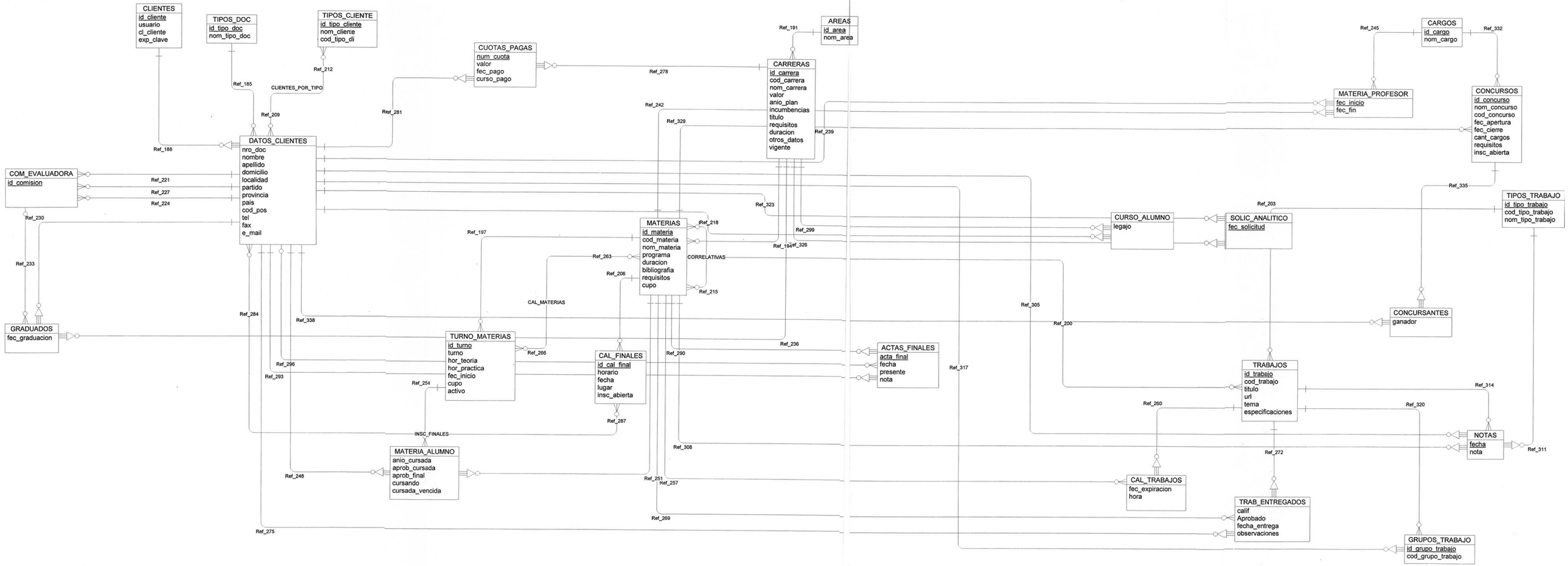
Nombre	Tipo
id_materia	NUMBER(10)
id_turno	NUMBER(10)
turno	VARCHAR2(2)
hor_teoria	VARCHAR2(100)
hor_practica	VARCHAR2(100)
fec_inicio	DATE
cupo	NUMBER(6)
activo	VARCHAR2(1)

D - Referencias de las Relaciones

Nombre Relación	Entidad 1	Entidad 2	Cd.
CAL_MATERIAS	MATERIAS (id_materia)	TURNOS_ MATERIAS (id_materia)	N:M
CLIENTES_POR_TIPO	DATOS_CLIENES (id_cliente)	TIPOS_CLIENTES(id_cliente)	N:M
CORRELATIVAS	MATERIAS (id_materia)	MATERIAS (id_materia)	N:M
INSC_FINALES	DATOS_CLIENTES(id_cliente)	CAL_FINALES(id_cliente)	N:M
Ref_185	TIPOS_DOC(id_tipo_doc)	DATOS_CLIENTES(id_tipo_doc)	1:N
Ref_188	CLIENTES(id_cliente)	DATOS_CLIENTES(id_cliente)	1:N
Ref_191	AREAS(id_area)	CARRERAS(id_area)	1:N
Ref_194	CARRERAS(id_carrera)	MATERIAS(id_carrera)	1:N
Ref_197	MATERIAS(id_materia)	TURNO_MATERIAS (id_materia)	1:N
Ref_200	MATERIAS(id_materia)	TRABAJOS(id_materia)	1:N
Ref_203	TIPOS_TRABAJO(id_tipo_trabajo)	TRABAJOS(id_tipo_trabajo)	1:N
Ref_206	MATERIAS (id_materia)	CAL_FINALES (id_materia)	1:N
Ref_221	DATOS_CLIENTES(id_cliente)	COM_EVALUADORA(id_jurado1)	1:N
Ref_224	DATOS_CLIENTES(id_cliente)	COM_EVALUADORA(id_jurado2)	1:N
Ref_227	DATOS_CLIENTES(id_cliente)	COM_EVALUADORA(id_jurado3)	1:N
Ref_230	DATOS_CLIENTES(id_cliente)	GRADUADOS(id_cliente)	1:N
Ref_233	COM_EVALUADORA(id_comisión)	GRADUADOS(id_com_evaluadora)	1:N
Ref_236	CARRERAS(id_carrera)	GRADUADOS(id_carrera)	1:N
Ref_239	DATOS_CLIENTES(id_cliente)	MATERIA_PROFESOR(id_cliente)	1:N
Ref_242	MATERIAS (id_materia)	MATERIA_PROFESOR (id_materia)	1:N
Ref_245	CARGOS(id_cargo)	MATERIA_PROFESOR(id_cargo)	1:N
Ref_248	DATOS_CLIENTES(id_cliente)	MATERIA_ALUMNO(id_cliente)	1:N
Ref_251	MATERIAS (id_materia)	MATERIA_ALUMNO (id_materia)	1:N
Ref_254	TURNO_MATERIAS(id_turno)	MATERIA_ALUMNO(id_turno)	1:N
Ref_257	MATERIAS (id_materia)	CAL_TRABAJOS (id_materia)	1:N
Ref_260	TRABAJOS(id_trabajo)	CAL_TRABAJOS(id_trabajo)	1:N
Ref_269	MATERIAS (id_materia)	TRAB_ENTREGADOS (id_materia)	1:N
Ref_272	TRABAJOS(id_trabajo)	TRAB_ENTREGADOS(id_trabajo)	1:N
Ref_275	DATOS_CLIENTES(id_cliente)	TRAB_ENTREGADOS(id_cliente)	1:N
Ref_278	CARRERAS(id_carrera)	CUOTAS_PAGAS(id_carrera)	1:N
Ref_281	DATOS_CLIENTES(id_cliente)	CUOTAS_PAGAS(id_cliente)	1:N
Ref_290	MATERIAS (id_materia)	ACTAS_FINALES(id_materia)	1:N
Ref_293	DATOS_CLIENTES(id_cliente)	ACTAS_FINALES(id_cliente)	1:N

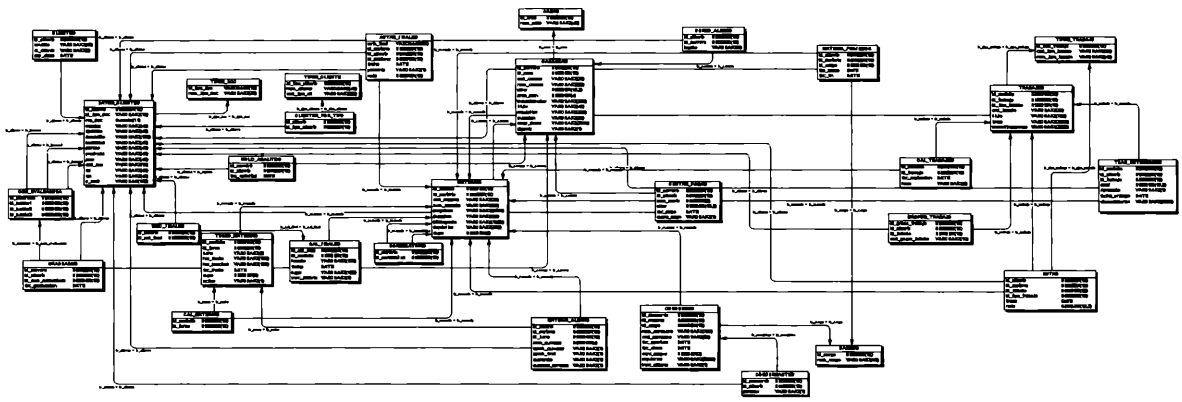
Nombre Relación	Entidad 1	Entidad 2	Cd.
Ref_296	DATOS_CLIENTES(id_cliente)	ACTAS_FINALES(id_profesor)	1:N
Ref_299	CARRERAS(id_carrera)	CURSO_ALUMNO(id_carrera)	1:N
Ref_302	DATOS_CLIENTES(id_cliente)	CURSO_ALUMNO(id_cliente)	1:N
Ref_305	DATOS_CLIENTES(id_cliente)	NOTAS(id_cliente)	1:N
Ref_308	MATERIAS (id_materia)	NOTAS (id_materia)	1:N
Ref_311	TIPOS_TRABAJO(id_tipo_trabajo)	NOTAS(id_tipo_trabajo)	1:N
Ref_314	TRABAJOS(id_trabajo)	NOTAS(id_trabajo)	1:N
Ref_317	DATOS_CLIENTES(id_cliente)	GRUPOS_TRABAJOS(id_cliente)	1:N
Ref_320	TRABAJOS(id_trabajo)	GRUPOS_TRABAJOS(id_trabajo)	1:N
Ref_323	DATOS_CLIENTES(id_cliente)	SOLIC_ANALITICO(id_cliente)	1:N
Ref_326	CARRERAS(id_carrera)	SOLIC_ANALITICO(id_carrera)	1:N
Ref_329	MATERIAS (id_materia)	CONCURSOS (id_materia)	1:N
Ref_332	CARGOS(id_cargo)	CONCURSOS(id_cargo)	1:N
Ref_335	CONCURSOS(id_concurso)	CONCURSANTES(id_concurso)	1:N
Ref_338	DATOS_CLIENTES(id_cliente)	CONCURSANTES(id_cliente)	1:N

Nota: Entre paréntesis se encuentra el atributo que interviene en la relación.


CLIENTES
id_cliente
usuario
cl_cliente
exp_clave
TIPOS_DOC
id_tipo_doc
nom_tipo_doc
TIPOS_CLIENTE
id_tipo_cliente
nom_cliente
cod_tipo_cli
CUOTAS_PAGAS
num_cuota
valor
fec_pago
curso_pago
AREAS
id_area
nom_area
CARRERAS
id_carrera
cod_carrera
nom_carrera
valor
anio_plan
incumbencias
titulo
requisitos
duracion
otros_datos
vigente
CARGOS
id_cargo
nom_cargo
MATERIA_PROFESOR
fec_inicio
fec_fin
CONCURSOS
id_concurso
nom_concurso
cod_concurso
fec_apertura
fec_cierre
cant_cargos
requisitos
insc_abierta
TIPOS_TRABAJO
id_tipo_trabajo
cod_tipo_trabajo
nom_tipo_trabajo
DATOS_CLIENTES
nro_doc
nombre
apellido
domicilio
localidad
partido
provincia
pais
cod_pos
tel
fax
e_mail
COM_EVALUADORA
id_comision
MATERIAS
id_materia
cod_materia
nom_materia
programa
duracion
bibliografia
requisitos
cupo
CURSO_ALUMNO
legajo
SOLIC_ANALITICO
fec_solicitud
GRADUADOS
fec_graduacion
TURNO_MATERIAS
id_turno
turno
hor_teoria
hor_practica
fec_inicio
cupo
activo
CAL_FINALES
id_cal_final
horario
fecha
lugar
insc_abierta
ACTAS_FINALES
acta_final
fecha
presente
nota
CONCURSANTES
ganador
TRABAJOS
id_trabajo
cod_trabajo
titulo
url
tema
especificaciones
NOTAS
fecha
nota
MATERIA_ALUMNO
anio_cursada
aprob_cursada
aprob_final
cursando
cursada_vencida
CAL_TRABAJOS
fec_expiracion
hora
TRAB_ENTREGADOS
calif
Aprobado
fecha_entrega
observaciones
GRUPOS_TRABAJO
id_grupo_trabajo
cod_grupo_trabajo
CLIENTES_POR_TIPO
CORRELATIVAS
CAL_MATERIAS
INSC_FINALES
Ref_185
Ref_188
Ref_191
Ref_197
Ref_200
Ref_203
Ref_206
Ref_209
Ref_212
Ref_218
Ref_221
Ref_224
Ref_227
Ref_230
Ref_233
Ref_236
Ref_239
Ref_242
Ref_245
Ref_248
Ref_251
Ref_254
Ref_257
Ref_260
Ref_263
Ref_266
Ref_269
Ref_272
Ref_275
Ref_278
Ref_281
Ref_284
Ref_287
Ref_290
Ref_293
Ref_296
Ref_299
Ref_305
Ref_308
Ref_311
Ref_314
Ref_317
Ref_320
Ref_323
Ref_326
Ref_329
Ref_332
Ref_335
Ref_338

MODELO FÍSICO

A - Gráfico del Modelo Físico



Ver gráfico ampliado más adelante

B – Tablas de la Base de Datos

- ACTAS_FINALES
- AREAS
- CAL_FINALES
- CAL_MATERIAS
- CAL_TRABAJOS
- CARGOS
- CARRERAS
- CLIENTES
- CLIENTES_POR_TIPO
- COM_EVALUADORA
- CONCURSANTES
- CONCURSOS
- CORRELATIVAS
- CUOTAS_PAGAS
- CURSO_ALUMNO
- DATOS_CLIENTES
- GRADUADOS
- GRUPOS_TRABAJO

- INSC_FINALES
- MATERIA_ALUMNO
- MATERIA_PROFESOR
- MATERIAS
- NOTAS
- SOLIC_ANALITICO
- TIPOS_CLIENTE
- TIPOS_DOC
- TIPOS_TRABAJO
- TRAB_ENTREGADOS
- TRABAJOS
- TURNO_MATERIAS

C - Script para crear la Base de Datos

```
==========================================================
Nombre de la Base de Datos:  ORCL
Nombre del DBMS:             ORACLE Version 7.x
==========================================================
```

BEGIN SCRIPT

```
==========================================================
 Tabla: CLIENTES
==========================================================
 create table CLIENTES
 (
   id_cliente      NUMBER(10)          not null,
   usuario         VARCHAR2(50)        not null,
   cl_cliente      VARCHAR2(8)         not null,
   exp_clave       DATE                null   ,
   constraint clientes_PK_cliente primary key (id_cliente)
 )
 /
==========================================================
 Tabla: TIPOS_DOC
==========================================================
 create table TIPOS_DOC
 (
   id_tipo_doc     VARCHAR2(10)        not null,
   nom_tipo_doc    VARCHAR2(20)        not null,
   constraint tipos_doc_PK_tipo_doc primary key (id_tipo_doc)
 )
 /
==========================================================
 Tabla: TIPOS_CLIENTE
==========================================================
 create table TIPOS_CLIENTE
 (
   id_tipo_cliente  NUMBER(10)         not null,
   nom_cliente      VARCHAR2(40)       not null,
   cod_tipo_cli     VARCHAR2(20)       not null,
   constraint tipo_cliente_PK_tipo_cliente primary key (id_tipo_cliente)
 )
 /
==========================================================
 Tabla: AREAS
==========================================================
 create table AREAS
 (
   id_area         NUMBER(10)          not null,
```

```
    nom_area        VARCHAR2(40)        not null,
    constraint areas_PK_area primary key (id_area)
)
/
```

Tabla: CARGOS

```
create table CARGOS
(
    id_cargo        NUMBER(10)          not null,
    nom_cargo       VARCHAR2(40)        not null,
    constraint cargos_PK_cargo primary key (id_cargo)
)
/
```

Tabla: TIPOS_TRABAJO

```
create table TIPOS_TRABAJO
(
    id_tipo_trabajo     NUMBER(10)          not null,
    cod_tipo_trabajo    VARCHAR2(20)        not null,
    nom_tipo_trabajo    VARCHAR2(40)        not null,
    constraint tipos_trabajo_PK_tipo_trabajo primary key (id_tipo_trabajo)
)
/
```

Tabla: DATOS_CLIENTES

```
create table DATOS_CLIENTES
(
    id_cliente      NUMBER(10)          not null,
    id_tipo_doc     VARCHAR2(10)        not null,
    nro_doc         NUMBER(11)          not null,
    nombre          VARCHAR2(40)        not null,
    apellido        VARCHAR2(40)        not null,
    domicilio       VARCHAR2(100)       null    ,
    localidad       VARCHAR2(40)        null    ,
    partido         VARCHAR2(40)        null    ,
    provincia       VARCHAR2(40)        null    ,
    pais            VARCHAR2(40)        null    ,
    cod_pos         VARCHAR2(10)        null    ,
    tel             VARCHAR2(40)        null    ,
    fax             VARCHAR2(40)        null    ,
    e_mail          VARCHAR(100)        null    ,
    constraint datos_clientes_PK_cliente primary key (id_cliente)
)
/
```

Tabla: CARRERAS

```
create table CARRERAS
(
   id_carrera        NUMBER(10)        not null,
   id_area           NUMBER(10)        not null,
   cod_carrera       VARCHAR2(20)      not null,
   nom_carrera       VARCHAR2(80)      not null,
   valor             NUMBER(15,2)      null   ,
   anio_plan         NUMBER(4)         not null,
   incumbencias      VARCHAR2(200)     null   ,
   titulo            VARCHAR2(40)      not null,
   requisitos        VARCHAR2(200)     null   ,
   duracion          VARCHAR2(20)      null   ,
   otros_datos       VARCHAR2(2000)    null   ,
   vigente           VARCHAR2(1)       not null,
   constraint carreras_PK_carrera primary key (id_carrera),
   constraint CK_car_vigente check (vigente IN ('T','F'))
)
/
```

Tabla: MATERIAS

```
create table MATERIAS
(
   id_carrera        NUMBER(10)        not null,
   id_materia        NUMBER(10)        not null,
   cod_materia       VARCHAR2(20)      not null,
   nom_materia       VARCHAR2(80)      not null,
   programa          VARCHAR2(2000)    null   ,
   duracion          VARCHAR2(20)      null   ,
   bibliografia      VARCHAR2(1000)    null   ,
   requisitos        VARCHAR2(200)     null   ,
   cupo              NUMBER(6)         null   ,
   constraint materias_PK_materia primary key (id_materia)
)
/
```

Tabla: TURNO_MATERIAS

```
create table TURNO_MATERIAS
(
   id_materia        NUMBER(10)        not null,
   id_turno          NUMBER(10)        not null,
   turno             VARCHAR2(2)       not null,
   hor_teoria        VARCHAR2(100)     not null,
```

```
    hor_practica      VARCHAR2(100)    not null,
    fec_inicio        DATE             null  ,
    cupo              NUMBER(6)        null  ,
    activo            VARCHAR2(1)      not null,
    constraint turno_materias_PK_maturn primary key (id_turno),
    constraint CK_activo check (activo IN ('T','F'))
)
/
```

Tabla: TRABAJOS

```
create table TRABAJOS
(
    id_materia        NUMBER(10)       not null,
    id_trabajo        NUMBER(10)       not null,
    id_tipo_trabajo   NUMBER(10)       not null,
    cod_trabajo       VARCHAR2(20)     not null,
    titulo            VARCHAR2(100)    null  ,
    tema              VARCHAR2(2000)   null  ,
    especificaciones  VARCHAR2(2000)   null  ,
    constraint trabajos_PK_trabajo primary key (id_trabajo)
)
/
```

Tabla: CAL_FINALES

```
create table CAL_FINALES
(
    id_cal_final      NUMBER(10)       not null,
    id_materia        NUMBER(10)       not null,
    horario           VARCHAR2(100)    null  ,
    fecha             DATE             not null,
    lugar             VARCHAR2(100)    null  ,
    insc_abierta      VARCHAR2(1)      not null,
    constraint cal_finales_PK_mat_feci primary key (id_cal_final),
    constraint CK_cal_finales_abierta check (insc_abierta IN ('T','F'))
)
/
```

Tabla: CLIENTES_POR_TIPO

```
create table CLIENTES_POR_TIPO
(
    id_cliente        NUMBER(10)       not null,
    id_tipo_cliente   NUMBER(10)       not null,
    constraint clie_por_tipo_PK_cliente_tipo primary key (id_cliente, id_tipo_cliente)
)
```

```
/
==========================================================
 Tabla: CORRELATIVAS
==========================================================
 create table CORRELATIVAS
 (
   id_materia      NUMBER(10)      not null,
   id_correlativa  NUMBER(10)      not null,
   constraint correlativas_PK_mat_cor primary key (id_materia, id_correlativa)
 )
 /
==========================================================
 Tabla: COM_EVALUADORA
==========================================================
 create table COM_EVALUADORA
 (
   id_comision     NUMBER(10)      not null,
   id_jurado1      NUMBER(10)      not null,
   id_jurado2      NUMBER(10)      not null,
   id_jurado3      NUMBER(10)      not null,
   constraint com_evaluadora_PK_com primary key (id_comision)
 )
 /
==========================================================
 Tabla: GRADUADOS
==========================================================
 create table GRADUADOS
 (
   id_carrera          NUMBER(10)      not null,
   id_cliente          NUMBER(10)      not null,
   fec_graduacion      DATE            not null,
   id_com_evaluadora   NUMBER(10)      null,
   constraint graduados_PK_car_cli primary key (id_carrera, id_cliente)
 )
 /
==========================================================
 Tabla: MATERIA_PROFESOR
==========================================================
 create table MATERIA_PROFESOR
 (
   id_cliente      NUMBER(10)      not null,
   id_materia      NUMBER(10)      not null,
   id_cargo        NUMBER(10)      not null,
   fec_inicio      DATE            not null,
   fec_fin         DATE            null   ,
   constraint mat_prof_PK_cli_mat_feci primary key (id_cliente, id_materia, fec_inicio)
 )
```

```
/
==========================================================
Tabla: MATERIA_ALUMNO
==========================================================
create table MATERIA_ALUMNO
(
   id_cliente        NUMBER(10)        not null,
   id_materia        NUMBER(10)        not null,
   id_turno          NUMBER(10)        not null,
   anio_cursada      NUMBER(4)         not null,
   aprob_cursada     VARCHAR2(1)       not null,
   aprob_final       VARCHAR2(1)       not null,
   cursando          VARCHAR2(1)       not null,
   cursada_vencidaVARCHAR2(1)          not null,
   constraint materia_alumno_PK_cli_mat primary key (id_cliente, id_materia),
   constraint CK_aprob_cursada check ( aprob_cursada IN ('T','F')),
   constraint CK_cur_ven check (cursada_vencida IN ('T','F')),
   constraint CK_cursando check (cursando IN ('T','F')),
   constraint CK_aprob_final check (aprob_final  IN ('T','F'))
)
/
==========================================================
Tabla: CAL_TRABAJOS
==========================================================
create table CAL_TRABAJOS
(
   id_materia        NUMBER(10)        not null,
   id_trabajo        NUMBER(10)        not null,
   fec_expiracion    DATE              null   ,
   hora              VARCHAR2(5)       null   ,
   constraint cal_trabajo_PK_trabajo primary key (id_trabajo)
)
/
==========================================================
Tabla: CAL_MATERIAS
==========================================================
create table CAL_MATERIAS
(
   id_materia        NUMBER(10)        not null,
   id_turno          NUMBER(10)        not null,
   constraint cal_materias_PK_mat_feci primary key (id_materia, id_turno)
)
/
==========================================================
Tabla: TRAB_ENTREGADOS
==========================================================
create table TRAB_ENTREGADOS
```

```
(
   id_materia      NUMBER(10)      not null,
   id_cliente      NUMBER(10)      not null,
   id_trabajo      NUMBER(10)      not null,
   calif           NUMBER(15,2)    null   ,
   Aprobado        VARCHAR2(1)     null   ,
   fecha_entrega   DATE            not null,
   observaciones   VARCHAR2(2000)  null   ,
   constraint trab_entregados_PK_cli_trab primary key (id_cliente, id_trabajo),
   constraint CK_aprobado check ( aprobado IN ('T','F'))
)
/
```

Tabla: CUOTAS_PAGAS

```
create table CUOTAS_PAGAS
(
   id_carrera      NUMBER(10)      not null,
   id_cliente      NUMBER(10)      not null,
   num_cuota       NUMBER(2)       not null,
   valor           NUMBER(15,2)    not null,
   fec_pago        DATE            not null,
   curso_pago      VARCHAR2(1)     not null,
   constraint cuotas_pagas_PK_car_cli_cuo primary key (id_carrera, id_cliente,
num_cuota ),
   constraint CK_curso_pago check (curso_pago IN ('T','F'))
)
/
```

Tabla: INSC_FINALES

```
create table INSC_FINALES
(
   id_cliente      NUMBER(10)      not null,
   id_cal_final    NUMBER(10)      not null,
   constraint insc_finales_PK_mat_feci primary key (id_cliente, id_cal_final )
)
/
```

Tabla: ACTAS_FINALES

```
create table ACTAS_FINALES
(
   acta_final      VARCHAR2(50)    not null,
   id_materia      NUMBER(10)      not null,
   id_cliente      NUMBER(10)      not null,
   id_profesor     NUMBER(10)      null   ,
```

```
   fecha          DATE           not null,
   presente       VARCHAR2(1)    null   ,
   nota           NUMBER(10)     null   ,
   constraint actas_finales_PK_acta_mat_cli primary key (acta_final, id_materia,
id_cliente),
   constraint CK_presente check ( presente IN ('T','F'))
)
/
```

Tabla: CURSO_ALUMNO

```
create table CURSO_ALUMNO
(
   id_cliente     NUMBER(10)     not null,
   id_carrera     NUMBER(10)     not null,
   legajo         VARCHAR2(8)    null   ,
   constraint carr_alumno_PK_cli_car primary key (id_cliente, id_carrera)
)
/
```

Tabla: NOTAS

```
create table NOTAS
(
   id_cliente     NUMBER(10)     not null,
   id_materia     NUMBER(10)     not null,
   id_trabajo     NUMBER(10)     not null,
   id_tipo_trabajo NUMBER(10)    not null,
   fecha          DATE           not null,
   nota           NUMBER(10,2)   not null,
   constraint notas_PK_climattp primary key (id_cliente, id_materia, id_tipo_trabajo,
fecha)
)
/
```

Tabla: GRUPOS_TRABAJO

```
create table GRUPOS_TRABAJO
(
   id_grupo_trabajo     NUMBER(10)     not null,
   id_cliente           NUMBER(10)     not null,
   id_trabajo           NUMBER(10)     not null,
   cod_grupo_trabajo    VARCHAR2(10)   not null,
   constraint grupos_trab_PK_grupos_trab primary key (id_grupo_trabajo, id_cliente)
)
/
```

Tabla: SOLIC_ANALITICO

```
create table SOLIC_ANALITICO
(
   id_carrera      NUMBER(10)      not null,
   id_cliente      NUMBER(10)      not null,
   fec_solicitud   DATE not null,
   constraint sol_ana_PK_car_cli primary key (id_carrera, id_cliente, fec_solicitud)
)
/
```

Tabla: CONCURSOS

```
create table CONCURSOS
(
   id_concurso     NUMBER(10)      not null,
   id_materia      NUMBER(10)      not null,
   id_cargo        NUMBER(10)      not null,
   nom_concurso    VARCHAR2(100)   not null,
   cod_concurso    VARCHAR2(20)    null   ,
   fec_apertura    DATE            null   ,
   fec_cierre      DATE            null   ,
   cant_cargos     NUMBER(2)       not null,
   requisitos      VARCHAR2(2000)  not null,
   insc_abierta    VARCHAR2(1)     not null,
   constraint conc_PK_concurso primary key (id_concurso),
   constraint CK_conc_vigente check (insc_abierta IN ('T','F'))
)
/
```

Tabla: CONCURSANTES

```
create table CONCURSANTES
(
   id_concurso     NUMBER(10)      not null,
   id_cliente      NUMBER(10)      not null,
   ganador         VARCHAR2(1)     not null,
   constraint conc_PK_conccli primary key (id_concurso, id_cliente),
   constraint CK_conc_gan check ( ganador IN ('T','F'))
)
/
```

Trigger: TR_SOLIC_ANALITICO

```
create or replace trigger  TR_SOLIC_ANALITICO
before insert
on SOLIC_ANALITICO
```

```
for each row
begin
   :NEW.FEC_SOLICITUD:= SYSDATE;
end;
/
```

