UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA

UNAN-LEÓN

FACULTAD DE CIENCIAS

DEPARTAMENTO DE COMPUTACIÓN

TESIS PARA OPTAR AL TÍTULO DE INGENIEROS EN SISTEMAS DE INFORMACIÓN

TEMA:

“AUTOMATIZACIÓN DEL LABORATORIO CLÍNICO HEODRA BAJO ENTORNO LINUX”.

Presentado por:

- Br. Wilmer Javier Arteaga Sandoval.
- Br. Gabriel Ernesto Larios Rivera.
- Br. Melvin José Lezama Benavides.

TUTOR:

Lic. Rina Araúz

Octubre 2006

León, Nicaragua

INDICE

Introducción ***.....4***
Antecedentes ***.....5***
Justificación ***.....6***
Objetivo general ***.....7***
Marco teórico ***.....8***
Servicios del laboratorio clinico ***.....8***
Bioseguridad ***.....9***
Banco de sangre ***.....10***
Uroanálisis ***.....10***
Recolección de la muestra ***.....11***
Exámen físico ***.....11***
Sistema ***.....12***
Tipos de Sistemas ***.....12***
Interface: ***.....13***
Clave foránea ***.....15***
Estructura Cliente-Servidor ***.....16***
Base de Datos cliente servidor ***.....17***
Características más importantes que distinguen cliente/servidor ***.....18***
Componentes cliente – servidor ***.....17***
Elementos principales ***.....18***
La Era de la arquitectura cliente servidor ***.....20***
¿Que es un Cliente? ***.....21***
¿Que es un Servidor? ***.....21***
Las Comunicaciones ***.....21***
Infraestructura de comunicaciones ***.....21***
Infraestructura de redes ***.....21***
Características del Modelo Cliente/Servidor ***.....22***
Tipos de Clientes ***.....23***
Tipos de Servidor ***.....24***

Software y herramientas para la implementación del sistema ***.....25***
Java ***.....25***
Historia ***.....26***
Características del lenguaje Java ***.....27***
Máquina virtual Java ***.....30***
Instalar JDK ***.....32***
Herramientas necesarias ***.....35***
JDBC ***.....35***
Cargar el controlador JDBC ***.....37***
Establecer la conexión ***.....38***
Ejecución de consultas ***.....41***
PostgresSQL ***.....45***
Historia de PostgreSQL ***.....46***
Las principales mejoras en PostgreSQL ***.....49***
Pasos para la instalación de postgres ***.....49***
Autentificacion del cliente ***.....50***
¿Por qué Debian? ***.....54***
Data visión ***.....55***
Funcionamiento de la aplicación ***.....58***
Esquema del funcionamiento de la aplicación ***.....59***
Diseño metodológico ***.....60***
Materiales ***.....61***
Método de Análisis de Requisitos ***.....62***
Método del proceso de desarrollo ***.....62***
Forma de ciclo de vida en cascada incremental ***.....63***
D*efiniciones, Acrónimos Y Abreviaturas* ***.....64***
Análisis ***.....64***
F*unciones de la Aplicación* ***.....65***
Aplicación Principal Solicitud ***.....66***
Área de Uroanalisis ***.....68***

Banco de Sangre ***.....74***
Diagrama de flujo de datos ***.....80***
Diccionario de datos ***.....84***
Conclusiones ***.....88***
Recomendaciones ***.....89***
Bibliografía ***.....90***

1. INTRODUCCIÓN

En la actualidad son muchas las necesidades para automatizar la gestión de la información. Nosotros como estudiantes de Ingeniería en Sistemas de Información participaremos en el desarrollo de una aplicación donde pondremos en práctica los conocimientos adquiridos.

Esta aplicación deberá recepcionar datos de solicitudes de pacientes, almacenarlos en una base de datos, acceder desde las diferentes áreas para procesar los datos referente a cada una de ellas, procesar la información en cada área, realizar reportes estadísticos e impresión de resultados de exámenes de manera rápida y eficiente en el laboratorio en el Laboratorio Clínico HEODRA. Para llevar a cabo el desarrollo de dicho software será necesario especificar los requisitos software que el sistema deberá cumplir.

2. ANTECEDENTES

El laboratorio clínico HEODRA no cuenta con un sistema automatizado que permita el acceso inmediato y efectivo a información que es de vital importancia procesar, para agilizar la atención que se brinda a pacientes de este laboratorio. Este proceso actualmente es llevado a cabo manualmente.

3. JUSTIFICACIÓN

La computadora es una herramienta necesaria en la actualidad, siendo esta un elemento indispensable y necesario para el procesamiento de datos en cualquier área. La mayoría de las computadoras están dotadas de programas que permiten facilidad de manejo de la información, esto la hace una herramienta poderosa que se puede adecuar o moldear a las necesidades de cada usuario.

4. OBJETIVO GENERAL:

- Desarrollar un software para el laboratorio clínico del HEODRA, bajo entorno Linux, utilizando las siguientes herramientas: como gestor de base de datos PostgreSql, como lenguaje de programación JAVA 2 y para generar salidas de reportes DataVision. Dicho software tendrá como objetivo principal agilizar las tareas de atención al paciente y entrega de reportes estadísticos.

OBJETIVOS ESPECÍFICOS:

- Generar reportes estadísticos por área y por examen mensualmente.
- Generar reportes finales de las operaciones realizadas en el laboratorio.
- Automatizar la entrega de resultados a los pacientes.
- Control de calidad.

Estos objetivos serán alcanzados mediante la creación de una aplicación visual, que estará interactuando con una base de datos residente en un servidor, todo esto desarrollado con las herramientas antes mencionadas, para tratar dar una solución al problema de los registros en el laboratorio clínico HEODRA.

5. MARCO TEÓRICO

Laboratorio Clínico: Es una herramienta primordial para el área médica, ya que por medio de este se diagnostican diferentes patologías y además se realizan estudios para establecer el tipo de tratamiento que se debe administrar al paciente, al igual que el seguimiento del mismo.



El paciente o usuario llega al Laboratorio para realizarse sus exámenes clínicos, del Bacteriólogo y del Auxiliar depende que este usuario reciba el servicio adecuado en todo sentido, ya sea científico o humano, el profesional de la salud debe estar en condiciones de proporcionar una ayuda integral.

Esta investigación pretende ser una guía básica para comprender y despejar las dudas que se puedan presentar, se espera cumplir con las expectativas de fructificar y enriquecer **el conocimiento** en el área de sistemas hospitalarios.

5.1 SERVICIOS DEL LABORATORIO CLINICO

Cada examen de laboratorio clínico debe ser realizado a los pacientes de forma individual, guiándose siempre por los parámetros profesionales y éticos. Básicamente, **el trabajo** en el laboratorio clínico se clasifica en tres grandes **grupos** temáticos:

1. Toma de muestras.
2. **Análisis** de las muestras.
3. Entrega de resultados.

En cada uno de estos temas, se requiere de numerosas medidas de **atención** y cuidado, con el fin de minimizar al máximo los errores factibles de ser cometidos en la práctica diaria. Se debe enfatizar que **el trabajo** en el laboratorio clínico, como cualquier tipo de trabajo, es realizado por seres humanos y no se esta exento de cometer equivocaciones.

Pero estas equivocaciones pueden ser erradicadas de los laboratorios clínicos, si se mantienen eficientes **actitudes** éticas, profesionales y de **procedimiento**.

5.2 RAZONES PARA UTILIZAR Y AUTOMATIZAR LOS SERVICIOS DEL LABORATORIO CLÍNICO

1. Descubrir **enfermedades** en etapas subclínicas.
2. Ratificar un diagnostico sospechado clínicamente.
3. Obtener **información** sobre el pronóstico de una enfermedad.
4. Establecer un **diagnóstico** basado en una sospecha bien definida.
5. Vigilar un tratamiento o conocer una determinada respuesta terapéutica.
6. Evitar pérdidas de exámenes que anualmente asciende a más de 1,000.
7. Trabajar con mayor rapidez y eficiencia.
8. Obtener mejor acceso a la información.

6. BIOSEGURIDAD

Son todos los** procedimientos **y** acciones **que garantizan una mejor** calidad de vida**, tanto del profesional, del paciente y del** medio ambiente**.

Actualmente el Laboratorio Clínico del Hospital cuenta con 8 Áreas, en donde se realizan todos los diferentes tipos de exámenes, cada una de estas áreas esta dirigida por un responsable, existe una Unidad de Gestión de la Calidad que regula el buen funcionamiento de todo el Laboratorio. En nuestra aplicación trataremos con dos áreas **Banco de Sangre y Uroanálisis,** debido a las limitantes en equipos que esta institución pública posee, hemos seleccionado las dos areas más concurridas que producen mayor volumen de información, a continuación daremos una pequeña descripción de cada una de estas áreas.

7. BANCO DE SANGRE

RECOLECCION DE MUESTRAS DE SANGRE

Para una gran cantidad de estudios que requieren muestras sanguíneas, en algunos casos se debe conservar condiciones de ayuno, el cual puede prolongarse como mínimo seis horas y en ocasiones durante doce horas. En cualquiera de los casos, deben seguirse las siguientes indicaciones generales, a saber:

La sangre debe recolectarse en tubos de **vidrio** o **plástico** estériles (preferiblemente tubos al vacío). En caso de recolectar la sangre con jeringa y agujas estériles, deben llenarse los tubos con precisión y agilidad, evitando en todo momento realizar procedimientos bruscos que puedan producir rompimiento de las células sanguíneas (hemólisis). En otro tipo de estudios, la sangre no se deposita en tubos, sino en otro tipo de recipientes (frascos de hemocultivo).

Al recolectar la sangre, debe permitirse que se coagule, si es el caso, o someter los tubos con la muestra a ciertas maniobras recomendadas para evitar su coagulación.

En otras ocasiones, tan solo se colocan unas pequeñas gotas de sangre en láminas portaobjetos de vidrio, (extendidos de sangre periférica), en capilares de vidrio o en placas de vidrio o **plástico** de origen comercial para la realización de algunos estudios.

8. *UROANÁLISIS*

Orina, líquido excretado por los riñones a través de las vías urinarias, con el cual se eliminan sustancias innecesarias para el organismo. Desempeña un **papel** importante en la regulación del balance de líquidos y electrolitos y del **equilibrio** entre **ácidos** y bases. La cantidad de orina producida diariamente es de 1 a 1,5 litros, valor que aumenta si se ingieren muchos líquidos y disminuye en caso de sudoración intensa. Las muestras de orina son biopsias líquidas de los tejidos del tracto urinario, recolectadas en forma indolora que permiten tener **información** rápida y económica.

9. RECOLECCIÓN DE LA MUESTRA:

La muestra se recoge normalmente por micción espontanea, tener en cuenta que se debe recoger la primera de la mañana, el paciente debe levantarse, asearse muy bien los genitales y en un recipiente estéril recoger la micción intermedia.

10. *EXÁMEN FÍSICO:*

Aspecto: Es considerado como normal un aspecto transparente, pero es aceptado hasta un aspecto ligeramente turbio ya que este puede ser debido a contaminaciones. El aspecto de una orina turbia ya es considerado como anormal, esto puede ser debido a presencia de leucocitos, glóbulos rojos, bacterias, cristales, grasa (Por obstrucción de linfáticos).

Color: En condiciones normales el **color** de la orina va de amarillo hasta ámbar. Se pueden encontrar **colores** anormales debido a la presencia de elementos anormales en la orina como por ejemplo sangre, medicamentos, alimentos y otros pigmentos.

Incolora: se conoce como **HIDRURICA** característica de una **diabetes** insípida se presenta por baja en la **producción** de Hormona antidiurética.

Rosado o Rojo: Se presenta por la presencia aumentada de Urobilinogeno, porfobilinogeno.

Azul: después de procesos quirúrgicos.

Amarillo intenso: pigmentos biliares.

Negro: melanomas productores de melanina.**pH:** Es el reflejo de la capacidad del riñón para mantener la concentración normal de hidrogeniones. El **pH** normal va de 5.5 - 6.5. Influyendo el régimen dietético en cada paciente.

11. ***Sistema:*** un conjunto de elementos dinámicamente relacionados formando una actividad para alcanzar un **objetivo** operando sobre **datos** para proveer **información**.

Características de los sistemas: **s**istema es un todo organizado y complejo; un conjunto o combinación de cosas o partes que forman un todo complejo o unitario. Es un conjunto de objetos unidos por alguna forma de interacción o interdependencia. Los **límites** o fronteras entre el sistema y su ambiente admiten cierta arbitrariedad.

Según Bertalanffy, sistema es un conjunto de unidades recíprocamente relacionadas. De ahí se deducen dos conceptos: propósito (u objetivo) y globalismo (o totalidad).

- ***Propósito u objetivo:*** todo sistema tiene uno o algunos propósitos. Los elementos (u objetos), como también las relaciones, definen una **distribución** que trata siempre de alcanzar un objetivo.
- ***Globalismo o totalidad:*** un **cambio** en una de las unidades del sistema, con **probabilidad** producirá cambios en las otras. El efecto total se presenta como un ajuste a todo el sistema. Hay una relación de causa/efecto. De estos cambios y ajustes, se derivan dos fenómenos: **entropía** y homeostasia.
- ***Entropía:*** es la tendencia de los sistemas a desgastarse, a desintegrarse, para el relajamiento de los estándares y un aumento de la aleatoriedad. La **entropía** aumenta con el correr del **tiempo**. Si aumenta la información, disminuye la entropía, pues la información es la base de la configuración y del orden. De aquí nace la negentropía, o sea, la información como medio o instrumento de ordenación del sistema.
- ***Homeostasia:*** es el **equilibrio** dinámico entre las partes del sistema. Los sistemas tienen una tendencia a adaptarse con el fin de alcanzar un **equilibrio** interno frente a los cambios externos del entorno.

12. Tipos de Sistemas

En cuanto a su **constitución**, pueden ser físicos o abstractos:

- Sistemas físicos o concretos: compuestos por equipos, maquinaria, objetos y cosas reales. El **hardware**.

- Sistemas abstractos: compuestos por conceptos, planes, **hipótesis** e ideas. Muchas veces solo existen en el **pensamiento** de las personas. Es el **software**.

En cuanto a su naturaleza, pueden cerrados o abiertos:

- Sistemas cerrados: no presentan intercambio con el **medio ambiente** que los rodea, son herméticos a cualquier influencia ambiental. No reciben ningún **recurso** externo y nada producen que sea enviado hacia fuera. En rigor, no existen sistemas cerrados. Se da el nombre de sistema cerrado a aquellos sistemas cuyo **comportamiento** es determinístico y programado y que opera con muy pequeño intercambio de energía y materia con el ambiente. Se aplica el término a los sistemas completamente estructurados, donde los elementos y relaciones se combinan de una manera peculiar y rígida produciendo una salida invariable, como las **máquinas**.
- Sistemas abiertos: presentan intercambio con el ambiente, a través de entradas y salidas. Intercambian energía y materia con el ambiente. Son adaptativos para sobrevivir. Su estructura es óptima cuando el conjunto de elementos del sistema se organiza, aproximándose a una operación adaptativa. La adaptabilidad es un continuo **proceso** de **aprendizaje** y de auto-organización.

Los sistemas abiertos no pueden vivir aislados. Los sistemas cerrados, cumplen con el segundo principio de la **termodinámica** que dice que "una cierta cantidad llamada entropía, tiende a aumentar al máximo".

Existe una tendencia general de los **eventos** en la naturaleza **física** en **dirección** a un **estado** de máximo desorden. Los sistemas abiertos evitan el aumento de la entropía y pueden desarrollarse en **dirección** a un **estado** de creciente orden y organización (entropía negativa). Los sistemas abiertos restauran su propia energía y reparan pérdidas en su propia organización. El concepto de sistema abierto se puede aplicar a diversos niveles de enfoque: al nivel del individuo, del **grupo**, de **la organización** y de la **sociedad**.

13. ***Interfaces:*** Forma en la que el ordenador establece la comunicación con el usuario, actualmente casi todas son de modo "gráfico" donde se nos presentan en forma de gráficos o iconos, los elementos o acciones que podemos realizar con el ordenador,

antiguamente por ejemplo con MS-DOS la interface era de "línea de comandos", donde el operador escribía el nombre del comando (dir, copy, format, etc...) para que el ordenador ejecutara esa orden.

***Interfaz de usuario*:** es la parte del programa que permite a éste interactuar con el usuario, pueden adoptar muchas formas, que van desde la simple línea de comandos hasta las interfaces gráficas que proporcionan las aplicaciones más modernas.

La interfase de usuario es el aspecto más importante de cualquier aplicación. Una aplicación sin un interfaz fácil, impide que los usuarios saquen el máximo rendimiento del programa. Java proporciona los elementos básicos para construir *decentes* interfaces de usuario a través del AWT (Abstract Windows toolkit) y JFC (Java Foundation Class).

Aplicación de consola vs. Aplicación con Interfaz Gráfica de Usuario

Los elementos que componen la interfaz gráfica son elementos gráficos, y a través de ellos el usuario puede interactuar con la aplicación. En esta interacción el usuario introduce datos que el programa necesita para llevar a cabo su funcionalidad y obtiene los resultados de procesar dichos datos. Por ejemplo, las ventanas, los botones, las imágenes, etc. Son elementos gráficos. Una diferencia clara entre una aplicación de consola y una aplicación con interfaz gráfica de usuario, es que la primera no tiene ningún elemento gráfico, mientras que en la segunda éstos si existen. Por otra parte, un evento es la notificación que hace un elemento gráfico cuando el usuario interactúa con él. Por lo tanto, si se realiza alguna acción sobre algún elemento de la interfaz, se dice que se ha generado un evento en dicho elemento. Otra diferencia destacable entre una aplicación de consola y una con interfaz gráfica de usuario está relacionada con los eventos y su gestión, y afecta notablemente a la hora de programar la aplicación. En una aplicación de consola el programa decide cuándo necesita datos del usuario, y es en ese momento cuando los lee de la cadena de entrada. Sin embargo, una aplicación con interfaz gráfica siempre se encuentra a la espera de una entrada de datos por parte del usuario. Éste, en cualquier momento, puede realizar alguna acción sobre algún elemento de la interfaz (por ejemplo, pulsar con el ratón sobre un botón, introducir un carácter por teclado, etcétera).

Para poder atender las acciones realizadas sobre los elementos de la interfaz gráfica de usuario, es necesario asociar código a los eventos que se puedan generar como consecuencia de dichas acciones. De esta manera, si se asocia código a un evento concreto, éste se ejecutará cuando se realice la acción que genera el evento sobre un elemento de la interfaz. Al código asociado a los eventos se le denomina código de gestión de eventos. A la hora de programar una aplicación, dependiendo si ésta es de consola o con interfaz gráfica, dadas las diferencias existentes entre ellas (existencia o no de elementos gráficos y tratamiento de eventos), los pasos a seguir serán bastante distintos.

14. ***Clave foránea:*** una clave foránea es un atributo de una relación, cuyos valores se corresponden con los de una clave primaria en otra o en la misma relación. Este mecanismo se usa para establecer interrelaciones.