==========

FOREIGN KEY DE LAS TABLAS CREADAS

==========

```
alter table DATOS_CLIENTES
   add constraint datos_clientes_FK_tipo_doc foreign key  (id_tipo_doc)
     references TIPOS_DOC (id_tipo_doc)
/

alter table CLIENTES
   add constraint clientes_FK_datos_clientes foreign key  (id_cliente)
     references DATOS_CLIENTES (id_cliente)
/

alter table CARRERAS
   add constraint carreras_FK_area foreign key  (id_area)
     references AREAS (id_area)
/

alter table MATERIAS
   add constraint materias_FK_carrera foreign key  (id_carrera)
     references CARRERAS (id_carrera)
/
alter table TURNO_MATERIAS
   add constraint turno_materias_FK_materia foreign key  (id_materia)
     references MATERIAS (id_materia)
/

alter table TRABAJOS
   add constraint trabajos_FK_materia foreign key  (id_materia)
     references MATERIAS (id_materia)
/

alter table TRABAJOS
   add constraint trabajos_FK_tipo_trabajo foreign key  (id_tipo_trabajo)
     references TIPOS_TRABAJO (id_tipo_trabajo)
/

alter table CAL_FINALES
   add constraint cal_finales_FK_materia foreign key  (id_materia)
     references MATERIAS (id_materia)
/
```

```
alter table CLIENTES_POR_TIPO
   add constraint clie_por_tipo_FK_cliente foreign key  (id_cliente)
     references DATOS_CLIENTES (id_cliente)
/

alter table CLIENTES_POR_TIPO
   add constraint clie_por_tipo_FK_tipo_cliente foreign key  (id_tipo_cliente)
     references TIPOS_CLIENTE (id_tipo_cliente)
/

alter table CORRELATIVAS
   add constraint correlativas_FK_correlativa foreign key  (id_correlativa)
     references MATERIAS (id_materia)
/

alter table CORRELATIVAS
   add constraint correlativas_FK_materia foreign key  (id_materia)
     references MATERIAS (id_materia)
/

alter table COM_EVALUADORA
   add constraint com_evaluadora_FK_jur1 foreign key  (id_jurado1)
     references DATOS_CLIENTES (id_cliente)
/

alter table COM_EVALUADORA
   add constraint com_evaluadora_FK_jur2 foreign key  (id_jurado2)
     references DATOS_CLIENTES (id_cliente)
/

alter table COM_EVALUADORA
   add constraint com_evaluadora_FK_jur3 foreign key  (id_jurado3)
     references DATOS_CLIENTES (id_cliente)
/

alter table GRADUADOS
    add constraint graduados_FK_cliente foreign key (id_cliente)
     references DATOS_CLIENTES (id_cliente)
/

alter table GRADUADOS
    add constraint graduados_FK_com_eval foreign key (id_com_evaluadora)
      references COM_EVALUADORA (id_comision)
/

alter table GRADUADOS
```

```
    add constraint graduados_FK_carrera foreign key (id_carrera)
     references CARRERAS (id_carrera)
/

alter table MATERIA_PROFESOR
   add constraint mat_prof_FK_cliente foreign key  (id_cliente)
     references DATOS_CLIENTES (id_cliente)
/

alter table MATERIA_PROFESOR
   add constraint mat_prof_FK_materia foreign key  (id_materia)
     references MATERIAS (id_materia)
/

alter table MATERIA_PROFESOR
   add constraint mat_prof_FK_cargo foreign key  (id_cargo)
     references CARGOS (id_cargo)
/

alter table MATERIA_ALUMNO
   add constraint materia_alumno_FK_cliente foreign key  (id_cliente)
     references DATOS_CLIENTES (id_cliente)
/

alter table MATERIA_ALUMNO
   add constraint materia_alumno_FK_materia foreign key  (id_materia)
     references MATERIAS (id_materia)
/

alter table MATERIA_ALUMNO
   add constraint materia_alumno_FK_turno foreign key  (id_turno)
     references TURNO_MATERIAS (id_turno)
/

alter table CAL_TRABAJOS
   add constraint cal_trabajo_FK_materia foreign key  (id_materia)
     references MATERIAS (id_materia)
/

alter table CAL_TRABAJOS
   add constraint cal_trabajo_FK_trabajo foreign key  (id_trabajo)
     references TRABAJOS (id_trabajo)
/

alter table CAL_MATERIAS
   add constraint cal_materias_FK_materia foreign key  (id_materia)
     references MATERIAS (id_materia)
```

```
/

alter table CAL_MATERIAS
   add constraint cal_materias_FK_turno_mat foreign key  (id_turno)
     references TURNO_MATERIAS (id_turno)
/

alter table TRAB_ENTREGADOS
   add constraint trab_entregados_FK_id_materia foreign key  (id_materia)
     references MATERIAS (id_materia)
/

alter table TRAB_ENTREGADOS
   add constraint trab_entregados_FK_trabajo foreign key  (id_trabajo)
     references TRABAJOS (id_trabajo)
/

alter table TRAB_ENTREGADOS
   add constraint trab_entregados_FK_cliente foreign key  (id_cliente)
     references DATOS_CLIENTES (id_cliente)
/

alter table CUOTAS_PAGAS
   add constraint cuotas_pagas_FK_carrera foreign key  (id_carrera)
     references CARRERAS (id_carrera)
/

alter table CUOTAS_PAGAS
   add constraint cuotas_pagas_FK_cliente foreign key  (id_cliente)
     references DATOS_CLIENTES (id_cliente)
/

alter table INSC_FINALES
   add constraint insc_finales_FK_cliente foreign key  (id_cliente)
     references DATOS_CLIENTES (id_cliente)
/

alter table INSC_FINALES
   add constraint insc_finales_FK_cal_final foreign key  (id_cal_final)
     references CAL_FINALES (id_cal_final)
/

alter table ACTAS_FINALES
   add constraint actas_finales_FK_materia foreign key  (id_materia)
     references MATERIAS (id_materia)
/
```

```
alter table ACTAS_FINALES
   add constraint actas_finales_FK_cliente foreign key  (id_cliente)
     references DATOS_CLIENTES (id_cliente)
/

alter table ACTAS_FINALES
   add constraint actas_finales_FK_profesor foreign key  (id_profesor)
     references DATOS_CLIENTES (id_cliente)
/

alter table CURSO_ALUMNO
   add constraint carr_alumno_FK_curso foreign key  (id_carrera)
     references CARRERAS (id_carrera)
/

alter table CURSO_ALUMNO
   add constraint carr_alumno_FK_alumno foreign key (id_cliente)
     references DATOS_CLIENTES (id_cliente)
/

alter table NOTAS
   add constraint notas_FK_alumno foreign key  (id_cliente)
     references DATOS_CLIENTES (id_cliente)
/

alter table NOTAS
   add constraint notas_FK_materia foreign key  (id_materia)
     references MATERIAS (id_materia)
/

alter table NOTAS
   add constraint notas_FK_tp foreign key  (id_tipo_trabajo)
     references TIPOS_TRABAJO (id_tipo_trabajo)
/

alter table NOTAS
   add constraint notas_FK_trab foreign key  (id_trabajo)
     references TRABAJOS (id_trabajo)
/

alter table GRUPOS_TRABAJO
   add constraint grupos_trab_FK_alumno foreign key  (id_cliente)
     references DATOS_CLIENTES (id_cliente)
/

alter table GRUPOS_TRABAJO
   add constraint grupos_trab_FK_trab foreign key  (id_trabajo)
```

```
      references TRABAJOS (id_trabajo)
/

alter table SOLIC_ANALITICO
   add constraint sol_ana_FK_cliente foreign key  (id_cliente)
     references DATOS_CLIENTES (id_cliente)
/

alter table SOLIC_ANALITICO
   add constraint sol_ana_FK_carrera foreign key  (id_carrera)
     references CARRERAS (id_carrera)
/

alter table CONCURSOS
   add constraint concursos_FK_materia foreign key  (id_materia)
     references MATERIAS (id_materia)
/

alter table CONCURSOS
   add constraint concursos_FK_cargos foreign key  (id_cargo)
     references CARGOS (id_cargo)
/

alter table CONCURSANTES
   add constraint concursantes_FK_concursos foreign key  (id_concurso)
     references CONCURSOS (id_concurso)
/

alter table CONCURSANTES
   add constraint concursantes_FK_cliente foreign key  (id_cliente)
     references DATOS_CLIENTES (id_cliente)
/
END SCRIPT
```

D – Formato de las Tablas y Atributos que intervienen en las Relaciones

- **ACTAS_FINALES**

Nombre	Tipo	PK	Nulo
acta_final	VARCHAR2(50)	Si	Si
id_materia	NUMBER(10)	Si	Si
id_cliente	NUMBER(10)	Si	Si
id_profesor	NUMBER(10)	No	No
fecha	DATE	No	Si
presente	VARCHAR2(1)	No	No
nota	NUMBER(10)	No	No

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_materia	MATERIAS	id_materia
id_cliente	DATOS_CLIENTES	id_cliente
id_cliente	DATOS_CLIENTES	id_profesor

- **AREAS**

Nombre	Tipo	PK	Nulo
id_area	NUMBER(10)	Si	Si
nom_area	VARCHAR2(40)	No	Si

Referencias

Referenciado por	Clave Primaria	Clave Foránea
CARRERAS	id_area	id_area

- **CAL_FINALES**

Nombre	Tipo	PK	Nulo
id_cal_final	NUMBER(10)	Si	Si
id_materia	NUMBER(10)	No	Si
horario	VARCHAR2(100)	No	No
fecha	DATE	No	Si
lugar	VARCHAR2(100)	No	No
insc_abierta	VARCHAR2(1)	No	Si

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_materia	MATERIAS	id_materia

Referenciado por	Clave Primaria	Clave Foránea
INSC_FINALES	id_cal_final	id_cal_final

- **CAL_MATERIAS**

Nombre	Tipo	PK	Nulo
id_materia	NUMBER(10)	Si	Si
id_turno	NUMBER(10)	Si	Si

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_materia	MATERIAS	id_materia
id_turno	TURNO_MATERIAS	id_turno

- **CAL_TRABAJOS**

Nombre	Tipo	PK	Nulo
id_materia	NUMBER(10)	No	Si
id_trabajo	NUMBER(10)	Si	Si
fec_expiracion	DATE	No	No
hora	VARCHAR2(5)	No	No

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_materia	MATERIAS	id_materia
id_trabajo	TRABAJOS	id_trabajo

- **CARGOS**

Nombre	Tipo	PK	Nulo
id_cargo	NUMBER(10)	Si	Si
nom_cargo	VARCHAR2(40)	No	Si

Referencias

Referenciado por	Clave Primaria	Clave Foránea
MATERIA_PROFESOR	id_cargo	id_cargo
CONCURSOS	id_cargo	id_cargo

- **CARRERAS**

Nombre	Tipo	PK	Nulo
id_carrera	NUMBER(10)	Si	Si
id_area	NUMBER(10)	No	Si
cod_carrera	VARCHAR2(20)	No	Si
nom_carrera	VARCHAR2(80)	No	Si
valor	NUMBER(15,2)	No	No
anio_plan	NUMBER(4)	No	Si
incumbencias	VARCHAR2(200)	No	No
titulo	VARCHAR2(40)	No	Si
requisitos	VARCHAR2(200)	No	No
duracion	VARCHAR2(20)	No	No
otros_datos	VARCHAR2(2000)	No	No
vigente	VARCHAR2(1)	No	Si

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_area	AREAS	id_area

Referenciado por	Clave Primaria	Clave Foránea
MATERIAS	id_carrera	id_carrera
GRADUADOS	id_carrera	id_carrera
CUOTAS_PAGAS	id_carrera	id_carrera
CURSO_ALUMNO	id_carrera	id_carrera
SOLIC_ANALITICO	id_carrera	id_carrera

- **CLIENTES**

Nombre	Tipo	PK	Nulo
id_cliente	NUMBER(10)	Si	Si
usuario	VARCHAR2(50)	No	Si
cl_cliente	VARCHAR2(8)	No	Si
exp_clave	DATE	No	No

Referencias

Referenciado por	Clave Primaria	Clave Foránea
DATOS_CLIENTES	id_cliente	id_cliente

- **CLIENTES_POR_TIPO**

Nombre	Tipo	PK	Nulo
id_cliente	NUMBER(10)	Si	Si
id_tipo_cliente	NUMBER(10)	Si	Si

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_cliente	DATOS_CLIENTES	id_cliente
id_tipo_cliente	TIPOS_CLIENTE	id_tipo_cliente

- **COM_EVALUADORA**

Nombre	Tipo	PK	Nulo
id_comision	NUMBER(10)	Si	Si
id_jurado1	NUMBER(10)	No	Si
id_jurado2	NUMBER(10)	No	Si
id_jurado3	NUMBER(10)	No	Si

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_cliente	DATOS_CLIENTES	id_jurado1
id_cliente	DATOS_CLIENTES	id_jurado2
id_cliente	DATOS_CLIENTES	id_jurado3

Referenciado por	Clave Primaria	Clave Foránea
GRADUADOS	id_comision	id_com_evaluadora

- **CONCURSANTES**

Nombre	Tipo	PK	Nulo
id_concurso	NUMBER(10)	Si	Si
id_cliente	NUMBER(10)	Si	Si
ganador	VARCHAR2(1)	No	Si

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_concurso	CONCURSOS	id_concurso
id_cliente	DATOS_CLIENTES	id_cliente

- **CONCURSOS**

Nombre	Tipo	PK	Nulo
id_concurso	NUMBER(10)	Si	Si
id_materia	NUMBER(10)	No	Si
id_cargo	NUMBER(10)	No	Si
nom_concurso	VARCHAR2(100)	No	Si
cod_concurso	VARCHAR2(20)	No	No
fec_apertura	DATE	No	No
fec_cierre	DATE	No	No
cant_cargos	NUMBER(2)	No	Si
requisitos	VARCHAR2(2000)	No	Si
insc_abierta	VARCHAR2(1)	No	Si

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_materia	MATERIAS	id_materia
id_cargo	CARGOS	id_cargo

Referenciado por	Clave Primaria	Clave Foránea
CONCURSANTES	id_concurso	id_concurso

- **CORRELATIVAS**

Nombre	Tipo	PK	Nulo
id_materia	NUMBER(10)	Si	Si
id_correlativa	NUMBER(10)	Si	Si

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_materia	MATERIAS	id_correlativa
id_materia	MATERIAS	id_materia

- **CUOTAS_PAGAS**

Nombre	Tipo	PK	Nulo
id_carrera	NUMBER(10)	Si	Si
id_cliente	NUMBER(10)	Si	Si
num_cuota	NUMBER(2)	Si	Si
valor	NUMBER(15,2)	No	Si
fec_pago	DATE	No	Si
curso_pago	VARCHAR2(1)	No	Si

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_carrera	CARRERAS	id_carrera
id_cliente	DATOS_CLIENTES	id_cliente

- **CURSO_ALUMNO**

Nombre	Tipo	PK	Nulo
id_cliente	NUMBER(10)	Si	Si
id_carrera	NUMBER(10)	Si	Si
legajo	VARCHAR2(8)	No	No

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_carrera	CARRERAS	id_carrera
id_cliente	DATOS_CLIENTES	id_cliente

- **DATOS_CLIENTES**

Nombre	Tipo	PK	Nulo
id_cliente	NUMBER(10)	Si	Si
id_tipo_doc	VARCHAR2(10)	No	Si
nro_doc	Numeric(11)	No	Si
nombre	VARCHAR2(40)	No	Si
apellido	VARCHAR2(40)	No	Si
domicilio	VARCHAR2(100)	No	No
localidad	VARCHAR2(40)	No	No
partido	VARCHAR2(40)	No	No
provincia	VARCHAR2(40)	No	No
pais	VARCHAR2(40)	No	No
cod_pos	VARCHAR2(10)	No	No
tel	VARCHAR2(40)	No	No
fax	VARCHAR2(40)	No	No
e_mail	VARCHAR(100)	No	No

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_tipo_doc	TIPOS_DOC	id_tipo_doc
id_cliente	CLIENTES	id_cliente

Es Referenciada por	Clave Primaria	Clave Foránea
CLIENTES_POR_TIPO	id_cliente	id_cliente
COM_EVALUADORA	id_cliente	id_jurado1
COM_EVALUADORA	id_cliente	id_jurado2
COM_EVALUADORA	id_cliente	id_jurado3
GRADUADOS	id_cliente	id_cliente
MATERIA_PROFESOR	id_cliente	id_cliente
MATERIA_ALUMNO	id_cliente	id_cliente
TRAB_ENTREGADOS	id_cliente	id_cliente
CUOTAS_PAGAS	id_cliente	id_cliente
INSC_FINALES	id_cliente	id_cliente
ACTAS_FINALES	id_cliente	id_cliente
ACTAS_FINALES	id_cliente	id_profesor
CURSO_ALUMNO	id_cliente	id_cliente
NOTAS	id_cliente	id_cliente
GRUPOS_TRABAJO	id_cliente	id_cliente
SOLIC_ANALITICO	id_cliente	id_cliente
CONCURSANTES	id_cliente	id_cliente

- **GRADUADOS**

Nombre	Tipo	PK	Nulo
id_carrera	NUMBER(10)	Si	Si
id_cliente	NUMBER(10)	Si	Si
fec_graduacion	DATE	No	Si
id_com_evaluadora	NUMBER(10)	No	No

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_cliente	DATOS_CLIENTES	id_cliente
id_comision	COM_EVALUADORA	id_com_evaluadora
id_carrera	CARRERAS	id_carrera

- **GRUPOS_TRABAJO**

Nombre	Tipo	PK	Nulo
id_grupo_trabajo	NUMBER(10)	Si	Si
id_cliente	NUMBER(10)	Si	Si
id_trabajo	NUMBER(10)	No	Si
cod_grupo_trabajo	VARCHAR2(10)	No	Si

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_cliente	DATOS_CLIENTES	id_cliente
id_trabajo	TRABAJOS	id_trabajo

- **INSC_FINALES**

Nombre	Tipo	PK	Nulo
id_cliente	NUMBER(10)	Si	Si
id_cal_final	NUMBER(10)	Si	Si

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_cliente	DATOS_CLIENTES	id_cliente
id_cal_final	CAL_FINALES	id_cal_final

- **MATERIA_ALUMNO**

Nombre	Tipo	PK	Nulo
id_cliente	NUMBER(10)	Si	Si
id_materia	NUMBER(10)	Si	Si
id_turno	NUMBER(10)	No	Si
anio_cursada	NUMBER(4)	No	Si
aprob_cursada	VARCHAR2(1)	No	Si
aprob_final	VARCHAR2(1)	No	Si
cursando	VARCHAR2(1)	No	Si
cursada_vencida	VARCHAR2(1)	No	Si

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_cliente	DATOS_CLIENTES	id_cliente
id_materia	MATERIAS	id_materia
id_turno	TURNO_MATERIAS	id_turno

- **MATERIA_PROFESOR**

Nombre	Tipo	PK	Nulo
id_cliente	NUMBER(10)	Si	Si
id_materia	NUMBER(10)	Si	Si
id_cargo	NUMBER(10)	No	Si
fec_inicio	DATE	Si	Si
fec_fin	DATE	No	No

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_cliente	DATOS_CLIENTES	id_cliente
id_materia	MATERIAS	id_materia
id_cargo	CARGOS	id_cargo

- **MATERIAS**

Nombre	Tipo	PK	Nulo
id_carrera	NUMBER(10)	No	Si
id_materia	NUMBER(10)	Si	Si
Cod_materia	VARCHAR2(20)	No	Si
Nom_materia	VARCHAR2(80)	No	Si
Programa	VARCHAR2(2000)	No	No
Duracion	VARCHAR2(20)	No	No
Bibliografia	VARCHAR2(1000)	No	No
Requisitos	VARCHAR2(200)	No	No
Cupo	NUMBER(6)	No	No

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_carrera	CARRERAS	id_carrera

Referenciado por	Clave Primaria	Clave Foránea
TURNO_MATERIAS	id_materia	id_materia
TRABAJOS	id_materia	id_materia
CAL_FINALES	id_materia	id_materia
CORRELATIVAS	id_materia	id_correlativa
CORRELATIVAS	id_materia	id_materia
MATERIA_PROFESOR	id_materia	id_materia
MATERIA_ALUMNO	id_materia	id_materia
CAL_TRABAJOS	id_materia	id_materia
CAL_MATERIAS	id_materia	id_materia
TRAB_ENTREGADOS	id_materia	id_materia
ACTAS_FINALES	id_materia	id_materia
NOTAS	id_materia	id_materia
CONCURSOS	id_materia	id_materia

- **NOTAS**

Nombre	Tipo	PK	Nulo
id_cliente	NUMBER(10)	Si	Si
id_materia	NUMBER(10)	Si	Si
id_trabajo	NUMBER(10)	No	Si
id_tipo_trabajo	NUMBER(10)	Si	Si
fecha	DATE	Si	Si
nota	NUMBER(10,2)	No	Si

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_cliente	DATOS_CLIENTES	id_cliente
id_materia	MATERIAS	id_materia
id_tipo_trabajo	TIPOS_TRABAJO	id_tipo_trabajo
id_trabajo	TRABAJOS	id_trabajo

- **SOLIC_ANALITICO**

Nombre	Tipo	PK	Nulo
id_carrera	NUMBER(10)	Si	Si
id_cliente	NUMBER(10)	Si	Si
fec_solicitud	DATE	Si	Si

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_cliente	DATOS_CLIENTES	id_cliente
id_carrera	CARRERAS	id_carrera

- **TIPOS_CLIENTE**

Nombre	Tipo	PK	Nulo
id_tipo_cliente	NUMBER(10)	Si	Si
nom_cliente	VARCHAR2(40)	No	Si
cod_tipo_cli	VARCHAR2(20)	No	Si

Referencias

Referenciado por	Clave Primaria	Clave Foránea
CLIENTES_POR_TIPO	id_tipo_cliente	id_tipo_cliente

- **TIPOS_DOC**

Nombre	Tipo	PK	Nulo
id_tipo_doc	VARCHAR2(10)	Si	Si
nom_tipo_doc	VARCHAR2(20)	No	Si

Referencias

Referenciado por	Clave Primaria	Clave Foránea
DATOS_CLIENTES	id_tipo_doc	id_tipo_doc

- **TIPOS_TRABAJO**

Nombre	Tipo	PK	Nulo
id_tipo_trabajo	NUMBER(10)	Si	Si
cod_tipo_trabajo	VARCHAR2(20)	No	Si
nom_tipo_trabajo	VARCHAR2(40)	No	Si

Referencias

Referenciado por	Clave Primaria	Clave Foránea
TRABAJOS	id_tipo_trabajo	id_tipo_trabajo
NOTAS	id_tipo_trabajo	id_tipo_trabajo

- **TRAB_ENTREGADOS**

Nombre	Tipo	PK	Nulo
id_materia	NUMBER(10)	No	Si
id_cliente	NUMBER(10)	Si	Si
id_trabajo	NUMBER(10)	Si	Si
calif	NUMBER(15,2)	No	No
Aprobado	VARCHAR2(1)	No	No
fecha_entrega	DATE	No	Si
observaciones	VARCHAR2(2000)	No	No

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_materia	MATERIAS	id_materia
id_trabajo	TRABAJOS	id_trabajo
id_cliente	DATOS_CLIENTES	id_cliente

- **TRABAJOS**

Nombre	Tipo	PK	Nulo
id_materia	NUMBER(10)	No	Si
id_trabajo	NUMBER(10)	Si	Si
id_tipo_trabajo	NUMBER(10)	No	Si
cod_trabajo	VARCHAR2(20)	No	Si
titulo	VARCHAR2(100)	No	No
tema	VARCHAR2(2000)	No	No
especificaciones	VARCHAR2(2000)	No	No

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_materia	MATERIAS	id_materia
id_tipo_trabajo	TIPOS_TRABAJO	id_tipo_trabajo

Referenciado por	Clave Primaria	Clave Foránea
CAL_TRABAJOS	id_trabajo	id_trabajo
TRAB_ENTREGADOS	id_trabajo	id_trabajo
NOTAS	id_trabajo	id_trabajo
GRUPOS_TRABAJO	id_trabajo	id_trabajo

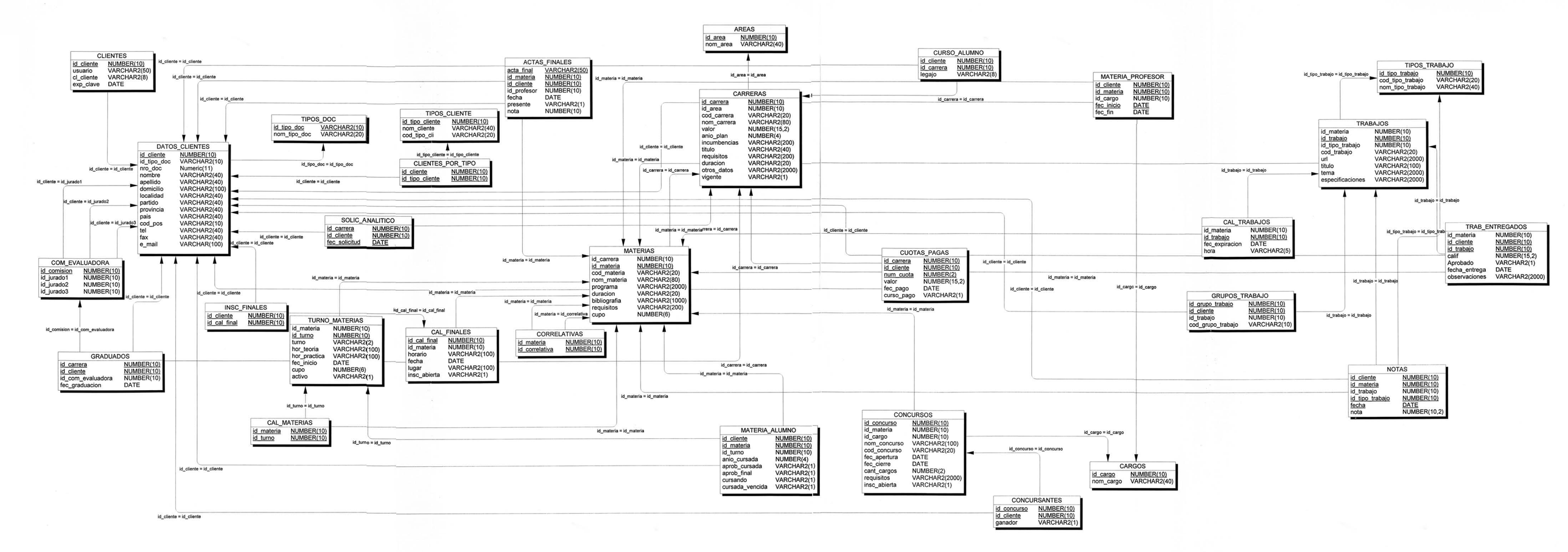
- **TURNO_MATERIAS**

Nombre	Tipo	PK	Nulo
id_materia	NUMBER(10)	No	Si
id_turno	NUMBER(10)	Si	Si
turno	VARCHAR2(2)	No	Si
hor_teoria	VARCHAR2(100)	No	Si
hor_practica	VARCHAR2(100)	No	Si
fec_inicio	DATE	No	No
cupo	NUMBER(6)	No	No
activo	VARCHAR2(1)	No	Si

Referencias

Clave Primaria	Referencia a	Clave Foránea
id_materia	MATERIAS	id_materia

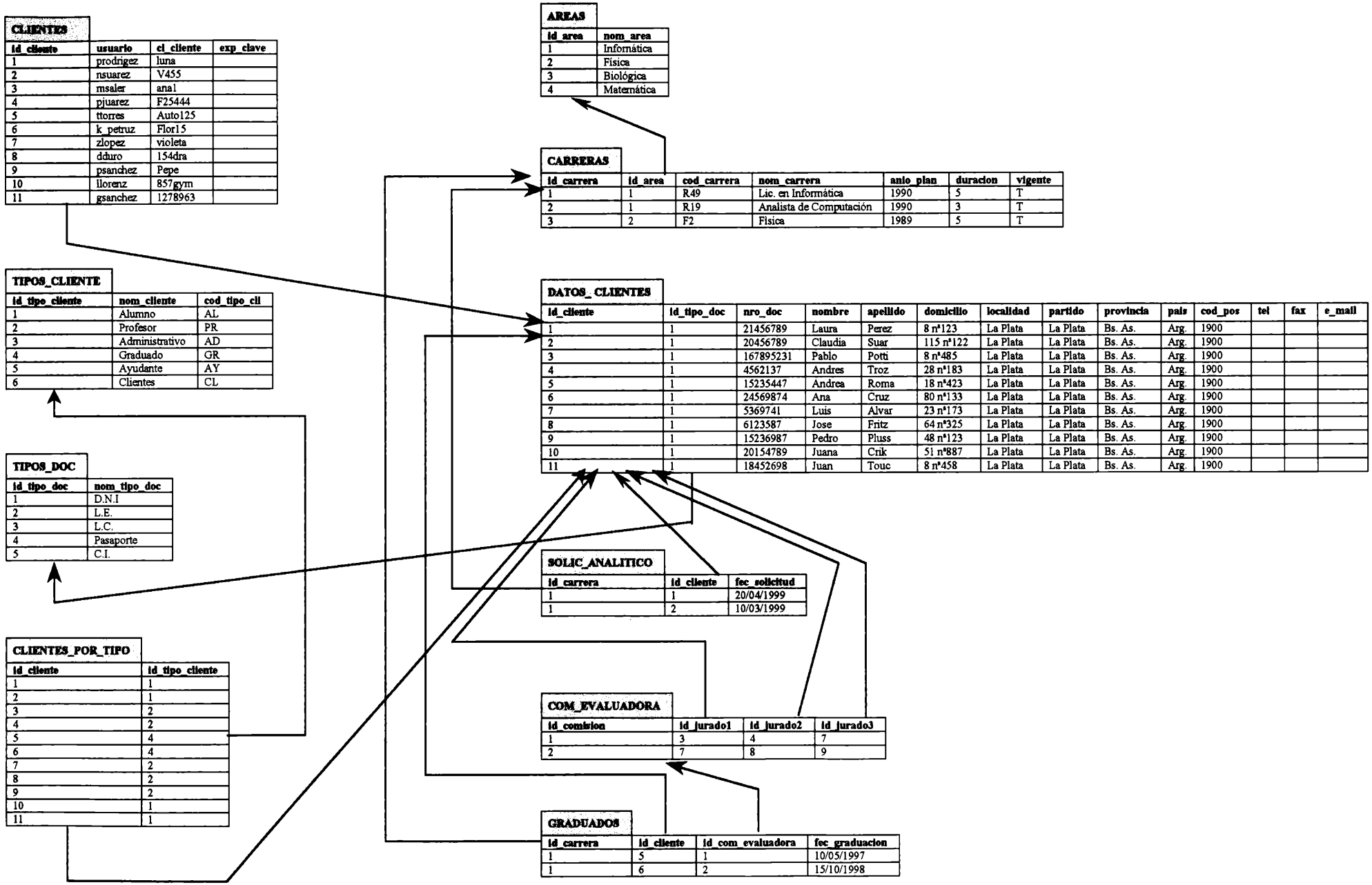
Referenciado por	Clave Primaria	Clave Foránea
MATERIA_ALUMNO	id_turno	id_turno
CAL_MATERIAS	id_turno	id_turno


AREAS
id_area NUMBER(10)
nom_area VARCHAR2(40)
CLIENTES
id_cliente NUMBER(10)
usuario VARCHAR2(50)
cl_cliente VARCHAR2(8)
exp_clave DATE
ACTAS_FINALES
acta_final VARCHAR2(50)
id_materia NUMBER(10)
id_cliente NUMBER(10)
id_profesor NUMBER(10)
fecha DATE
presente VARCHAR2(1)
nota NUMBER(10)
CURSO_ALUMNO
id_cliente NUMBER(10)
id_carrera NUMBER(10)
legajo VARCHAR2(8)
MATERIA_PROFESOR
id_cliente NUMBER(10)
id_materia NUMBER(10)
id_cargo NUMBER(10)
fec_inicio DATE
fec_fin DATE
TIPOS_TRABAJO
id_tipo_trabajo NUMBER(10)
cod_tipo_trabajo VARCHAR2(20)
nom_tipo_trabajo VARCHAR2(40)
TIPOS_DOC
id_tipo_doc VARCHAR2(10)
nom_tipo_doc VARCHAR2(20)
TIPOS_CLIENTE
id_tipo_cliente NUMBER(10)
nom_cliente VARCHAR2(40)
cod_tipo_cli VARCHAR2(20)
CARRERAS
id_carrera NUMBER(10)
id_area NUMBER(10)
cod_carrera VARCHAR2(20)
nom_carrera VARCHAR2(80)
valor NUMBER(15,2)
anio_plan NUMBER(4)
incumbencias VARCHAR2(200)
titulo VARCHAR2(40)
requisitos VARCHAR2(200)
duracion VARCHAR2(20)
otros_datos VARCHAR2(2000)
vigente VARCHAR2(1)
TRABAJOS
id_materia NUMBER(10)
id_trabajo NUMBER(10)
id_tipo_trabajo NUMBER(10)
cod_trabajo VARCHAR2(20)
url VARCHAR2(2000)
titulo VARCHAR2(100)
tema VARCHAR2(2000)
especificaciones VARCHAR2(2000)
DATOS_CLIENTES
id_cliente NUMBER(10)
id_tipo_doc VARCHAR2(10)
nro_doc Numeric(11)
nombre VARCHAR2(40)
apellido VARCHAR2(40)
domicilio VARCHAR2(100)
localidad VARCHAR2(40)
partido VARCHAR2(40)
provincia VARCHAR2(40)
pais VARCHAR2(40)
cod_pos VARCHAR2(10)
tel VARCHAR2(40)
fax VARCHAR2(40)
e_mail VARCHAR(100)
CLIENTES_POR_TIPO
id_cliente NUMBER(10)
id_tipo_cliente NUMBER(10)
SOLIC_ANALITICO
id_carrera NUMBER(10)
id_cliente NUMBER(10)
fec_solicitud DATE
CAL_TRABAJOS
id_materia NUMBER(10)
id_trabajo NUMBER(10)
fec_expiracion DATE
hora VARCHAR2(5)
TRAB_ENTREGADOS
id_materia NUMBER(10)
id_cliente NUMBER(10)
id_trabajo NUMBER(10)
calif NUMBER(15,2)
Aprobado VARCHAR2(1)
fecha_entrega DATE
observaciones VARCHAR2(2000)
MATERIAS
id_carrera NUMBER(10)
id_materia NUMBER(10)
cod_materia VARCHAR2(20)
nom_materia VARCHAR2(80)
programa VARCHAR2(2000)
duracion VARCHAR2(20)
bibliografia VARCHAR2(1000)
requisitos VARCHAR2(200)
cupo NUMBER(6)
CUOTAS_PAGAS
id_carrera NUMBER(10)
id_cliente NUMBER(10)
num_cuota NUMBER(2)
valor NUMBER(15,2)
fec_pago DATE
curso_pago VARCHAR2(1)
GRUPOS_TRABAJO
id_grupo_trabajo NUMBER(10)
id_cliente NUMBER(10)
id_trabajo NUMBER(10)
cod_grupo_trabajo VARCHAR2(10)
COM_EVALUADORA
id_comision NUMBER(10)
id_jurado1 NUMBER(10)
id_jurado2 NUMBER(10)
id_jurado3 NUMBER(10)
INSC_FINALES
id_cliente NUMBER(10)
id_cal_final NUMBER(10)
TURNO_MATERIAS
id_materia NUMBER(10)
id_turno NUMBER(10)
turno VARCHAR2(2)
hor_teoria VARCHAR2(100)
hor_practica VARCHAR2(100)
fec_inicio DATE
cupo NUMBER(6)
activo VARCHAR2(1)
CAL_FINALES
id_cal_final NUMBER(10)
id_materia NUMBER(10)
horario VARCHAR2(100)
fecha DATE
lugar VARCHAR2(100)
insc_abierta VARCHAR2(1)
CORRELATIVAS
id_materia NUMBER(10)
id_correlativa NUMBER(10)
NOTAS
id_cliente NUMBER(10)
id_materia NUMBER(10)
id_trabajo NUMBER(10)
id_tipo_trabajo NUMBER(10)
fecha DATE
nota NUMBER(10,2)
GRADUADOS
id_carrera NUMBER(10)
id_cliente NUMBER(10)
id_com_evaluadora NUMBER(10)
fec_graduacion DATE
CAL_MATERIAS
id_materia NUMBER(10)
id_turno NUMBER(10)
MATERIA_ALUMNO
id_cliente NUMBER(10)
id_materia NUMBER(10)
id_turno NUMBER(10)
anio_cursada NUMBER(4)
aprob_cursada VARCHAR2(1)
aprob_final VARCHAR2(1)
cursando VARCHAR2(1)
cursada_vencida VARCHAR2(1)
CONCURSOS
id_concurso NUMBER(10)
id_materia NUMBER(10)
id_cargo NUMBER(10)
nom_concurso VARCHAR2(100)
cod_concurso VARCHAR2(20)
fec_apertura DATE
fec_cierre DATE
cant_cargos NUMBER(2)
requisitos VARCHAR2(2000)
insc_abierta VARCHAR2(1)
CARGOS
id_cargo NUMBER(10)
nom_cargo VARCHAR2(40)
CONCURSANTES
id_concurso NUMBER(10)
id_cliente NUMBER(10)
ganador VARCHAR2(1)

MODELO DE INSTANCIA

A continuación presentamos un modelo de instancia de Apliades, que contiene todas las tablas de la Base de Datos de la aplicación.

Modelo de Comunicación



CLIENTES

id_cliente	usuario	cl_cliente	exp_clave
1	prodrigez	luna	
2	nsuarez	V455	
3	msaler	ana1	
4	pjuarez	F25444	
5	ttorres	Auto125	
6	k_petruz	Flor15	
7	zlopez	violeta	
8	dduro	154dra	
9	psanchez	Pepe	
10	llorenz	857gym	
11	gsanchez	1278963	

AREAS

id_area	nom_area
1	Informática
2	Física
3	Biológica
4	Matemática

CARRERAS

id_carrera	id_area	cod_carrera	nom_carrera	anio_plan	duracion	vigente
1	1	R49	Lic. en Informática	1990	5	T
2	1	R19	Analista de Computación	1990	3	T
3	2	F2	Física	1989	5	T

TIPOS_CLIENTE

id_tipo_cliente	nom_cliente	cod_tipo_cli
1	Alumno	AL
2	Profesor	PR
3	Administrativo	AD
4	Graduado	GR
5	Ayudante	AY
6	Clientes	CL

DATOS_CLIENTES

id_cliente	id_tipo_doc	nro_doc	nombre	apellido	domicilio	localidad	partido	provincia	pais	cod_pos	tel	fax	e_mail
1	1	21456789	Laura	Perez	8 nº123	La Plata	La Plata	Bs. As.	Arg.	1900			
2	1	20456789	Claudia	Suar	115 nº122	La Plata	La Plata	Bs. As.	Arg.	1900			
3	1	167895231	Pablo	Potti	8 nº485	La Plata	La Plata	Bs. As.	Arg.	1900			
4	1	4562137	Andres	Troz	28 nº183	La Plata	La Plata	Bs. As.	Arg.	1900			
5	1	15235447	Andrea	Roma	18 nº423	La Plata	La Plata	Bs. As.	Arg.	1900			
6	1	24569874	Ana	Cruz	80 nº133	La Plata	La Plata	Bs. As.	Arg.	1900			
7	1	5369741	Luis	Alvar	23 nº173	La Plata	La Plata	Bs. As.	Arg.	1900			
8	1	6123587	Jose	Fritz	64 nº325	La Plata	La Plata	Bs. As.	Arg.	1900			
9	1	15236987	Pedro	Pluss	48 nº123	La Plata	La Plata	Bs. As.	Arg.	1900			
10	1	20154789	Juana	Crik	51 nº887	La Plata	La Plata	Bs. As.	Arg.	1900			
11	1	18452698	Juan	Touc	8 nº458	La Plata	La Plata	Bs. As.	Arg.	1900			

TIPOS_DOC

id_tipo_doc	nom_tipo_doc
1	D.N.I
2	L.E.
3	L.C.
4	Pasaporte
5	C.I.

SOLIC_ANALITICO

id_carrera	id_cliente	fec_solicitud
1	1	20/04/1999
1	2	10/03/1999

CLIENTES_POR_TIPO

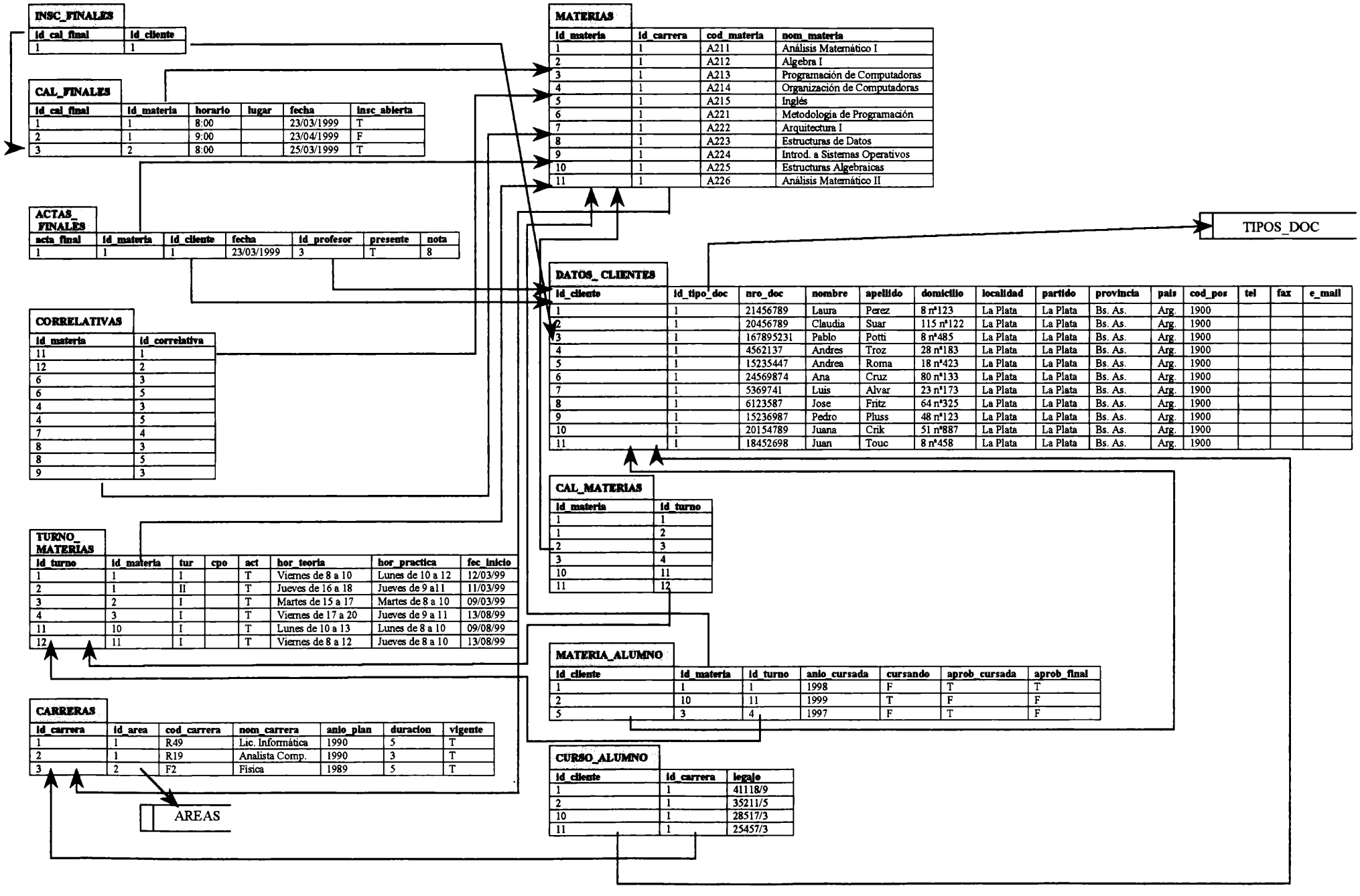
id_cliente	id_tipo_cliente
1	1
2	1
3	2
4	2
5	4
6	4
7	2
8	2
9	2
10	1
11	1

COM_EVALUADORA

id_comision	id_jurado1	id_jurado2	id_jurado3
1	3	4	7
2	7	8	9

GRADUADOS

id_carrera	id_cliente	id_com_evaluadora	fec_graduacion
1	5	1	10/05/1997
1	6	2	15/10/1998



INSC_FINALES

id_cal_final	id_cliente
1	1

CAL_FINALES

id_cal_final	id_materia	horario	lugar	fecha	insc_abierta
1	1	8:00		23/03/1999	T
2	1	9:00		23/04/1999	F
3	2	8:00		25/03/1999	T

ACTAS_FINALES

acta_final	id_materia	id_cliente	fecha	id_profesor	presente	nota
1	1	1	23/03/1999	3	T	8

CORRELATIVAS

id_materia	id_correlativa
11	1
12	2
6	3
6	5
4	3
4	5
7	4
8	3
8	5
9	3

TURNO_MATERIAS

id_turno	id_materia	tur	cpo	act	hor_teoria	hor_practica	fec_inicio
1	1	I		T	Viernes de 8 a 10	Lunes de 10 a 12	12/03/99
2	1	II		T	Jueves de 16 a 18	Jueves de 9 a11	11/03/99
3	2	I		T	Martes de 15 a 17	Martes de 8 a 10	09/03/99
4	3	I		T	Viernes de 17 a 20	Jueves de 9 a 11	13/08/99
11	10	I		T	Lunes de 10 a 13	Lunes de 8 a 10	09/08/99
12	11	I		T	Viernes de 8 a 12	Jueves de 8 a 10	13/08/99

CARRERAS

id_carrera	id_area	cod_carrera	nom_carrera	anio_plan	duracion	vigente
1	1	R49	Lic. Informática	1990	5	T
2	1	R19	Analista Comp.	1990	3	T
3	2	F2	Física	1989	5	T

AREAS

MATERIAS

id_materia	id_carrera	cod_materia	nom_materia
1	1	A211	Análisis Matemático I
2	1	A212	Algebra I
3	1	A213	Programación de Computadoras
4	1	A214	Organización de Computadoras
5	1	A215	Inglés
6	1	A221	Metodología de Programación
7	1	A222	Arquitectura I
8	1	A223	Estructuras de Datos
9	1	A224	Introd. a Sistemas Operativos
10	1	A225	Estructuras Algebraicas
11	1	A226	Análisis Matemático II

TIPOS_DOC

DATOS_CLIENTES

id_cliente	id_tipo_doc	nro_doc	nombre	apellido	domicilio	localidad	partido	provincia	pais	cod_pos	tel	fax	e_mail
1	1	21456789	Laura	Perez	8 nº123	La Plata	La Plata	Bs. As.	Arg.	1900			
2	1	20456789	Claudia	Suar	115 nº122	La Plata	La Plata	Bs. As.	Arg.	1900			
3	1	167895231	Pablo	Potti	8 nº485	La Plata	La Plata	Bs. As.	Arg.	1900			
4	1	4562137	Andres	Troz	28 nº183	La Plata	La Plata	Bs. As.	Arg.	1900			
5	1	15235447	Andrea	Roma	18 nº423	La Plata	La Plata	Bs. As.	Arg.	1900			
6	1	24569874	Ana	Cruz	80 nº133	La Plata	La Plata	Bs. As.	Arg.	1900			
7	1	5369741	Luis	Alvar	23 nº173	La Plata	La Plata	Bs. As.	Arg.	1900			
8	1	6123587	Jose	Fritz	64 nº325	La Plata	La Plata	Bs. As.	Arg.	1900			
9	1	15236987	Pedro	Pluss	48 nº123	La Plata	La Plata	Bs. As.	Arg.	1900			
10	1	20154789	Juana	Crik	51 nº887	La Plata	La Plata	Bs. As.	Arg.	1900			
11	1	18452698	Juan	Touc	8 nº458	La Plata	La Plata	Bs. As.	Arg.	1900			

CAL_MATERIAS

id_materia	id_turno
1	1
1	2
2	3
3	4
10	11
11	12

MATERIA_ALUMNO

id_cliente	id_materia	id_turno	anio_cursada	cursando	aprob_cursada	aprob_final
1	1	1	1998	F	T	T
2	10	11	1999	T	F	F
5	3	4	1997	F	T	F

CURSO_ALUMNO

id_cliente	id_carrera	legajo
1	1	41118/9
2	1	35211/5
10	1	28517/3
11	1	25457/3

TIPOS_DOC

CARGOS

id_cargo	nom_cargo
1	Profesor Titular
2	Profesor Adjunto
3	Ayudante Alumno
4	Ayudante Diplomado

DATOS_CLIENTES

id_cliente	id_tipo_doc	nro_doc	nombre	apellido	domicilio	localidad	partido	provincia	pais	cod_pos	tel	fax	e_mail
1	1	21456789	Laura	Perez	8 nº123	La Plata	La Plata	Bs. As.	Arg.	1900			
2	1	20456789	Claudia	Suar	115 nº122	La Plata	La Plata	Bs. As.	Arg.	1900			
3	1	167895231	Pablo	Potti	8 nº485	La Plata	La Plata	Bs. As.	Arg.	1900			
4	1	4562137	Andres	Troz	28 nº183	La Plata	La Plata	Bs. As.	Arg.	1900			
5	1	15235447	Andrea	Roma	18 nº423	La Plata	La Plata	Bs. As.	Arg.	1900			
6	1	24569874	Ana	Cruz	80 nº133	La Plata	La Plata	Bs. As.	Arg.	1900			
7	1	5369741	Luis	Alvar	23 nº173	La Plata	La Plata	Bs. As.	Arg.	1900			
8	1	6123587	Jose	Fritz	64 nº325	La Plata	La Plata	Bs. As.	Arg.	1900			
9	1	15236987	Pedro	Pluss	48 nº123	La Plata	La Plata	Bs. As.	Arg.	1900			
10	1	20154789	Juana	Crik	51 nº887	La Plata	La Plata	Bs. As.	Arg.	1900			
11	1	18452698	Juan	Touc	8 nº458	La Plata	La Plata	Bs. As.	Arg.	1900			

MATERIA_PROFESOR

id_cliente	id_materia	fec_inicio	id_cargo	fec_fin
3	1	01/01/1998	1	
4	1	01/01/1997	2	
7	2	01/01/1997	1	

CONCURSANTES

id_concurso	id_cliente
1	1
1	2

CONCURSOS

id_concurso	id_materia	cod_concurso	fec_apertura	fec_cierre	id_cargo	cant_cargos	requisitos	ins_abierta
1	1	C1	01/01/1999	30/03/1999	3	2	Análisis I Aprobado	T
2	2	C2	10/05/1999	30/06/1999	2	1	Lic. en Informática	T

MATERIAS

id_materia	id_carrera	cod_materia	nom_materia
1	1	A211	Análisis Matemático I
2	1	A212	Algebra I
3	1	A213	Programación de Computadoras
4	1	A214	Organización de Computadoras
5	1	A215	Inglés
6	1	A221	Metodología de Programación
7	1	A222	Arquitectura I
8	1	A223	Estructuras de Datos
9	1	A224	Introd. a Sistemas Operativos
10	1	A225	Estructuras Algebraicas
11	1	A226	Análisis Matemático II

AREAS

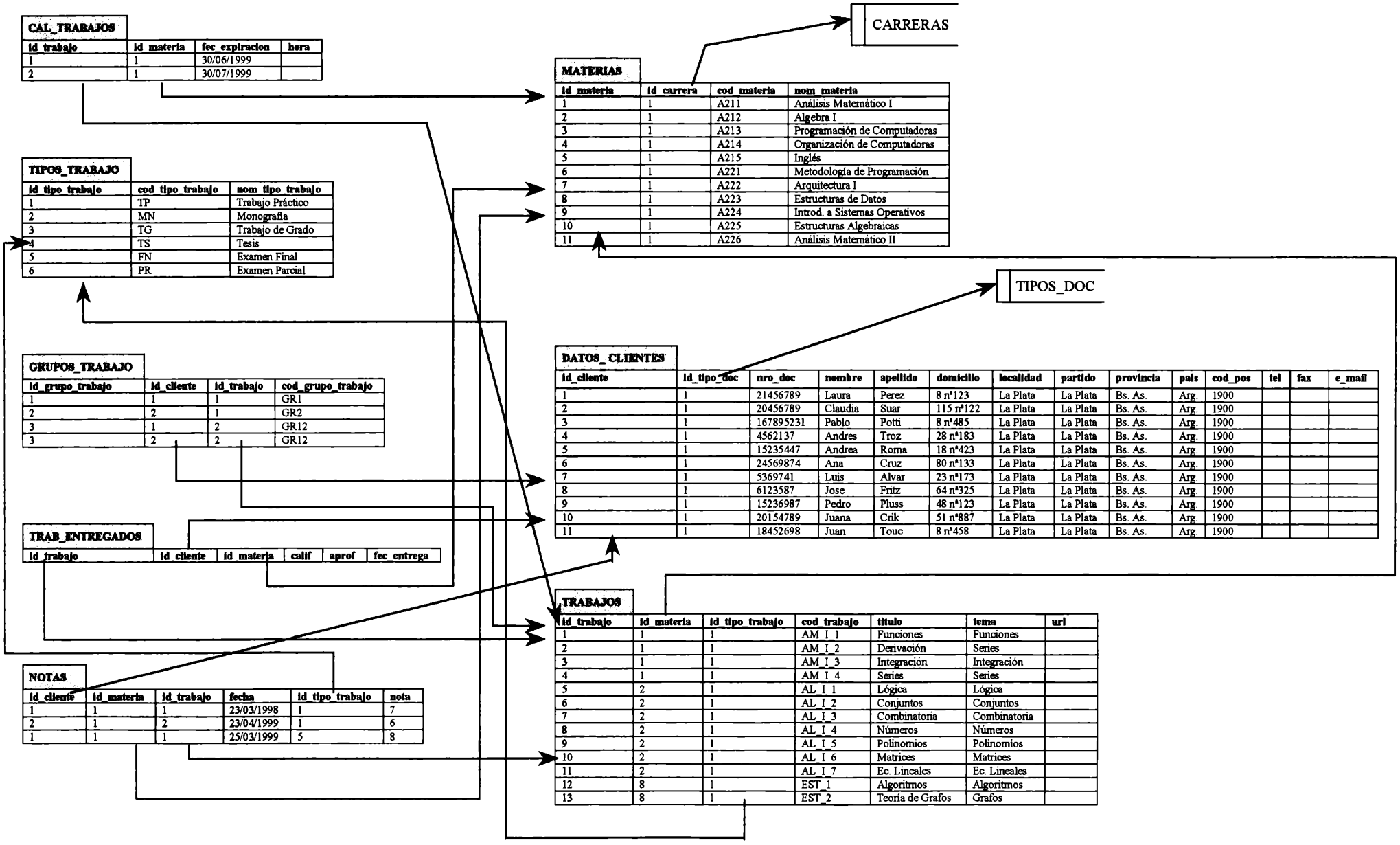
id_area	nom_area
1	Informática
2	Física
3	Biológica
4	Matemática

CARRERAS

id_carrera	id_area	cod_carrera	nom_carrera	anio_plan	duracion	vigente
1	1	R49	Lic. en Informática	1990	5	T
2	1	R19	Analista de Computación	1990	3	T
3	2	F2	Fisica	1989	5	T

CUOTAS_PAGAS

id_carrera	id_cliente	num_cuota	valor	fec_pago	curso_pago
1	1	1	100	02/03/1999	F



CAL_TRABAJOS

id_trabajo	id_materia	fec_expiracion	hora
1	1	30/06/1999	
2	1	30/07/1999	

CARRERAS

MATERIAS

id_materia	id_carrera	cod_materia	nom_materia
1	1	A211	Análisis Matemático I
2	1	A212	Algebra I
3	1	A213	Programación de Computadoras
4	1	A214	Organización de Computadoras
5	1	A215	Inglés
6	1	A221	Metodología de Programación
7	1	A222	Arquitectura I
8	1	A223	Estructuras de Datos
9	1	A224	Introd. a Sistemas Operativos
10	1	A225	Estructuras Algebraicas
11	1	A226	Análisis Matemático II

TIPOS_TRABAJO

id_tipo_trabajo	cod_tipo_trabajo	nom_tipo_trabajo
1	TP	Trabajo Práctico
2	MN	Monografía
3	TG	Trabajo de Grado
4	TS	Tesis
5	FN	Examen Final
6	PR	Examen Parcial

TIPOS_DOC

DATOS_CLIENTES

id_cliente	id_tipo_doc	nro_doc	nombre	apellido	domicilio	localidad	partido	provincia	pais	cod_pos	tel	fax	e_mail
1	1	21456789	Laura	Perez	8 nº123	La Plata	La Plata	Bs. As.	Arg.	1900			
2	1	20456789	Claudia	Suar	115 nº122	La Plata	La Plata	Bs. As.	Arg.	1900			
3	1	167895231	Pablo	Potti	8 nº485	La Plata	La Plata	Bs. As.	Arg.	1900			
4	1	4562137	Andres	Troz	28 nº183	La Plata	La Plata	Bs. As.	Arg.	1900			
5	1	15235447	Andrea	Roma	18 nº423	La Plata	La Plata	Bs. As.	Arg.	1900			
6	1	24569874	Ana	Cruz	80 nº133	La Plata	La Plata	Bs. As.	Arg.	1900			
7	1	5369741	Luis	Alvar	23 nº173	La Plata	La Plata	Bs. As.	Arg.	1900			
8	1	6123587	Jose	Fritz	64 nº325	La Plata	La Plata	Bs. As.	Arg.	1900			
9	1	15236987	Pedro	Pluss	48 nº123	La Plata	La Plata	Bs. As.	Arg.	1900			
10	1	20154789	Juana	Crik	51 nº887	La Plata	La Plata	Bs. As.	Arg.	1900			
11	1	18452698	Juan	Touc	8 nº458	La Plata	La Plata	Bs. As.	Arg.	1900			

GRUPOS_TRABAJO

id_grupo_trabajo	id_cliente	id_trabajo	cod_grupo_trabajo
1	1	1	GR1
2	2	1	GR2
3	1	2	GR12
3	2	2	GR12

TRAB_ENTREGADOS

id_trabajo	id_cliente	id_materia	calif	aprof	fec_entrega

TRABAJOS

id_trabajo	id_materia	id_tipo_trabajo	cod_trabajo	titulo	tema	url
1	1	1	AM_I_1	Funciones	Funciones	
2	1	1	AM_I_2	Derivación	Series	
3	1	1	AM_I_3	Integración	Integración	
4	1	1	AM_I_4	Series	Series	
5	2	1	AL_I_1	Lógica	Lógica	
6	2	1	AL_I_2	Conjuntos	Conjuntos	
7	2	1	AL_I_3	Combinatoria	Combinatoria	
8	2	1	AL_I_4	Números	Números	
9	2	1	AL_I_5	Polinomios	Polinomios	
10	2	1	AL_I_6	Matrices	Matrices	
11	2	1	AL_I_7	Ec. Lineales	Ec. Lineales	
12	8	1	EST_1	Algoritmos	Algoritmos	
13	8	1	EST_2	Teoría de Grafos	Grafos	

NOTAS

id_cliente	id_materia	id_trabajo	fecha	id_tipo_trabajo	nota
1	1	1	23/03/1998	1	7
2	1	2	23/04/1999	1	6
1	1	1	25/03/1999	5	8

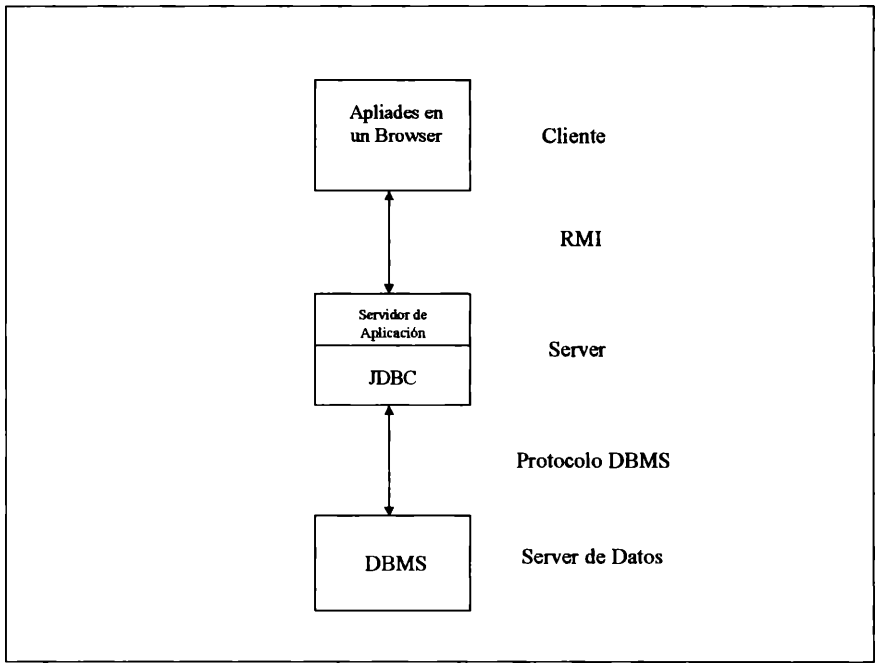
MODELOS DE COMUNICACION

MODELO DE COMUNICACIÓN CLIENTE/SERVIDOR

Si bien el API JDBC soporta los modelos de comunicación two-tier y three-tier para el acceso a las bases de datos, el modelo que adoptamos para Apliades es el three-tier.

En el modelo Three-tier, los comandos son enviados al Server de aplicación, el cual es el encargado de enviar las sentencias SQL a la Base de Datos. La Base de Datos procesa las sentencias SQL y envía los resultados al Server, el cual los envía al usuario.

Este modelo fue elegido porque permite llevar un control de los accesos y un control de los tipos de modificaciones que se pueden realizar sobre la Base de Datos, mientras que en el caso del two-tier, se pierde el control, ya que la comunicación del cliente con la Base de Datos es directa, sin tener al Server como intermediario. Además, al darle toda la inteligencia al Server, la complejidad del cliente disminuye, convirtiéndolo en un cliente "thin".



Modelo Tier de Apliades

MODELO DE COMUNICACIÓN EN EL CLIENTE

Cuando un usuario se conecta a la aplicación, todos los servicios que se llevan a cabo en esa sesión, serán específicos para el usuario conectado. Esto genera la necesidad de conocer la identificación del usuario que se conecta.

Como se mencionó anteriormente, el lenguaje no permite que los applets utilicen los recursos de la máquina donde se ejecutan, impidiendo guardar la identificación del usuario en una variable del sistema o archivo temporal. Como todos los applets se ejecutan en una misma página web, corren en la misma máquina virtual de Java y pueden comunicarse entre sí, permiten solucionar el problema descripto.

Dos o más applets en una página web pueden comunicarse entre sí, pero pueden hacerlo de diferentes formas, aquí se presentan dos de los modelos de comunicación.