Las situaciones donde puede violarse la integridad referencial es en el borrado de tuplas o en la modificación de claves principales. Si se elimina una tupla cuya clave primaria se usa como clave foránea en otra relación, las tuplas con esos valores de clave foránea contendrán valores sin referenciar.

Existen varias formas de asegurarse de que se conserva la integridad referencial:

Restringir operaciones: borrar o modificar tuplas cuya clave primaria es clave foránea en otras tuplas, sólo estará permitido si no existe ninguna tupla con ese valor de clave en ninguna otra relación.

- Es decir, si el valor de una clave primaria en una tupla es "clave1", sólo podremos eliminar esa tupla si el valor "clave1" no se usa en ninguna otra tupla, de la misma relación o de otra, como valor de clave foránea.
- ***Transmisión en cascada:*** borrar o modificar tuplas cuya clave primaria es clave foránea en otras implica borrar o modificar las tuplas con los mismos valores de clave foránea.

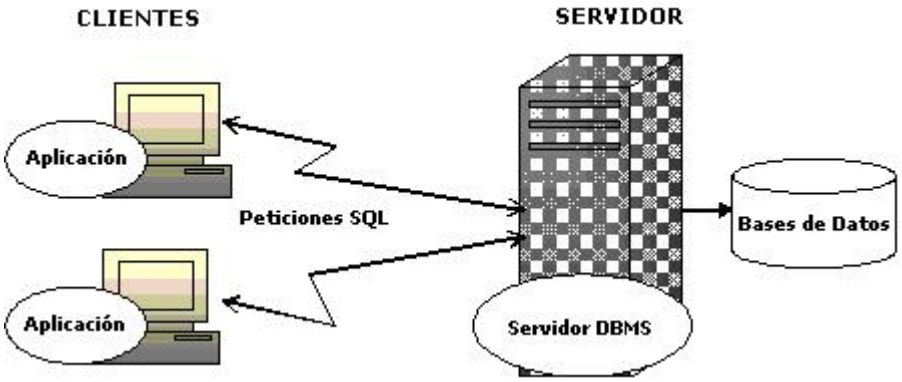
- Si en el caso anterior, modificamos el valor de clave primaria "clave1" por "clave2", todas las apariciones del valor "clave1" en donde sea clave foránea deben ser sustituidos por "clave2".
- Poner a nulo: cuando se elimine una tupla cuyo valor de clave primaria aparece en otras relaciones como clave foránea, se asigna el valor NULL a dichas claves foráneas.

  De nuevo, siguiendo el ejemplo anterior, si eliminamos la tupla con el valor de clave primaria "clave1", en todas las tuplas donde aparezca ese valor como clave foránea se sustituirá por NULL.

15. ***Estructura Cliente-Servidor:*** Cualquier combinación de **sistemas** que pueden colaborar entre si para dar a los usuarios toda la información que ellos necesiten sin que tengan que saber donde esta ubicada.

- Es una arquitectura de procesamientos cooperativo donde uno de los componentes pide **servicios** a otro.
- Es un **procesamiento de datos** de índole colaborativo entre dos o más **computadoras** conectadas a **una red**.
- El término cliente/servidor es originalmente aplicado a la arquitectura de **software** que describe el procesamiento entre dos o más **programas**: una aplicación y un **servicio** soportante.
- IBM define al **modelo** Cliente/Servidor. "Es la **tecnología** que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del **grupo** de trabajo y/o, a través de **la organización**, en múltiples plataformas. El modelo soporta un **medio ambiente** distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o "**clientes**", resultan en un trabajo realizado por otros computadores llamados **servidores**".
- "Es un modelo para construir **sistemas de información**, que se sustenta en la idea de repartir el tratamiento de la información y los datos por todo el **sistema** informático, permitiendo mejorar el rendimiento del sistema global de información"

16. Base de Datos cliente servidor



El cliente envía mensajes que representados en solicitudes SQL hacia el servidor de bases de datos. Los resultados de cada orden de SQL son devueltos al cliente. El DBMS se encarga de recolectar los datos desde su base de datos, no envía los registros completos, teniéndose un uso mucho más eficiente de la capacidad de procesamiento distribuida. Es usual que se generen aplicaciones en el cliente y en el servidor. Los servidores de bases de datos constituyen el fundamento de los sistemas de apoyo de decisiones que precisan de consultas específicas y reportes flexibles.

16.1 Componentes cliente – servidor

Como su nombre lo indica esta compuesta por Clientes y Servidores, pero además oculto en este nombre se encuentra los mecanismos de interacción entre ellos, clave en las aplicaciones de este tipo. Orfali lo asocia al / de Cliente / Servidor y se denomina Middleware.

"***Middleware*** es un término vago que abarca a todo el software distribuido necesario para el soporte de interacciones entre clientes y servidores. Imagínelo como el software que ocupa la parte intermedia del sistema de cliente/servidor. Es el enlace que permite que un cliente obtenga un servicio de un servidor. ¿Dónde empieza y dónde termina el middleware? Empieza en el módulo de API de la parte del cliente que se emplea para invocar un servicio y comprende la transmisión de la solicitud por la red y la respuesta resultante. Pero no incluye al software que presta el servicio real; esto pertenece a los dominios del servidor. Tampoco a la interfaz del usuario ni a la lógica de la aplicación, en los dominios del cliente".

Clientes y servidores son entidades lógicas independientes que operan en conjunto a través de una red para realizar una tarea. Se prefiere cambiar la palabra red, por mecanismos de comunicación. Pues clientes y servidores pueden estar en la misma máquina.

16.2 Las características más importantes que se distinguen cliente/servidor son:

- ***Orientado a servicios.*** El servidor los ofrece y el cliente los consume.

- ***Compartición de recursos.*** Servicios ofrecidos a muchos clientes. Un servidor puede atender muchos clientes que solicitan esos servicios.

- ***Transparencia de ubicación.*** El servidor es un proceso que puede residir en el mismo aparato que el cliente o en un aparato distinto a lo largo de una red. Un programa puede ser un servidor en un momento y convertirse en un cliente posteriormente.

- ***Mezcla e igualdad***. Tal vez de las más importantes ventajas de este paradigma. Una aplicación cliente/servidor, idealmente es independiente del hardware y de sistemas operativos; mezclando e igualando estas plataformas.

- Interacción a través de mensajes, para envío y respuesta de servicios.
- Servicios encapsulados, exponiendo los servicios a través de interfaces, lo que facilita la sustitución de servidores son afectar los clientes; permitiendo a la vez una fácil escalabilidad.

16.3 Elementos principales

"Los elementos principales de la arquitectura cliente servidor son justamente el elemento llamado cliente y el otro elemento llamado servidor". Por ejemplo dentro de un **ambiente multimedia**, el elemento cliente seria el dispositivo que puede observar el vídeo, cuadros y **texto**, o reproduce el audio distribuido por el elemento servidor.

Por otro lado el cliente también puede ser una **computadora personal** o una **televisión** inteligente que posea la capacidad de entender datos digitales. Dentro de este caso el elemento servidor es el depositario del vídeo digital, audio, fotografías digitales y texto

y los distribuye bajo **demanda** de ser una maquina que cuenta con la capacidad de almacenar los datos y ejecutar todo el software que brinda éstos al cliente.

Con el **objetivo** de definir y delimitar el modelo de referencia de una arquitectura Cliente/Servidor, debemos identificar los componentes que permitan articular dicha arquitectura, considerando que toda aplicación de un **sistema de información** está caracterizada por tres componentes básicos:

- **Presentación/Captación de Información**
- **Procesos**
- **Almacenamiento de la Información**

Algunos antecedentes, ¿porque fue creado?

Existen diversos puntos de vista sobre la manera en que debería efectuarse el procesamiento de datos, aunque la mayoría que opina, coincide en que nos encontramos en medio de un **proceso** de **evolución** que se prolongará todavía por algunos años y que cambiará la forma en que obtenemos y utilizamos la información almacenada electrónicamente.

El principal motivo detrás de esta evolución es la necesidad que tienen las **organizaciones** (**empresas** o **instituciones** públicas o privadas), de realizar sus **operaciones** más ágil y eficientemente, debido a la creciente **presión** competitiva a la que están sometidas, lo cual se traduce en la necesidad de que su personal sea mas productivo, que se reduzcan los **costos** y **gastos** de operación, al mismo **tiempo** que se generan **productos** y servicios más rápidamente y con mejor **calidad**.

En este contexto, es necesario establecer una infraestructura de procesamiento de información, que cuente con los elementos requeridos para proveer información adecuada, exacta y oportuna en la **toma de decisiones** y para proporcionar un mejor servicio a los clientes.

El modelo Cliente/Servidor reúne las características necesarias para proveer esta infraestructura, independientemente del tamaño y complejidad de las operaciones de las

organizaciones públicas o privadas y, consecuentemente desempeña un **papel** importante en este proceso de evolución.

16.4 La Era de la arquitectura cliente servidor

"En esta arquitectura la computadora de cada uno de los usuarios, llamada cliente, produce una demanda de información a cualquiera de las computadoras que proporcionan información, conocidas como servidores", estos últimos responden a la demanda del cliente que la produjo.

Los clientes y los servidores pueden estar conectados a una **red** local o una red amplia, como la que se puede implementar en **una empresa** o a una red mundial como lo es la **Internet**.

Bajo este modelo cada usuario tiene la **libertad** de obtener la información que requiera en un momento dado proveniente de una o varias **fuentes** locales o distantes y de procesarla como según le convenga. Los distintos servidores también pueden intercambiar información dentro de esta arquitectura.

16.5 ¿Que es una Arquitectura?

Una arquitectura es un entramado de componentes funcionales que aprovechando diferentes estándares, convenciones, reglas y procesos, permite integrar una amplia gama de productos y servicios informáticos, de manera que pueden ser utilizados eficazmente dentro de la **organización**.

Debemos señalar que para seleccionar el modelo de una arquitectura, hay que partir del contexto tecnológico y organizativo del momento y, que la arquitectura Cliente/Servidor requiere una determinada especialización de cada uno de los diferentes componentes que la integran.

16.6 ¿Que es un Cliente?

Es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de **redes LAN** o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

16.7 ¿Que es un Servidor?

Es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de **redes** LANs o WANs, para proveer de múltiples servicios a los clientes y ciudadanos tales como impresión, acceso a bases de datos, **fax**, procesamiento de **imágenes**, etc.

16.8 Las Comunicaciones

En sus dos vertientes:

- Infraestructura de redes
- Infraestructura de comunicaciones

16.9 Infraestructura de redes

Componentes **Hardware** y Software que garantizan la conexión **física** y la transferencia de datos entre los distintos equipos de la red.

16.10 Infraestructura de comunicaciones

Componentes Hardware y Software que permiten **la comunicación** y su **gestión**, entre los clientes y los servidores.

La arquitectura Cliente/Servidor es el resultado de la **integración** de dos culturas. Por un lado, la del Mainframe que aporta capacidad de **almacenamiento**, integridad y acceso a la información y, por el otro, la del computador que aporta facilidad de uso (**cultura** de PC), bajo **costo**, presentación atractiva (aspecto lúdico) y una amplia **oferta** en productos y aplicaciones.

16.11 Características del Modelo Cliente/Servidor

En el modelo Cliente/Servidor podemos encontrar las siguientes características:

1. El Cliente y el Servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas, realizando actividades o tareas independientes.

2. Las **funciones** de Cliente y Servidor pueden estar en plataformas separadas, o en la misma plataforma.

Para ver el gráfico seleccione la opción ¨Descargar trabajo¨ del menú superior

3. Un servidor da servicio a múltiples clientes en forma concurrente.

4. Cada plataforma puede ser escalable independientemente. Los cambios realizados en las plataformas de los Clientes o de los Servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.

5. La interrelación entre el hardware y el software están basados en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no **muestra** la complejidad de los diferentes tipos de formatos de datos y de los **protocolos**.

6. Un sistema de servidores realiza múltiples funciones al mismo tiempo que presenta una imagen de un solo sistema a las estaciones Clientes**.** Esto se logra combinando los recursos de cómputo que se encuentran físicamente separados en un solo sistema lógico, proporcionando de esta manera el servicio más efectivo para el usuario final.

También es importante hacer notar que las funciones Cliente/Servidor pueden ser dinámicas. Ejemplo, un servidor puede convertirse en cliente cuando realiza la solicitud de servicios a otras plataformas dentro de la red.

Su capacidad para permitir integrar los equipos ya existentes en una organización, dentro de una arquitectura **informática** descentralizada y heterogénea.

7. Además se constituye como el nexo de unión mas adecuado para reconciliar los sistemas de información basados en mainframes o minicomputadores, con aquellos otros sustentados en entornos informáticos pequeños y estaciones de trabajo.

8. Designa un modelo de **construcción** de sistemas informáticos de **carácter** distribuido.

1. Su representación típica es un centro de trabajo (PC), en donde el usuario dispone de sus propias aplicaciones de **oficina** y sus propias bases de datos, sin dependencia directa del sistema central de información de la organización, al tiempo que puede acceder a los
2. recursos de este host central y otros sistemas de la organización ponen a su servicio.

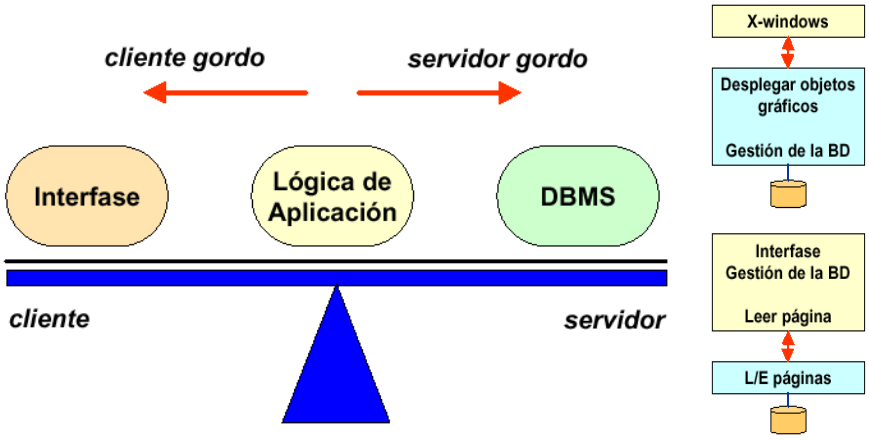
16.12 Tipos de Clientes

1. "Cliente flaco":

- Servidor rápidamente saturado.
- Gran circulación de datos de interfase en la red.

1. **"Cliente gordo":**

- Casi todo **el trabajo** en el cliente.
- No hay **centralización** de la gestión de la BD.
- Gran circulación de datos inútiles en la red.



16.13 Tipos de Servidor

Servidores de archivos

Servidor donde se almacena archivos y aplicaciones de **productividad** como por ejemplo **procesadores de texto**, **hojas de cálculo**, etc.

Servidores de bases de datos

Servidor donde se almacenan las bases de datos, tablas, índices. Es uno de los servidores que más carga tiene.

Servidores de transacciones

Servidor que cumple o procesa todas las transacciones. Valida primero y recién genera un pedido al servidor de bases de datos.

Servidores de Groupware

Servidor utilizado para el seguimiento de operaciones dentro de la red.

Servidores de objetos

Contienen objetos que deben estar fuera del servidor de base de datos. Estos objetos pueden ser videos, imágenes, objetos multimedia en general.

Servidores Web

Se usan como una forma inteligente para **comunicación** entre empresas a través de Internet.Este servidor permite transacciones con el acondicionamiento de un browser específico.En conclusión, Cliente/Servidor puede incluir múltiples plataformas, bases de datos, redes y **sistemas operativos**. Estos pueden ser de distintos **proveedores**, en arquitecturas propietarias y no propietarias y funcionando todos al mismo tiempo. Por lo tanto, su implantación involucra diferentes tipos de estándares: APPC, **TCP/IP, OSI**, NFS, DRDA corriendo sobre DOS, OS/2, **Windows** o PC **UNIX**, en TokenRing, **Ethernet**, FDDI o medio coaxial, sólo por mencionar algunas de las posibilidades.

17. SOFTWARE Y HERRAMIENTAS PARA LA IMPLEMENTACIÓN DEL SISTEMA

17.1.3 JAVA

Es un lenguaje de programación orientado a objetos, actualmente una de las tecnologías informáticas con una mayor difusión dentro de este sector, desarrollado por James Gosling y sus compañeros de Sun Microsystems al inicio de la década de 1990, para un uso en dispositivos electrónicos de consumo. Sin embargo, pronto se entendió que sus características se adaptaban perfectamente a las condiciones de Internet, y fue aqui, donde finalmente fue aplicado. En los pocos años que lleva de vida, ha sufrido un gran desarrollo y aceptación .A diferencia de los lenguajes de programación convencionales, que generalmente están diseñados para ser compilados a código nativo, Java es compilado en un bytecode que es ejecutado por una máquina virtual Java.

El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos mucho más simple y elimina herramientas de bajo nivel como punteros.

Java está sólo lejanamente emparentado con JavaScript, aunque tengan nombres similares y compartan una sintaxis al estilo de C algo parecida.

El componente básico que se necesita para programar Java es su kit de desarrollo (formado fundamentalmente por: una librería de clases, compilador, máquina virtual y otras herramientas). Este kit es conocido como JDK (Java Development Kit).

- **Servlets y JSP**, para programar aplicaciones web.
- **EJB**, para el desarrollo de componentes transaccionales y multiplaforma en el servidor.
- **JNDI**, para acceder a servicios de directorio (p.e. LDAP).
- **JDBC**, para acceder a base de datos.
- **JMS**, para utilizar todo tipo de servicios de mensajería (comunicación asíncrona).
- **Java 3D**, para el desarrollo de interfaces tridimensionales.
- **J2ME**, es la plataforma Java para el desarrollo de aplicaciones para pequeños dispositivos electrónicos.

17.1.3 Historia

Orígenes

La **plataforma Java** y el lenguaje Java empezaron como un proyecto interno de **Sun Microsystems** en diciembre de 1990. Patrick Naughton, ingeniero de Sun, estaba decepcionado con el estado de C++ y la **API** de C y sus herramientas. Mientras consideraba migrar a **NeXT**, Naughton recibió la oferta de trabajar en una nueva tecnología, y así comenzó el proyecto *Stealth*.

El Proyecto Stealth fue rebautizado como *Green Project* (o *Proyecto Verde*) cuando James Gosling y Mike Sheridan se unieron a Naughton. Con la ayuda de otros ingenieros, empezaron a trabajar en una pequeña oficina en Sand Hill Road en Menlo Park, California. Intentaban desarrollar una nueva tecnología para programar la siguiente generación de *dispositivos inteligentes*, en los que Sun veía un campo nuevo a explotar.

El equipo pensó al principio usar C++, pero se descartó por varias razones. Al estar desarrollando un sistema empotrado con recursos limitados, C++ no es adecuado por necesitar mayor potencia además de que su complejidad conduce a errores de desarrollo. La ausencia de un *recolector de basura* (*garbage collector*) obligaba a los programadores a manejar manualmente el sistema de memoria, una tarea peligrosa y proclive a fallos. El equipo también se encontró con problemas por la falta de herramientas portables en cuanto a seguridad, programación distribuida, y programación concurrente. Finalmente abogaban por una plataforma que fuese fácilmente portable a todo tipo de dispositivo.