Comunicación entre Applets - Modelo I

En este modelo, cada applet puede invocar métodos de otro applet, debido a que están ejecutándose en el mismo contexto, es decir en la misma máquina virtual de Java. La idea es lograr que los applets entren en contacto entre sí y esto es posible a través de la clase AppletContext que tiene métodos para localizar applets por el nombre u obtener todos los applets que se encuentran ejecutándose en el mismo ambiente.

Una vez que dos applets se han comunicado, la comunicación es similar a la de cualquier objeto del sistema. Esto significa que pueden invocar los métodos del otro, compartir arreglos entre ellos o intercambiar información a través de pipes.

Un pipe de streams es un par de streams: uno es un stream de entrada y el otro es un stream de salida. Los datos escritos en el stream de entrada, son leídos desde el stream de salida al cual se encuentra conectado. Se puede utilizar este mecanismo para intercambiar datos entre dos applets o entre dos objetos del sistema. Un objeto crea el PipedInputStream y el PipedOutputStream, y luego pasa un extremo del pipe al otro objeto.

El objeto que posee el extremo de salida del stream puede escribir en el pipe, mientras que el objeto que posee el extremo de entrada puede leer los datos enviados.

Los pipes de streams son útiles para enviar mensajes en secuencia. A veces un objeto necesita indicarle al otro una serie de tareas que debe realizar en un determinado

orden. En este caso, le enviaría las tareas a realizar a través del PipedOutputStream y continuaría con sus propias tareas, mientras el objeto que realiza las tareas, lee los mensajes del PipedInputStream, realiza el trabajo y luego lee el próximo mensaje del pipe. De esta forma, se garantiza que los mensajes se lean en el orden que fueron enviados.

El applet escritor, debe realizar un loop para esperar a que el applet lector se convierta en un applet activo. El applet lector tiene un problema similar, tiene que esperar a que el escritor le envíe el extremo del pipe para comenzar la transmisión.

Comunicación entre Applets - Modelo II

Dado que los applets se ejecutan en la misma instancia de máquina virtual, todas las variables de clase estática pueden ser compartidas entre ellos. Esto significa que se pueden crear clases "singleton" que son compartidas por todos los applets. Un "singleton" es una clase que tiene una única instancia. Su constructor se encuentra oculto, impidiendo que otras clases pueden crear nuevas instancias de la clase singleton. Se accede a la instancia de clase a través del método instance() y se crea una clase singleton que actúa como un registro para todos los applets del entorno. Si el singleton se implementa como un observable, puede monitorear los applets que se encuentran activos. Esto permite a dos applets sincronizarse sin la necesidad de utilizar constantemente las técnicas de poolling para testear los applets que se registran como tales.

Por lo tanto se crea una clase AppletRegistry que permite registrar los applets. Esta clase utiliza una tabla para guardar los applets por nombre. Cada vez que se desea encontrar un applet, se llama al método findApplet con el nombre del applet que se está buscando:

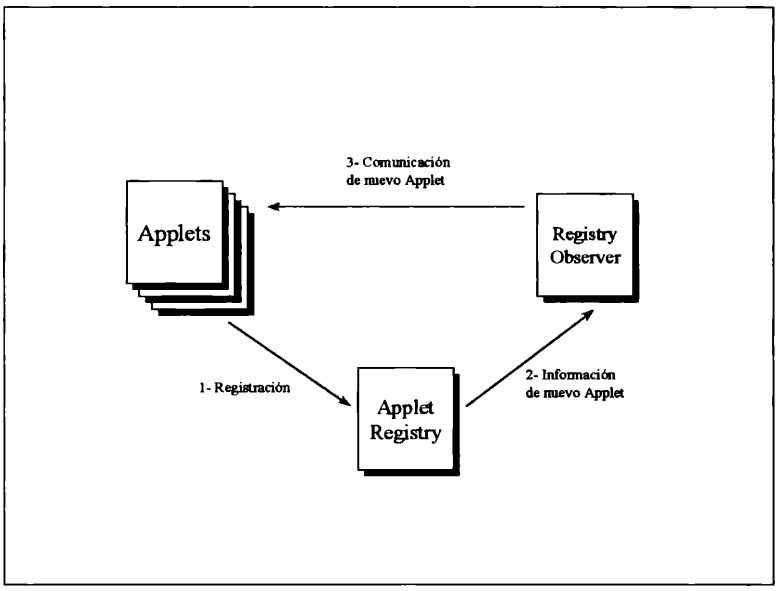
```
Applet findit = AppletRegistry.instance().findApplet("miApplet")
```

Cuando un applet comienza a ejecutarse, llama al método addApplet:

```
AppletRegistry.instance().addApplet("miApplet", this)
```

El AppletRegistry es una clase observable. Cada vez que un applet es agregado o removido, envía un update a las clases observers. Esto implica que cada objeto debe implementar la interfaz observer para enterarse cuando un applet se registra. El registry

envía una notificación cada vez que un applet es agregado o removido, utilizando un mecanismo de evento simple para enviar la notificación. El objeto pasado en el método update en la clase observer, es una instancia de la clase AppletRegistryEvent. Este objeto contiene información específica del evento (el nombre del applet, el applet y el evento de agregado o removido).



Modelo de Comunicación entre Applets

Modelo elegido

El primer modelo descripto presenta dos inconvenientes, el primero es la limitación en la longitud de los datos que pueden transmitirse a través del pipe (1 caracter) y el otro es que los applets utilizan constantemente las técnicas de poolling para testear qué applets se activan, disminuyendo la performance del sistema. Por esta razón se utilizó el segundo modelo de comunicación entre applets.

En Apliades encontramos clases que extienden Observer y clases que implementan Observable.

Resultaría engorroso y poco práctico que el usuario deba ingresar su identificación cada vez que un servicio necesita dicha información. La idea es que mientras dure la sesión de conexión, los datos del usuario activo puedan ser accedidos por cualquier servicio.

Ejemplo de comportamiento del modelo en el servicio Consultar Notas de Apliades

Dado que un usuario sólo puede obtener información acerca de sus propias notas, se debe conocer su identificación para poder efectuar la consulta. Aquí surge la necesidad de comunicación entre los applets conexión y consulta de notas.

El applet conexión, al ser el primer applet que se carga en el browser, crea la instancia del AppletRegistry y se agrega con los datos del usuario conectado. Cuando se ejecuta el servicio consulta de notas, el applet correspondiente se registra en el AppletRegistry, busca en dicho registro la instancia del applet conexión y toma de la misma la identificación del usuario conectado para ejecutar la consulta requerida.

A continuación se presenta la codificación de los applets y las clases involucradas en el ejemplo:

Clase AppletRegistry

El AppletRegistry implementa un registro de Applets que permite la localización entre los mismos.

```
import java.applet.Applet;
import java.util.*;
import com.sun.java.swing.*;

public class AppletRegistry extends Observable{
    protected static AppletRegistry registry;
    protected Hashtable applets;
    protected int nextUnique;

protected AppletRegistry() {
    applets = new Hashtable();
    nextUnique = 0;
}

public synchronized void addApplet(String name, JApplet newApplet){
    applets.put(name,newApplet);
    setChanged();
    notifyObservers(new AppletRegistryEvent(AppletRegistryEvent.ADD_APPLET, name,
    newApplet));
    }
```

```
public synchronized void addApplet(String name, JApplet newApplet, boolean uniqueName){
    if (!uniqueName && (applets.get(name) != null)){
            name = name + "<"+ nextUnique+">";
            nextUnique++;
            }
    applets.put(name, newApplet);
    setChanged();
    notifyObservers(new AppletRegistryEvent(AppletRegistryEvent.ADD_APPLET, name,
    newApplet));
    }

 public JApplet findApplet (String name){
    return(JApplet) applets.get(name);
    }
    public Enumeration getApplets(){
            return applets.elements();
            }

public synchronized static AppletRegistry instance(){
    if (registry == null){
            registry = new AppletRegistry();
            }
    return registry;
    }

public synchronized void removeApplet( JApplet applet){
    Enumeration e = applets.keys();
    while (e.hasMoreElements()){
      Object key = e.nextElement();
      if (applets.get(key) == applet){
            applets.remove(key);
            setChanged();
            notifyObservers(new AppletRegistryEvent(AppletRegistryEvent.REMOVE_APPLET,
            (String) key, applet));
            return;
             }
            }
}

public synchronized void removeApplet( String name){
```

```
        JApplet applet = (JApplet) applets.get(name);
        if (applet == null) return;
        applets.remove(name);
        setChanged();
        notifyObservers(new AppletRegistryEvent(AppletRegistryEvent.REMOVE_APPLET, name,
        applet));
    }
}
```

Clase AppletRegistryEvent

Permite crear un objeto que usará el AppletRegistry. Mantiene los datos del applet e indica si se agregó o quitó el applet dado.

```
import java.applet.Applet;
import com.sun.java.swing.*;
public class AppletRegistryEvent extends Object{
    public final static int ADD_APPLET = 1;
    public final static int REMOVE_APPLET = 2;
    public int id;
    public String appletName;
    public JApplet applet;

public AppletRegistryEvent(int id,String appletName, JApplet japplet) {
    this.id=id;
    this.appletName=appletName;
    this.applet=japplet;
    }
}
```

Clase Conexión

Permite iniciar la sesión de conexión de un usuario con Apliades.

```
import java.applet.*;
import java.awt.*;
import java.rmi.*;
import java.net.*;
import Servidor.Resultado;
import java.net.URL;
import java.sql.*;
import java.lang.*;
```

```
import com.sun.java.swing.*;
import java.io.*;
import java.util.*;

public class Conexion extends JApplet implements java.awt.event.ActionListener, Observer {
    private JPanel ivjJAppletContentPane = null;
    private JLabel ivjJLabel1 = null;
    private JLabel ivjJLabel11 = null;
    private JLabel ivjJLabel12 = null;
    private JComboBox ivjComboCat = null;
    private JPasswordField ivjPassClave = null;
    private JTextField ivjTFUsr = null;
    String vusuario;
    private int idcliente;
    String vmsg = "";
    private JOptionPane ivjJOptionPane1 = null;
    private JButton ivjBtAce = null;
    private JButton ivjBtCan = null;
    private java.util.Hashtable ht ;
    Servidor.SampServerIf obj;
    private JLabel ivjJLabel2 = null;

public Conexion() {
    super(); }
/*-----------------------------------------------------------------------------------------------------------------
Verifica que la Clave y la categoría correspondan al Usuario, y que éste exista para esa categoría
Si OK. entonces establece la conexión
-----------------------------------------------------------------------------------------------------------------*/
public void aceConexion() {
vmsg ="";
int resu = -1;
vusuario = getTFUsr().getText();
String clave = getPassClave().getText();
String sel = String.valueOf(getComboCat().getSelectedItem());
int idcat = ((Integer) ht.get(sel)).intValue();
if ((vusuario.equals("")) || (clave.equals(""))){
    vmsg = "Debe completar los Datos!";
} else {
    if (clave.equals(null)) {
    vmsg = "Debe ingresar la Clave!";
  } else {
                if(vusuario.equals(null) ){
```

```
                        vmsg = "Debe ingresar el Usuario!";
                    } else {
                        try {  resu = obj.esUsuValido(vusuario,clave,idcat);
                         } catch (RemoteException e1){
                       System.out.println("Error en la conexión RMI!");
                       System.out.println("Excepción "+e1+": \n"+e1.getMessage());
                     }//RemoteException e1
                    if(resu == -1){
                            vmsg ="El Usuario ingresado no se encuentra registrado!";
                    }
                    if(resu == -2){
                            vmsg ="La Clave ingresada es incorrecta!";
                    }
                    if(resu == -3){
                            vmsg ="La Categoría no corresponde al Usuario ingresado!";
                    }
                    if(resu == -4){
                            vmsg ="La Conexión no puede establecerse. Intente Nuevamente!";
                    }
                }
            }
        }
        if (vmsg.equals("")){
        try {  idcliente = obj.codUsuario(vusuario);
          } catch (RemoteException e2){
              System.out.println("Error en conexión RMI!");
              System.out.println("Exception "+e2+" caught: \n"+e2.getMessage());
                    }//RemoteException e2
            //Para Comunicación
            AppletRegistry.instance().addApplet("conexion",this);
            AppletRegistry.instance().addObserver(this);
        } else  {
            JOptionPane.showMessageDialog(null, vmsg);
            }
        }

        public void actionPerformed(java.awt.event.ActionEvent e) {
            if ((e.getSource() == getBtAce()) ) {
                    connEtoC1(e);
            }
```

```
    if ((e.getSource() == getBtCan()) ) {
            connEtoC2(e);
    }
}

private void connEtoC1(java.awt.event.ActionEvent arg1) {
    try {  this.aceConexion();
    } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
    }
}

private void connEtoC2(java.awt.event.ActionEvent arg1) {
    try { this.salir(arg1);
    } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
    }
}

public String getAppletInfo() {
    return "Usuario.Conexion"; }

private JButton getBtAce() {
    if (ivjBtAce == null) {
            try {   ivjBtAce = new JButton();
                    ivjBtAce.setName("BtAce");
                    ivjBtAce.setOpaque(false);
                    ivjBtAce.setText("");
                    ivjBtAce.setFocusPainted(false);
                    ivjBtAce.setHorizontalTextPosition(SwingConstants.CENTER);
                    ivjBtAce.setIcon(new ImageIcon("C:\\Tesis\\ace.JPG"));
                    ivjBtAce.setBorderPainted(false);
                    ivjBtAce.setBounds(90, 176, 47, 40);
                    ivjBtAce.setPressedIcon(new ImageIcon("C:\\Tesis\\aceo.JPG"));
            } catch (java.lang.Throwable ivjExc) {
                    handleException(ivjExc);
            }
    };
    return ivjBtAce;
}
```

```
private JButton getBtCan() {
    if (ivjBtCan == null) {
            try {   ivjBtCan = new JButton();
                    ivjBtCan.setName("BtCan");
                    ivjBtCan.setOpaque(false);
                    ivjBtCan.setText("");
                    ivjBtCan.setFocusPainted(false);
                    ivjBtCan.setHorizontalTextPosition(SwingConstants.CENTER);
                    ivjBtCan.setIcon(new ImageIcon("C:\\Tesis\\Can.JPG"));
                    ivjBtCan.setBorderPainted(false);
                    ivjBtCan.setBounds(143, 176, 47, 40);
                    ivjBtCan.setPressedIcon(new ImageIcon("C:\\Tesis\\Cano.JPG"));
            } catch (java.lang.Throwable ivjExc) {
                    handleException(ivjExc);
            }
    };
    return ivjBtCan;
}

private JComboBox getComboCat() {
    if (ivjComboCat == null) {
            try {   ivjComboCat = new JComboBox();
                    ivjComboCat.setName("ComboCat");
                    ivjComboCat.setToolTipText("Seleccionar Categoría de Usuario");
                    ivjComboCat.setBackground(new java.awt.Color(250,250,250));
                    ivjComboCat.setBounds(76, 108, 185, 23);
                    ivjComboCat.setForeground(new java.awt.Color(104,25,20));
            } catch (java.lang.Throwable ivjExc) {
                    handleException(ivjExc);
            }
    };
    return ivjComboCat; }

public int getidcliente() {
    return this.idcliente;
}

private JPanel getJAppletContentPane() {
    if (ivjJAppletContentPane == null) {
            try {   ivjJAppletContentPane = new JPanel();
```

```
                    ivjJAppletContentPane.setName("JAppletContentPane");
                    ivjJAppletContentPane.setBorder(new plaf.metal.Flush3DBorder());
                    ivjJAppletContentPane.setLayout(null);
                    ivjJAppletContentPane.setBackground(new java.awt.Color(190,190,190));
                    getJAppletContentPane().add(getComboCat(), getComboCat().getName());
                    getJAppletContentPane().add(getJLabel12(), getJLabel12().getName());
                    getJAppletContentPane().add(getJLabel1(), getJLabel1().getName());
                    getJAppletContentPane().add(getJLabel11(), getJLabel11().getName());
                    getJAppletContentPane().add(getBtAce(), getBtAce().getName());
                    getJAppletContentPane().add(getPassClave(), getPassClave().getName());
                    getJAppletContentPane().add(getBtCan(), getBtCan().getName());
                    getJAppletContentPane().add(getTFUsr(), getTFUsr().getName());
                    getJAppletContentPane().add(getJLabel2(), getJLabel2().getName());
            } catch (java.lang.Throwable ivjExc) {
                    handleException(ivjExc);
            }
    };
    return ivjJAppletContentPane;
}

private JLabel getJLabel1() {
    if (ivjJLabel1 == null) {
            try {   ivjJLabel1 = new JLabel();
                    ivjJLabel1.setName("JLabel1");
                    ivjJLabel1.setText("Usuario");
                    ivjJLabel1.setBounds(14, 32, 45, 15);
                    ivjJLabel1.setForeground(new java.awt.Color(159,3,24));
            } catch (java.lang.Throwable ivjExc) {
                    handleException(ivjExc);
            }
    };
    return ivjJLabel1;
}

private JLabel getJLabel11() {
    if (ivjJLabel11 == null) {
            try {   ivjJLabel11 = new JLabel();
                    ivjJLabel11.setName("JLabel11");
                    ivjJLabel11.setText("Clave");
                    ivjJLabel11.setBounds(14, 74, 45, 15);
```

```
                ivjJLabel11.setForeground(new java.awt.Color(159,3,24));
            } catch (java.lang.Throwable ivjExc) {
                handleException(ivjExc);
            }
        };
        return ivjJLabel11;
    }

    private JLabel getJLabel12() {
        if (ivjJLabel12 == null) {
            try {   ivjJLabel12 = new JLabel();
                ivjJLabel12.setName("JLabel12");
                ivjJLabel12.setText("Categoría");
                ivjJLabel12.setBounds(15, 117, 67, 15);
                ivjJLabel12.setForeground(new java.awt.Color(159,3,24));
            } catch (java.lang.Throwable ivjExc) {
                handleException(ivjExc);
            }
        };
        return ivjJLabel12;
    }

    private JLabel getJLabel2() {
        if (ivjJLabel2 == null) {
            try {   ivjJLabel2 = new JLabel();
                ivjJLabel2.setName("JLabel2");
                ivjJLabel2.setIcon(new ImageIcon("C:\\Tesis\\linlarga1.jpg"));
                ivjJLabel2.setText("");
                ivjJLabel2.setBounds(5, 158, 271, 10);
                ivjJLabel2.setHorizontalAlignment(SwingConstants.CENTER);
            } catch (java.lang.Throwable ivjExc) {
                handleException(ivjExc);
            }
        };
        return ivjJLabel2;
    }

    private JOptionPane getJOptionPane1() {
        if (ivjJOptionPane1 == null) {
            try {   ivjJOptionPane1 = new JOptionPane();
```

```
                ivjJOptionPane1.setName("JOptionPane1");
                ivjJOptionPane1.setBounds(335, 89, 65, 68);
            } catch (java.lang.Throwable ivjExc) {
                handleException(ivjExc);
            }
        };
        return ivjJOptionPane1;
    }

    private JPasswordField getPassClave() {
        if (ivjPassClave == null) {
            try {   ivjPassClave = new JPasswordField();
                ivjPassClave.setName("PassClave");
                ivjPassClave.setToolTipText("Ingresar Clave de Acceso");
                ivjPassClave.setBorder(new plaf.basic.BasicFieldBorder());
                ivjPassClave.setBackground(new java.awt.Color(250,250,250));
                ivjPassClave.setBounds(76, 69, 185, 19);
                ivjPassClave.setForeground(new java.awt.Color(104,25,20));
            } catch (java.lang.Throwable ivjExc) {
                handleException(ivjExc);
            }
        };
        return ivjPassClave;
    }

    private JTextField getTFUsr() {
        if (ivjTFUsr == null) {
            try {   ivjTFUsr = new JTextField();
                ivjTFUsr.setName("TFUsr");
                ivjTFUsr.setToolTipText("Ingresar Usuario");
                ivjTFUsr.setBorder(new plaf.basic.BasicFieldBorder());
                ivjTFUsr.setBackground(new java.awt.Color(250,250,250));
                ivjTFUsr.setBounds(76, 27, 185, 19);
                ivjTFUsr.setForeground(new java.awt.Color(104,25,20));
            } catch (java.lang.Throwable ivjExc) {
                handleException(ivjExc);
            }
        };
        return ivjTFUsr;
    }
```

```
private void handleException(Throwable exception) {
}

public synchronized void init() {
    try {       setName("Conexion");
                setSize(280, 230);
                setContentPane(getJAppletContentPane());
                initConnections();
                // Para registrarse en el AppletRegistry-----
                 AppletRegistry.instance().addApplet("conexion",this);
                 AppletRegistry.instance().addObserver(this);
                //-----------------------------------------
                try{  URL url= getDocumentBase();
                  String host= url.getHost();
                  if (host.length()==0) {
                            host = getParameter("host");
                            if (host==null)
                            {host="10.10.93.1";}
                            }
                  obj= (Servidor.SampServerIf)Naming.lookup("rmi://"+host+"/Server");
                  Resultado res = obj.listaCatUsr();
                  int cod_res=res.getres_op();
                  if (cod_res == -1) {
                             System.out.println("No hay datos de Categorías!");
                             };
                  // Cargamos el pop up de Categorías
                    ht= new java.util.Hashtable();
                    ht= (java.util.Hashtable) res.getobj();
                    try{ for (java.util.Enumeration e = ht.elements(), k= ht.keys(); e.hasMoreElements(); ){
                                    getComboCat().addItem( k.nextElement());
                                    }
                            } catch (java.util.NoSuchElementException e2){ }//RemoteException e2
                    repaint();
                   }
            catch (RemoteException e1)
             {   System.out.println("Error con la conexión RMI!");
                     System.out.println("Excepción "+e1+": \n"+e1.getMessage());
                   }//RemoteException e1
            catch (Exception e)
```

```
                {System.out.println("Excepción"+e+": \n"+e.getMessage());
                    showStatus("Exception");
            }//Exception e
        } catch (java.lang.Throwable ivjExc) {
        handleException(ivjExc);
        }
}

private void initConnections() {
    getBtAce().addActionListener(this);
    getBtCan().addActionListener(this);
}

public void salir(java.awt.event.ActionEvent actionEvent) {
    System.exit(0);
}

public synchronized void update (Observable obs, Object ob) {
// Este método es invocado con el notifyObservers
System.out.println("update de Conexion");
  }
}
```

Consultar Notas

Permite a un alumno consultar sus notas de una materia determinada.

```
import java.applet.*;
import java.awt.*;
import java.rmi.*;
import java.net.*;
import Servidor.Resultado;
import java.net.URL;
import java.sql.*;
import java.lang.*;
import java.io.*;
import com.sun.java.swing.*;
import java.util.*;

public class CNotas extends JApplet implements java.awt.event.ActionListener, Observer {
    private JPanel ivjJAppletContentPane = null;
    private JLabel ivjJLabel1 = null;
```

```
    private JLabel ivjJLabel11 = null;
    private JLabel ivjJLabel21211 = null;
    private JComboBox ivjComboCurso = null;
    private JComboBox ivjComboMat = null;
    private JButton ivjBtimp = null;
    private JButton ivjBtSoliciana = null;
    private JScrollPane ivjScrollCalif = null;
    private JTextArea ivjTextACalif = null;
    private java.util.Hashtable ht ;
    private java.util.Hashtable htMat;
    private java.util.Hashtable htCalif;
    Servidor.SampServerIf obj;
    int idcliente ;
    private JOptionPane ivjJOptionPane1 = null;

public CNotas() {
    super();
}

public void actionPerformed(java.awt.event.ActionEvent e) {
    if ((e.getSource() == getComboCurso()) ) {
            car_Mat(e);
    }
    if ((e.getSource() == getComboCurso()) ) {
            connEtoC1(e);
    }
    if ((e.getSource() == getComboMat()) ) {
            connEtoC2(e);
    }
    if ((e.getSource() == getBtSoliciana()) ) {
            connEtoC3(e);
    }
}

/*---------------------------------------------------------------------
 * Carga Notas de Materia elegida
---------------------------------------------------------------------*/
public void calif(java.awt.event.ActionEvent actionEvent) {
  getTextACalif().selectAll();
  getTextACalif().cut();
```

```
        String sel = String.valueOf(getComboMat().getSelectedItem());
        int idmat = ((Integer) htMat.get(sel)).intValue();
        java.util.Date vfecha;
        int vidtipotrabajo;
        int vnota;
        String vnom_tipo_trabajo;
        String vtitulo;
    try{  Resultado res = obj.notasMatAlumno(idmat,idcliente);
                int cod_res = res.getres_op();
                if (cod_res == -1) {
                        System.out.println("No hay datos!");   };
                // Cargamos el pop up de Cursos
                  htCalif= new java.util.Hashtable();
                  htCalif= (java.util.Hashtable) res.getobj();
                  String sfecha;
                   try{for (java.util.Enumeration e = htCalif.elements(), k= htCalif.keys();
                                                e.hasMoreElements(); ){
                           Servidor.NotasAlumno calific = (Servidor.NotasAlumno) k.nextElement();
                           vfecha=calific.getfecha();
                           java.text.SimpleDateFormat formatter = new java.text.SimpleDateFormat
                                                        ("dd/MM/yyyy");
                           sfecha = formatter.format(vfecha);
                           vnota=calific.getnota();
                           vnom_tipo_trabajo=calific.getnom_tipo_trabajo ();
                           vtitulo = calific.gettitulo();
                           getTextACalif().append(String.valueOf(" "+vnom_tipo_trabajo+"
                          "+vtitulo+"\n"));
                           getTextACalif().append(String.valueOf(" NOTA: "+vnota+" FECHA:
                           "+sfecha+"\n"+"\n"));
                                                        }
                     } catch (java.util.NoSuchElementException e2)
            {   ;
                    }//RemoteException e2
    }
          catch (RemoteException e1)
           {   System.out.println("Error con la conexión RMI!");
                  System.out.println("Excepción "+e1+": \n"+e1.getMessage());
                 }//RemoteException e1
        repaint();
        return; }
```

```
public void car_Mat(java.awt.event.ActionEvent actionEvent) {
    getComboMat().removeAllItems();
    String sel = String.valueOf(getComboCurso().getSelectedItem());
    int idcurso = ((Integer) ht.get(sel)).intValue();
try{
     Resultado res = obj.materiasdelAlumno(idcurso,idcliente);
                int cod_res = res.getres_op();
                if (cod_res == -1) {
                        System.out.println("No hay datos!");
                         };
                // Cargamos el pop up de Cursos
                  htMat= new java.util.Hashtable();
                  htMat= (java.util.Hashtable) res.getobj();
                   try{  for (java.util.Enumeration e = htMat.elements(),
                         k= htMat.keys(); e.hasMoreElements(); ){
                            getComboMat().addItem( k.nextElement());
                            }
                  } catch (java.util.NoSuchElementException e2)
          {   ;
                   }//RemoteException e2
}
         catch (RemoteException e1)
          {   System.out.println("Error en la conexión RMI!");
                 System.out.println("Excepción "+e1+": \n"+e1.getMessage());
                }//RemoteException e1
      repaint();
      return;
}

private void connEtoC1(java.awt.event.ActionEvent arg1) {
    try { this.car_Mat(arg1);
    } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
    }
}

private void connEtoC2(java.awt.event.ActionEvent arg1) {
    try { this.calif(arg1);
    } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
```

```
    }
}

private void connEtoC3(java.awt.event.ActionEvent arg1) {
    try { this.solicitarAnalitico();
    } catch (java.lang.Throwable ivjExc) {
            handleException(ivjExc);
    }
}

public String getAppletInfo() {
    return "Usuario.Notas";
}

private JButton getBtimp() {
    if (ivjBtimp == null) {
            try {   ivjBtimp = new JButton();
                    ivjBtimp.setName("Btimp");
                    ivjBtimp.setToolTipText("Imprimir");
                    ivjBtimp.setOpaque(false);
                    ivjBtimp.setText("");
                    ivjBtimp.setFocusPainted(false);
                    ivjBtimp.setHorizontalTextPosition(SwingConstants.CENTER);
                    ivjBtimp.setIcon(new ImageIcon("C:\\TESIS\\Imp190.JPG"));
                    ivjBtimp.setBorderPainted(false);
                    ivjBtimp.setBounds(441, 15, 44, 41);
                    ivjBtimp.setPressedIcon(new ImageIcon("C:\\TESIS\\Imp190o.JPG"));
            } catch (java.lang.Throwable ivjExc) {
                    handleException(ivjExc);
            }
    };
    return ivjBtimp;
}

private JButton getBtSoliciana() {
    if (ivjBtSoliciana == null) {
            try {   ivjBtSoliciana = new JButton();
                    ivjBtSoliciana.setName("BtSoliciana");
                    ivjBtSoliciana.setToolTipText("Enviar Pedido de Analítico");
                    ivjBtSoliciana.setOpaque(false);
```

```
                ivjBtSoliciana.setText("");
                ivjBtSoliciana.setFocusPainted(false);
                ivjBtSoliciana.setHorizontalTextPosition(SwingConstants.CENTER);
                ivjBtSoliciana.setIcon(new ImageIcon("C:\\Tesis\\solici.JPG"));
                ivjBtSoliciana.setBorderPainted(false);
                ivjBtSoliciana.setBounds(333, 257, 158, 31);
                ivjBtSoliciana.setPressedIcon(new ImageIcon("C:\\Tesis\\Solicio.JPG"));
            } catch (java.lang.Throwable ivjExc) {
                handleException(ivjExc);
            }
        };
        return ivjBtSoliciana;
    }

    private JComboBox getComboCurso() {
        if (ivjComboCurso == null) {
            try {   ivjComboCurso = new JComboBox();
                ivjComboCurso.setName("ComboCurso");
                ivjComboCurso.setBackground(new java.awt.Color(250,250,250));
                ivjComboCurso.setBounds(11, 25, 204, 23);
                ivjComboCurso.setForeground(new java.awt.Color(104,25,20));
            } catch (java.lang.Throwable ivjExc) {
                handleException(ivjExc);
            }
        };
        return ivjComboCurso;
    }

    private JComboBox getComboMat() {
        if (ivjComboMat == null) {
            try {   ivjComboMat = new JComboBox();
                ivjComboMat.setName("ComboMat");
                ivjComboMat.setBackground(new java.awt.Color(250,250,250));
                ivjComboMat.setBounds(227, 25, 204, 23);
                ivjComboMat.setForeground(new java.awt.Color(104,25,20));
            } catch (java.lang.Throwable ivjExc) {
                handleException(ivjExc);
            }
        };
        return ivjComboMat; }
```

```
private JPanel getJAppletContentPane() {
    if (ivjJAppletContentPane == null) {
            try {   ivjJAppletContentPane = new JPanel();
                    ivjJAppletContentPane.setName("JAppletContentPane");
                    ivjJAppletContentPane.setBorder(new plaf.metal.Flush3DBorder());
                    ivjJAppletContentPane.setLayout(null);
                    ivjJAppletContentPane.setBackground(new java.awt.Color(190,190,190));
                    getJAppletContentPane().add(getComboCurso(),
                    getComboCurso().getName());
                    getJAppletContentPane().add(getJLabel11(), getJLabel11().getName());
                    getJAppletContentPane().add(getComboMat(), getComboMat().getName());
                    getJAppletContentPane().add(getJLabel21211(), getJLabel21211().getName());
                    getJAppletContentPane().add(getBtSoliciana(), getBtSoliciana().getName());
                    getJAppletContentPane().add(getBtimp(), getBtimp().getName());
                    getJAppletContentPane().add(getScrollCalif(), getScrollCalif().getName());
                    getJAppletContentPane().add(getJLabel1(), getJLabel1().getName());
            } catch (java.lang.Throwable ivjExc) {
                    handleException(ivjExc);
            }
    };
    return ivjJAppletContentPane;
}

private JLabel getJLabel1() {
    if (ivjJLabel1 == null) {
            try {   ivjJLabel1 = new JLabel();
                    ivjJLabel1.setName("JLabel1");
                    ivjJLabel1.setText("Carrera - Curso");
                    ivjJLabel1.setBounds(12, 8, 95, 15);
                    ivjJLabel1.setForeground(new java.awt.Color(159,3,24));
            } catch (java.lang.Throwable ivjExc) {
                    handleException(ivjExc);
            }
    };
    return ivjJLabel1;
}

private JLabel getJLabel11() {
    if (ivjJLabel11 == null) {
            try {   ivjJLabel11 = new JLabel();
```

```
                ivjJLabel11.setName("JLabel11");
                ivjJLabel11.setText("Materia");
                ivjJLabel11.setBounds(227, 8, 51, 15);
                ivjJLabel11.setForeground(new java.awt.Color(159,3,24));
            } catch (java.lang.Throwable ivjExc) {
                handleException(ivjExc);
            }
        };
        return ivjJLabel11;
    }

    private JLabel getJLabel21211() {
        if (ivjJLabel21211 == null) {
            try {   ivjJLabel21211 = new JLabel();
                ivjJLabel21211.setName("JLabel21211");
                ivjJLabel21211.setText("Calificaciones");
                ivjJLabel21211.setBounds(12, 77, 112, 15);
                ivjJLabel21211.setForeground(new java.awt.Color(46,60,149));
            } catch (java.lang.Throwable ivjExc) {
                handleException(ivjExc);
            }
        };
        return ivjJLabel21211;
    }

    private JOptionPane getJOptionPane1() {
        if (ivjJOptionPane1 == null) {
            try {   ivjJOptionPane1 = new JOptionPane();
                ivjJOptionPane1.setName("JOptionPane1");
                ivjJOptionPane1.setBounds(542, 134, 75, 65);
            } catch (java.lang.Throwable ivjExc) {
                handleException(ivjExc);
            }
        };
        return ivjJOptionPane1;
    }

    private JScrollPane getScrollCalif() {
        if (ivjScrollCalif == null) {
            try {   ivjScrollCalif = new JScrollPane();
```

```
                    ivjScrollCalif.setName("ScrollCalif");
                    ivjScrollCalif.setBorder(new plaf.basic.BasicFieldBorder());
                    ivjScrollCalif.setBounds(12, 93, 478, 150);
                    getScrollCalif().setViewportView(getTextACalif());
                } catch (java.lang.Throwable ivjExc) {
                    handleException(ivjExc);
                }
    };
    return ivjScrollCalif;
}

private JTextArea getTextACalif() {
    if (ivjTextACalif == null) {
        try {   ivjTextACalif = new JTextArea();
                ivjTextACalif.setName("TextACalif");
                ivjTextACalif.setBorder(new plaf.basic.BasicMarginBorder());
                ivjTextACalif.setBackground(java.awt.Color.lightGray);
                ivjTextACalif.setBounds(0, 0, 11, 19);
                ivjTextACalif.setForeground(new java.awt.Color(67,59,96));
                ivjTextACalif.setEditable(false);
        } catch (java.lang.Throwable ivjExc) {
                handleException(ivjExc);
        }
    };
    return ivjTextACalif;
}

private void handleException(Throwable exception) {
}

public synchronized void init() {
    try {   setName("Notas");
            setSize(500, 300);
            setContentPane(getJAppletContentPane());
            initConnections();
            // Para Conexión
            Conexion app;
            AppletRegistry.instance().addApplet("cnotas",this);
            AppletRegistry.instance().addObserver(this);
            app= (Conexion) AppletRegistry.instance().findApplet("conexion");
```

```
                idcliente = app.getidcliente();
                try{  URL url= getDocumentBase();
                 String host= url.getHost();
                 if (host.length()==0) {
                        host = getParameter("host");
                        if (host==null)
                        {host="10.10.93.1";}
                        }
                 obj= (Servidor.SampServerIf)Naming.lookup("rmi://"+host+"/Server");
                 Resultado res = obj.listaCursoPosibles(idcliente);
                 int cod_res=res.getres_op();
                 if (cod_res == -1) {
                         System.out.println("No hay datos de Cursos!");
                         };
                 // Cargamos el pop up de Cursos
                  ht= new java.util.Hashtable();
                  ht= (java.util.Hashtable) res.getobj();
                  try{   for (java.util.Enumeration e = ht.elements(), k= ht.keys();
                        e.hasMoreElements(); ){
                                getComboCurso().addItem( k.nextElement());
                                }
                         } catch (java.util.NoSuchElementException e2){ }//RemoteException e2
                  repaint();
                 }
            catch (RemoteException e1)
             {   System.out.println("Error en la conexión RMI!");
                    System.out.println("Excepción "+e1+" \n"+e1.getMessage());
                  }//RemoteException e1

            catch (Exception e)
                {System.out.println("Exception "+e+" caught: \n"+e.getMessage());
                  showStatus("Exception");
            }//Exception e

        } catch (java.lang.Throwable ivjExc) {
                handleException(ivjExc);
        }
    }

    private void initConnections() {
```

```
    getComboCurso().addActionListener(this);
    getComboMat().addActionListener(this);
    getBtSoliciana().addActionListener(this);
}

public void solicitarAnalitico() {
int resu = -1;
String vmsg;
String sel = String.valueOf(getComboCurso().getSelectedItem());
int idcurso = ((Integer) ht.get(sel)).intValue();

try {  resu = obj.registrarSolicAnalitico(idcliente, idcurso);
   } catch (RemoteException e1){
         System.out.println("Error con la conexión RMI!");
               System.out.println("Excepción "+e1+": \n"+e1.getMessage());
               }//RemoteException e1
               if(resu == -1){
                        vmsg ="El pedido no se ha podido realizar. Intente Nuevamente";
                  }else{
            vmsg ="El pedido se ha realizado exitosamente!";
                  }
JOptionPane.showMessageDialog(null, vmsg);
}

public synchronized void update(Observable obs, Object ob) {
Conexion app;
if (!( ob instanceof AppletRegistryEvent)) return;
   AppletRegistryEvent evt  = (AppletRegistryEvent) ob;
 if (evt.appletName.equals("conexion")) {
     app= (Conexion) evt.applet;
    idcliente = app.getidcliente();
   }
}
}
```