Bill Joy había concebido un nuevo lenguaje que combinase lo mejor de *Mesa* y C. En un escrito titulado *Further* (más lejos), proponía a Sun que sus ingenieros crearan un entorno *orientado a objetos* basado en C++. Al principio Gosling intentó modificar y ampliar C++, a lo que llamó C++ ++ --, pero pronto descartó la idea para crear un lenguaje completamente nuevo, al que llamó *Oak*, en referencia al roble que tenía junto a su oficina.

El equipo dedicó largas horas de trabajo y en el verano de 1992 tuvieron lista algunas partes de la plataforma, incluyendo el Sistema Operativo Green, el lenguaje Oak, las librerías y el hardware. La primera prueba, llevada a cabo el 3 de Septiembre de 1992, se centró en construir una **PDA** (*Personal Digital Assistant* o Asistente Digital Personal) llamada *Star7*, que contaba con una interfaz gráfica y un asistente apodado "Duke" para guiar al usuario.

En noviembre de ese mismo año, el Proyecto Verde se convirtió en **FirstPerson, Inc**, una división propiedad de **Sun Microsystems**, y el equipo se trasladó a **Palo Alto** (California). El interés se centró entonces en construir dispositivos interactivos, hasta que Time Warner publicó una solicitud de oferta para un adaptador de televisión. Es decir, un aparato que se sitúa entre la televisión y una fuente de señal externa y que adapta el contenido de ésta (video, audio, páginas Web, etc.) para verse en la pantalla. Entonces, Firsperson cambió de idea y envió a Warner una propuesta para el dispositivo que deseaban. Sin embargo, la industria del cable consideró que esa propuesta daba demasiado control al usuario, con lo que FirstPerson perdió la puja a favor de **Silicon Graphics Incorporated**. Un trato con la empresa **3DO** para el mismo tipo de dispositivo tampoco llegó a buen puerto. Viendo que no había muchas posibilidades en la industria de la televisión, la compañía volvió al seno de Sun.

17.1.3Características del lenguaje Java

Lenguaje simple

Java posee una curva de aprendizaje muy rápida. Resulta relativamente sencillo escribir applets interesantes desde el principio. Todos aquellos familiarizados con C++ encontrarán que Java es más sencillo, ya que se han eliminado ciertas características, como los punteros. Debido a su semejanza con C y C++, y dado que la mayoría de la gente los conoce aunque sea de forma elemental, resulta muy fácil aprender Java. Los programadores experimentados en C++ pueden migrar muy rápidamente a Java y ser productivos en poco tiempo.

Orientado a objetos

Java fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos. La tendencia del futuro, a la que Java se suma, apunta hacia la programación orientada a objetos, especialmente en entornos cada vez más complejos y basados en red.

Distribuido

Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.

Interpretado y compilado a la vez

Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los bytecodes, semejantes a las instrucciones de ensamblador. Por otra parte, es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (run-time).

Robusto

Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria.

Seguro

Dada la naturaleza distribuida de Java, donde las applets se bajan desde cualquier punto de la Red, la seguridad se impuso como una necesidad de vital importancia. A nadie le gustaría ejecutar en su ordenador programas con acceso total a su sistema, procedentes

de fuentes desconocidas. Así que se implementaron barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real.

Indiferente a la arquitectura

Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows Nt, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. Para acomodar requisitos de ejecución tan variopintos, el compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura diseñado para transportar el código eficientemente a múltiples plataformas hardware y software. El resto de problemas los soluciona el intérprete de Java.

Portable

La indiferencia a la arquitectura representa sólo una parte de su portabilidad. Además, Java especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas.
Estas dos últimas características se conocen como la *Máquina Virtual Java* (JVM).

Multihebra

Hoy en día ya se ven como terriblemente limitadas las aplicaciones que sólo pueden ejecutar una acción a la vez. Java soporta sincronización de múltiples hilos de ejecución (*multithreading*) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario mientras otro presenta una animación en pantalla y otro realiza cálculos.

Dinámico

El lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas. Se pueden enlazar

nuevos módulos de código bajo demanda, procedente de fuentes muy variadas, incluso desde la Red.

Produce applets

Java puede ser usado para crear dos tipos de programas: aplicaciones independientes y applets.
Las aplicaciones independientes se comportan como cualquier otro programa escrito en cualquier lenguaje, como por ejemplo el navegador de Web HotJava, escrito íntegramente en Java. Por su parte, las applets son pequeños programas que aparecen embebidos en las páginas Web, como aparecen los gráficos o el texto, pero con la capacidad de ejecutar acciones muy complejas, como animar imágenes, establecer conexiones de red, presentar menús y cuadros de diálogo para luego emprender acciones, etc.

17.2.1Máquina virtual Java

El método tradicional de implementación de Java es por medio de un intérprete que corresponde a la máquina virtual de Java. Un compilador permite traducir el código fuente en archivos "class" que contienen instrucciones en bytecode (independientes de la máquina). Estos archivos "class" son recuperados e interpretados por la máquina virtual. Los servicios comunes son ofrecidos en forma de bibliotecas de clases o "so" archivos de bibliotecas compartidos. Las bibliotecas de clases proveen servicios a la JVM y a los programas en bytecode, en particular el soporte de lenguaje básico y las funcionalidades extendidas. Una biblioteca de tiempo de ejecución provee el elemento de bajo nivel llamado "Recolección de Desperdicios", el soporte de "hilos" y ejecuta directamente en el hardware de la máquina.

Fue pensada como un sistema muy sencillo, ya que debía ser albergado en dispositivos electrónicos de consumo, por lo que debemos alejarnos rápidamente de conceptos como los del PC, o los mini ordenadores, arquitecturas infinitamente más complejas que esta.

La JVM es un ordenador abstracto que es capaz de ejecutar programas Java compilados. El término virtual es debido a que inicialmente no existían arquitecturas reales Java, por lo la JVM se solía implementar por software sobre plataformas ya existentes. Pero poco a poco van apareciendo los chips Java, que darán apoyo a la aparición de máquinas “reales” Java. Como es de esperar al ser una máquina virtual, el rendimiento que logra en la ejecución del código Java está muy alejado de las aplicaciones desarrolladas para una plataforma específicamente, pero este factor que es muy importante, se ve compensado por otro factor aún más importante: *la portabilidad.*

Debido a la sencillez de diseño de la JVM, rápidamente se ha extendido a todas las arquitecturas existentes, logrando una capa software común para todas ellas. Este es un hecho clave ya que los programas Java pueden ser ejecutados indistintamente en cualquier plataforma, por lo que redes de ordenadores heterogéneas se esta convirtiendo en una piedra angular en el funcionamiento de las mismas.

17.2.1 **Instalar JDK**

La base para poder operar cualquier producto que utiliza java es el JDK de la plataforma correspondiente ("Write Once, Run Everywhere"), por lo cual antes de correr cualquier aplicacion java, se debe instalar el JDK, se puede descargar desde el sitio web de Sun que está en:

http://java.sun.com/j2se/

Una vez obtenigo el paquete, el usuario deberá colocarse en el directorio donde descargó el archivo j2sdk-1_4_1_02-linux-i586.bin.

Como será necesario escribir en algunos directorios que normalmente están restringidos al usuario normal, hay que cambiarse como usuario "root" (el administrador del sistema) con la siguiente instrucción:

debian:/home/usuario#su

NOTA:

Al ejecutar la instrucción "su -", el sistema operativo solicitará la contraseña correspondiente a root.

A continuación se deben cambiar los permisos del archivo para poderlo ejecutar, con la siguiente instrucción:

debian:/home/usuario#chmod 755 j2sdk-1_4_1_02-linux-i586.bin

Ahora ejecútelo, con la instrucción:

debian:/home/usuario# ./j2sdk-1_4_1_02-linux-i586.bin

Al ejecutar el programa, éste pregunta si el usuario está de acuerdo con los términos de la licencia. Aceptamos escribiento "yes".

El programa, al ser ejecutado, crea un directorio donde está la versión de java que vamos a instalar.

Se observa que el nombre del directorio es muy largo y difícil de recordar, por lo que se recomienda cambiarlo por uno más sencillo como por ejemplo:

debian:/home/usuario#mv j2sdk-1_4_1_02-linux-i586 java

El siguiente paso es mover el archivo al directorio donde se desea hacer la instalación.Ejemplo:

debian:/home/usuario# mv java /usr/local

Se colocaron los binarios de java en un directorio donde los usuarios pueden ejecutarlo. Ahora es necesario configurar la variable de ambiente PATH, agregando 3 líneas al final del archivo /etc/profile, para que sea válido para todos los usuarios del sistema de la siguiente manera:

debian: /home/usuario# vi /etc/profile

Y las líneas que se deben de agregar al final del archivo son:
export JAVA_HOME=/usr/local/java
export JAVA_BIN=/usr/local/java/bin
export PATH=$PATH:$JAVA_HOME:$JAVA_BIN

Para que los cambios tengan efecto se debe hacer que el procesador los lea mediante la siguiente instrucción:

source /etc/profile

Finalmente se prueba que java esté correctamente instalado, ejecutándolo de la siguiente forma:

debian:/home/usuario#java
Usage: java [-options] class [args...]
(to execute a class)
or java [-options] -jar jarfile [args...]
(to execute a jar file)

where options include:
-client to select the "client" VM

-server to select the "server" VM
-hotspot is a synonym for the "client" VM [deprecated]
The default VM is client.

-cp <class search path of directories and zip/jar files>
-classpath <class search path of directories and zip/jar files>
A : separated list of directories, JAR archives,
and ZIP archives to search for class files.
-D<name>=<value>
set a system property
-verbose[:class|gc|jni]
enable verbose output
-version print product version and exit
-version:<value>
require the specified version to run
-showversion print product version and continue
-jre-restrict-search | -jre-no-restrict-search
include/exclude user private JREs in the version search
-? -help print this help message
-X print help on non-standard options
-ea[:<packagename>...|:<classname>]
-enableassertions[:<packagename>...|:<classname>]
enable assertions
-da[:<packagename>...|:<classname>]
-disableassertions[:<packagename>...|:<classname>]
disable assertions
-esa | -enablesystemassertions
enable system assertions
-dsa | -disablesystemassertions
disable system assertions

Java debe responder y dar datos de sí mismo, como por ejemplo su versión. En caso de no ser así se deberan revisar cada una de las instrucciones y vuelver a probar, es

necesario que este paso esté correcto para poder comenzar a compilar y jecutar programas java.

17.3 JDBC

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

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

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

Los distribuidores de bases de datos suministran los controladores que implementan el API JDBC y que permiten acceder a sus propias implementaciones de bases de datos. De esta forma JDBC proporciona a los programadores de Java una interfaz de alto nivel y les evita el tener que tratar con detalles de bajo nivel para acceder a bases de datos. **En el caso del manejador de bases de datos para PostgreSQL es postgresql-8.1-407.jdbc2.jar, el driver JDBC oficial**.

17.3.1 Herramientas necesarias

- Un ambiente de desarrollo para Java, tal como el Java 2 SDK, el cual está disponible en **java.sun.com**. La versión estándar del SDK 1.4 ya incluye el API JDBC.

- Un servidor de bases de datos PostgreSQL al que se tenga acceso con un nombre de usuario y contraseña (esta puede ser opcional).

- El driver JDBC para PostgreSQL, **postgresql-8.1-407.jdbc2.jar**

El ejemplo que se mostrará a continuación hará referencia a la versión 2 del driver. Los procedimientos descritos aquí deben de ser prácticamente los mismos si se utiliza alguna otra versión del driver, incluso, si se usa alguna de las versiones en desarrollo.

Para nuestro ejemplo crearemos una base de datos nombrada **agendita** en la cual guardaremos una lista de contactos. Los datos que vamos a manejar son únicamente nombre, email y teléfono.

[**root@host** root]createdb agendita
[**root@host** root]psql agendita

Agendita#CREATE TABLE contactos (nombre varchar(80), telefono varchar(20), email varchar(60));

Agendita# INSERT INTO contactos VALUES ('Pepe','8282-7272','pepe@hotmail.net');

Agendita# INSERT INTO contactos VALUES ('Gabriel','2737-9212','gabriel@micorreo.com');

Agendita# INSERT INTO contactos VALUES ('Juan','7262-8292','juan@correo.com.cx');

Descargamos el driver para postgres desde la pagina web **http://jdbc.postgresql.org**, en la seccion de descargas escogemos el que mas se adecue a nuestra version de jdk (este archivo tendra una extensión .jar) Es necesario que este archivo este incluído en la variable de ambiente CLASSPATH.
export CLASSPATH=/home/usuario/driver/postgresql-8.1-407.jdbc2.jar:., esta linea puede ser agregada al archivo .bash_rc para que se reconozca la ruta del driver a través de la consola de Linux.

17.3.2 Cargar el controlador JDBC

Para trabajar con el API JDBC se tiene que importar el paquete **java.sql**

import java.sql;

En este paquete se definen los objetos que proporcionan toda la funcionalidad que se requiere para el acceso a bases de datos.

El siguiente paso después de importar el paquete java.sql consiste en cargar el controlador JDBC, es decir un objeto **Driver** específico para una base de datos que define cómo se ejecutan las instrucciones para esa base de datos en particular.

Hay varias formas de hacerlo, pero la más sencilla es utilizar el método **forName()** de la clase **Class**:

Class.forName("Controlador JDBC");
para el caso particular del controlador para PostgreSQL, se tiene lo siguiente:
Class.forName("org.postgresql.Driver");

Debe tenerse en cuenta que el método estático forName() definido por la clase Class genera un objeto de la clase especificada. Cualquier controlador JDBC tiene que incluir una parte de iniciación estática que se ejecuta cuando se carga la clase. En cuanto el cargador de clases carga dicha clase, se ejecuta la iniciación estática, que pasa a registrarse como un controlador JDBC en el **DriverManager**.

Es decir, el siguiente código:

Class.forName("Controlador JDBC");

es equivalente a:
Class c = Class.forName("Controlador JDBC");
Driver driver = (Driver)c.newInstance();
DriverManager.registerDriver(driver);

Algunos controladores no crean automáticamente una instancia cuando se carga la clase. Si forName() no crea por sí solo una instancia del controlador, se tiene que hacer esto de manera explícita:

Class.forName("Controlador JDBC").newInstance();

De nuevo, para el postgres:

Class.forName("org.postgresql.Driver").newInstance();

17.3.3 Establecer la conexión

Una vez registrado el controlador con el DriverManager, se debe especificar la fuente de datos a la que se desea acceder. En JDBC, una fuente de datos se especifica por medio de un URL con el prefijo de protocolo **jdbc:** la sintaxis y la estructura del protocolo es la siguiente:

jdbc:{subprotocolo}:{subnombre}

El {subprotocolo} expresa el tipo de controlador, normalmente es el nombre del sistema de base de datos, como db2, oracle o mysql.

El contenido y la sintaxis de {subnombre} dependen del {subprotocolo}, pero en general indican el nombre y la ubicación de la fuente de datos.

Por ejemplo, para acceder a una base de datos denominada *productos* en un sistema *Oracle* local, el URL sería de la siguiente manera:

String url = "jdbc:oracle:productos";

Si la misma base de datos estuviera en un sistema *DB2* en un servidor llamado *dbserver.ibm.com*, el URL sería el siguiente:

String url = "jdbc:db2:dbserver.ibm.com/productos";

El formato general para conectarse a PostgresSQL es:

jdbc:postgresql://[servidor][:puerto]/[base_de_datos][?param1=valor1][param2=valor2]. Si queremos acceder de manera local a la base de datos agendita creada anteriormente, el URL sería :

String url = "jdbc:postgresql://localhost/agendita";

Una vez que se ha determinado el URL, se puede establecer una conexión a una base de datos.

El objeto **Connection** es el principal objeto utilizado para proporcionar un vínculo entre las bases de datos y una aplicación Java. Connection proporciona métodos para manejar el procesamiento de transacciones, para crear objetos y ejecutar instrucciones SQL y para crear objetos para la ejecución de procedimientos almacenados.

Se puede emplear tanto el objeto Driver como el objeto DriverManager para crear un objeto Connection. Se utiliza el método **connect()** para el objeto Driver, y el método **getConnection()** para el objeto DriverManager.

El objeto Connection proporciona una conexión estática a la base de datos. Esto significa que hasta que se llame en forma explícita a su método **close()** para cerrar la conexión o se destruya el objeto Connection, la conexión a la base de datos permanecerá activa.

La manera más usual de establecer una conexión a una base de datos es invocando el método getConnection() de la clase DriverManager. A menudo, las bases de datos están protegidas con nombres de usuario (login) y contraseñas (password) para restringir el acceso a las mismas. El método getConnection() permite que el nombre de usuario y la contraseña se pasen también como parámetros.

```
String login = "bingo";
String password = "holahola";
Connection conn = DriverManager.getConnection(url,login,password);
```

Para ejecutar instrucciones SQL y procesar los resultados de las mismas, debemos hacer uso de un objeto **Statement**.

Los objetos Statement envían comandos SQL a la base de datos, que pueden ser de cualquiera de los tipos siguientes:

- Un comando de definición de datos como CREATE TABLE o CREATE INDEX.
- Un comando de manipulación de datos como INSERT, DELETE o UPDATE.
- Un sentencia SELECT para consulta de datos.

Un comando de manipulación de datos devuelve un contador con el número de filas (registros) afectados, o modificados, mientras una instrucción SELECT devuelve un conjunto de registros denominado conjunto de resultados (*result set*). La interfaz Statement no tiene un constructor, sin embargo, podemos obtener un objeto Statement al invocar el método **createStatement()** de un objeto Connection.

```
conn = DriverManager.getConnection(url,login,pasword);
Statement stmt = conn.createStatement();
```

Una vez creado el objeto Statement, se puede emplear para enviar consultas a la base de datos usando los métodos execute(), executeUpdate() o executeQuery(). La elección del método depende del tipo de consulta que se va a enviar al servidor de bases de datos:

Método	**Descripción**
execute( )	Se usa principalmente cuando una sentencia SQL regresa varios conjuntos de resultados. Esto ocurre principalmente cuando se está haciendo uso de procedimientos almacenados.
executeUpdate( )	Este método se utiliza con instrucciones SQL de manipulación de datos tales como INSERT, DELETE o UPDATE.
executeQuery( )	Se usa en las instrucciones del tipo SELECT.

Es recomendable que se cierren los objetos Connection y Statement que se hayan creado cuando ya no se necesiten. Lo que sucede es que cuando en una aplicación en Java se están usando recursos externos, como es el caso del acceso a bases de datos con el API JDBC, el recolector de basura de Java (*garbage collector*) no tiene manera de conocer cuál es el estado de esos recursos, y por lo tanto, no es capaz de liberarlos en el caso de que ya no sean útiles. Lo que sucede en estos casos es que pueden quedar almacenados en memoria grandes cantidades de recursos relacionados con la aplicación de bases de datos que se está ejecutando. Es por esto que se recomienda que se cierren de manera explícita los objetos Connection y Statement.

De manera similar a Connection, la interfaz Statement tiene un método **close()** que permite cerrar de manera explícita un objeto Statement. Al cerrar un objeto Statement se liberan los recursos que están en uso tanto en la aplicación Java como en el servidor de bases de datos.

```
Statement stmt = conn.createStatement();
    ....
  stmt.close();
```

17.3.4 Ejecución de consultas

Cuando se ejecutan sentencias SELECT usando el método executeQuery(), se obtiene como respuesta un conjunto de resultados, que en Java es representado por un objeto **ResultSet**.

```
Statement stmt = conn.createStatement();

ResultSet res  = stmt.executeQuery("SELECT * FROM agendita");
```

Se puede pensar en un conjunto de resultados como una tabla (filas y columnas) en la que están los datos obtenidos por una sentencia SELECT.

La información del conjunto de resultados se puede obtener usando el método **next()** y los diversos métodos **getXXX()** del objeto ResultSet. El método next() permite moverse fila por fila a través del ResultSet, mientras que los diversos métodos getXXX() permiten accesar a los datos de una fila en particular.

Los métodos getXXX() toman como argumento el índice o nombre de una columna, y regresan un valor con el tipo de datos especificado en el método. Así por ejemplo, getString() regresará una cadena, getBoolean() regresará un booleano y getInt() regresará un entero. Cabe mencionar que estos métodos deben tener una correspondencia con los tipos de datos que se tienen en el ResultSet, y que son a las vez los tipos de datos provenientes de la consulta SELECT en la base de datos, sin embargo, si únicamente se desean mostrar los datos se puede usar getString() sin importar el tipo de dato de la columna.

Por otra parte, si en estos métodos se utiliza la versión que toma el índice de la columna, se debe considerar que los índices empiezan a partir de 1, y no en 0 (cero) como en los arreglos, los vectores, y algunas otras estructuras de datos de Java.

Existe un objeto ResultSetMetaData que proporciona varios métodos para obtener información sobre los datos que están dentro de un objeto ResultSet. Estos métodos permiten entre otras cosas obtener de manera dinámica el número de columnas en el conjunto de resultados, así como el nombre y el tipo de cada columna.

```
ResultSet res  = stmt.executeQuery("SELECT * FROM agendita");
ResultSetMetaData metadata = res.getMetaData();

/*
Esta es una pequeña aplicación que muestra como obtener los
datos de una tabla usando una consulta SELECT. Se ejecuta
desde la línea de comandos del sistema operativo y los datos
obtenidos son mostrados en la consola.
*/
```

```
import java.sql.*;

public class MostrarAgendita
{
  static String login = "usuario";
  static String password = "usuario123";
  static String url = "jdbc:potgresql://localhost/agendita";

  public static void main(String[] args) throws Exception
  {
    Connection conn = null;
    try
    {
      Class.forName("com.mysql.jdbc.Driver").newInstance();
      conn = DriverManager.getConnection(url,login,password);

      if (conn != null)
      {
        Statement stmt = conn.createStatement();
        ResultSet res  = stmt.executeQuery("SELECT * FROM contactos");

        System.out.println("\nNOMBRE \t\t EMAIL \t\t\t TELEFONO \n");

        while(res.next())
        {
          String nombre  = res.getString("nombre");
          String email   = res.getString("email");
          String telefono= res.getString("telefono");

          System.out.println(nombre +" \t "+email+" \t "+telefono);
        }
```

```
            res.close();
            stmt.close();
            conn.close();
          }
        }
        catch(SQLException ex)
        {
          System.out.println(ex);
        }
        catch(ClassNotFoundException ex)
        {
          System.out.println(ex);
        }
    }
}
```

18. PostgresSQL

Postgres es un manejador de bases de datos (DBMS) de alto porte, desarrollado originalmente en el Departamento de Ciencias de Computación de la Universidad de Berkeley. Fue pionero de muchos de los conceptos referentes a objetos que ahora están disponibles en algunas bases de datos comerciales. Proporciona soporte a lenguaje SQL92/SQL93, integridad en transacciones y capacidad para extensión de tipos. PostgreSQL es un descendiente "open-source" de este código original de Berkeley.

Intenta ser un sistema de bases de datos de alto nivel, con unas prestaciones a la altura de Oracle, Sybase o Interbase. Es software libre, concretamente está liberado bajo la licencia BSD, lo que significa que cualquiera puede disponer de su código fuente, modificarlo a voluntad y redistribuirlo libremente, es más, la licencia BSD permite redistribuir el código modificado o no como software cerrado. Por su arquitectura de diseño, escala muy bien al aumentar el número de CPUs y la cantidad de RAM. Está considerado como la base de datos de código abierto más avanzada del mundo porque proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales tales como DB2 u Oracle. La siguiente es una breve lista de algunas de esas características:

- **Integridad Referencial:** PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.
- **Altamente_Extensible:** Soporta operadores, funcionaes métodos de acceso y tipos de datos definidos por el usuario.
- **MVCC:** Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios. Cuando hay un usuario escribiendo en la base de datos y otro leyendo el MVCC evita que el usuario escritor bloquee al usuario lector.
- **Cliente/Servidor:** usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.

18.1 Historia de PostgreSQL

El origen de SQL se remonta al año 1974, cuando el gigante azul, IBM, desarrollo el lenguaje SEQUEL "Structured English QUEry Language" (Lenguaje de Consulta Estructurado en Ingles). Luego Oracle lanzó su primer producto comercial, una implementación de SQL realizada en el año 1979. Dos años más tarde IBM sacó al mercado el producto SQL/DS y en 1983 el popular DB2.

El Sistema Gestor de Bases de Datos Relacionales Orientadas a Objetos conocida como PostgreSQL está derivado del paquete Postgres escrito en Berkeley. Con cerca de una década de desarrollo tras ella, PostgreSQL es la base de datos de código abierto más avanzada hoy día disponible, ofreciendo características propias de los más potentes motores de bases de datos comerciales.

La implementación del DBMS Postgres comenzó en 1986. Los conceptos iniciales para el sistema fueron presentados en The Design of Postgres y la definición del modelo de datos inicial apareció en The Postgres Data Model. El diseño del sistema de reglas fue descrito en ese momento en The Design of the Postgres Rules System. La lógica y arquitectura del gestor de almacenamiento fueron detalladas en The Postgres Storage System.

Postgres ha pasado por varias versiones desde entonces. El primer sistema de pruebas fue operacional en 1987 y fue mostrado en la Conferencia ACM-SIGMOD de 1988. La Versión 1, descrita en The Implementation of Postgres, fue lanzada a unos pocos usuarios externos en Junio de 1989. Despues de revisar el primer sistema de reglas éste fue rediseñado (On Rules, Procedures, Caching and Views in Database Systems) y la Versión 2 salió en Junio de 1990. La Versión 3 apareció en 1991 y añadió una implementación para múltiples gestores de almacenamiento, un ejecutor de consultas mejorado y un sistema de reescritura de reglas nuevo. En su mayor parte, las siguientes versiones hasta el lanzamiento de Postgres95 se centraron en portabilidad y fiabilidad.

Postgres ha sido usado para implementar muchas aplicaciones de investigación y producción. Entre ellas: un sistema de análisis de datos financieros, un paquete de monitorización de rendimiento de motores a reacción, una base de datos de seguimiento de asteroides y varios sistemas de información geográfica. Postgres se ha usado también como una herramienta educativa en varias universidades. Finalmente, Illustra Information Technologies (posteriormente absorbida por Informix) tomó el código y lo comercializó.

El tamaño de la comunidad de usuarios externos casi se duplicó durante 1993. El mantenimiento del código y las tareas de soporte ocupaban demasiado tiempo que debía dedicarse a la investigación, así que el proyecto terminó oficialmente con el lanzamiento de la Versión 4.2.

En 1994, Andrew Yu y Jolly Chen añadieron un intérprete de lenguage SQL a Postgres. Postgres95 fue lanzada a continuación a la Web para que encontrara su propio hueco en el mundo como un descendiente de dominio público y código abierto del código original Postgres de Berkeley.

El código de Postgres95 fue adaptado a ANSI C y reducido en tamaño en un 25%. Muchos cambios internos mejoraron el rendimiento y la facilidad de mantenimiento. Postgres95 v1.0.x se ejecutaba en torno a un 30-50% más rápido que Postgres v4.2. Además de corrección de errores, éstas fueron las principales mejoras:

* El lenguage de consultas Postquel fue reemplazado con SQL (implementado en el servidor).

* Además del programa de monitorización, se incluyó un nuevo programa (psql) para realizar consultas SQL interactivas usando la librería GNU readline.

* Una nueva librería de interfaz, libpgtcl, soportaba clientes basados en Tcl.

* Se distribuyó con el código fuente un breve tutorial introduciendo las características comunes de SQL y de Postgres95.

* Se utilizó GNU make (en vez de BSD make) para la compilación. Postgres95 también podía ser compilado con un gcc sin parches (la alineación de variables de longitud doble fue corregida).

En 1996 nace PostgreSQL, para reflejar la relación entre el Postgres original y las versiones más recientes con capacidades SQL. Los números de versión parten de la 6.0, volviendo a la secuencia seguida originalmente por el proyecto Postgres.

Postgres es un manejador de bases de datos (DBMS) de alto porte, desarrollado originalmente en el Departamento de Ciencias de Computación de la Universidad de Berkeley. Fue pionero de muchos de los conceptos referentes a objetos que ahora están disponibles en algunas bases de datos comerciales. Proporciona soporte a lenguaje SQL92/SQL93, integridad en transacciones y capacidad para extensión de tipos. PostgreSQL es un descendiente "**open-source**" de este código original de Berkeley.

Postgres intenta ser un sistema de bases de datos de alto nivel, con unas prestaciones a la altura de Oracle, Sybase o Interbase. Es **software libre**, concretamente está liberado bajo la **licencia BSD**, lo que significa que cualquiera puede disponer de su código fuente, modificarlo a voluntad y redistribuirlo libremente, es más, la licencia BSD permite redistribuir el código modificado o no como software cerrado. Por su arquitectura de diseño, escala muy bien al aumentar el número de CPUs y la cantidad de RAM. Está considerado como la base de datos de código abierto más avanzada del mundo porque proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales tales como DB2 u Oracle. La siguiente es una breve lista de algunas de esas características:

- Integridad Referencial**:** PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.
- Altamente Extensible: Soporta operadores, funcionales métodos de acceso y tipos de datos definidos por el usuario.
- MVCC: Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios. Cuando hay un usuario escribiendo en la base de datos y otro leyendo el MVCC evita que el usuario escritor bloquee al usuario lector.
- Cliente/Servidor: usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.

18.2 Las principales mejoras en PostgreSQL incluyen:

* Los bloqueos de tabla han sido sustituidos por el control de concurrencia multi-versión, el cual permite a los accesos de sólo lectura continuar leyendo datos consistentes durante la actualización de registros, y permite copias de seguridad en caliente desde pg_dump mientras la base de datos permanece disponible para consultas.

* Se han implementado importantes características del motor de datos, incluyendo subconsultas, valores por defecto, restricciones a valores en los campos (constraints) y disparadores (triggers).

* Se han añadido características adicionales que cumplen el estándar SQL92, incluyendo claves primarias, identificadores entrecomillados, forzado de tipos cadena literales, conversión de tipos y entrada de enteros binarios y hexadecimales.

* Los tipos internos han sido mejorados, incluyendo nuevos tipos de fecha/hora de rango amplio y soporte para tipos geométricos adicionales.

La velocidad del código del motor de datos ha sido incrementada aproximadamente en un 20-40%, y su tiempo de arranque ha bajado el 80% desde que la versión 6.0 fue lanzada.

En nuestro caso hemos utilizado postgresql 8.1.4, que puede ser descargado desde el sitio oficial de postgres **http://www.postgresql.org/download/**.

18.3 Pasos para la instalación de postgres

1. obtener las fuentes de este en el sitio web antes descrito
2. una vez descargado lo movemos y descomprimimos en la carpeta /usr/local

   [**root@host** root]#bzcat postgresql-8.1.4.tar.bz2 | tar -xvf-
3. se compila y se mueven los ejecutables a los lugares apropiados

   [**root@host** root]#cd postgresql-8.1.4

   [**root@host** root]#./configure

   [**root@host** root]#make

   [**root@host** root]#make install
4. si todo ha salido bien en este momento se tiene instalado PostgreSQL
5. se debera crear una cuenta de usuario UNIX que poseera y gestionara los archivos de

bases de datos, normalmente este usuario es llamado “postgres”, pero se le puede llamar como uno desee.

[**root@host** root]#useradd “usuario”

Y con el comando passwd se le asigna una contraseña si se desea.

6. vamos a crear el directorio donde se localice el cluster de base de datos y le cambiamos los permisos para que el usuario que se ha agregado sea el propietario de ese directorio

   [**root@host** root]#mkdir -p /var/pgql/data

   [**root@host** root]#chown “usuario” /var/pgsql/data”

7. nos cambiamos al “usuario” e inicializamos el cluster de base de datos con el comando initdb.

   [**root@host** root]#su - “usuario”

   [**root@host** root]#/usr/local/pgsql/bin/initdb -D /var/pgsql/data

8. el siguiente paso es inicializar el servidor de base de datos. Antes de esto, se recomienda agregar estas lineas al archivo .bash_profile del “usuario”, para evitar poner la ruta del cluster de la base de datos cada vez que inicializamos el servidor, tambien agregar a nuestro PATH el directorio donde estan los comandos de postgres

   PGDATA=/var/pgsql/data
   PATH=/usr/local/pgsql/bin:$PATH
   export PATH PGDATA

Ahora si como “usuario” inicializamos el servidor de bases de datos con el comando pg_ctl

9. [**usuario@host** root]:~$pg_ctl start

18.4 Autentificacion del cliente

Es una caracteristica central para postgres, sin ella podria sacrificar la conectividad remota o por el contrario podria permitir que caulquiera pudiera conectar a su base de datos y recibir, incluso modificar sus datos. Postgres tiene varios tipos de autentificacion de cliente a su disposicion, los accesos basados en maquina (host) se encuentran especificados en el archivo pg_hba.conf, este tipo de autentificacion es flexible, proporcionando una amplia variedad de opciones de configuracion, puede

restringir accesos a bases de datos a maquinas especificas, asi como permitir acceso a un rango de direcciones IP usando mascaras de red.

Dicho simplemente, el archivo pg_hba.conf le permite determinar quien esta autorizado para conectarse a que base de datos desde que maquinas, y para graduar como deben probar su autenticidad para obtener acceso.

Cuando una aplicación solicita una conexión, la peticion especificara un nombre de usuario postgres y la base de datos a la cual intenta conectar. Opcionalmente, puede ser proporcionada una contraseña, dependiendo de la configuracion esperada para la maquina que se conecta.

```
# PostgreSQL Client Authentication Configuration File
# ===================================================
# This file controls: which hosts are allowed to connect, how clients
# are authenticated, which PostgreSQL user names they can use, which
# databases they can access.  Records take one of these forms:
#
# local      DATABASE  USER  METHOD  [OPTION]
# host       DATABASE  USER  CIDR-ADDRESS  METHOD  [OPTION]
# hostssl    DATABASE  USER  CIDR-ADDRESS  METHOD  [OPTION]
# hostnossl  DATABASE  USER  CIDR-ADDRESS  METHOD  [OPTION]
#
# (The uppercase items must be replaced by actual values.)
#
# The first field is the connection type: "local" is a Unix-domain socket,
# "host" is either a plain or SSL-encrypted TCP/IP socket, "hostssl" is an
# SSL-encrypted TCP/IP socket, and "hostnossl" is a plain TCP/IP socket.
#
# DATABASE can be "all", "sameuser", "samerole", a database name, or
# a comma-separated list thereof.
#
# USER can be "all", a user name, a group name prefixed with "+", or
# a comma-separated list thereof.  In both the DATABASE and USER fields
```

```
# you can also write a file name prefixed with "@" to include names from
# a separate file.
#
# CIDR-ADDRESS specifies the set of hosts the record matches.
# It is made up of an IP address and a CIDR mask that is an integer
# (between 0 and 32 (IPv4) or 128 (IPv6) inclusive) that specifies
# the number of significant bits in the mask.  Alternatively, you can write
# an IP address and netmask in separate columns to specify the set of hosts.
#
# METHOD can be "trust", "reject", "md5", "crypt", "password",
# "krb5", "ident", or "pam".  Note that "password" sends passwords
# in clear text; "md5" is preferred since it sends encrypted passwords.
#
# OPTION is the ident map or the name of the PAM service, depending on
METHOD.
# This file is read on server startup and when the postmaster receives
# a SIGHUP signal.  If you edit the file on a running system, you have
# to SIGHUP the postmaster for the changes to take effect.  You can use
# "pg_ctl reload" to do that.

# CAUTION: Configuring the system for local "trust" authentication allows
# any local user to connect as any PostgreSQL user, including the database
# superuser. If you do not trust all your local users, use another
# authentication method.


# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD

# "local" is for Unix domain socket connections only
local   all         all                               trust
# IPv4 local connections:
host    all         all         127.0.0.1/32          trust
# IPv6 local connections:
```

```
host    all         all         ::1/128                 trust
host    all         all             0.0.0.0         0.0.0.0 reject
```

Cuando una conexión es inicializada, postgres lee cada entrada de este archivo, tan pronto como un registro coincidente sea encontrado, se abandona la busqueda y permitira o rechazara la conexión, en base a la entrada encontrada.

Por lo tanto, supongamos que solo queremos aceptar conexiones locales y del host con IP 192.168.1.68

```
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD

# "local" is for Unix domain socket connections only
local   all         all     md5                                     trust
# IPv4 local connections:
host    all         all         192.168.1.68        255.255.255.255     md5
host    all         all             0.0.0.0         0.0.0.0                 reject
```

```
/*****************************************************************
*****************************************************************
*****************************************************************/
```

19. ¿Por qué Debian?

Hemos elegido Debian como sistema operativo sobre el que montar nuestra aplicación por algunas de sus características, entre las que mencionaremos las siguientes:

Estabilidad: La liberación de una nueva versión no es determinada por razones comerciales sino solamente por la madurez técnica. Por eso Debian GNU/Linux es una de las distribuciones más estables.

Seguridad: Los permisos de acceso separan los datos de usuarios de los datos del sistema operativo. Se conceden solo los derechos necesarios a los usuarios y a los procesos. Este concepto garantiza una seguridad alta por lo cual no se encuentran virus peligrosos de GNU/Linux.

Soporte: Miles de usuarios de Debian en todo el mundo dan soporte ayudando rápidamente en caso de problemas en más de 80 listas de correo y más de 20 grupos de noticias.