Nota: Las porciones de código marcadas con el color bordó identifican la comunicación entre applets.

Interfaz Gráfica con el Usuario

GUI - INTERFAZ GRÁFICA CON EL USUARIO

DISEÑO DE INTERFAZ

Con el transcurso del tiempo la interfaz con el usuario ha ido tomando un lugar más relevante en el desarrollo de las aplicaciones, pasando de ser rígidas y orientadas a las necesidades del programador, a interfaces amigables, orientadas a las necesidades del usuario.

Considerando importante la interacción hombre - máquina, para el desarrollo de la aplicación, tuvimos en cuenta no sólo la programación, sino también la construcción de un sistema para producir una interfaz dinámica, ágil y funcional.

Nuestra idea inicial fue construir la interfaz utilizando elementos (botones, fondos) publicados en Internet, pero nada de lo que encontrábamos nos permitía volcar la idea de lo que queríamos transmitir. Los botones nos parecían duros, poco novedosos, e incluso en algunos casos el sistema se encontraba incompleto (por ejemplo faltaba la representación del botón imprimir). Decidimos, entonces, crear nuestro propio sistema para la construcción de la interfaz, utilizando como herramienta el Adobe Photoshop.

Aunque sólo programamos los servicios disponibles para los alumnos, nuestra propuesta de diseño de interfaz involucra a todos los perfiles de usuario. Cada perfil se identifica por su propio código cromático, utilizado en fondos, títulos y botones de servicios.

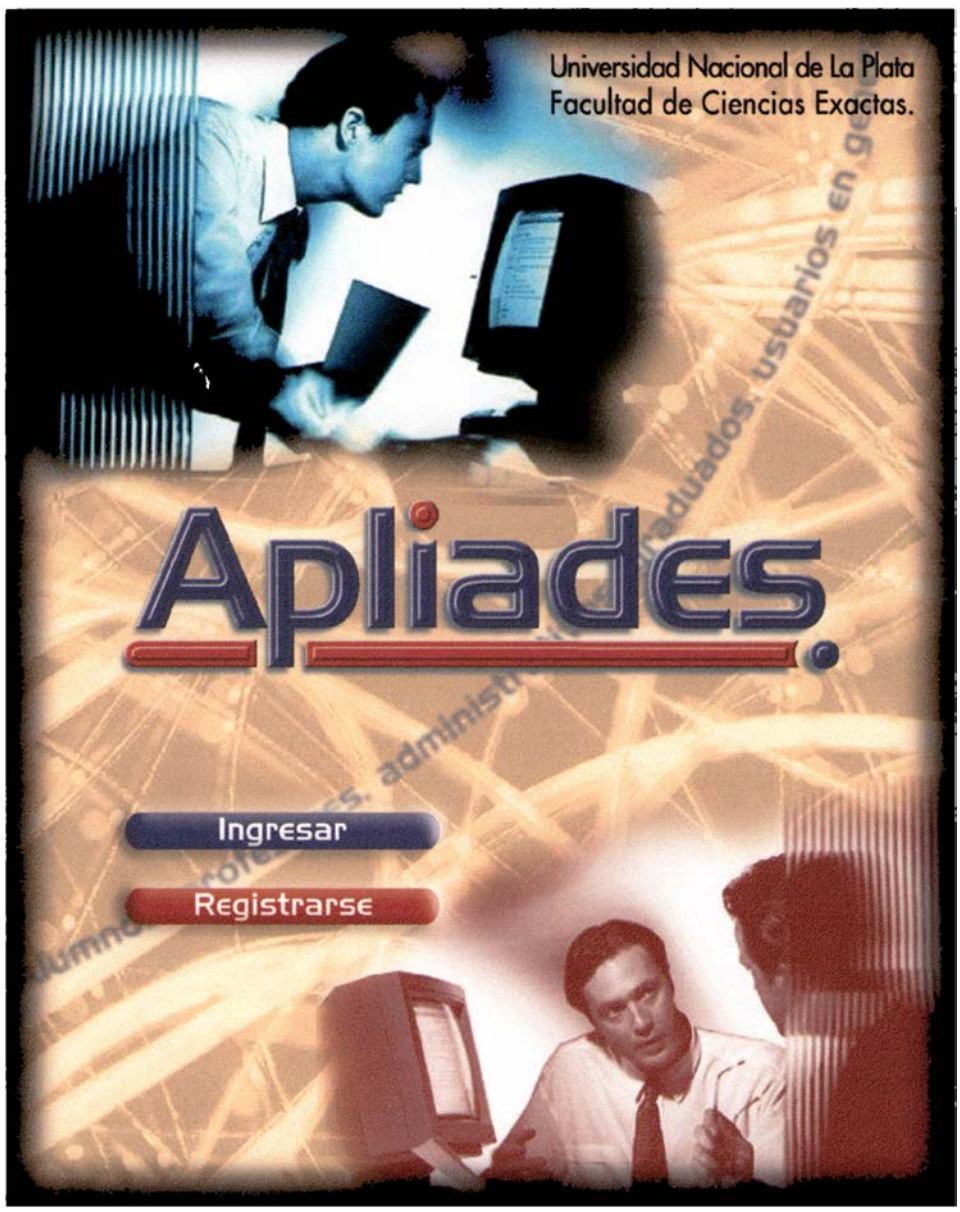
En cuanto a los botones se han usado botones con íconos y botones con texto. Elegimos dibujarlos con formas libres, de bordes redondeados, dándoles relieve y representación de luz y de sombra, pues consideramos que con estos efectos los botones darían más flexibilidad y ductilidad a la interfaz con el usuario.

PÁGINA PRINCIPAL DE APLIADES

Evolución de la página principal

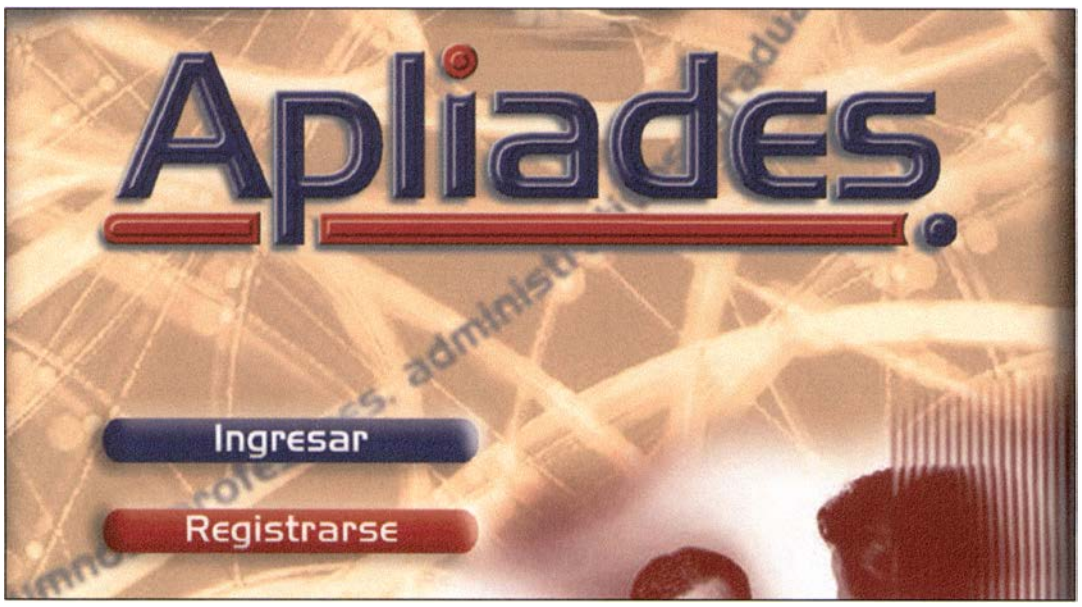
Al considerar la importancia de la página de presentación de la aplicación, realizamos varios diseños de dicha página hasta llegar a la que adoptamos definitivamente.

El primer diseño planteado fue el siguiente:



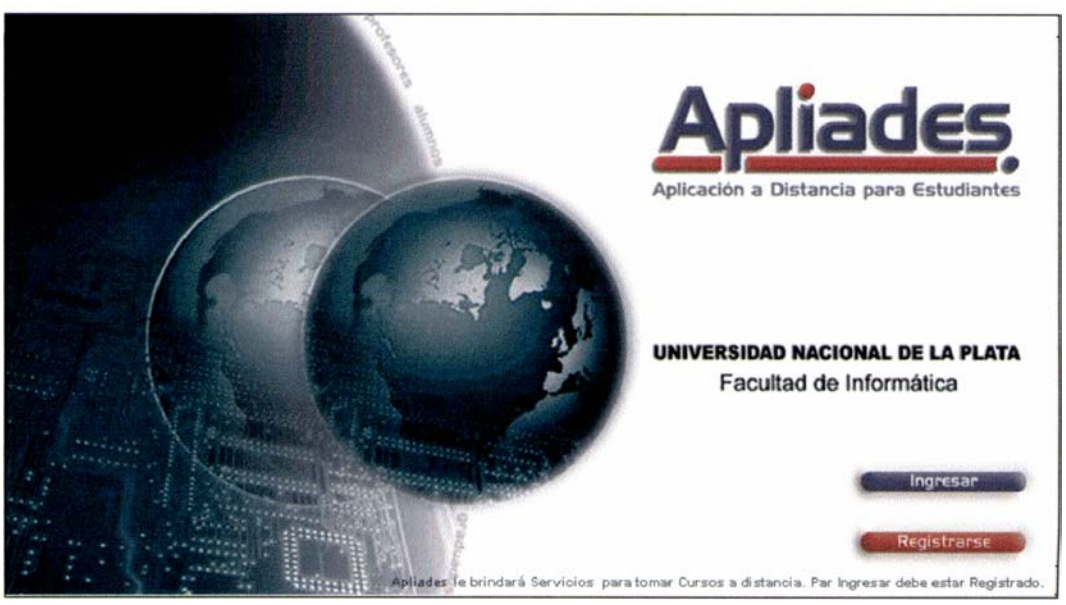
Este diseño lo descartamos pues transmitía una imagen de aplicación comercial y no educativa.

Luego decidimos modificar la portada anterior para ver si de este modo cambiábamos la imagen que se transmitía. Obtuvimos la siguiente página de presentación:

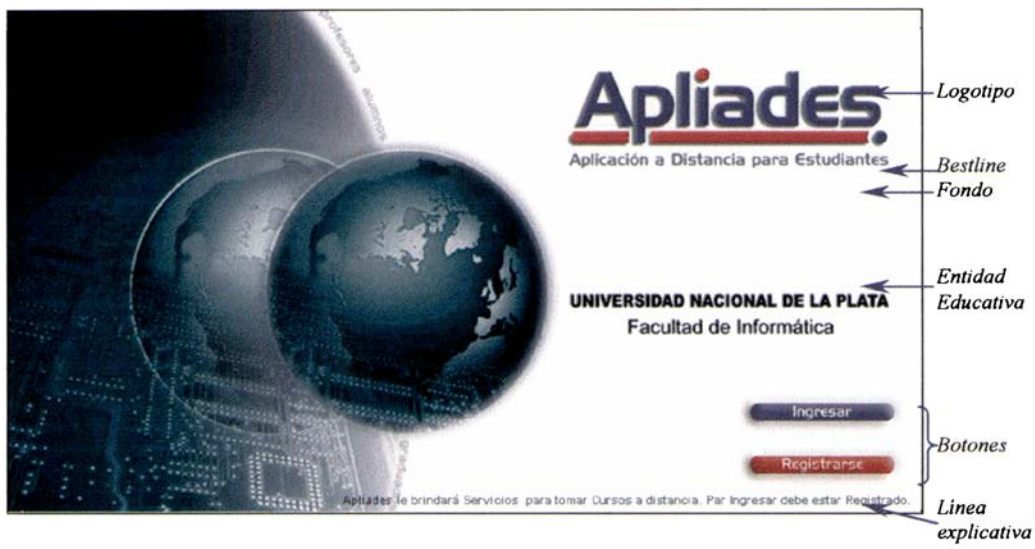


La página obtenida nos resultaba interesante, pero aún así no terminaba de representar lo que deseábamos comunicar.

Por último decidimos hacer un diseño totalmente distinto, en el cual logramos manifestar el hecho de que Apliades es una aplicación abierta a los usuarios del mundo con un trasfondo educativo. Finalmente, la página de presentación de Apliades quedó como se muestra a continuación:



Seguidamente se describen los elementos que componen la página de presentación. En esta se encuentran el logotipo, el fondo, la entidad educativa, los botones que permiten ingresar a las páginas de servicios o registrarse como usuario de Apliades y una línea con una breve explicación de la funcionalidad de la aplicación.



El logotipo de la aplicación se encuentra formado por el nombre que ésta fue denominada. Debajo del mismo aparece un bestline que indica la funcionalidad de la aplicación.

La escala cromática utilizada para la composición del logotipo es la misma que la que se empleo para nomenclar los datos de entrada y de salida que intervienen en cada uno de los applets de la aplicación.

Los botones que intervienen presentan los mismos colores que el logotipo, y son los que permiten acceder a las páginas de servicios, según el perfil de acceso, o registrarse como usuario de la aplicación.

	Permite ingresar a los servicios de Apliades
	Permite registrarse como usuario

PÁGINAS PARA EL INGRESO Y LA REGISTRACIÓN DE USUARIOS

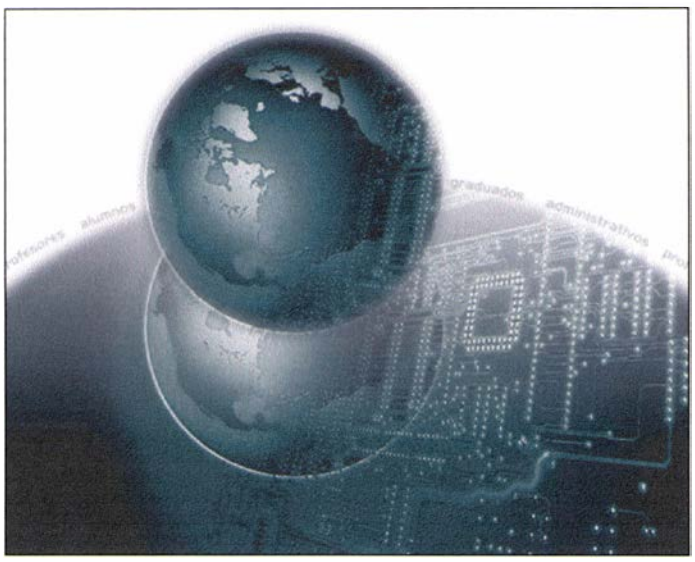
Estas páginas presentan el mismo aspecto para todos los perfiles de usuario, y contienen el mismo fondo, el título con el mismo aspecto, el botón de ayuda, el logotipo Apliades y el applet correspondiente a cada uno de los servicios.

Por ejemplo la página de ingreso a la aplicación tiene se presenta como sigue:

El botón ayuda se presenta con la misma apariencia a lo largo de toda la aplicación, independientemente del perfil de usuario y del servicio que se esté ejecutando.

Botón de Ayuda

Para estas páginas se utilizó un fondo diferente del presentado para las páginas de servicios de todos los perfiles.

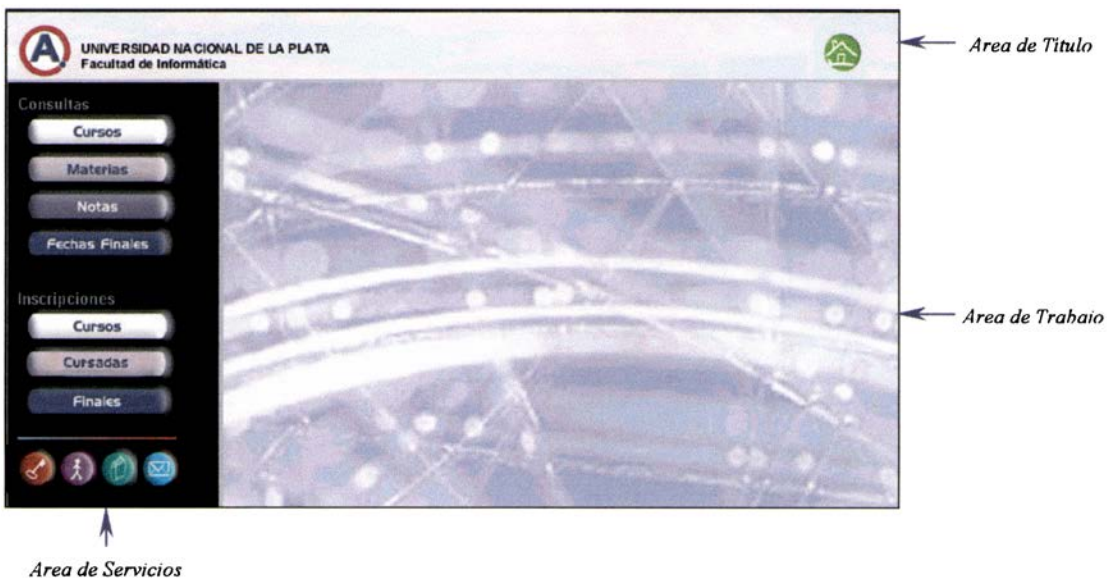


En cuanto a los títulos de las páginas de ingreso y de registración, se aplicaron efectos y colores diferentes a los de las páginas de servicios, esto se debe a que el fondo de sobre el cual se presentan es oscuro, impidiendo la visualización de las sombras. Sin embargo, la apariencia final lograda para ambos títulos es la misma en todas las páginas.

El título de la página de ingreso es el siguiente:

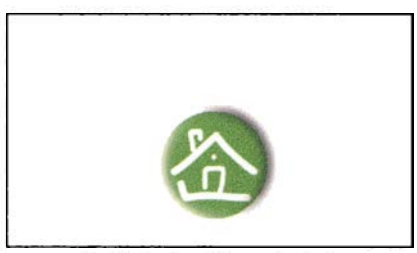
PÁGINA DE SERVICIOS

Areas que la componen



Area de título

Se encuentra en la parte superior de la página de servicios. Contiene el isotipo de la aplicación, el nombre de la entidad educativa y el botón que permite ir a la página principal.



Botón ir a la página principal

Además del logotipo, se ha realizado una aplicación del mismo con el fin de facilitar el uso y la visualización de Apliades. Al utilizar Ⓐ se evita perder el contexto de la indentidad de la aplicación a lo largo de la misma, sin sobrecargar las páginas.

Isotipo

Area de servicios

Se encuentra en la parte izquierda de la página. Contiene los botones que permiten el acceso a los servicios del perfil con el cual se ingresó a la aplicación.

En el perfil alumno intervienen botones de consulta y botones de inscripción, además de los botones genéricos, que son aquellos que se encuentran disponibles para todos los perfiles de usuario.

Los botones específicos y los genéricos están divididos por una línea en degradé en la gama de los colores que forman el logotipo.

Botones de consulta

Cursos	Permite la consulta de cursos o carreras
Materias	Permite la consulta de materias
Notas	Permite la consulta de notas
Fechas Finales	Permite la consulta de fechas de exámenes finales

Botones de inscripción

Cursos	Permite la inscripción en cursos o carreras
Cursadas	Permite la inscripción en cursadas
Finales	Permite la inscripción a exámenes finales

Botones de servicios generales

	Modificación de clave de usuario
	Modificación de datos personales
	Consulta de monografías
	Correo electrónico

Area de trabajo

Se encuentra en el lado derecho de la página, ocupando la mayor parte de la misma. Contiene el fondo que identifica el perfil de usuario, el botón de ayuda, el título del servicio y el applet correspondiente al servicio elegido.

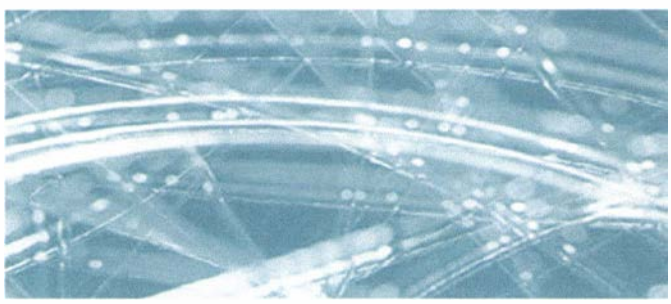
El botón de ayuda  se encuentra disponible en todas las páginas de servicios y presenta la misma apariencia, independientemente del perfil de acceso.

Los títulos de los servicios se construyeron utilizando la misma tipografía y los mismos efectos para todos los perfiles de usuario, aunque la escala cromática utlizada se corresponde con la que se utilizó en la construcción del fondo del perfil correspondiente.