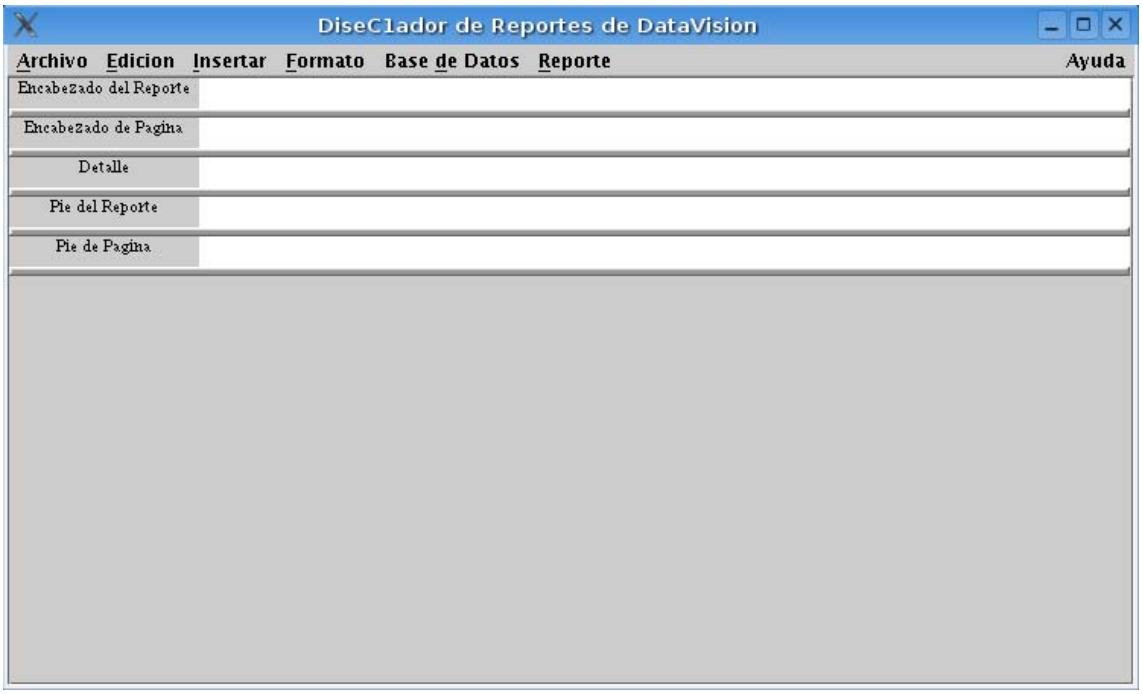
Multiusuario: Debian GNU/Linux permite a más de un usuario trabajar a la vez en la misma computadora (desde diferentes terminales). El sistema separa estrictamente los datos personales de los usuarios.

Multitarea: GNU/Linux permite ejecutar varios procesos simultáneamente. No hay ningún problema de dejar correr servidores de web, de mail, de FTP y otras aplicaciones al mismo tiempo en una sola computadora. Esto permite aprovechar el hardware eficientemente.

20. DATA VISIÓN

Es un generador de reportes Open Source similar a Crystal Reports. Con múltiples opciones de configuración que el usuario decidirá cual es la que se ajusta a sus necesidades, por ejemplo es capaz de realizar agrupaciones, conteos de registros, ordenar campos, nombre al reporte, pies de página, etc.

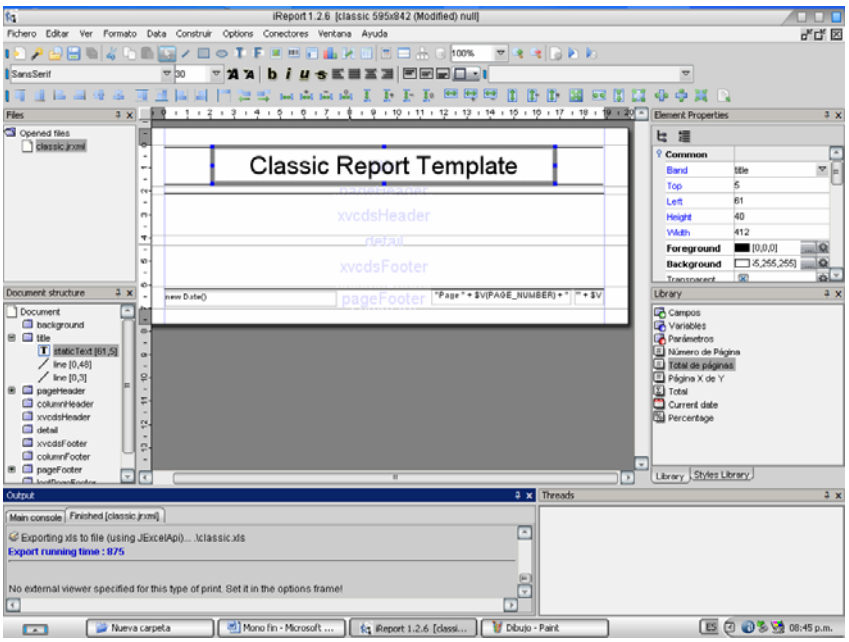
La creación de cualquier reporte dependerá de la “creatividad” del usuario final, debido a que una vez que se ha establecido la conexión con la base de datos, esta aplicación mostrara las tablas que contenga la base de datos con sus respectivos campos y con solo arrastrarlos a la parte de Detalle del proyecto quedara construido el esqueleto del informe con los atributos que serán mostrados en el reporte al momento de que este sea ejecutado.



Este esqueleto puede ser guardado para su posterior modificación. Una vez que el usuario ha decido que la estructura del reporte se adecua a sus necesidades puede exportarlo en múltiples formatos como por ejemplo: **HTML, XML, PDF, LaTeX, Doc, Txt.** Y así llevar ese reporte a otra plataforma sin verse envuelto en el inconveniente de la incompatibilidad de estas.

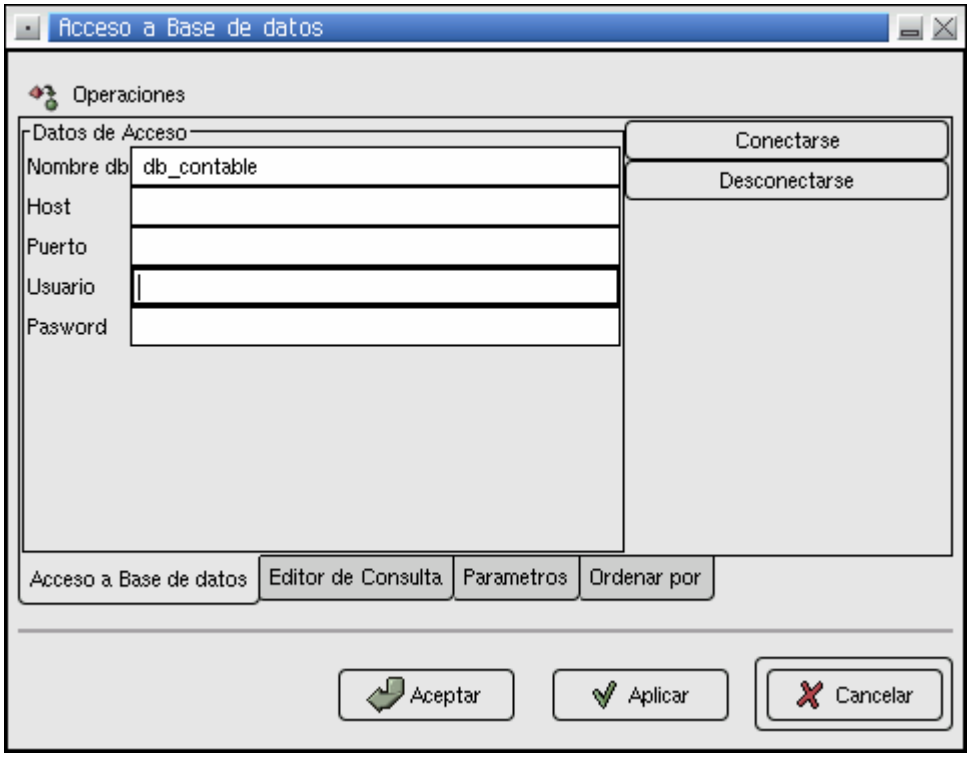
DataVision esta escrito en JAVA y trabaja perfectamente con JDBC, se conecta fácilmente a múltiples gestores de base de datos como: Oracle, PostgreSQL, MySQL, Informix, hsqldb, Microsoft Access, Progress, entre otros. Para que este entorno pueda ser ejecutado requiere la instalación de la maquina virtual java.

Otra herramienta utilizada en la creación de reportes es iReports que posee características parecidas a la herramienta utilizada para este proyecto: conexión con múltiples bases de datos, pies de reportes, inserción de campos especiales como imágenes, etc. Solo que esta herramienta posee muchas mas opciones configurables al gusto del usuario, cabe destacar que iReports esta dirigido meramente para administradores de bases de datos, debido a que el criterio de selección de registros se hace meramente con sentencias SQL utilizando su editor de consultas, por lo que el usuario de esta aplicación deberá poseer conocimientos sobre la materia. Las definiciones de las estructuras y componentes del informe se hacen en XML el cual es procesado por este gestor. Tanto Impresiones como previsualizaciones se hacen en múltiples formatos como Excel, PDF, HTML entre otros incluyendo la tecnología JAVA 2D.



moReports es otra herramienta para generar reportes con soporte para PostgreSQL(por el momento, esta pensado añadirle soporte para mas Servidores de Base de Datos)con salida en formato Postscript(también se añadirá soporte para otros formatos de salida). El formato de entrada para moReport se basa en el standar XML.

Posee las estructuras básicas para la generación de informes como: detalles, pie de reporte, encabezado, pie de página, etc. Su funcionamiento e interfaz es bastante parecido a Datavision, esta herramienta al igual que las anteriores posee un editor de consultas SQL para la selección de registros.



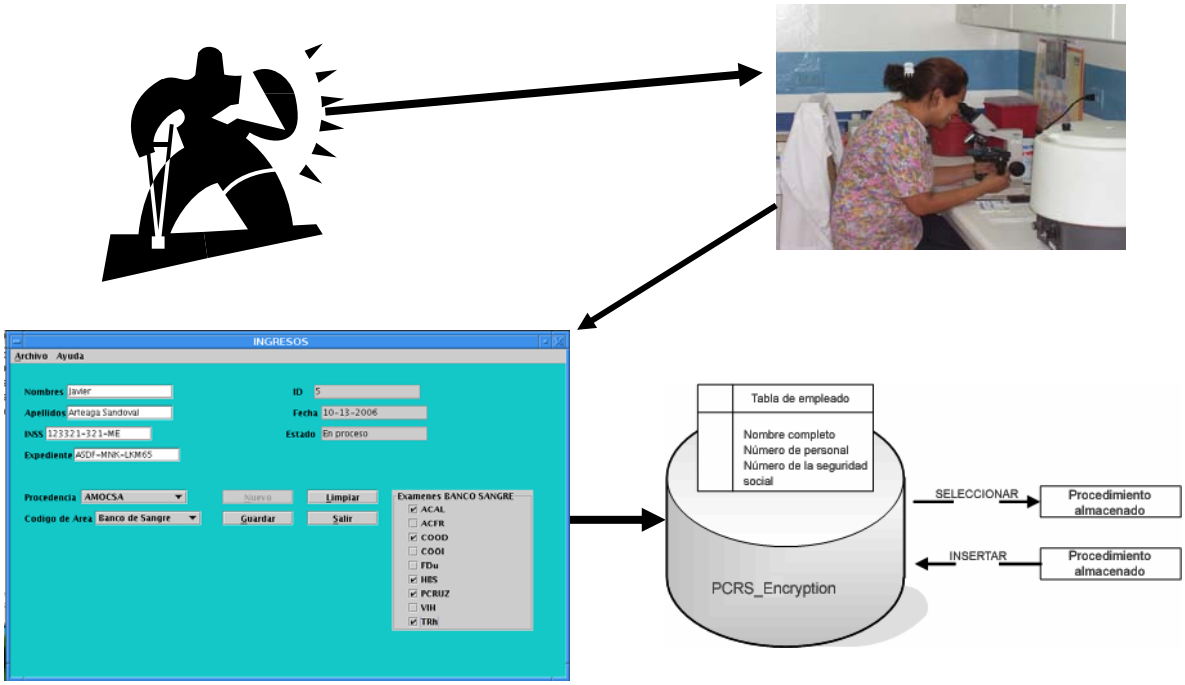
21. FUNCIONAMIENTO DE LA APLICACIÓN

Cuando X paciente se presenta al laboratorio clinico proporciona sus nombres, apellidos, expediente (si poseyera uno), numero de inss (si poseyera), procedencia que no es mas que el lugar de donde viene esa solicitud de examen (AMOCSA, SUMEDIO, BEMENIC, etc.) .Se debera seleccionar un area del laboratorio (Batereologia, Urologia, Banco de Sangre, etc.) Para que atraves de estas se pueda seleccionar el o los examenes a realizar, aquí es donde entra en juego la aplicación principal, que a su vez almacena estos detalles en una base de datos postgres y se sigue procesando la siguiente solicitud de examen. Las terminales que se encuentran en las distintas areas del laboratorio clinico accederan a los datos que se encuentran almacenados obteniendo solamente datos que resulten relevantes para un area en concreto como por ejemplo: nombres, apellidos, y los examenes que este paciente se realizara, una vez que el laboratorista a cargo de la realizacion de dichos examenes haya culminado de procesar esta solicitud procedera a guardar este registro, que no es mas que actualizar el estado de la solicitud del estado “en proceso” a estado de “finalizado”, se procede a la impresión de resultado de esa solicitud.

Para acceder a la pantalla principal de cada aplicación se debera proporcionar un nombre de usuario y una contraseña, solamente para la aplicación principal se debera proporcionar otro nombre de usuario y contraseña si lo que se desea es borrar datos que estan almacenados, al realizar esto la base de datos quedara vacia, esto se realizara para mantenimiento de la misma o cada vez y cuando se realicen reportes e informes de la misma por lo que ya no se necesitaran los datos que alli se encontraban.

Ura restriccion que cabe destacar es cada una de las terminales(Uroanalisis, Banco de Sangre, etc) accede solo a los datos que van dirigidos a ellas, en otras palabras Uroanalisis accede solamente a datos que son para Uroanalisis y asi con el resto de areas del laboratorio.

22. ESQUEMA DEL FUNCIONAMIENTO DE LA APLICACIÓN



23. DISEÑO METODOLÓGICO

El proceso de ingeniería de software se define como "un conjunto de etapas parcialmente ordenadas con la intención de logra un objetivo, en este caso, la obtención de un producto de software de calidad". El proceso de desarrollo de software "es aquel en que las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo". Concretamente "define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo".

El proceso de desarrollo de software requiere por un lado un conjunto de conceptos, una metodología y un lenguaje propio. A este proceso también se le llama el ciclo de vida del software que comprende cuatro grandes fases: concepción, elaboración, construcción y transición. La concepción define le alcance del proyecto y desarrolla un caso de negocio. La elaboración define un plan del proyecto, especifica las características y fundamenta la arquitectura. La construcción crea el producto y la transición transfiere el producto a los usuarios.

Es un procedimiento de recolección de información. Para la elaboración y diseño se utilizaran métodos de recolección de información tales como:

- Entrevistas con la Directora del laboratorios clínico Dra. Rosa Alonso
- Entrevistas con el personal que labora en el laboratorio en todo proceso de elaboración de exámenes.
- Revisión de documentos con los que laboran en el laboratorio (folletos, hojas de trabajo) y un estudio de los mismos.
- Elaboración de preguntas para la entrevista con el cliente Dra. Rosa Alonso.
- Consultas con nuestro Tutor y Asesor

23.1. Materiales

a. Hardware: En este proyecto a nivel de hardware emplearemos las siguientes herramientas:

- 2 PC con las siguientes características:
  - ✓ Memoria RAM de 256 MB
  - ✓ 20 GB en Disco Duro
  - ✓ Procesador Intel® Celeron™ CPU
  - ✓ Velocidad de procesamiento 2.2 Ghz

b. Software: Las herramientas software que emplearemos en nuestra aplicación son:

- ✓ Open Office
- ✓ Visio 2003
- ✓ Linux Debian (Sistema Operativo)
- ✓ PostgreSQL 8.1.3 (Sistema Gestor de Base de Datos)
- ✓ gedit
- ✓ Eclipse 3.1
- ✓ JVM 1.4.2_11
- ✓ DataVision 1.0.0
- ✓ Driver Postgresql -8.0-315.jdbc2.jar

23.2. Método de Análisis de Requisitos

Hemos hecho uso principalmente del análisis estructurado, ya que el sistema es grande y complejo, además nos permite crear modelos que representan el contenido y flujo de información.

23.3 Método del proceso de desarrollo

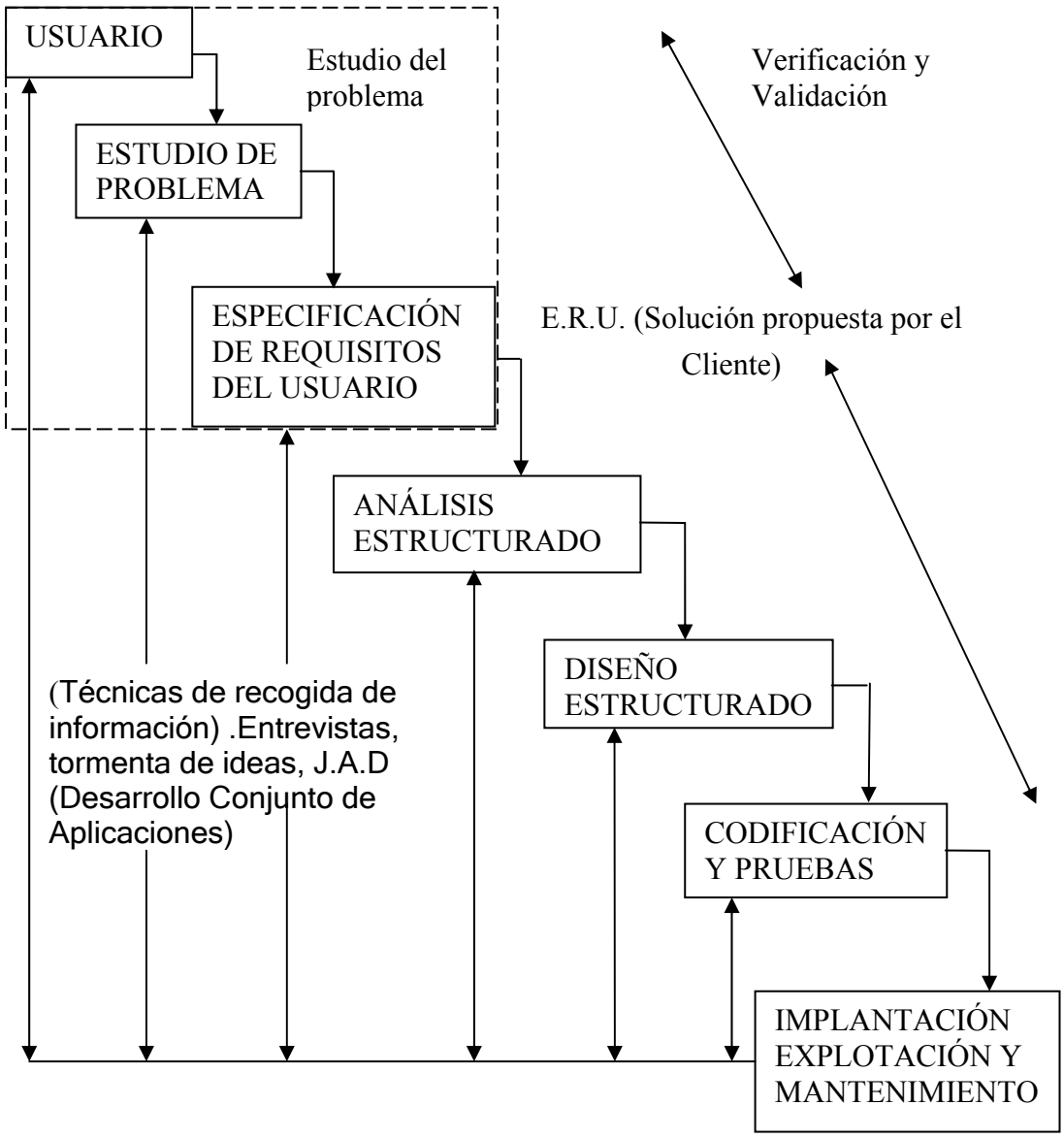
Para el proceso de desarrollo del sistema haremos uso del modelo de ciclo de vida clásico (Modelo en Cascada), el cual nos permitirá descomponer el proceso de desarrollo en diferentes fases o etapas permitiendo que las salidas de cada una de las fases sea la entrada la siguiente.

Fases generales de diseño

- Investigación: es una investigación detallada, donde se recogen los requisitos que deberá cumplir el sistema.
- Análisis de las necesidades del sistema: se determina que es lo que requiere que haga el sistema, permitirá observar los elementos lógicos (lo que hará el sistema) separados de los componentes físicos (terminales, etc).
- Diseño del sistema recomendado: parte del diseño lógico de información, es diseñar la interfaz de usuario, además es diseñar procedimientos precisos para la captura de datos, diseño de archivos o base de datos que guardaran los datos necesarios
- Codificación: se procederá a escribir el código de la aplicación, además de preparar la documentación, lo esencial para probar el sistema y llevar a cabo el mantenimiento una vez que la aplicación se encuentra instalada.
- Prueba: El sistema se emplea de manera experimental para asegurarse de que el software no tenga errores.
- Implementación y evaluación: Es el proceso de verificación de nuevo equipo, entrenar al personal que hará uso del sistema, instalar la aplicación y construir todos los archivos de datos necesarios para su utilización.

23.4 FORMA DE CICLO DE VIDA EN CASCADA INCREMENTAL

(Con una Metodología Estructurada).



23.5. ANÁLISIS

ESPECIFICACIÓN DE REQUISITOS SOFTWARE (ERS)

Introducción:

Propósito

Las especificaciones de requisitos del software que debe cumplir la aplicación “**LABCLI 1.0**” que consiste en automatizar los procesos de gestión del Laboratorio Clínico HEODRA.
Este documento esta dirigido a estudiantes interesados en la creación de aplicaciones visual en entorno Linux con acceso a base de datos.

23.6. Definiciones, acrónimos y abreviaturas:

Paciente: Atributo de la entidad solicitud, persona que solicita un o uno exámenes.

Solicitud: Entidad fuerte donde se hace la petición de los diferentes tipos de examen.

Área: Atributo de la entidad solicitud que corresponde al lugar físico donde se realizan los exámenes.

Examenes: Atributo de la entidad solicitud donde se especifica el examen o examenes que el paciente se va a realizar.

DESCRIPCIÓN GENERAL:

Relaciones del producto:

La aplicación se desarrollará en un equipo con las siguientes características:

- 40GB en Disco Duro.
- 128 MB RAM.
- Intel(R) Celeron(TM) CPU 1200MHz.
- Sistema Operativo: Linux Debian.

El equipo en que se implantará el sistema final es:

- 40GB en Disco Duro.
- 128 MB RAM.
- Intel(R) Celeron(TM) CPU 1200MHz.
- Sistema Operativo: Linux Debian.

Restricciones Generales:

El laboratorio Clinico del Hospital HEODRA usuaria de nuestra aplicación deberá tener como requisitos mínimos:

- Computadores con: 20GB de Disco Duro, 128MB de Memoria RAM, Procesadores con velocidad de 1.2 Mhz.
- Sistema Operativo Linux Debian.
- Red de Área Local.

23.7. Funciones de la Aplicación:

Nuestra aplicación debe realizar las siguientes funciones:

- En cada área deberá escribir su usuario y su password antes de iniciar la sus aplicaciones.
- Registrar las diferentes solicitudes de pacientes.
- Acceder a las solicitudes almacenadas en un servidor central.
- Procesar cada examen indicado en la solicitud.
- Imprimir resultado.
- Generar reportes estadísticos por examen.
- Generar reportes estadísticos por área para un rango de fecha.
- Generar reportes estadísticos.

24 REQUISITOS ESPECÍFICOS:

Requisitos Funcionales:

Aplicación Principal Solicitud

REQUISITOS ESPECÍFICOS:

Requisitos Funcionales:

NUEVA SOLICITUD

Especificación:

Introducción:

Esta función permitirá al usuario registrar en el sistema los datos correspondientes a uno o varios exámenes en particular correspondientes a una solicitud de examen.

Entradas:

Por pantalla:

Datos del Paciente: Esta agrupación de campos representa los datos generales del paciente.

- Nombre.
- Apellidos.
- Expediente.
- INSS.

Exámenes a Realizar: Esta agrupación de campos representa los posibles exámenes que un paciente puede realizarse.

- ACAL
- ACFR
- COOD
- COOI
- FDu
- HBS

- PCRUZ
- VIH
- TRh
- ESP
- EGO
- PBJ
- GCH
- R_ADD
- TIT_GCHCH

⇒ Áreas: Esta agrupación de campos representa las posibles áreas del laboratorio a las que puede ir una solicitud.

- Banco de Sangre
- Uroanalisis

⇒ Procedencia: Esta agrupación de campos representa la de un paciente.

- AMOCSA
- LA FRATERNIDAD
- BEMENIC
- SUMEDICO
- HEODRA

25 REQUISITOS ESPECÍFICOS:

Requisitos Funcionales:

Área de Uroanalisis

REQUISITOS ESPECÍFICOS:

Requisitos Funcionales:

NUEVO EXAMEN (Uroanalisis)

Especificación:

Introducción:

Esta función permitirá al usuario registrar en el sistema los datos correspondientes a uno o varios exámenes en particular correspondientes al área de uroanalisis.

Entradas:

Por pantalla:

⇒ Datos del examen físico: Esta agrupación de campos representa la características cualitativas o visuales del examen físico como los son:

- Color.
- Aspecto.
- Sedimento.
- Densidad.

⇒ Datos del examen Químico: Esta agrupación de campos representa cada una de las caracterizas de la composición química que tiene la muestra de orina como lo son:

- Proteínas.
- Hemoglobina.
- Cuerpos Cetonicos.
- PH

- Urobilinogeno.
- Glucosa.
- Bilirrubinas.
- Nitriros.

⇒ Datos del examen microscópico : Esta agrupación de campos representa cada una de las características microscópicas elaboradas a la muestra de orina como lo son:

- Células epiteliales.
- Leucocitos.
- Eritrocitos.
- Cilindros.
- Cristales.
- Otros.

⇒ Prueba de embarazo: Esta agrupación de campos es opcionala y representa las características para determinar en la muestra de orina si la paciente se encuentra o no en estado de embrazo. Sus características son:

- Datos cualitativos.
- Datos cuantitativos.

⇒ Datos del laboratorista: Esta agrupación de campos representa la información concerniente a quien realiza el examen así como la fecha en que se realiza. Sus características son:

- Nombre del laboratorista.
- Fecha entrega.

Proceso

Se mostrará al usuario la interfaz a través de la cual podrá ingresar en la agrupación de campos los datos concernientes a: datos del examen físico, datos del examen químico, datos del examen microscópico, prueba de embarazo (opcional) y datos del laboratorista.

Para guardar los datos introducidos en la base de datos se deberá pulsar el botón Guardar.

Salidas

Se modificara el registro actual de la solicitud cambiando el estado de la misma.

GUARDAR EXAMEN

Especificación:

Introducción:

Esta función permitirá al usuario guardar la información correspondiente a una solicitud de examen realizado en el laboratorio.

Entradas

Por el sistema:

⇒ Botón guardar: Para poder guardar el estado de la solicitud el usuario deberá dar click sobre el botón guardar.

Proceso

Se mostrará al usuario un dialogo estándar de información para informarle que los datos han sido guardados de manera correcta de lo contrario se mostrara un dialogo de error es decir que el usuario no ha introducido bien los datos. Estos cambios serán permanentes en la base de datos y se guardaran en la entidad solicitud.

Salidas

Se actualizará el registro modificado en la entidad Solicitud.

IMPRIMIR

Especificación:

Introducción:

Esta función permitirá a los usuarios de LabCli 1.0 poder imprimir el examen actual con los resultados del mismo una vez realizado y guardado.

Entradas

Por el sistema:

⇒ Botón imprimir: Para poder imprimir el resultado actual del examen el usuario pulsara sobre el botón imprimir.

Proceso

Se mostrará al usuario la interfaz principal de impresión del resultado del examen previamente guardado. Se le proporcionara al usuario varias opciones de impresión como son tipo de impresora, papel, si es a dos caras la impresión, etc.

Salidas

Ninguna

INGRESAR SISTEMA

Especificación:

Introducción:

Esta función permitirá a los usuarios registrados ingresar al sistema siempre y cuando el escriban su ID de usuario y contraseña válidos.

Entradas

Por el sistema:

⇒ login: Este campo representa el ID de cada usuario que sea válido en el sistema.

⇒ password: Representa la contraseña asociada al usuario que está intentando ingresar al sistema.

Proceso

Se mostrará al usuario la Frame principal de acceso al sistema LabCli, donde deberá introducir el nombre de usuario y su contraseña, en la cual el sistema verificará si los datos introducidos son válidos, de ser así el usuario podrá ingresar, de lo contrario se presentará un mensaje indicándole que el ID de usuario o contraseña son incorrectos.

Salidas

Ninguna

AYUDA

Especificación:

Introducción:

Esta función permitirá a los usuarios obtener una guía para el correcto manejo y funcionamiento de LabCli 1.0 así como su versión.

Entradas

Por el sistema:

⇒ Botón ayuda o menú Ayuda: Para poder acceder a la función ayuda de LabCli 1.0 el usuario deberá dar click en botón ayuda o en el menú ayuda.

Proceso

Se mostrará al usuario una breve guía acerca del correcto funcionamiento y manejo de LabCli así como un dialogo con la versión actual de LabCli. La guía contendrá información general acerca de cada uno de los componentes de la aplicación además se mostrara la versión actual del sistema LabCli.

Salidas

Ninguna

SALIR SISTEMA LABCLI

Especificación:

Introducción:

Esta función permitirá a los usuarios salir del sistema una vez terminada todas las tareas.

Entradas

Por el sistema:

⇒ Botón salir o menú salir: Este componente representa la función salir del sistema, para poder acceder a ella se deberá dar clic o pulsar en el botón salir o menú salir.

Proceso

Se mostrará al usuario un dialogo pidiendo la confirmacion de que si realmente desea salir de LabCli segun la eleccion del usuario se saldra del sistema o podra continuar trabajando.

Salidas

Ninguna

Banco de Sangre

26 REQUISITOS ESPECÍFICOS:

Requisitos Funcionales:

NUEVO EXAMEN (Banco de sangre)
Especificación:
Introducción:
Esta función permitirá al usuario registrar en el sistema los datos correspondientes a uno o varios exámenes en particular correspondientes al área de Banco de sangre.

Entradas:

Por pantalla:

⇒ Datos generales: Esta agrupación de campos representa la características generales de la sangre como los son:

- Grupo sanguineo.
- Factor RH.

⇒ Datos del examen: Esta agrupación de campos representa cada una de las caracterizas de los datos de los exámenes correspondientes al área de banco de sangre.

- Prueba cruzada.
- Compatible.
- Unidad No.

⇒ Datos de la prueba de Coombs : Esta agrupación de campos representa cada una de las características asociadas a la prueba de coombs como lo son:

- Coombs Directo.
- Coombs Indirecto.

⇒ Otras pruebas: Esta agrupación de campos es opcional y representa alguna pruebas adicionales realizadas a la muestra de la sangre como lo son:

- Otras pruebas.

⇒ Datos del laboratorista: Esta agrupación de campos representa la información concerniente a quien realiza el examen así como la fecha en que se realiza. Sus características son:

- Nombre del laboratorista.
- Fecha entrega.

Proceso

Se mostrará al usuario la interfaz a través de la cual podrá ingresar en la agrupacion de campos los datos concernientes a: datos generales, datos del examen, datos de la prueba de coombs y otras pruebas(opcional).

Para guardar los datos introducidos en la base de datos se deberá pulsar el botón Guardar.

Salidas

Se modificara el registro actual de la solicitud cambiando el estado de la misma.

GUARDAR EXAMEN

Especificación:

Introducción:

Esta función permitirá al usuario guardar la informacion correspondiente a una solicitud de examen realizado en el laboratorio.

Entradas

Por el sistema:

⇒ Boton guardar: Para poder guardar el estado de la solicitud el usuario debera dar click sobre el boton guardar .

Proceso

Se mostrará al usuario un dialogo estandar de informacion para informarle que los datos han sido guardados de manera correcta de lo contrario se mostrara un dialogo de error es decir que el usuario no ha introducido bien los datos. Estos cambios seran permanentes en la base de datos y se guardaran en la entidad solicitud.

Salidas

Se actualizará el registro modificado en la entidad Solicitud.

IMPRIMIR

Especificación:

Introducción:

Esta función permitirá a los usuarios de LabCli poder imprimir el examen actual con los resultados del mismo una vez realizado y guardado.

Entradas

Por el sistema:

⇒ Botón imprimir: Para poder imprimir el resultado actual del examen el usuario pulsara sobre el botón imprimir.

Proceso

Se mostrará al usuario la interfaz principal de impresión del resultado del examen previamente guardado. Se le proporcionara al usuario varias opciones de impresión como son tipo de impresora, papel, si es a dos caras la impresión, etc.

Salidas

Ninguna

INGRESAR SISTEMA

Especificación:

Introducción:

Esta función permitirá a los usuarios registrados ingresar al sistema siempre y cuando el escriban su ID de usuario y contraseña válidos.

Entradas

Por el sistema:

- ⇒ login: Este campo representa el ID de cada usuario que sea válido en el sistema.
- ⇒ password: Representa la contraseña asociada al usuario que está intentando ingresar al sistema.

Proceso

Se mostrará al usuario la Frame principal de acceso al sistema LabCli 1.0, donde deberá introducir el nombre de usuario y su contraseña, en la cual el sistema verificará si los datos introducidos son válidos, de ser así el usuario podrá ingresar, de lo contrario se presentará un mensaje indicándole que el ID de usuario o contraseña son incorrectos.

Salidas

Ninguna

AYUDA

Especificación:

Introducción:

Esta función permitirá a los usuarios obtener una guía para el correcto manejo y funcionamiento de LabCli asi como su versión.

Entradas

Por el sistema:

⇒ Boton ayuda o menú Ayuda: Para poder acceder a la función ayuda de LabCli el usuario deberá dar click en botón ayuda o en el menú ayuda.

Proceso

Se mostrará al usuario una breve guía acerca del correcto funcionamiento y manejo de LabCli así como un dialogo con la versión actual de LabCli. La guía contendrá información general acerca de cada uno de los componentes de la aplicación además se mostrara la versión actual del sistema LabCli 1.0.

Salidas

Ninguna

SALIR SISTEMA LABCLI

Especificación:

Introducción:

Esta función permitirá a los usuarios salir del sistema una vez terminada todas las tareas.

Entradas

Por el sistema:

⇒ Boton salir o menú salir: Este componente representa la función salir del sistema, para poder acceder a ella se deberá dar clic o pulsar en el botón salir o menú salir.

Proceso

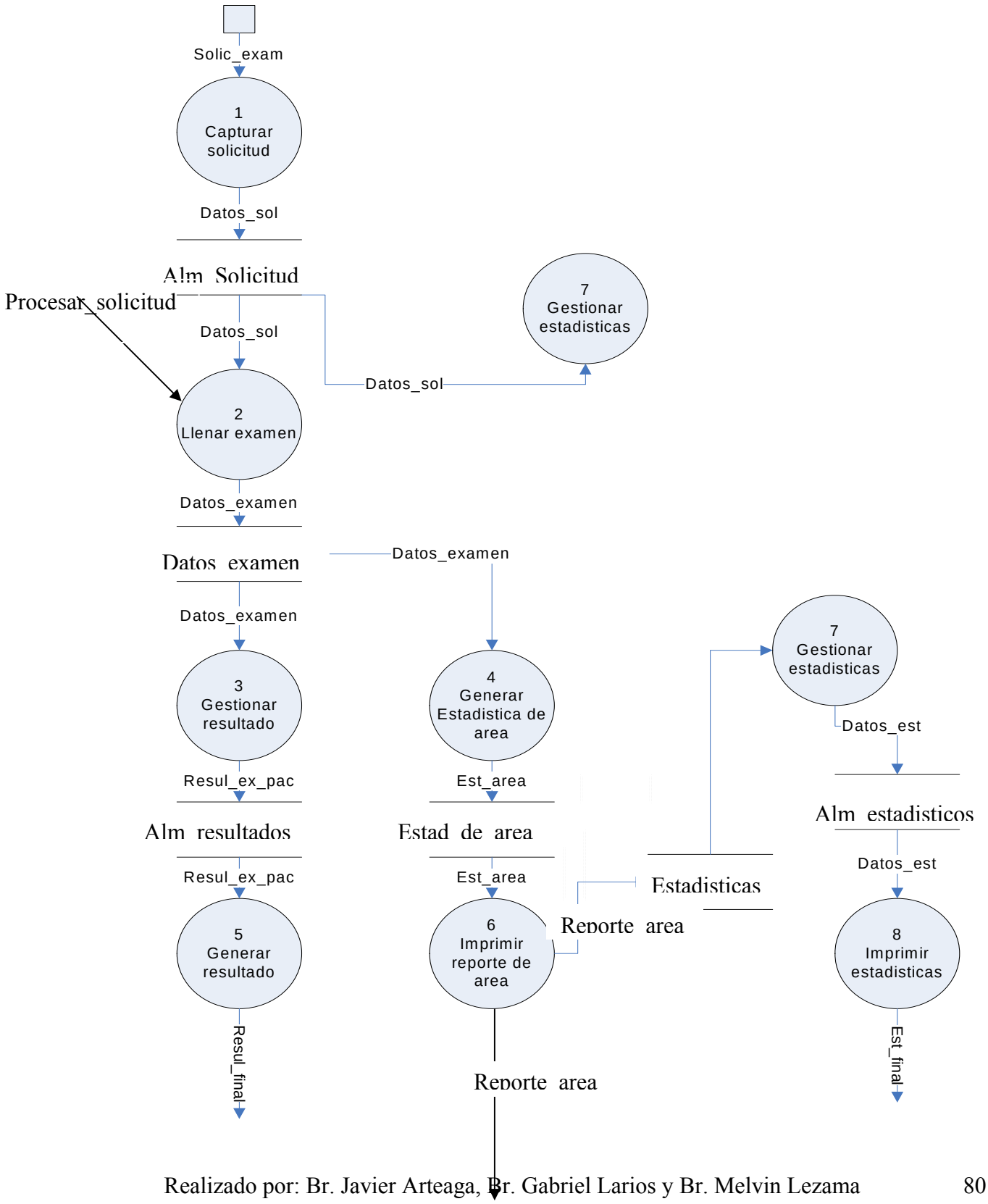
Se mostrará al usuario un dialogo pidiendo la confirmación de que si realmente desea salir de LabCli según la elección del usuario se saldrá del sistema o podrá continuar trabajando.