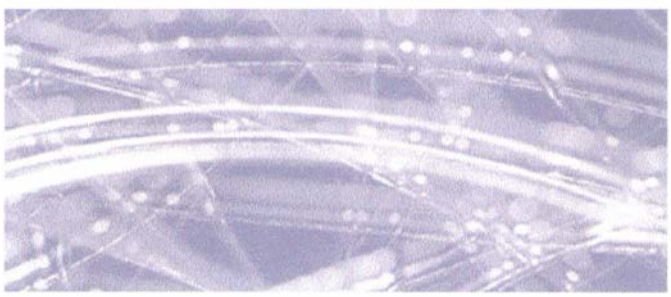
Por ejemplo en el perfil alumno los títulos se presentan con el aspecto que se muestra a continuación:

Si observamos los títulos del perfil usuario general, se verán como el que sigue:

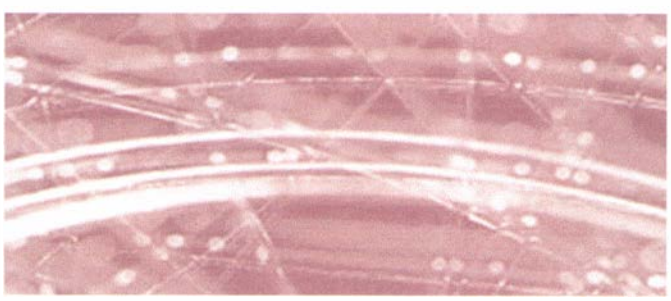
Los fondos, utilizados para la identificación de perfiles de usuario son los siguientes:



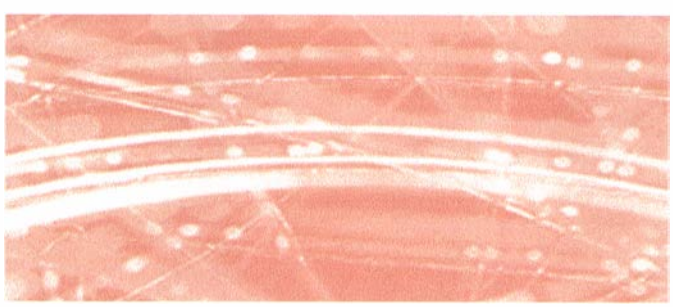
Profesor



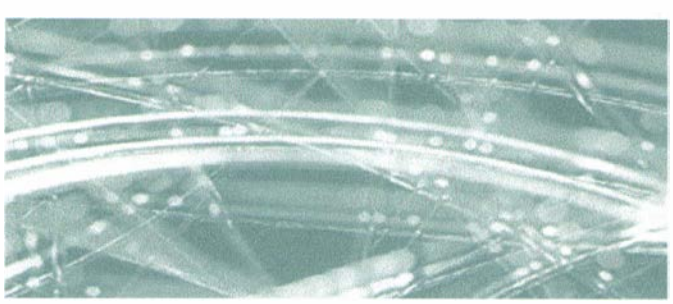
Alumno



Graduado

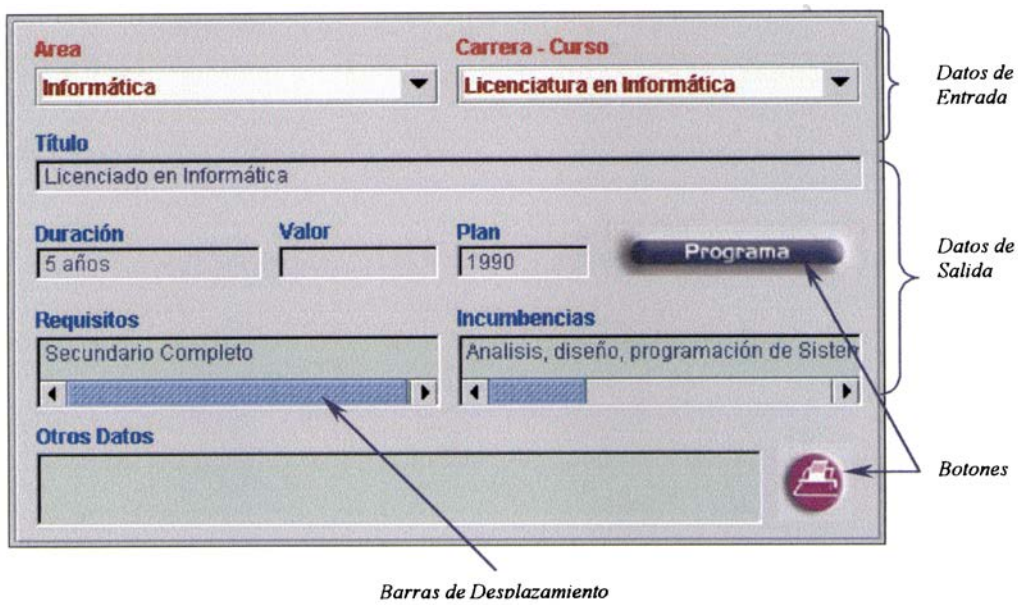


Administrativo



Usuario General

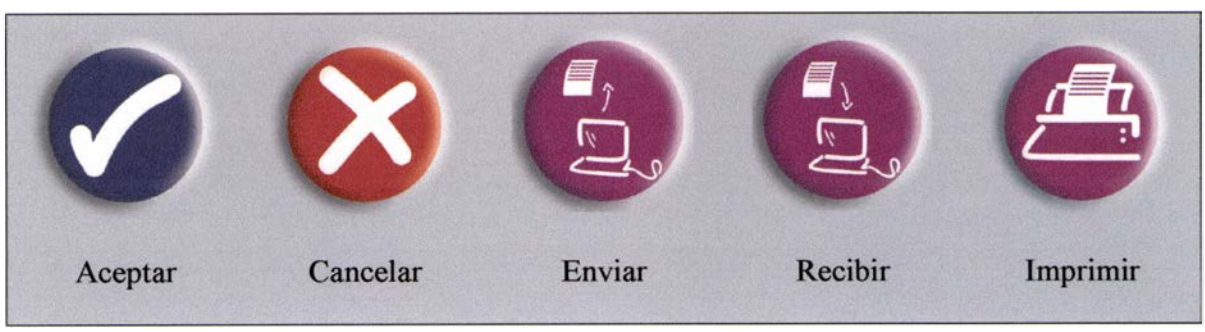
Los applets presentan la siguiente apariencia:



Con respecto a los colores, se ha utilizado la gama del azul para representar los datos de salida; la del rojo para los datos de entrada y la del gris para los fondos de los objetos y para los del applet mismo. Las barras de desplazamiento son de color lila.

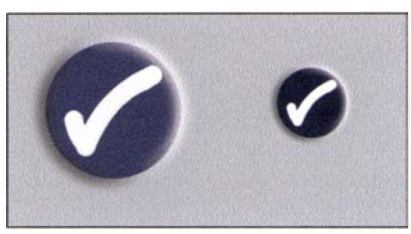
En cuanto a los botones, como se mencionó anteriormente, se han usado botones con íconos y botones con texto.

Los botones icónicos que se utilizaron dentro de los diferentes applets son los siguientes:

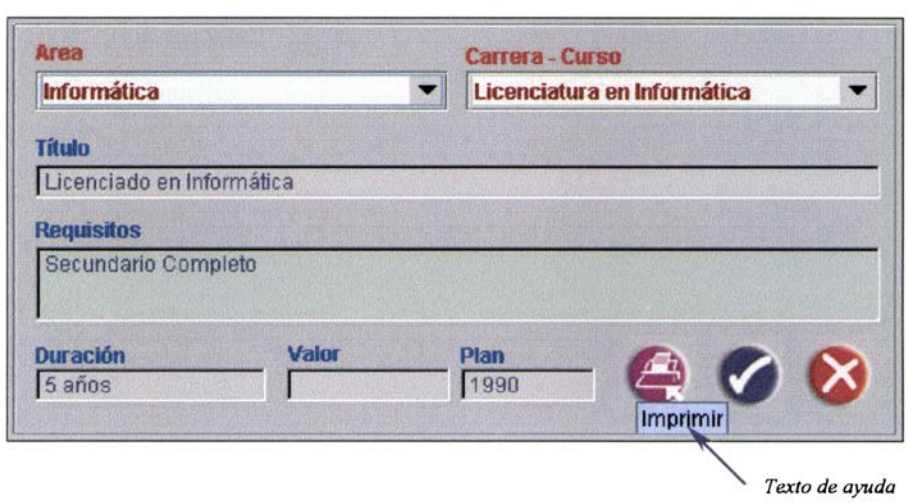


Algunos de los botones con texto utilizados en los applets son los siguientes:

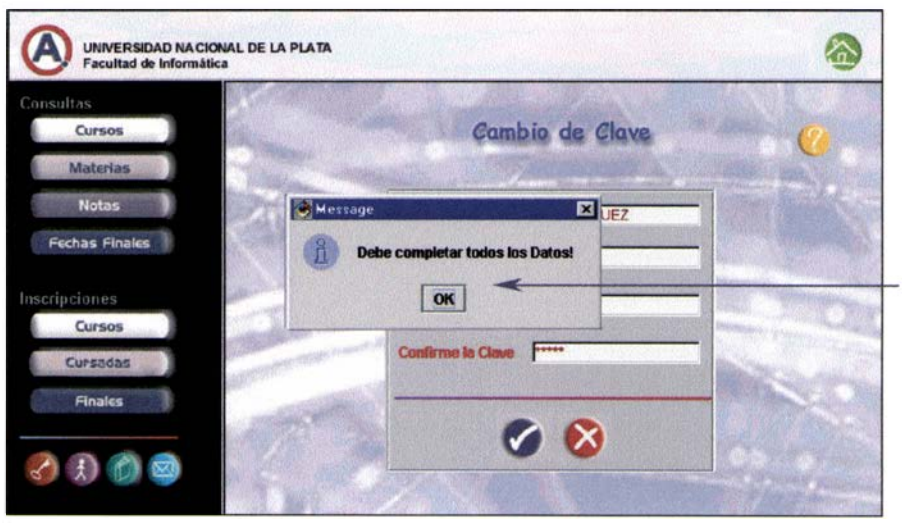
Al pulsar un botón dentro de un applet, se produce un efecto de "rebote". Este efecto fue logrado intercambiando dos imágenes con el mismo ícono, pero de color y tamaño diferentes. Por ejemplo el botón aceptar tiene asociados los siguientes dibujos:



Además de la página de ayuda que posee cada servicio, los objetos que se encuentran dentro de los applets muestran un texto de ayuda al posicionar el mouse sobre ellos. Como se muestra a continuación:



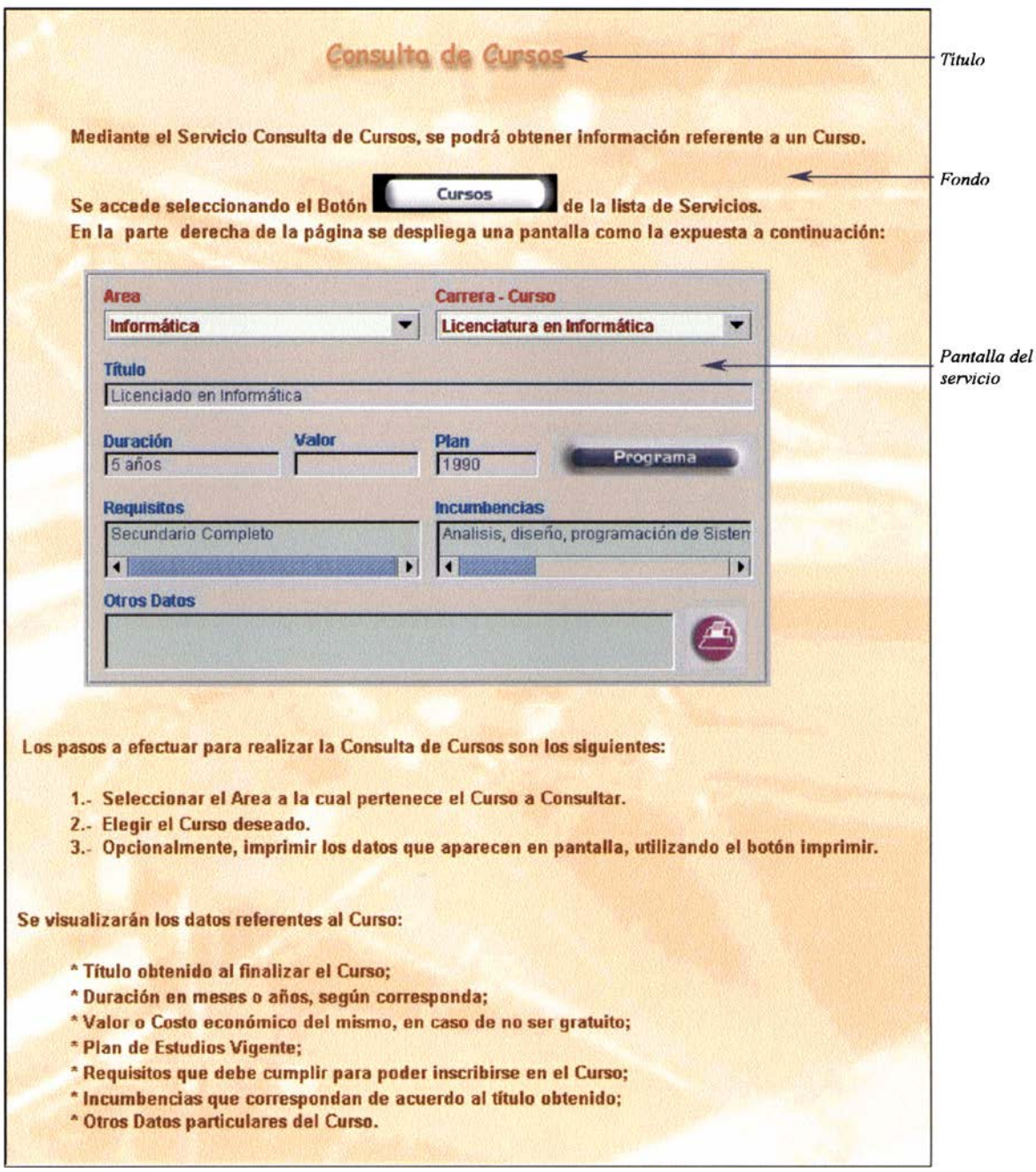
Los mensajes se presentan dentro de un cuadro, el cual contiene un texto con información de error o de confirmación de operación. Siempre se ubican en el mismo lugar de la página.



Cuadro de mensaje

PÁGINAS DE AYUDA

Se encuentran formadas por un título que identifica el servicio desde el cual fue invocada la ayuda, una explicación de cómo llevar a cabo el mismo, la pantalla mediante la cual se efectiviza éste y un fondo, el cual presenta el mismo color para todos los servicios de todos los perfiles de usuario.



Funcionalidad de la Aplicación

FUNCIONALIDAD DE LA APLICACIÓN

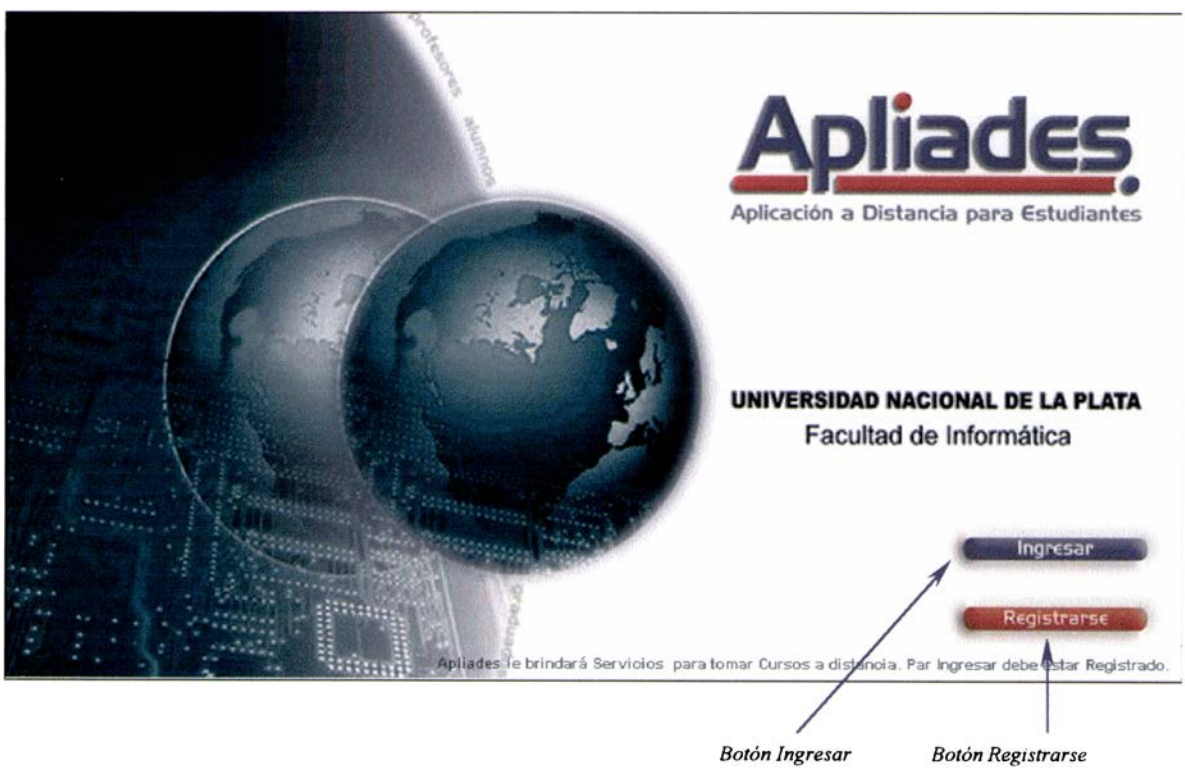
APLIADES – APLICACIÓN A DISTANCIA PARA ESTUDIANTES

Apliades es una aplicación que permite a los alumnos tomar cursos a distancia.

Otorga servicios que automatizan las tareas que realizan en la Facultad los profesores, alumnos, graduados, empleados administrativos y otras personas ajenas a la misma. A través del uso de los servicios que brinda Apliades se logra mayor eficiencia y rapidez en la realización de dichas tareas.

Página de presentación de Apliades

Al ingresar a esta aplicación se visualiza la página de presentación de Apliades, a través de la cual se puede acceder a la página de ingreso(1) o a la de registración(2). No se permite el acceso a usuarios no registrados.



Página de ingreso a Apliades

Al presionar el *botón ingresar* de la página de presentación, se ingresa a la página que permite establecer la conexión para acceder a los servicios que brinda Apliades.

Se debe completar la identificación de usuario, la clave de acceso y seleccionar la categoría del perfil de acceso de la lista desplegable. Ninguno de estos datos puede ser omitido.

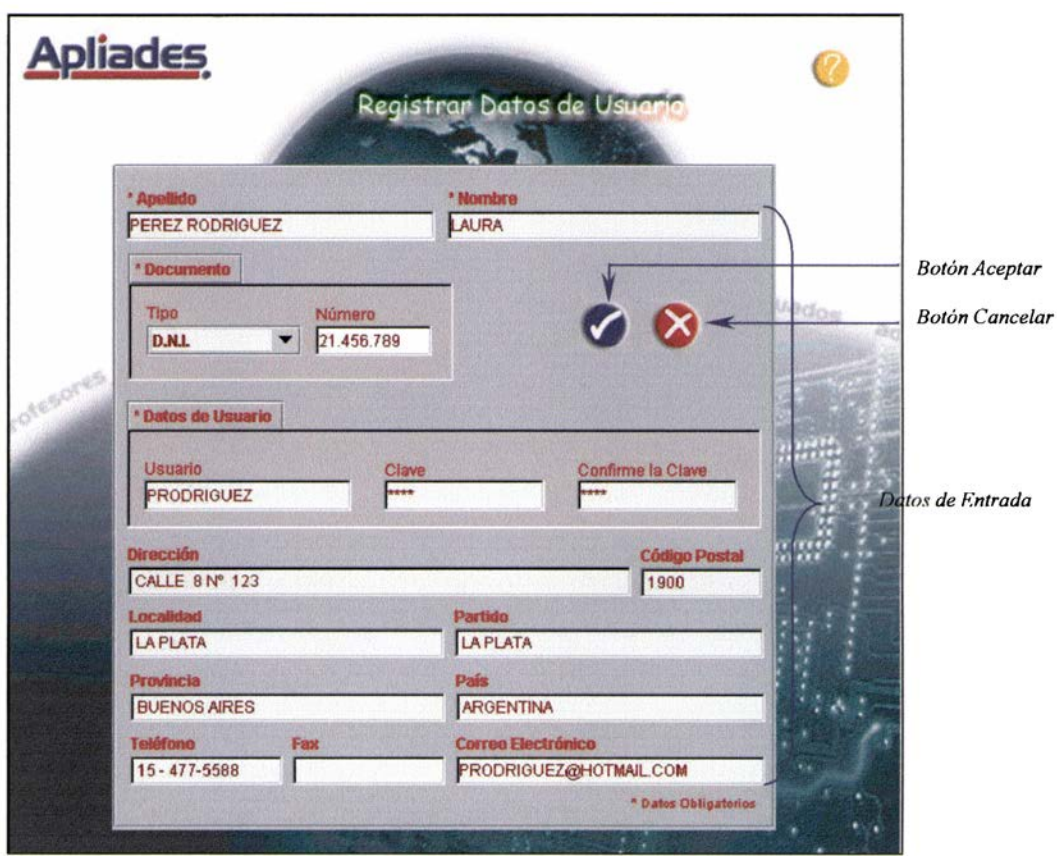
Al presionar el *botón aceptar*, previa validación y autenticación de los datos ingresados, se accede a la página de servicios que corresponde al perfil elegido. Con el *botón cancelar* se regresa a la página de presentación.

Página de registración de datos de usuario

Al presionar el *botón registrarse* de la página de presentación, se ingresa a la página que permite registrarse como usuario de Apliades.

Se deben completar en forma obligatoria el apellido, nombre, número de documento, identificación de usuario y clave de acceso y elegir el tipo de documento de la lista desplegable. El domicilio, el número de teléfono, el número de fax y la dirección de correo electrónico, completan la información del usuario, pero no se exigen para llevar a cabo la registración. Pueden ser ingresados en otro momento utilizando el servicio denominado Modificación de Datos Personales.

Al presionar el *botón aceptar* se efectiviza el registro de datos, previa validación de los mismos. Se asigna automáticamente la categoría de usuario general, y se ingresa a la página de servicios para dicha categoría. Con el *botón cancelar* se regresa a la página de presentación.



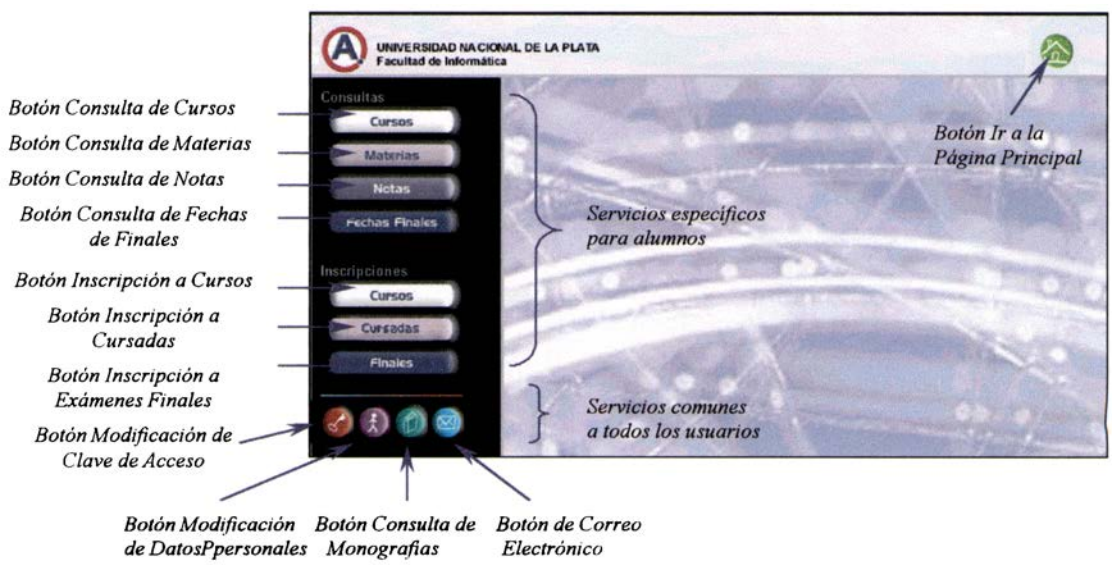
SERVICIOS PARA EL PERFIL ALUMNO

Al ingresar a Apliades con el perfil alumno, se presenta la lista de servicios que se mencionan a continuación:

- Consulta de cursos;
- Consulta de materias, envío y recepción de trabajos;
- Consulta de notas;
- Consulta de fechas de exámenes finales;
- Inscripción a cursos;
- Inscripción a cursadas;
- Inscripción a exámenes finales;
- Modificación de clave de acceso;
- Modificación de datos personales de usuario;
- Consulta de monografías publicadas;
- Acceso a servicio de correo electrónico.

Los cuatro últimos servicios mencionados se encuentran disponibles para todas las categorías de usuarios.

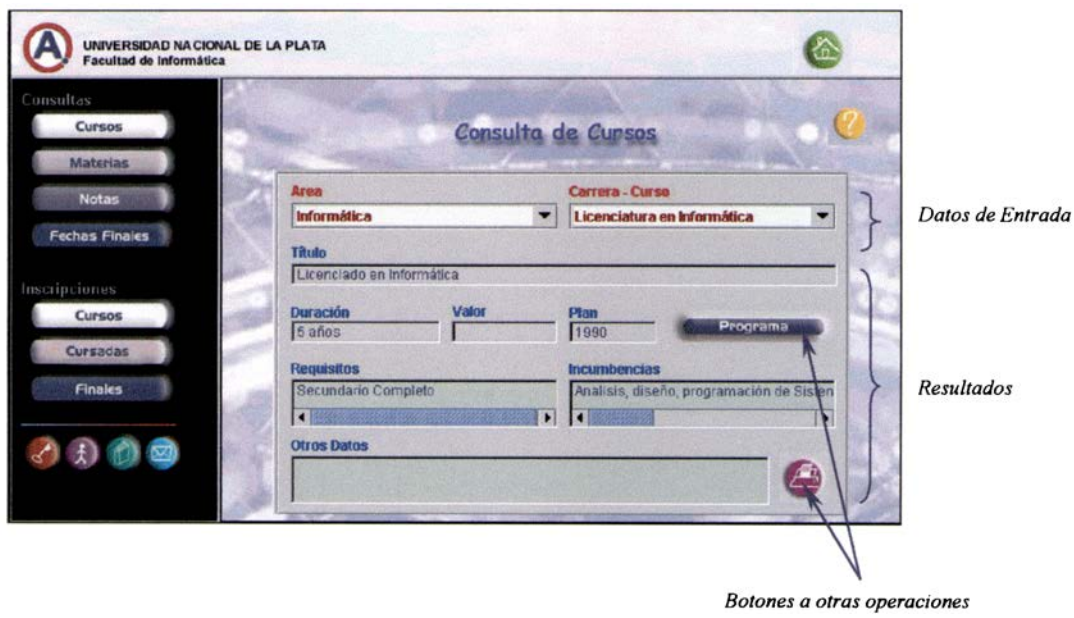
A los servicios se accede presionando el botón correspondiente de la lista de botones que se muestran en el área de servicios de la página.



Página de consulta de cursos

Al presionar el *botón consulta de cursos* de la lista de servicios disponibles, se presenta en el área de trabajo la pantalla que permite obtener información acerca de las carreras y cursos que se encuentran vigentes en la Facultad.

Se deben seleccionar el área y la carrera o curso correspondientes, obteniéndose como resultado el título que se otorga al finalizar, la duración, el valor o costo si lo tuviera, el plan vigente, los requisitos necesarios para inscribirse, las incumbencias y otros datos referentes al mismo. Además se puede consultar el programa del curso o carrera, e imprimir los datos que se visualizan en la página.

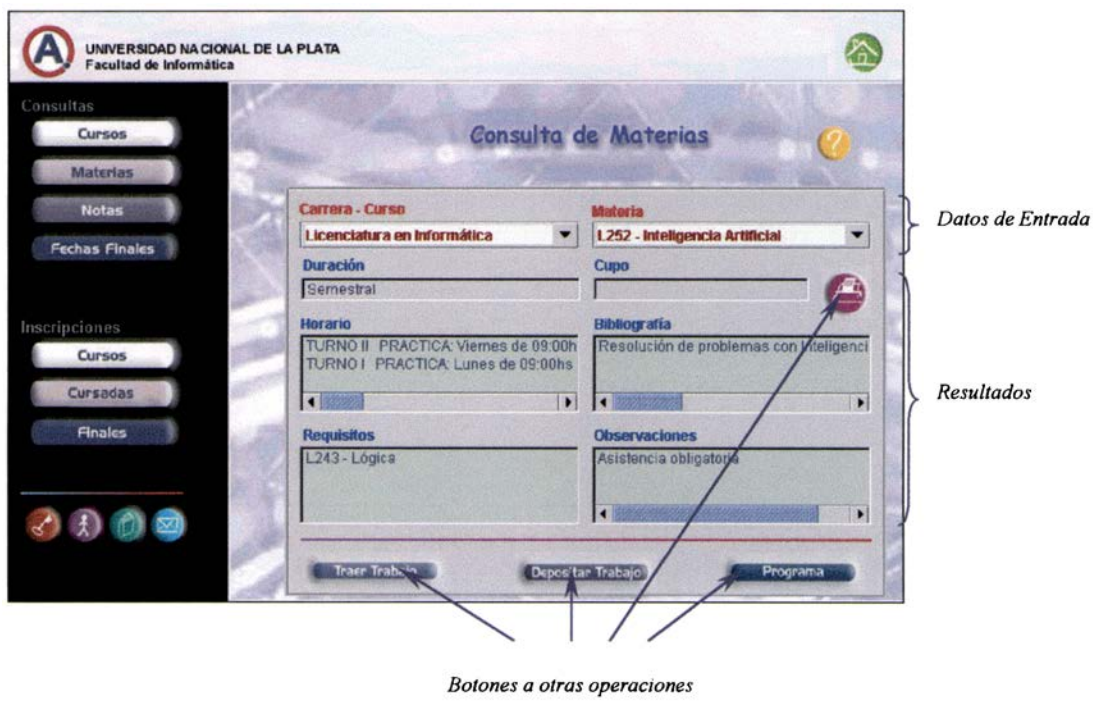


Página de consulta de materias

Al presionar el *botón consulta de materias* de la lista de servicios disponibles, se presenta en el área de trabajo la pantalla que permite obtener información acerca de las materias que se dictan en la Facultad.

Se deben seleccionar la carrera o curso, y la materia correspondiente. Se obtienen como resultado los datos referentes a la materia elegida: la duración, el cupo, el horario de teoría y de práctica en que se cursa, la bibliografía recomendada, los requisitos para poder cursarla y otros datos adicionales de la misma.

Además se puede consultar el programa de la materia e imprimir los datos que se visualizan en la página. Si el alumno se encuentra cursando la materia, desde esta página puede enviar y recibir trabajos de la misma.

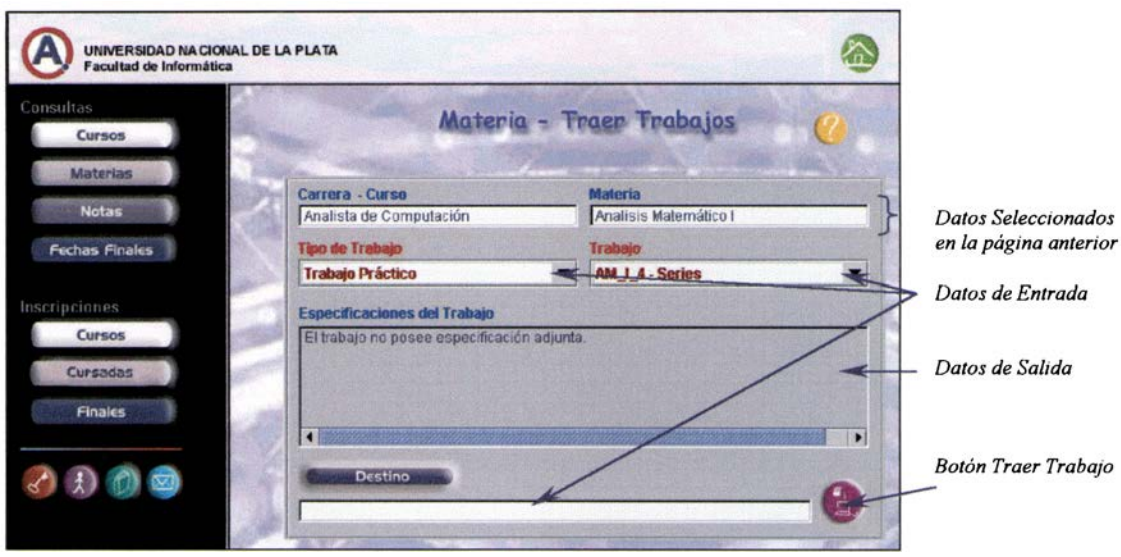


Página traer trabajos

Al presionar el *botón traer trabajo* de la página consulta de materias, se presenta en el área de trabajo la pantalla que permite obtener trabajos de la materia seleccionada en la página anterior.

Se debe seleccionar el tipo de trabajo y el trabajo deseado de la listas desplegables que se presentan. Se visualizan las especificaciones del trabajo elegido.

En el cuadro destino se debe indicar la ruta donde se guardará el trabajo y presionar el *botón traer trabajo* para concluir la operación.

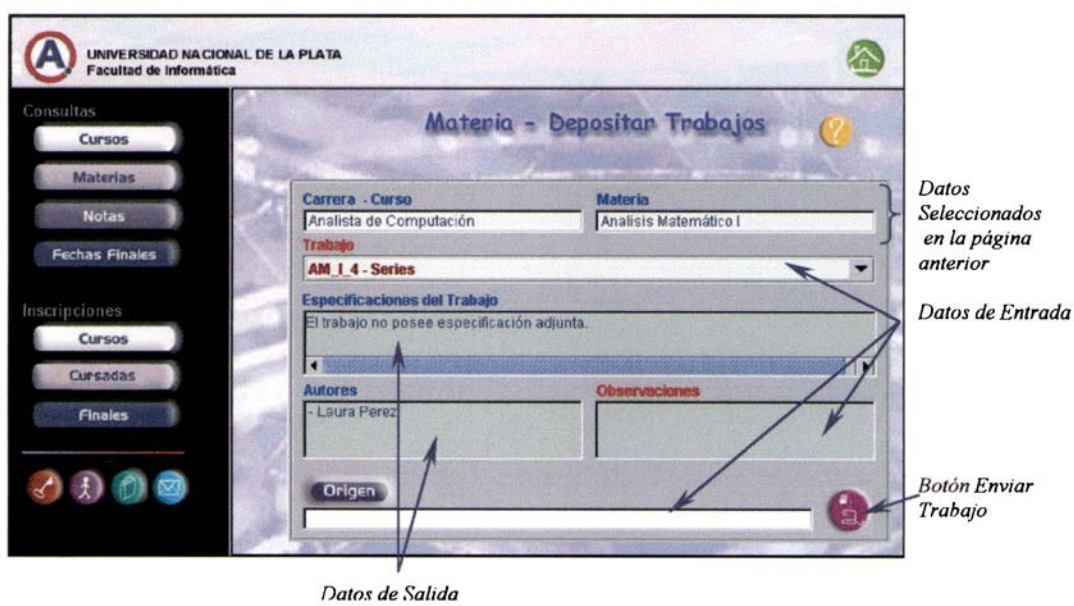


Página depositar trabajos

Al presionar el *botón depositar trabajo* de la página consulta de materias, se presenta en el área de trabajo la pantalla que permite enviar trabajos resueltos a la cátedra de la materia seleccionada en la página anterior.

Se debe seleccionar el trabajo a entregar de la lista que se presenta con todos los trabajos de la materia. Se visualizan las especificaciones y los autores del trabajo elegido. Cada cátedra armará un grupo de trabajo de uno o varios alumnos, quedando a partir de ese momento habilitado para enviar trabajos. En el cuadro de observaciones, el alumno puede indicar particularidades acerca del trabajo que está entregando como así también cualquier notificación acerca del grupo.

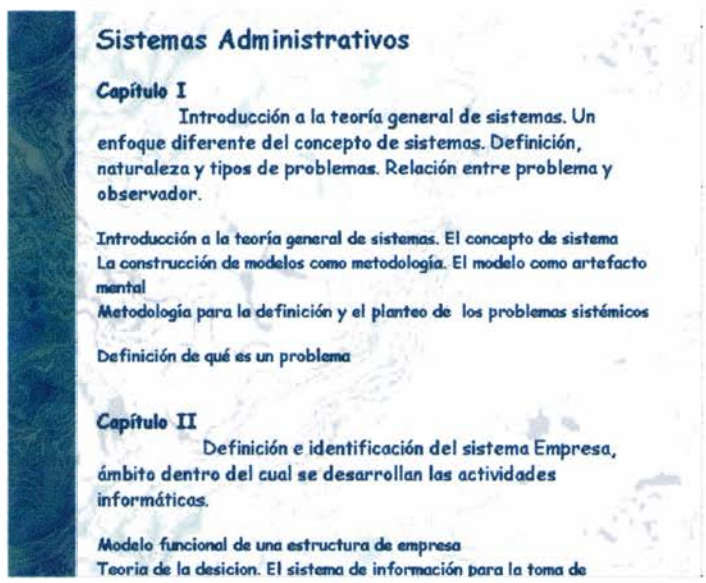
Una vez elegidos los datos, en el cuadro origen se debe indicar la ruta donde se encuentra el trabajo a enviar y posteriormente presionar el *botón enviar trabajo* para finalizar la operación.



Página programa de materia

Al presionar el *botón programa* de la página consulta de materias, se presenta en el área de trabajo el programa de la materia seleccionada.

Por ejemplo, si seleccionamos la materia Sistemas Administrativos, se visualiza el siguiente programa.



Sistemas Administrativos

Capítulo I

Introducción a la teoría general de sistemas. Un enfoque diferente del concepto de sistemas. Definición, naturaleza y tipos de problemas. Relación entre problema y observador.

Introducción a la teoría general de sistemas. El concepto de sistema
La construcción de modelos como metodología. El modelo como artefacto mental
Metodología para la definición y el planteo de los problemas sistémicos

Definición de qué es un problema

Capítulo II

Definición e identificación del sistema Empresa, ámbito dentro del cual se desarrollan las actividades informáticas.

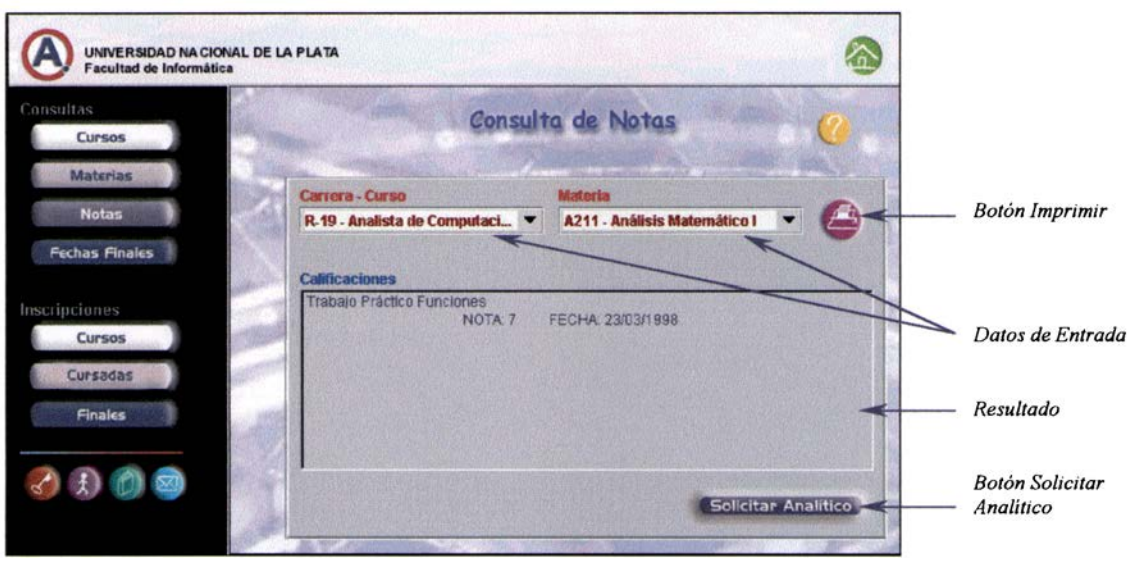
Modelo funcional de una estructura de empresa
Teoria de la desicion. El sistema de información para la toma de

Página de consulta de notas

Al presionar el *botón consulta de notas* de la lista de servicios disponibles, se presenta en el área de trabajo la pantalla que permite obtener información acerca de las calificaciones que el alumno obtuvo en una materia.

Al seleccionar la carrera o curso, y la materia de las listas desplegables que se presentan, se visualizan el tipo de trabajo, el trabajo, la nota y la fecha de obtención de la misma. Estos datos pueden imprimirse presionando el *botón imprimir*.

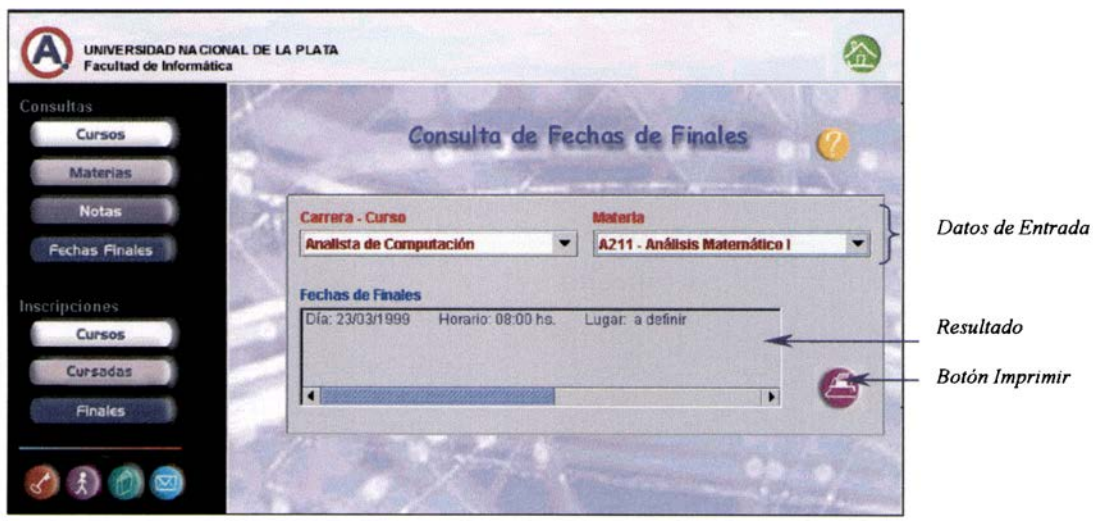
Además desde esta página se puede realizar el pedido del certificado analítico del alumno presionando el *botón solicitar analítico*.



Página de consulta de fechas de exámenes finales

Al presionar el *botón consulta de fechas de exámenes finales* de la lista de servicios disponibles, se presenta en el área de trabajo la pantalla que permite obtener información acerca de las fechas de exámenes finales.

Al seleccionar la carrera o curso, y la materia de las listas correspondientes, se obtienen como resultado las fechas de exámenes finales, el horario y el lugar dónde se llevará a cabo el mismo. Estos datos pueden imprimirse presionando el *botón imprimir*.



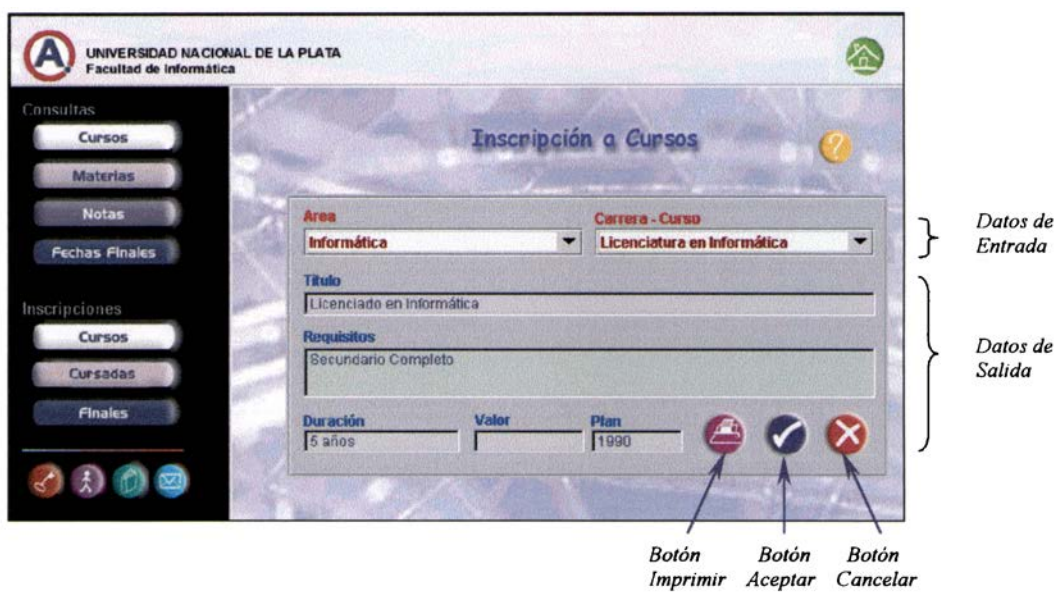
Página de inscripción a cursos

Al presionar el *botón inscripción a cursos* de la lista de servicios disponibles, se presenta en el área de trabajo la pantalla que permite realizar la inscripción a cursos o carreras que se dictan en la Facultad.

Se debe seleccionar el área y la carrera o curso de las listas que se presentan. Se obtienen el título, los requisitos, la duración, el valor y el plan del mismo.

Al presionar el *botón aceptar* se confirma la inscripción, si se desea abandonar el servicio sin efectivizarla se deberá presionar el *botón cancelar*.

Estos datos pueden imprimirse utilizando el *botón imprimir*.



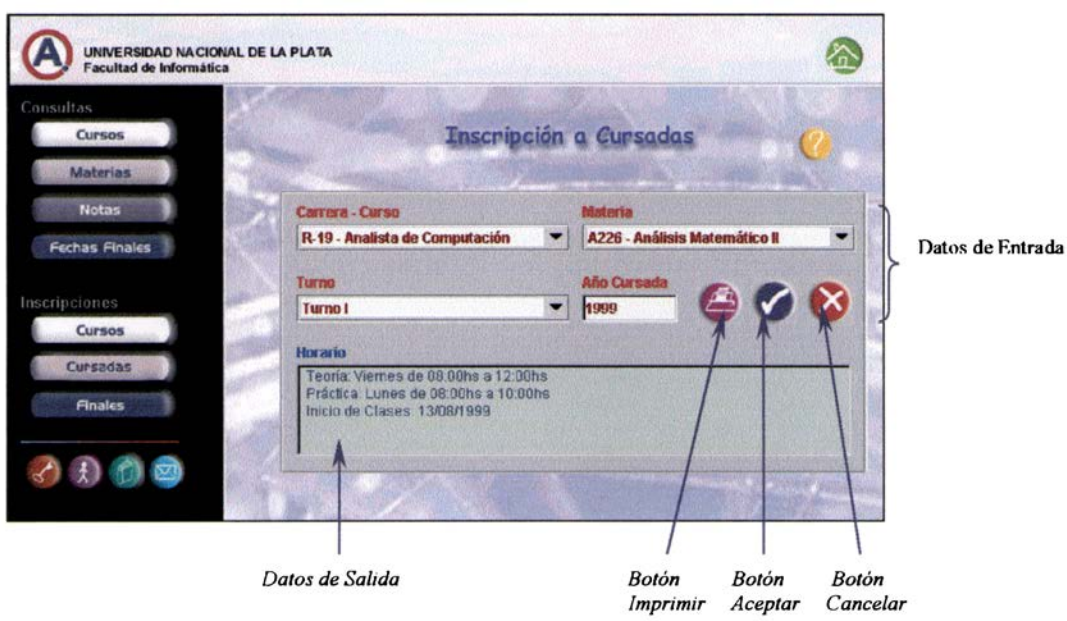
Página de inscripción a cursadas

Al presionar el *botón inscripción a cursadas* de la lista de servicios disponibles, se presenta en el área de trabajo la pantalla que permite realizar la inscripción en una materia que se dicta en la Facultad.

Se debe elegir la carrera o curso, de la lista de cursos de los cuales el usuario es alumno, la materia, para la cual está habilitado a cursar y el turno de la lista de turnos disponibles para dicha materia. Ingresar además el año de cursada donde se desea inscribir. Se visualizan los datos del turno elegido.

Al presionar el *botón aceptar* se confirma la inscripción, si se desea abandonar el servicio sin efectivizar la misma, se deberá utilizar el *botón cancelar*.

Estos datos pueden imprimirse presionando el *botón imprimir*.

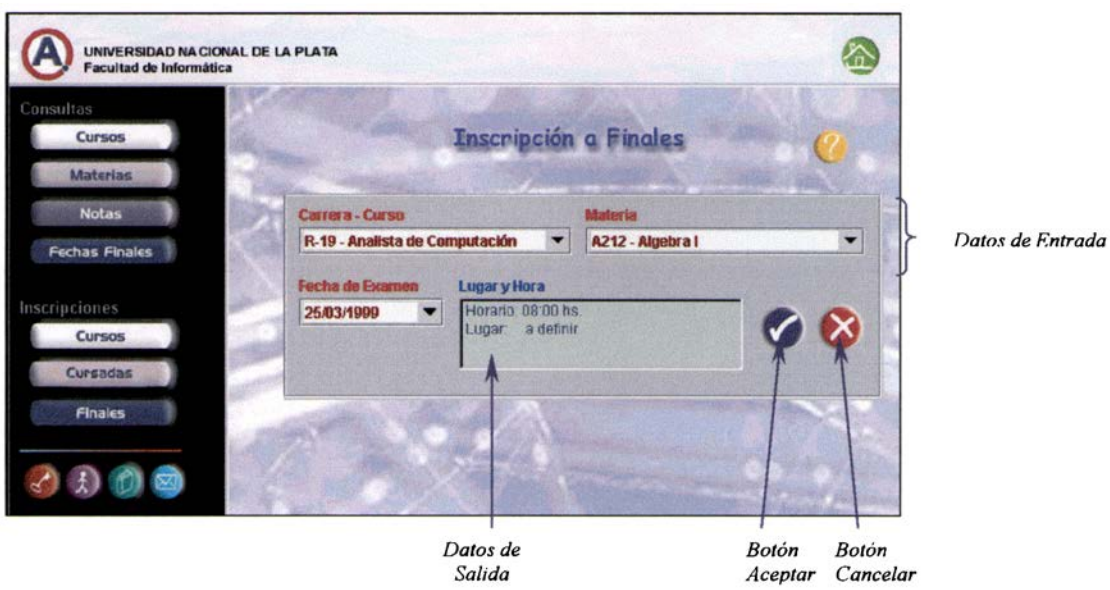


Página de inscripción a exámenes finales

Al presionar el *botón inscripción a exámenes finales* de la lista de servicios disponibles, se presenta en el área de trabajo la pantalla que permite realizar la inscripción para rendir el examen final de una materia.

Se debe elegir la carrera o curso de la lista de cursos de los cuales es alumno, la materia de la lista de materias en las cuales se encuentra habilitado para rendir y por último seleccionar la fecha en la cual desea rendir el examen, de la lista de fechas abiertas para la inscripción. Se visualiza el lugar y la hora del examen.

Al presionar el *botón aceptar* se confirma la inscripción y automáticamente se imprime un comprobante de inscripción. Si se desea abandonar el servicio sin efectivizar la misma, se deberá presionar el *botón cancelar*.

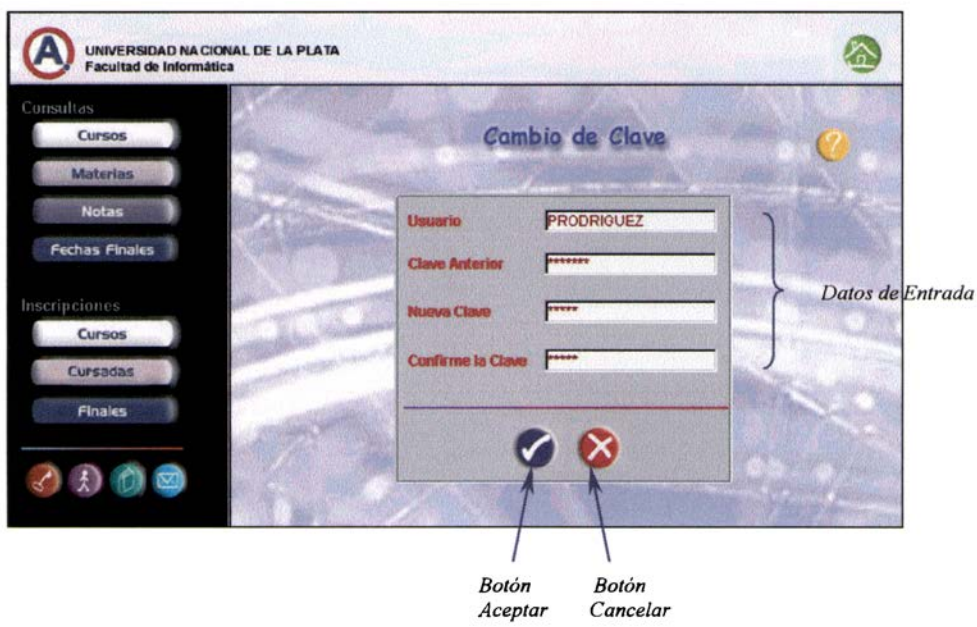


Página de modificación de clave de acceso

Al presionar el *botón modificación de clave de acceso* de la página de servicios, se ingresa a la página que permite cambiar la clave de acceso a Apliades.

Se debe completar la identificación de usuario, la clave anterior de acceso e ingresar dos veces la clave nueva. Ninguno de estos datos puede ser omitido.

Al presionar el *botón aceptar*, previa validación de los datos ingresados, se confirma la modificación, si se desea abandonar la operación se debe presionar el *botón cancelar*.

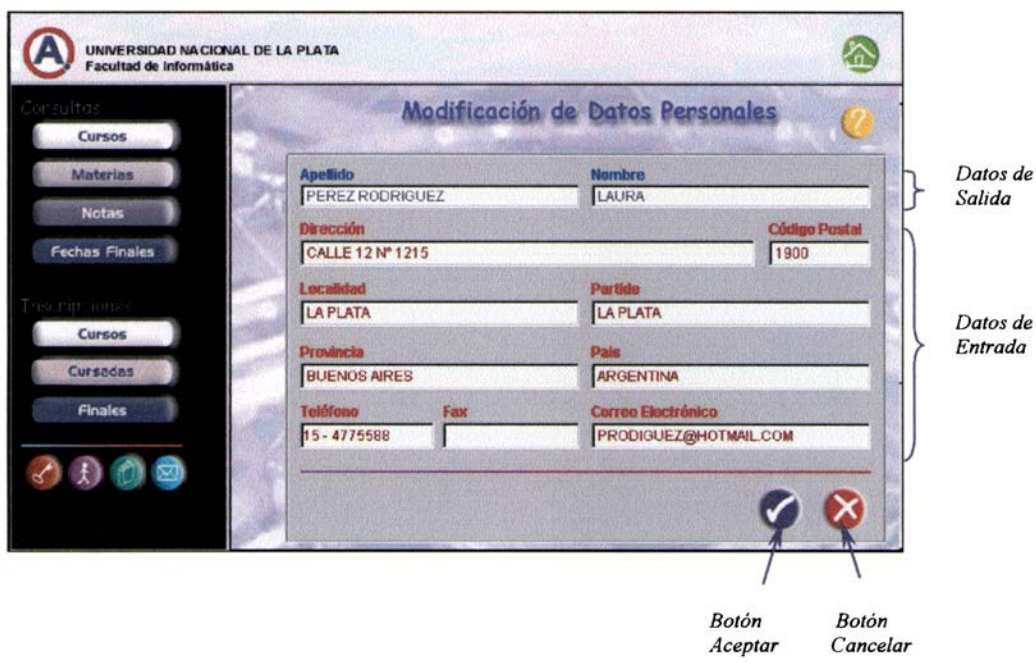


Página de modificación de datos personales

Al presionar el *botón modificación de datos personales* de la lista de servicios, se presenta la página que permite completar y modificar los datos personales del usuario.

Se presentan el apellido y nombre del usuario y los datos que ha ingresado en el momento de la registración. Puede modificar el domicilio, el número de teléfono, el número de fax y la dirección de correo electrónico o completar los datos faltantes.

Al presionar el *botón aceptar* se efectiviza el registro de datos, si no se desea realizar la actualización se debe pulsar el *botón cancelar*.

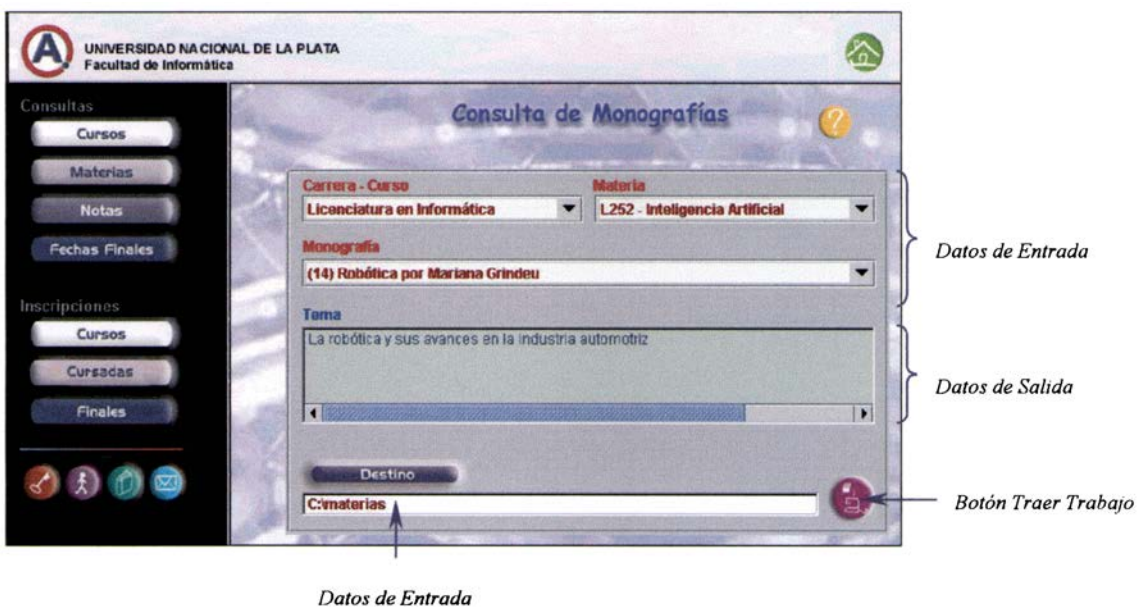


Página de consulta de monografías

Al presionar el *botón consulta de monografías* del conjunto de servicios disponibles, se presenta en el área de trabajo la pantalla que permite obtener información acerca de las monografías que se han publicado en la Facultad.

Se deben seleccionar la carrera o curso, la materia y la monografía de las listas desplegables que se presentan. Se visualiza el tema de la monografía elegida.

Se permite "bajar" la monografía, indicando en el cuadro destino la ruta donde se guardará la misma y presionando luego el *botón traer trabajo.*



SERVICIOS PARA EL PERFIL USUARIO GENERAL

Debido a que en la registración de datos de usuario que ofrece Apliades, se asigna automáticamente el perfil de usuario general, consideramos conveniente implementar parte de los servicios para dicho perfil, logrando de este modo presentar la página de servicios generales cuando el usuario efectiviza su registración.

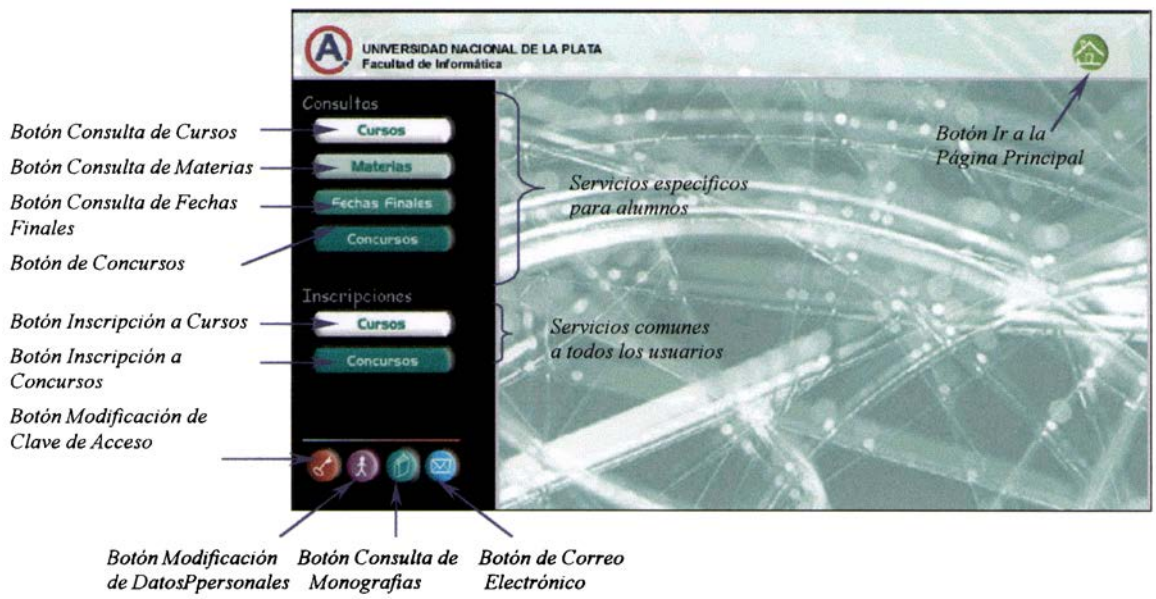
Con respecto a los servicios generales se implementaron aquellos que coinciden con el conjunto de servicios del perfil alumno.

La lista de servicios es la que se presenta a continuación:

- Consulta de cursos;
- Consulta de materias;
- Consulta de fechas de exámenes finales;
- Consulta de concursos;
- Inscripción a cursos;
- Inscripción a concursos;
- Modificación de clave de acceso;
- Modificación de datos personales de un usuario;
- Consulta de monografías publicadas;
- Acceso a servicio de correo electrónico.

Los cuatro últimos servicios mencionados se encuentran disponibles para todas las categorías de usuarios.

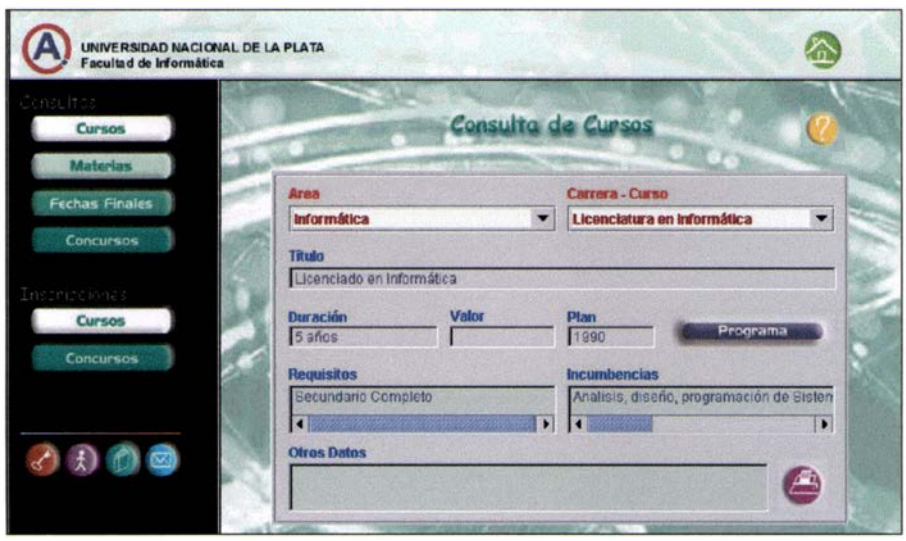
A los servicios se accede presionando el botón correspondiente de la lista de botones que se muestran en el área de servicios de la página.