Salidas

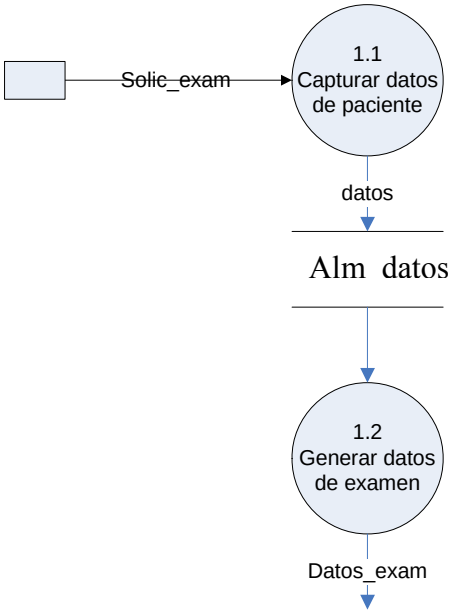
Ninguna

27 DIAGRAMA DE FLUJO DE DATOS

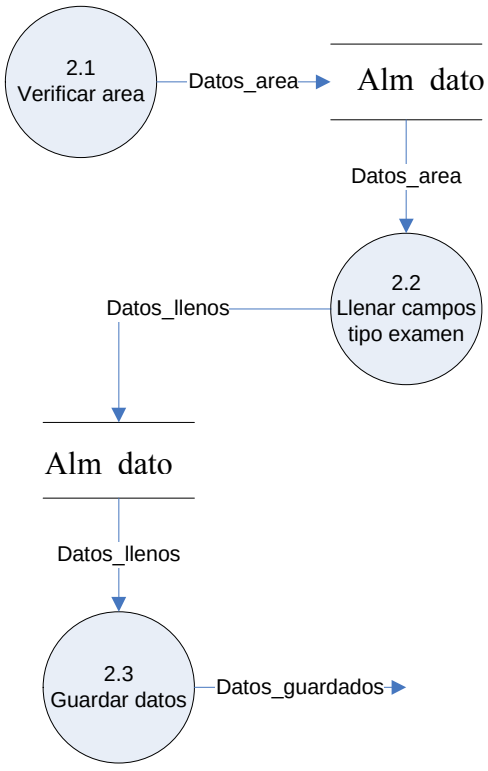
Nivel 1



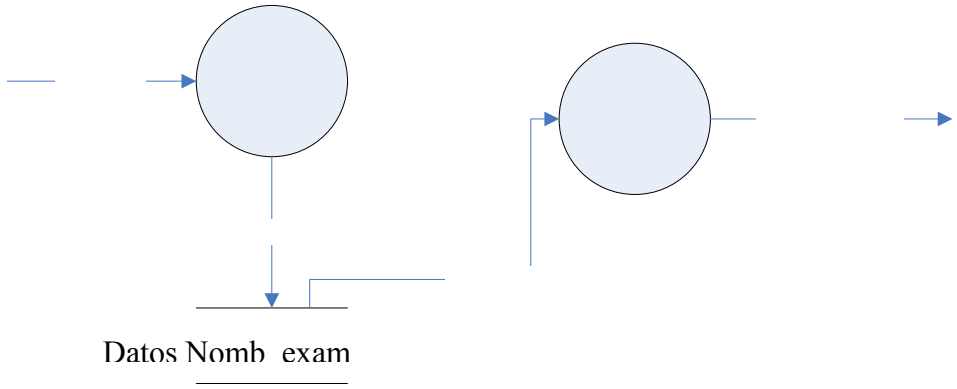
Nivel 2 proceso 1



Nivel 2 proceso 2



Nivel 2 proceso 3

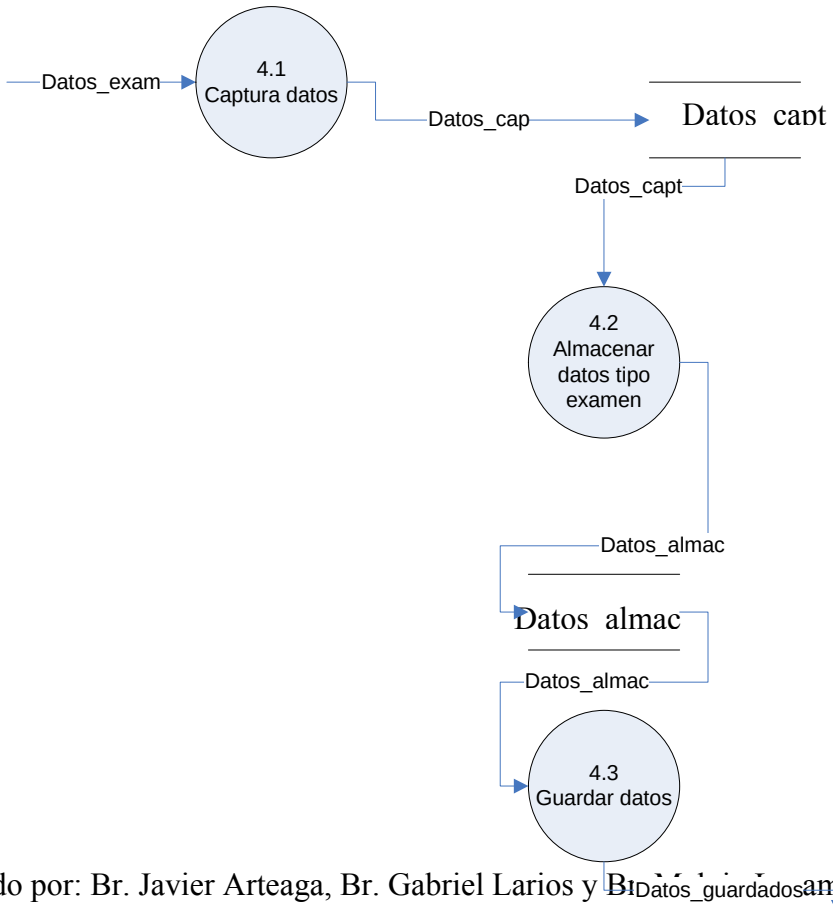


Datos_ex

3.1
Nombre
examen

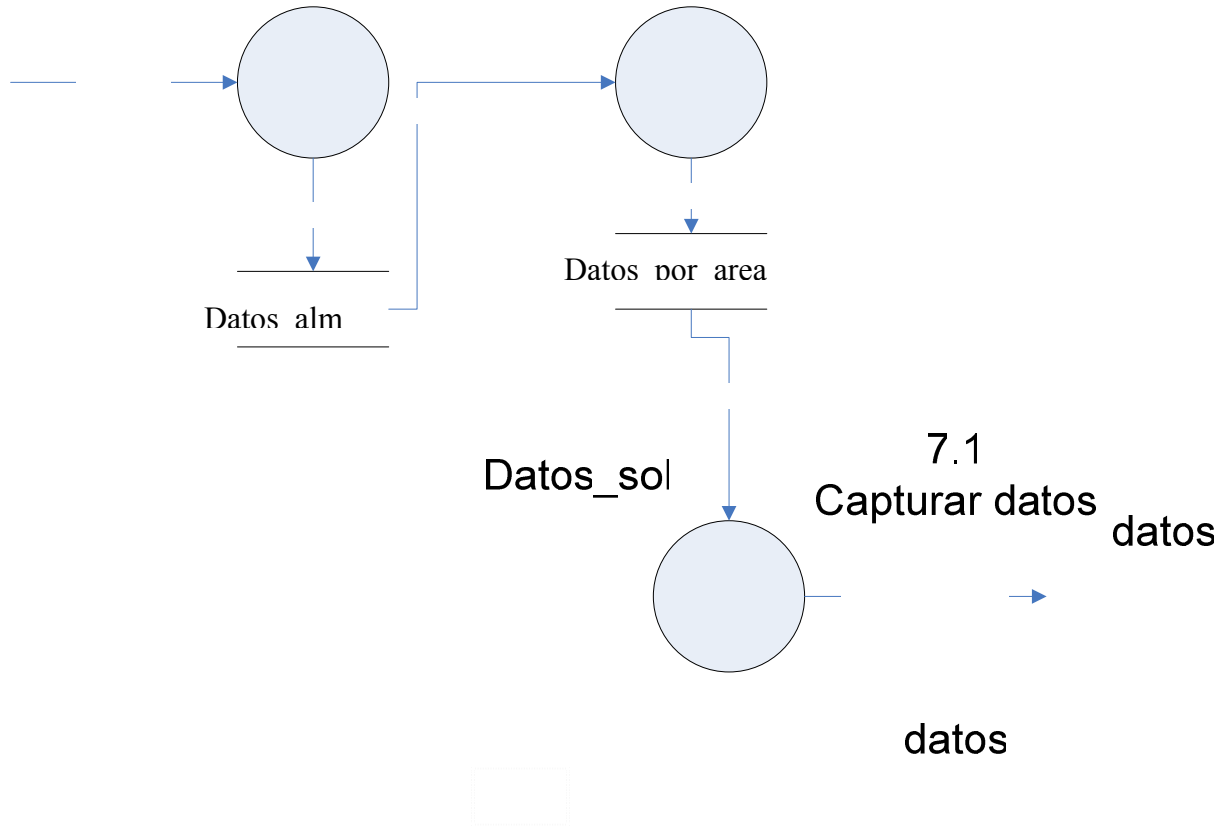
Nivel 2 proceso 4

Datos_nom



Datos_nom

Nivel 2 proceso 7



28 DICCIONARIO DE DATOS

Sol_examen = num_exa+fecha+hora+nombres+apellidos+Nº_expediente+Nº_INSS+ Procedencia+afiliación+esp_medica+examen

Procedencia = medic_interna +SCIN+UCIN+PED+UC+UMI+labor_parto+GIN ORTO+CIRA+CIRB+privado+emp_heodra+emerg+cons_ext+otros

Est_final_hema = tipo_examen + procedencia

Tipo_examen = CEL-FAC+CEL-LE+E-PER+HCT+HGB+MALAR+R-LEU+R-DIF+R-EOS+R-ERI+R-RET+R-TROM+VSG+CLAN+FIB+RET-COAG+TC+TPT+TP+TS

Procedencia = medic_interna +SCIN+UCIN+PED+UC+UMI+labor_parto+GIN ORTO+CIRA+CIRB+privado+emp_heodra+emerg+cons_ext+otros

Est_final_bacter = tipo_examen + procedencia

Tipo_examen = ABG+BAAR+CLA+COPRO+CESP+CLCR+FCO+KOH EFAR+EVU+HEMO+BIOQU+SECR+GRAM+TWR+URO

Procedencia = medic_interna +SCIN+UCIN+PED+UC+UMI+labor_parto+GIN ORTO+CIRA+CIRB+privado+emp_heodra+emerg+cons_ext+otros

Est_final_bsangre = tipo_examen + procedencia

Tipo_examen = ACAL+ACFR+COOD+COOI+FDu+HBS+PCRUZ+VIH+ TRh+HCS+Chagas+hematrocrito+extracciones

Procedencia = medic_interna +SCIN+UCIN+PED+UC+UMI+labor_parto+GIN ORTO+CIRA+CIRB+privado+emp_heodra+emerg+cons_ext+otros

Est_final_bioquimica = tipo_examen + procedencia

Tipo_examen = URIC+ALB+AMY+BIL-D+BIL-T+Ca+CI+COL+COL-HDL +CLE+CPK+CREA-O+CREA-S+FAC+FAL+P+GGT+GLU+Fe+LDH+Mg+BUN+K +PROT-O+PROT-S+RELA-G+Na+T3+T4+TGO+TGP+TRIG+TSH

Procedencia = medic_interna +SCIN+UCIN+PED+UC+UMI+labor_parto+GIN ORTO+CIRA+CIRB+privado+emp_heodra+emerg+cons_ext+otros

Est_final_uro_copro = tipo_examen + procedencia

Tipo_examen = ESP+ÈGO+PBJ+GCH+R-ADD+TIT-GCH+EGH+SOC+ Ph+AZ-RED

Procedencia = medic_interna SCIN+UCIN+PED+UC+UMI+labor_parto+GIN ORTO+CIRA+CIRB+privado+emp_heodra+emerg+cons_ext+otros

Consolidado = total_exam_area + procedencia

total_exam_area = bioquímica+bactereologia+hematologia+banco_sangre+ copro_uro

Procedencia = medic_interna +SCIN+UCIN+PED+UC+UMI+labor_parto+GIN ORTO+CIRA+CIRB+privado+emp_heodra+emerg+cons_ext+otros

Definiciones, acrónimos y abreviaturas:

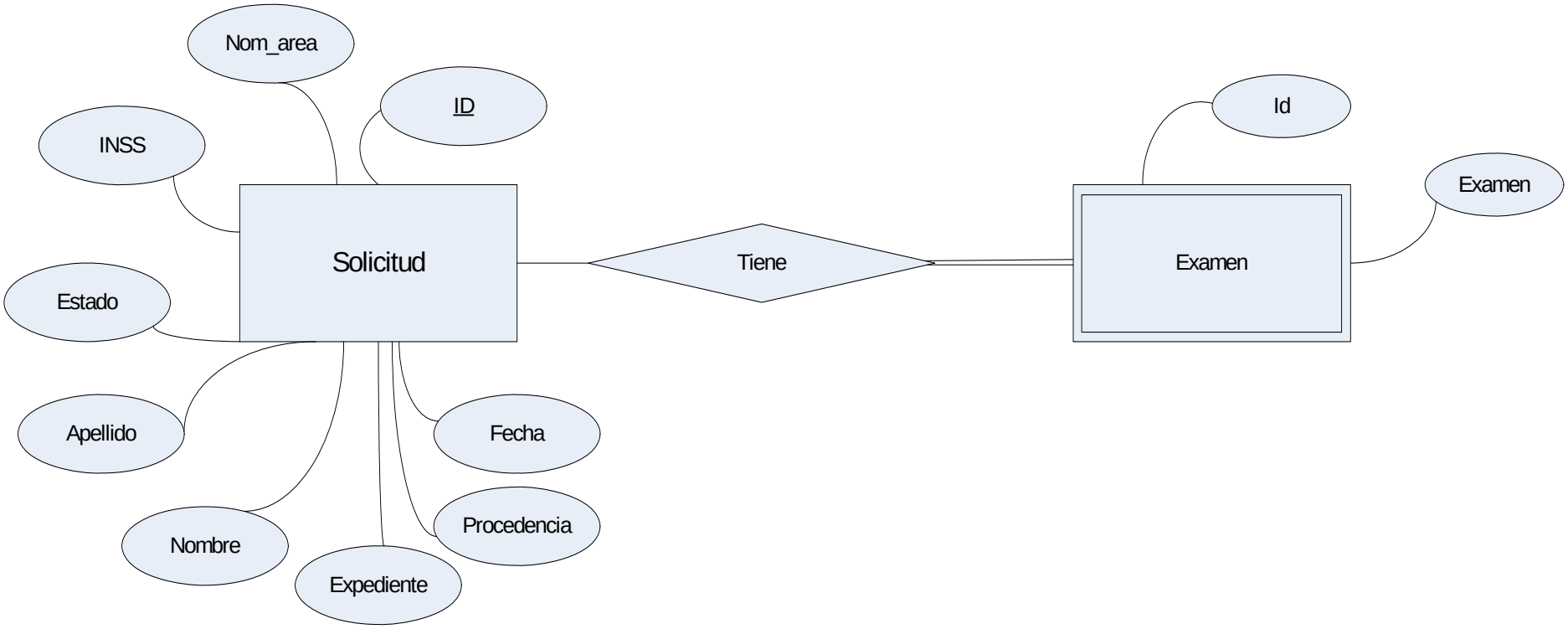
Paciente: Atributo de la entidad solicitud, persona que solicita un o uno exámenes.

Solicitud: Entidad fuerte donde se hace la petición de los diferentes tipos de examen.

Área: Atributo de la entidad solicitud que corresponde al lugar físico donde se realizan los exámenes.

Examen: Atributo de la entidad solicitud donde se especifica el examen o examenes que el paciente se va a realizar.

29 DIAGRAMA DE FLUJO DE DATOS



DISEÑO

DISEÑO DE DATOS

ESTRUCTURA DE BASE DE DATOS POSTGRES

create table solicitud(id integer ,nombres varchar(20),apellidos varchar(20),inss varchar(20),expe varchar(20), procedencia varchar(20),estado varchar(15),cod_area varchar(20),fecha date,primary key(id));

TABLA SOLICITUD		
Nombre Atributo	**Tipo**	**Tamaño**
Id	Integer	Autonumerico
Nombres	Varchar	20
Apellidos	Varchar	20
INSS	Varchar	20
Expe	Varchar	20
Procedencia	Varchar	20
Estado	Varchar	15
Cod_area	Varchar	20
Fecha	date	

create table examen(id integer,examen varchar(30),foreign key(id) references solicitud(id) on delete cascade);

TABLA EXAMEN		
Nombre Atributo	**Tipo**	**Tamaño**
Id	Integer	Autonumerico
Examen	Varchar	20

30 CONCLUSIONES

Al concluir nuestro trabajo monografico, desarrollando nuestro Sistema LabCli 1.0 y haber utilizado **Linux Debian** (Sistema Operativo), **JAVA** (Lenguaje de Programación), **PostgreSQL** (Sistema Gestor de Base de Datos), **Gedit** (Editor de código), **Eclipse** (Entorno de Desarrollo), **DataVision (**Generador de Reportes) concluimos que son herramientas potentes que permiten crear de forma sencilla y rápida sistemas de cualquier índole.

Gracias al esfuerzo realizado y a la utilización de estas herramientas (JAVA, JDBC, PostgreSQL, Datavision, Gedit) se concluyó con el desarrollo de la aplicación para “La Automatización del Laboratorio Clínico en entorno Linux”, propuesta como objetivo fundamental de nuestro trabajo monográfico.

31 RECOMENDACIONES

Al concluir nuestro trabajo monográfico recomendamos:

- Promover entre los estudiantes del Departamento de Computación el desarrollo de sistemas a través de Software Libre.

- Asignar un método de prioridad a un examen donde sea necesario que no tenga que esperar en situaciones que se ameriten.

- Implementar un protocolo de transferencia de datos a través de la red que garantice la integridad de los mismos.