Debido a que la funcionalidad de las páginas implementadas para el perfil usuario general es la misma que la que poseen las páginas de alumnos, no se explica el funcionamiento de las mismas, sino que sólo se presentan con el fin de visualizar el aspecto que poseen.

Página de consulta de cursos

Al presionar el *botón consulta de cursos* de la lista de servicios disponibles, se presenta en el área de trabajo la pantalla que permite obtener información acerca de las carreras y cursos que se encuentran vigentes en la Facultad.

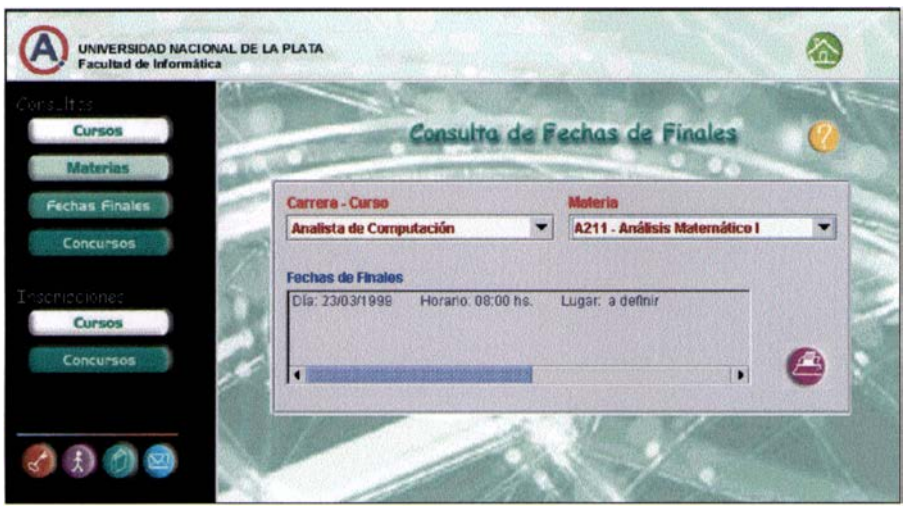


Página de consulta de materias

Al presionar el *botón consulta de materias* de la lista de servicios disponibles, se presenta en el área de trabajo la pantalla que permite obtener información acerca de las materias que se dictan en la Facultad.

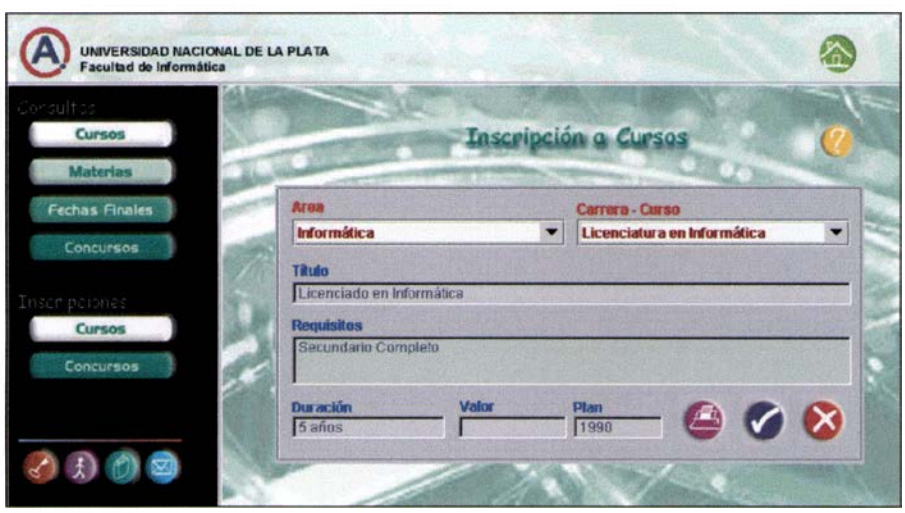
Página de consulta de fechas de exámenes finales

Al presionar el *botón consulta de fechas de exámenes finales* de la lista de servicios disponibles, se presenta en el área de trabajo la pantalla que permite obtener información acerca de las fechas de exámenes finales.



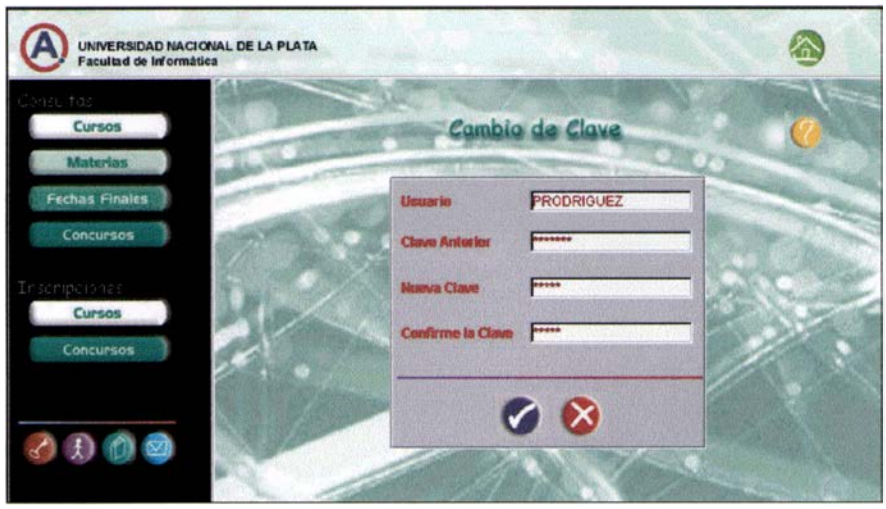
Página de inscripción a cursos

Al presionar el *botón inscripción a cursos* de la lista de servicios disponibles, se presenta en el área de trabajo la pantalla que permite realizar la inscripción a cursos o carreras que se dictan en la Facultad.



Página de modificación de clave de acceso

Al presionar el *botón modificación de clave de acceso* de la página de presentación, se ingresa a la página que permite cambiar la clave de acceso a Apliades.

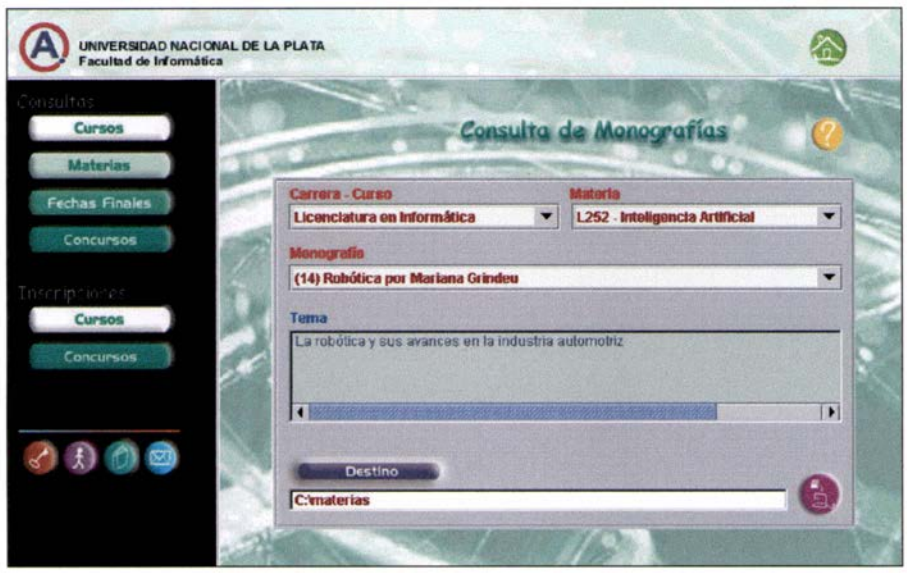


Página de modificación de datos personales

Al presionar el *botón modificación de datos personales* de la lista de servicios, se presenta la página que permite completar y modificar los datos personales del usuario.

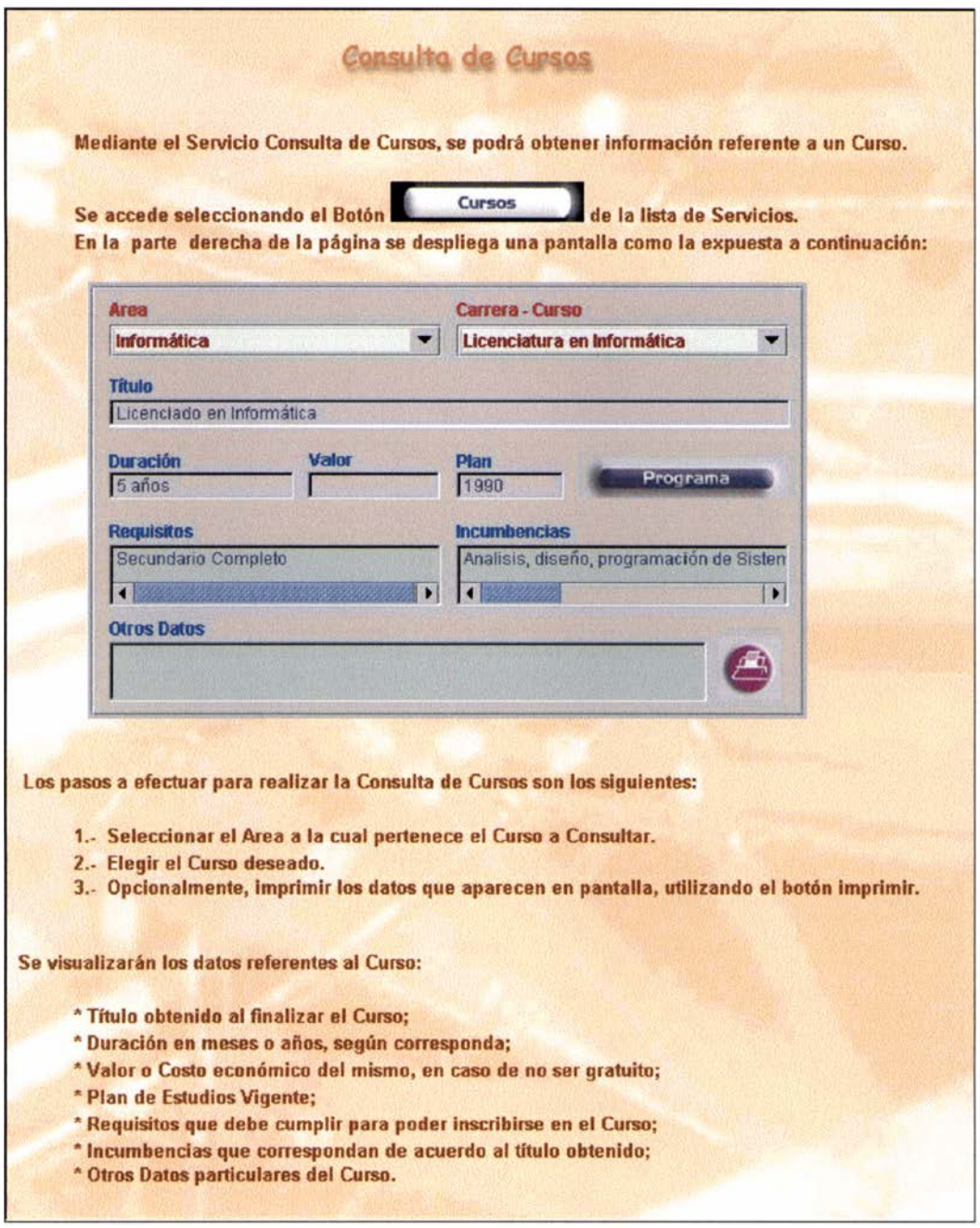
Página de consulta de monografías

Al presionar el *botón consulta de monografías* del conjunto de servicios disponibles, se presenta en el área de trabajo la pantalla que permite obtener información acerca de las monografías que se han publicado en la Facultad.



SERVICIOS DE AYUDA

Se accede a las páginas de ayuda a través del botón del mismo nombre que se presenta en cada uno de los servicios ofrecidos. Brinda una explicación de cómo llevar a cabo el servicio desde el cual se invocó la ayuda.



PRINCIPIOS DE NIELSEN

EVALUACIÓN DE INTERFACES

Si bien puede pensarse que con la evolución de Interfaces de estilo Windows o utilización de Páginas Web, los principios de Nielsen se tornan obsoletos, no es así. Se pueden aplicar, tanto a las interfaces orientadas a texto, como a las interfaces orientadas a gráficos, ya que existe una correspondencia entre el lenguaje textual y el lenguaje visual.

Los nueve principios básicos a seguir para el desarrollo de Interfaces de Usuario son los que se detallan a continuación: (ref. www.useit.com)

1. Utilizar diálogo simple y natural;
2. Emplear el lenguaje del usuario;
3. Minimizar el uso de la memoria del usuario;
4. Brindar consistencia;
5. Proveer feedback;
6. Proporcionar salidas evidentes;
7. Proveer short-cuts;
8. Mostrar buenos mensajes de error;
9. Prevenir errores;

La interfaz de Apliades cumple con los Principios de Nielsen.

1 - Utilización de diálogo simple y natural

Siempre se muestra información esencial, ya que toda información extra en un diálogo, compite con las unidades relevantes de información, disminuyendo su visibilidad.

Toda la información aparece de una manera lógica y natural, ya que no se usa exclusivamente mayúsculas, no se utilizan abreviaturas, no hay errores de tipeo, y la distribución de los datos es la más adecuada para un ingreso ordenado y completo por parte del usuario, sin considerar el orden más cómodo para la computación del proceso.

Principios de Nielsen

6 - Proporcionar salidas evidentes

Si bien los servicios que realizan transacciones proveen la alternativa de cancelar la operación, no es necesario disponer de esta opción en todas las páginas, pues las consultas no la necesitan.

7 - Proveer short-cuts

Dado que el grado de interacción entre los servicios para los alumnos de Apliades y los usuarios es mínimo, no se necesita tener atajos para llevar a cabo distintas operaciones, con la excepción del botón que conduce desde cualquier servicio a la página principal.

8 - Mostrar buenos mensajes de error

Los mensajes de error provistos no intimidan al usuario y describen en forma clara el problema ocurrido.

9 - Prevenir errores

Más importante que emitir mensajes de error es prevenirlos. Para evitar errores de tipeo, se han incluido en todas las pantallas selectores de datos, minimizando de esta manera el ingreso incorrecto del dato solicitado.

2 – Emplear el Lenguaje del Usuario

Los términos utilizados son expresados con palabras y frases familiares para el usuario. No se utilizan palabras extranjeras, ni vocablos de la jerga informática.

3 – Minimizar el uso de la memoria del usuario

La Aplicación fue diseñada sin tener más de dos niveles de anidamiento, evitando con ésto que el usuario deba realizar operaciones innecesarias para utilizar el servicio deseado.

Además, una vez que se ha ingresado a Apliades, los servicios disponibles para el perfil especificado, aparecen en el lado izquierdo de la pantalla durante toda la conexión, y el contexto en el cual se está trabajando siempre es visible a través del título de la página activa.

En todos los servicios se dispone de una ayuda que indica los pasos a seguir para efectuar el mismo.

4 - Brindar consistencia

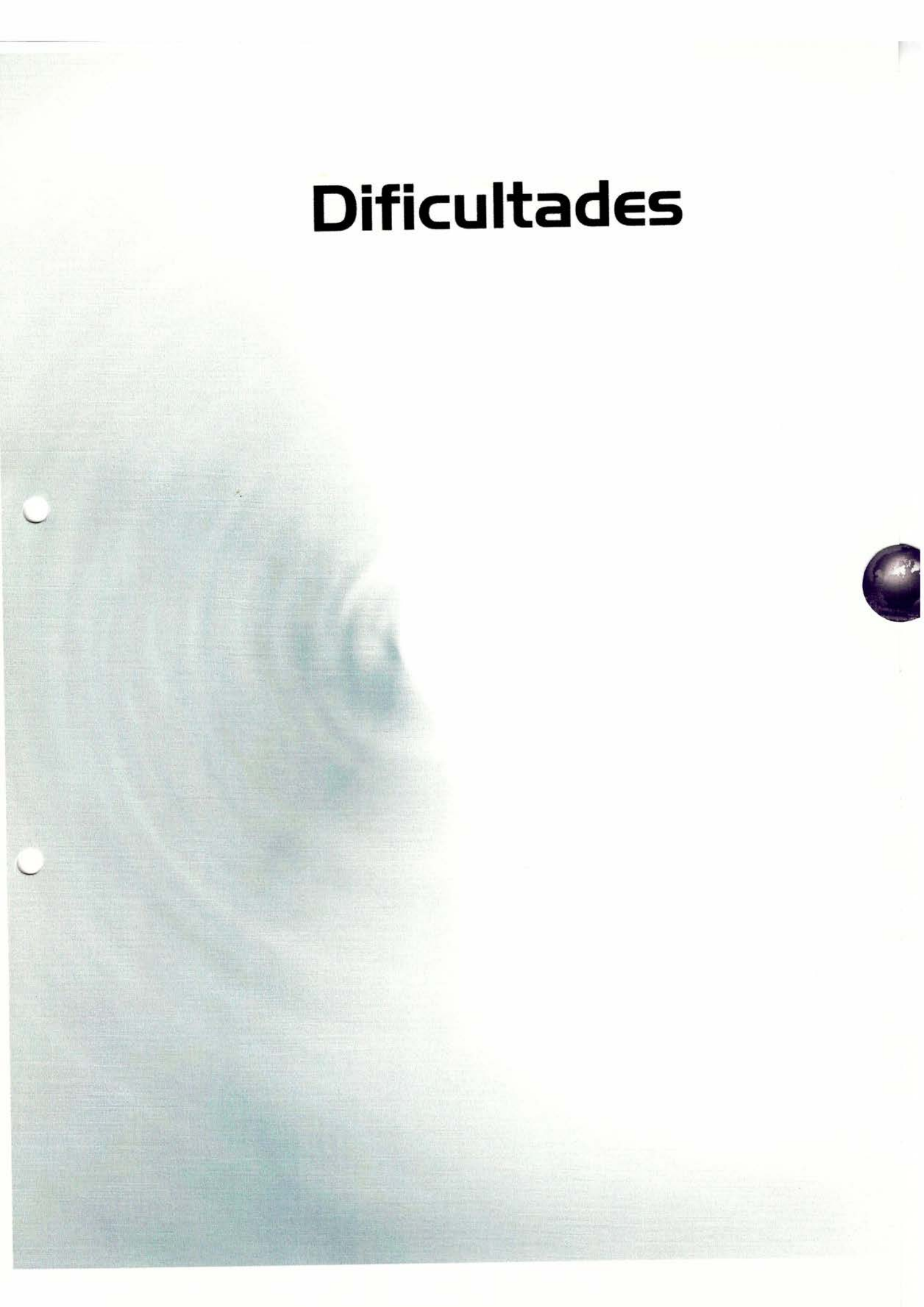
Apliades brinda consistencia semántica, ya que cada término tiene un único significado y no hay diferentes situaciones o acciones que representen distintos conceptos. Los botones que se utilizan tienen un único sentido para una misma acción a lo largo de toda la aplicación, y viceversa, es decir que para realizar una misma acción no se dispone de más de un botón.

Provee además consistencia visual, ya que, como se explicó anteriormente, los datos de entrada se presentan en color rojo, y los de salida en color azul en todos los casos. Los mensajes de error aparecen siempre en el mismo lugar de la pantalla.

5 - Proveer feedback

Si bien la mayoría de los servicios de alumnos son de consulta, en los servicios donde se efectúa actualización de datos, se informa al usuario el resultado de dicho proceso. En los servicios de consulta la intervención del usuario es mínima, luego de que éste ingresa los datos de selección, automáticamente se ejecuta la consulta.

Dificultades



DIFICULTADES

CONSIDERACIONES ACERCA DE LOS PROBLEMAS ENCONTRADOS

Incompatibilidad entre ambientes de desarrollo

Cuando comenzamos el estudio del lenguaje la versión existente del Visual Age for Java era la 1.0. En este ambiente de desarrollo las clases del AWT no permitían incluir imágenes con los objetos, y en nuestra especificación de interfaz los botones contenían imágenes. Por este motivo intentamos varias formas de subsanar el problema. Una fue trabajando los botones en otros ambientes de desarrollo (JBuilder, ButtonCool, NetObject) para luego importarlos y manipularlos en el Visual Age, pero la importación no tuvo los resultados esperados, ya que estos objetos utilizaban clases que el Visual Age no poseía. Además intentamos importar las clases que en el Symantec permitían la construcción de objetos con imágenes, pero si bien pudimos hacerlo no nos dejaba editarlos, ni asociar eventos a los mismos, pues en la importación sólo se incluían las clases, pero no el código de éstas.

Al aparecer en el mercado una nueva versión de Visual Age for Java se nos indica realizar la actualización de la herramienta de desarrollo para lograr el aprovechamiento de las nuevas posibilidades que ésta brinda. Esto trajo aparejado algunas soluciones a los problemas planteados anteriormente, pero además nos creó una serie de inconvenientes adicionales. Entre las soluciones podemos mencionar la que nos proporcionó el conjunto de clases Swing, ya que otorgaba una funcionalidad similar a la propuesta por AWT con la ventaja de incorporar métodos y propiedades específicas para el manejo de imágenes en los objetos. El mayor inconveniente con el que nos encontramos fue la incompatibilidad anunciada entre AWT (principal soporte de la interfaz de usuario de nuestra aplicación hasta ese momento) y SWING (ref. pag. 47 -Tomo I). Al no existir una herramienta en el ambiente que nos permitiera migrar interfaces basadas en AWT a su equivalente funcional en SWING, nos vimos en la obligación de recodificarlas.

Cabe aclarar que asumimos el costo de la recodificación basándonos en la mayor riqueza visual que las interfaces basadas en Swing podían ofrecer.

Por lo tanto al utilizar el Visual Age for Java 2.0, pudimos codificar las interfaces respetando el diseño original de las mismas.

Testeos de funcionalidad dentro del ambiente

Otro problema que tuvimos fue obtener la configuración correcta del ambiente Visual Age para que funcionen, dentro del mismo, RMI, JDBC y nuestra aplicación en forma conjunta. Esto fue debido a la escasa documentación disponible hasta ese momento respecto de este tema. Pero realizando distintas pruebas, logramos configurarlo. Importamos las clases JDBC de Oracle, seteamos las variables del classpath que posee el ambiente, incluyendo estas clases y las referencias a las clases propias de la aplicación.

Exportación de la aplicación con composición visual

Otro inconveniente fue que, a pesar de ejecutar los pasos indicados, no encontrábamos la forma de exportar una clase y que al importarla en otra instalación de la misma herramienta conservara la composición visual. Encontramos como solución momentanea copiar los cuatro archivos correspondientes a la composición visual, al ambiente y seteos del mismo, pero la desventaja de ésto era que el ambiente de la máquina actualizada se perdía.

Con el Visual Age 2.0 for Java, descubrimos la alternativa de exportar sólo el paquete desesado, e importarlo en cualquier otra máquina sin problemas. Esto fue posible generando un archivo export.dat.

Mantenimiento de estado de usuario

Otro tema que nos preocupó bastante fue cómo hacer para mantener los datos del usuario que ingresó a la aplicación, a lo largo de toda la sesión de conexión. Ya que todos los servicios dependen del usuario conectado. Como se explicó en el tomo I, los applets no pueden acceder a ningún recurso de la máquina cliente, impidiéndonos de esta forma guardar en una variable de memoria o en un archivo local, los datos del usuario. Pensamos en obtener la dirección IP de la máquina, y dentro del server asociar el usuario con dicha dirección, pero no podíamos leer los datos de la máquina cliente. La única alternativa que nos quedaba era lograr que los applets se comunicaran entre sí, intercambiando la información requerida. Esto se logró gracias a que los applets pueden comunicarse a través

del ambiente del browser, permiténdonos darle una solución óptima a este problema (ver Modelo de comunicación en el cliente, pág. 51 - Tomo II).

Portabilidad

Si bien, el eslogan de Java es "write once and run it anywhere", en la práctica no es tan portable como se anuncia. Si bien no se debía recodificar la programación para trasaladar un programa de una máquina a otra, cuando se ejecutaba la aplicación en distintas máquinas que tenían el mismo software de base, los resultados eran diversos. Por ejemplo con respecto a la visualización de los objetos con imágenes, en algunas máquinas el objeto desaparecía, en otras se veía sin la imagen asociada y en otras la visualización era la esperada. Esto es debido a que el browser sin las clases o plugins específicos no muestra los applets completos.

Con respecto a este problema se deben considerar las versiones del browser, del plugin y del JRE (Java Runtime Environment), ya que éstas deben ser compatibles con la versión de las Swing utilizadas para el desarrollo de la aplicación.

Con la última versión del JDK, la portabilidad aumentó, pero la migración a esta versión tenía el costo de la reprogramación de la aplicación, costo que ya habíamos asumido una vez, y que no estabamos en condiciones de volver a asumir por una cuestión de tiempos. Como es sabido, cuando se instala una nueva versión de un software, ésta soporta las versiones anteriores, pero increiblemente, no sucede ésto con Java. Esto se debe a la reestructuración de paquetes que se realizaron en las componentes Swing, moviendo las clases de un paquete a otro, y cambiando el paquete java.sun.com, principal contenedor de las Swings, por el paquete javax.

La versión del browser a utilizar debe ser, en el Netscape Navigator, superior a la 4.04, ya que ésta no es compatible con el Java Development Kit (JDK) 1.1.2, y en el Microsoft Internet Explorer 4.01 o superior. Además se debe instalar el plugin 1.1.1 o, si se utliza el Netscape, pueden copiarse los archivos swingall.jar, swing.jar, motif.jar, multi.jar, beaninfo.jar, windows.jar en el directorio Netscape\communicator \program\java\classes\. De este modo podrán visualizarse las aplicaciones realizadas con las swing 1.0.3 (versión utilizada para el desarrollo de Apliades).

Configuración del ambiente de producción

Por último, otro punto bastante problemático, fue la configuración del server donde se instaló la aplicación.

Se debió instalar y configurar el protocolo TCP/IP. En dicha configuración tuvimos que deshabilitar DNS, y asignar una dirección IP a la máquina.

En el autoexec.bat se realizaron los seteos de las varibles Path y Classpath. En la primera se indicó la ruta en la cual se encuentra instalada la aplicación y en la segunda la ruta de las clases que permiten la comunicación vía JDBC con la Base de Datos.

Además tuvimos que instalar y configurar la Base de Datos. La configuración la realizamos a través del utilitario SQL Net EasyConfiguration. Debimos asignarle un alias a la Base de Datos ya que a través del mismo el Server de la Aplicación realiza la conexión y el acceso a los datos de la misma, asignamos la dirección IP del host e indicamos el protocolo de comunicación que en este caso es TCP/IP.

RMI (Métodos de Invocación Remota)

Un problema importante que tenemos es la comunicación vía RMI. Desde el ambiente Visual Age for Java, realizando el correcto seteo de los parámetros y variables correspondientes, funciona de la forma esperada, pero fuera del ambiente de desarrollo no es así.

Para solucionar este problema realizamos varias pruebas. Una de ellas fue el seteo de la variable classpath del archivo autoexec.bat. Al hacerlo se pierde la visualización de las imágenes, y no se establece la conexión de los applets con el host. Al quitar el seteo de classpath, el servidor no se inicializa, ya que no encuentra las clases Stub y Skeleton necesarias para la conexión vía RMI.

Otra prueba realizada fue la instalación del Plugin 1.1.1 y la conversión de los archivos .html para que reconociera el plugin, pero, aunque la documentación aseguraba su funcionamiento, no fue así.

Conclusiones

CONCLUSIONES

El desarrollo de Apliades nos permitió acceder a nuevas herramientas de desarrollo, configurar protocolos de comunicación y ambientes de ejecución, y profundizar en el tema de lenguaje SQL para el manejo de Base de Datos. Además con este trabajo, logramos integrar las herramientas y conocimientos obtenidos durante toda la carrera, con respecto a la implementación completa de un sistema, cumpliendo con todas las etapas que habíamos ido aprendiendo en forma separada a través de las distintas materias.

De acuerdo a las dificultades mencionadas (pag. 119), podemos concluir que Java, si bien es la mejor opción en lenguajes de este tipo, no es tan portable como lo presentan.

Apliades queda abierta para futuros tesistas. Por un lado puede completarse la aplicación, ya que sólo se implementó el perfil alumno y parte del perfil usuario general, y por otro, puede realizarse la integración con el modelo de seguridad Kerberos.

Se establecieron lineamientos para la creación de las interfaces de todos los perfiles de usuario. La base de datos contiene tablas para la implementación de otros servicios de Apliades, no sólo aquellas que fueron usadas para el perfil alumno. Para el desarrollo de los perfiles restantes, deberían definirse los servicios propios de cada uno, y realizar un examen exhaustivo de la base de datos para hacer las modificaciones que se consideren convenientes para soportar las nuevas definiciones.

Debido al alcance de las funcionalidades implementadas en el prototipo desarrollado, se presentó un único usuario de base de datos, quedando para futuros tesistas completar el esquema de protección de la base de datos en la medida en que la aplicación incremente su funcionalidad.

TRABAJO DE GRADO

Aplicación a distancia para estudiantes

Antonia Maraviglia – Mariángeles Pucci

Director: Lic. Javier Díaz

Facultad de Informática

Universidad Nacional de La Plata

INDICE

LENGUAJE JAVA

AMBIENTES DE DESARROLLO

BASE DE DATOS

PRÓLOGO

En este informe se presentan los temas que se estudiaron y utilizaron para el desarrollo de la aplicación.

Se divide en tres partes: el Lenguaje Java, el Ambiente de Desarrollo y la Base de Datos.

La primer parte comprende desde conceptos básicos del lenguaje hasta conceptos avanzados del mismo, y permite, a través de su lectura, obtener los conocimientos necesarios para la programación en lenguaje Java.

Dado que el lenguaje no cuenta con un ambiente de desarrollo, se realizó una comparación entre los ambientes más renombrados para elegir uno. Se optó por el Visual Age for Java 2.0. Se dispone de esta información en la segunda parte del informe.

En la última parte se encuentran las principales características del motor de base de datos Oracle.