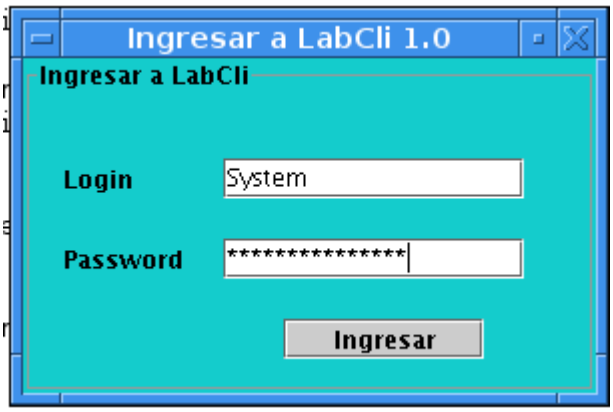
32 BIBLIOGRAFÍA

- **http://es.wikipedia.org/wiki/PostgreSQL**
- **http://es.tldp.org/Postgresql-es/web/navegable/tutorial/x56.html**
- **http://www.gsl.unt.edu.ar/desc/Festival/diapositivas/PostgreSQL.pdf**
- **www.postgresql.org**
- **www.milonic.com**
- **www.nvu.com**
- **www.apache.org**
- **www.php.net**
- **www.desarrolloweb.com**
- **http://es.wikipedia.org/wiki/Internet_Information_Server**
- **http://www.java.net**
- **http://www.javahispano.com**
- **http://www.mysqlhispano.org**
- **http://www.programacion.com/java**
- **http://www.lawebdelprogramador.com**
- **http://www.itapizaco.com.mx**
- **http://www.postgresql.cl**
- **Acceso a base de datos a través de JDBC 2.0. Editorial EIDOS.**
- **JAVA SWING O`REILLY.**
- **JAVA in 60 minutes a day - 2003 – wiley.**
- **Practical Postgresql O`REILLY.**

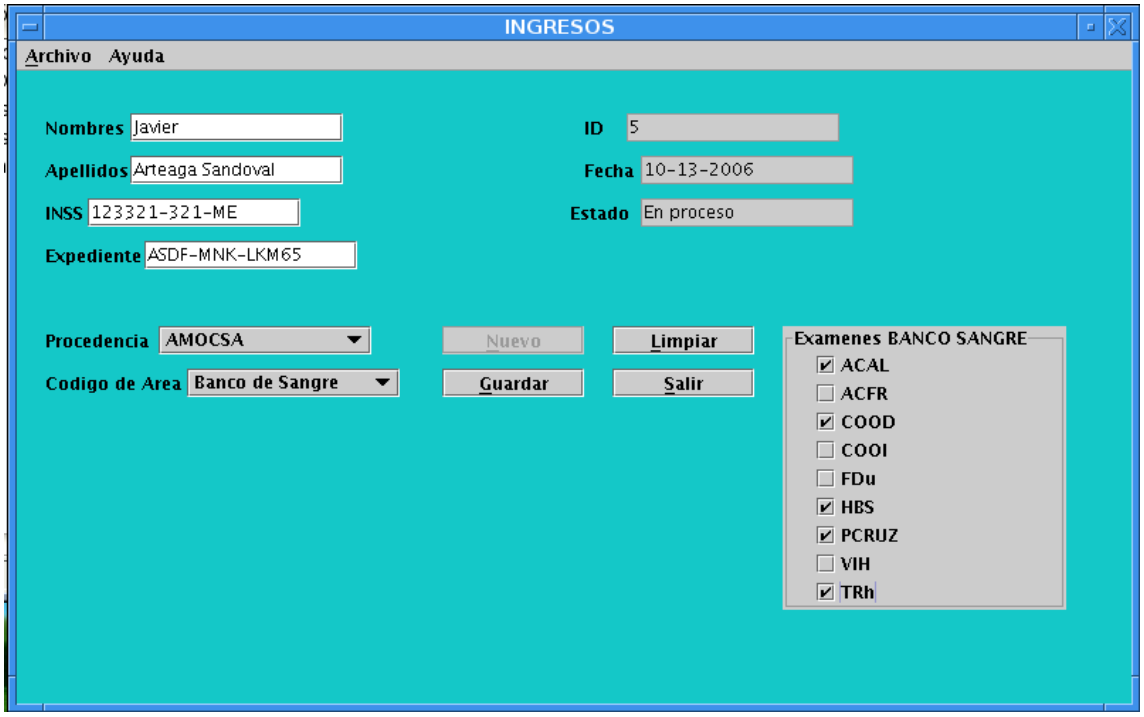
Anexos

Pantalla de la aplicación Principal

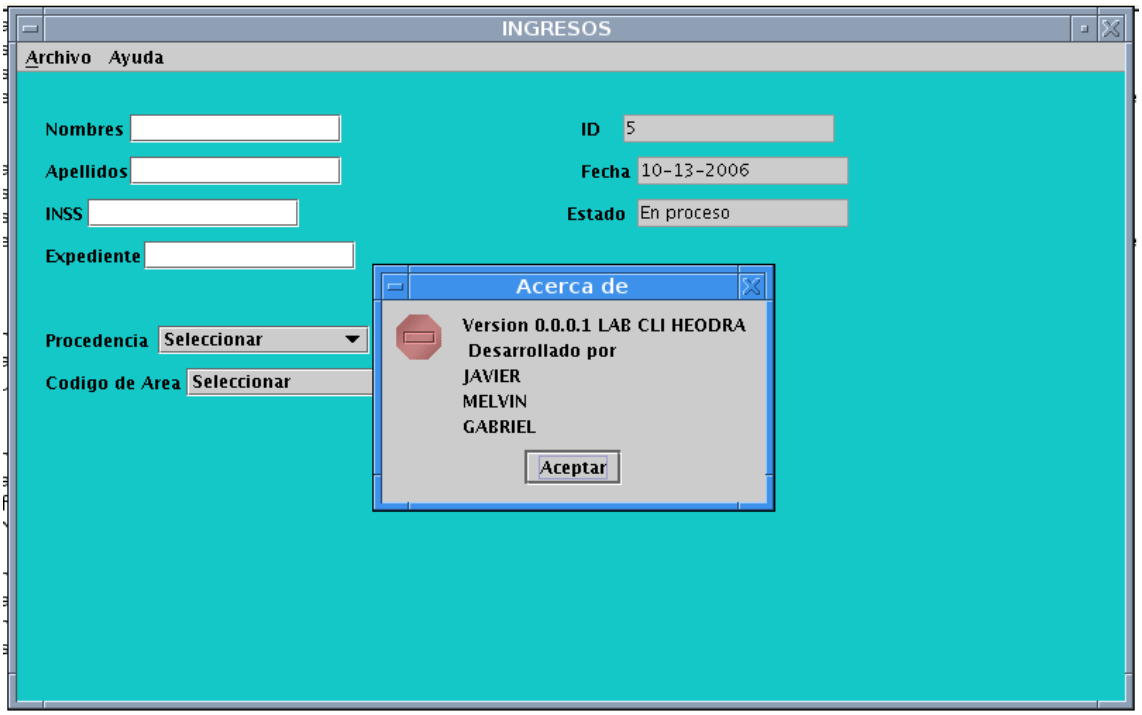
Acceso del Usuario



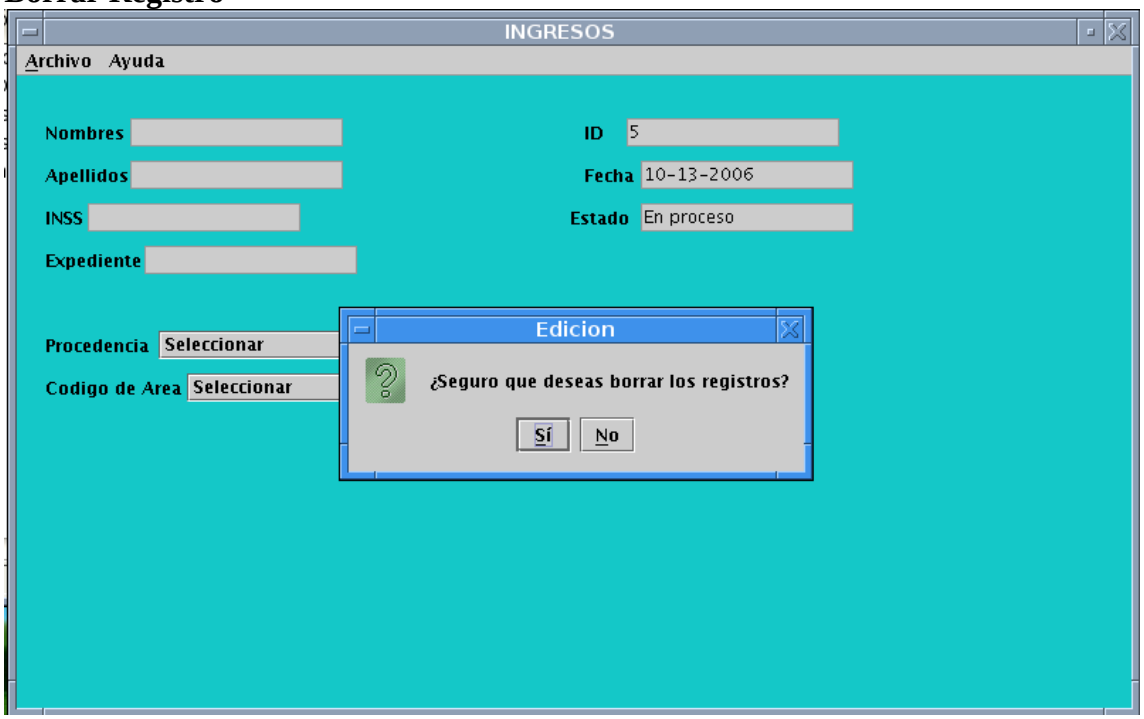
Pantalla de Ingresos



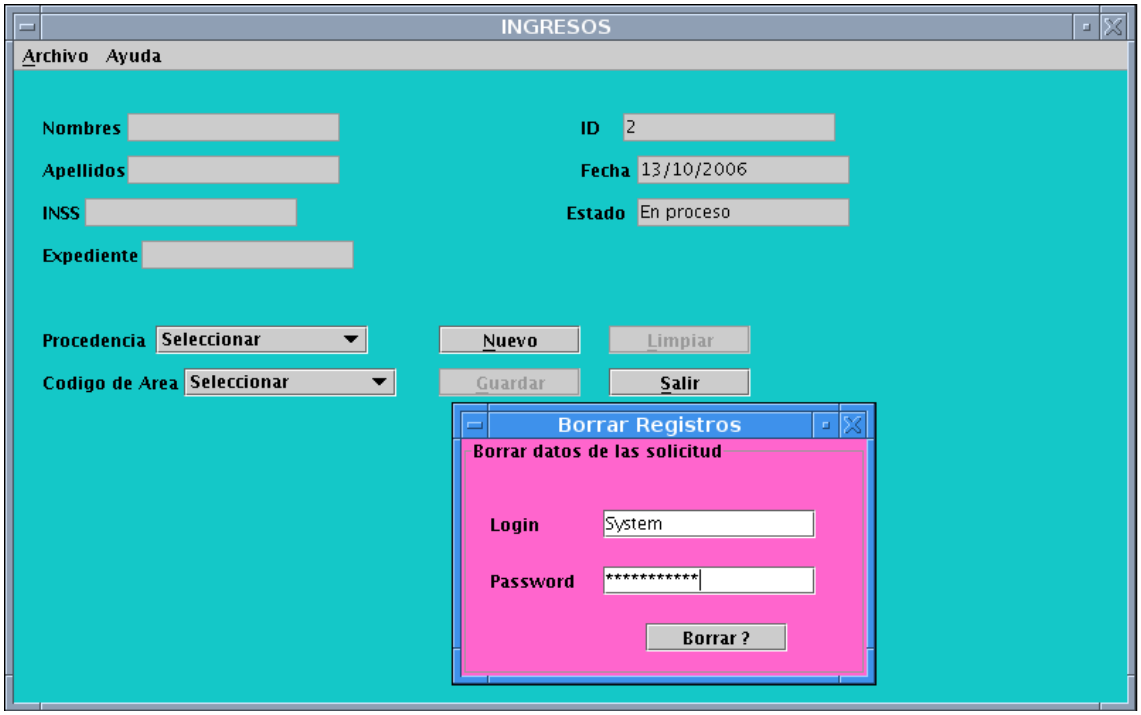
Pantalla de Créditos



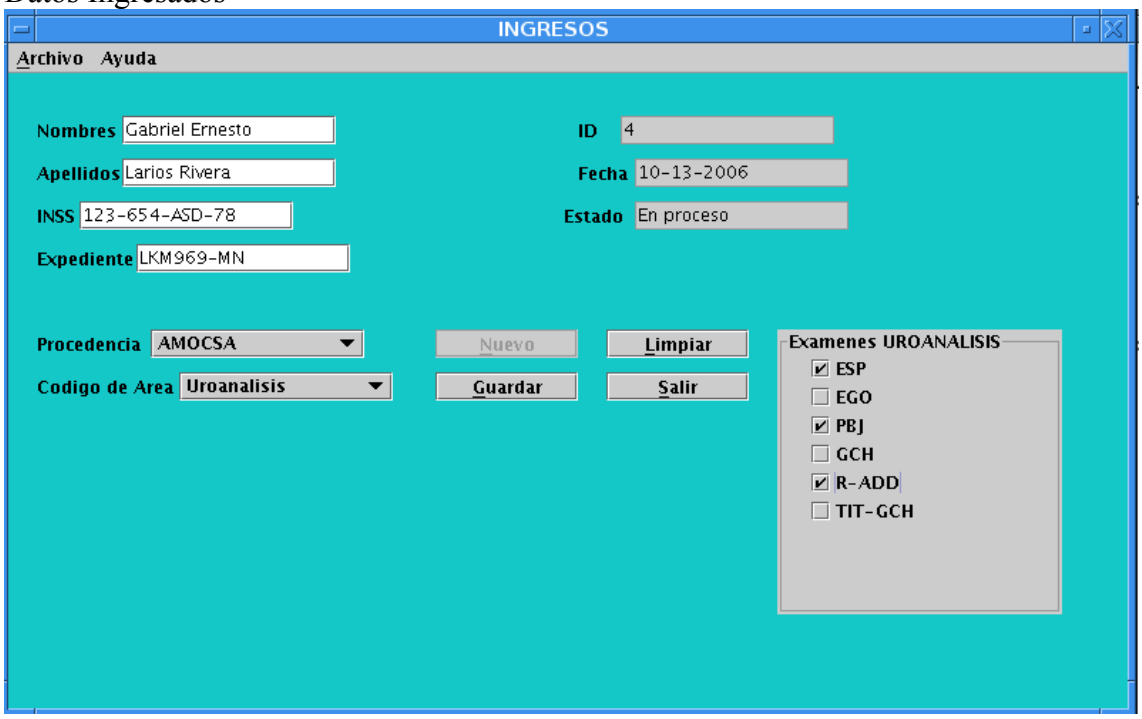
Borrar Registro



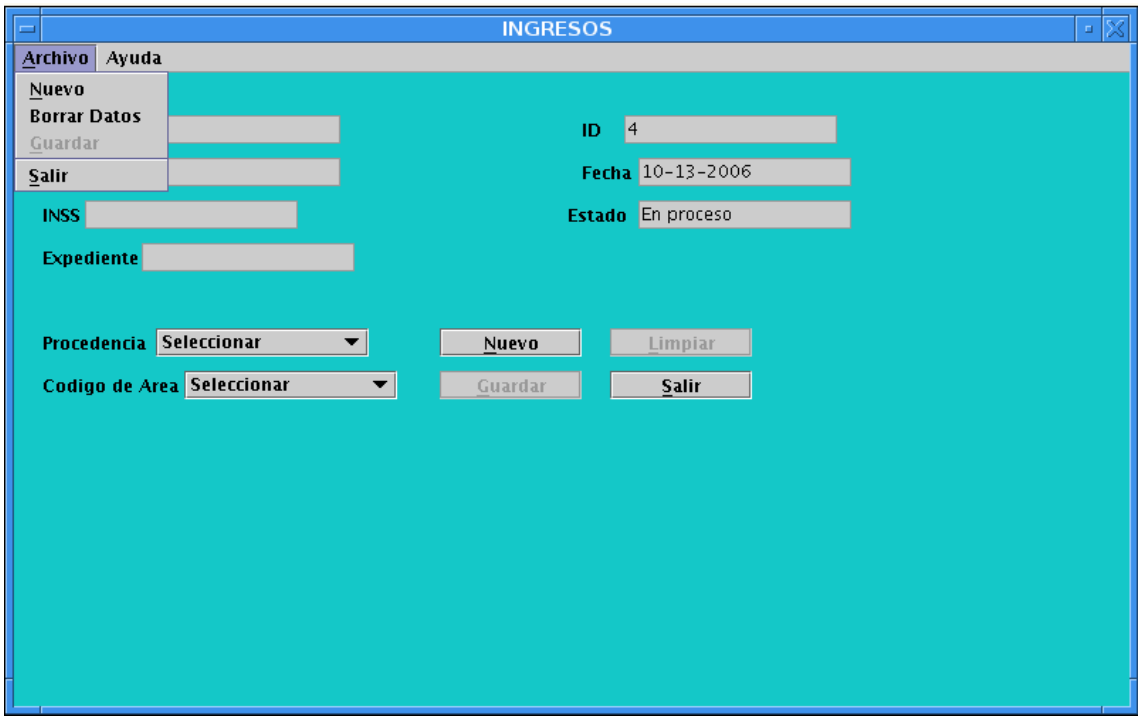
Seguridad al borrar Registros de la Base de Datos

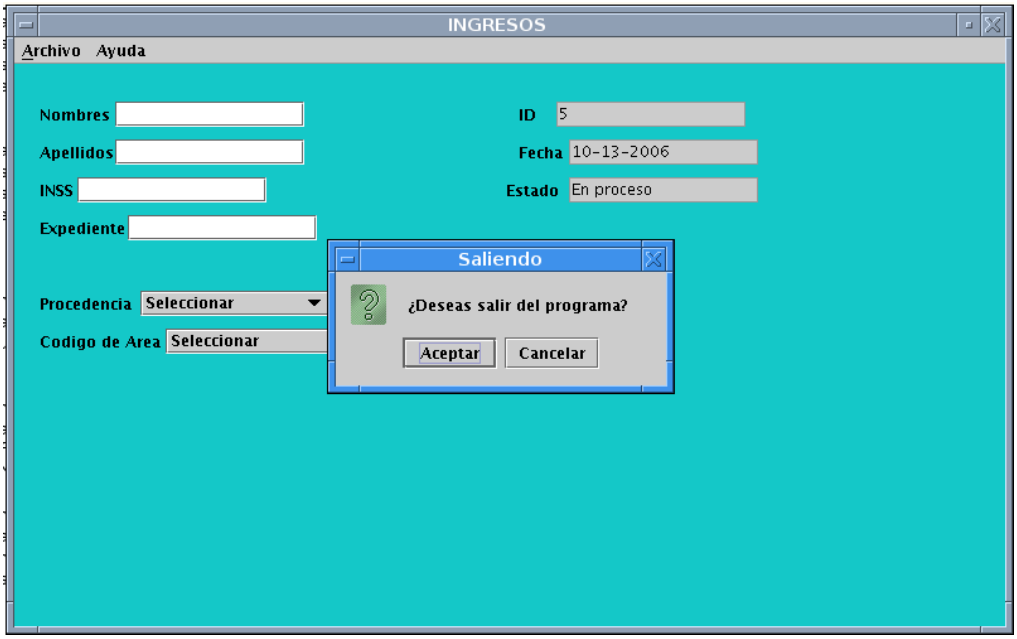


Datos Ingresados



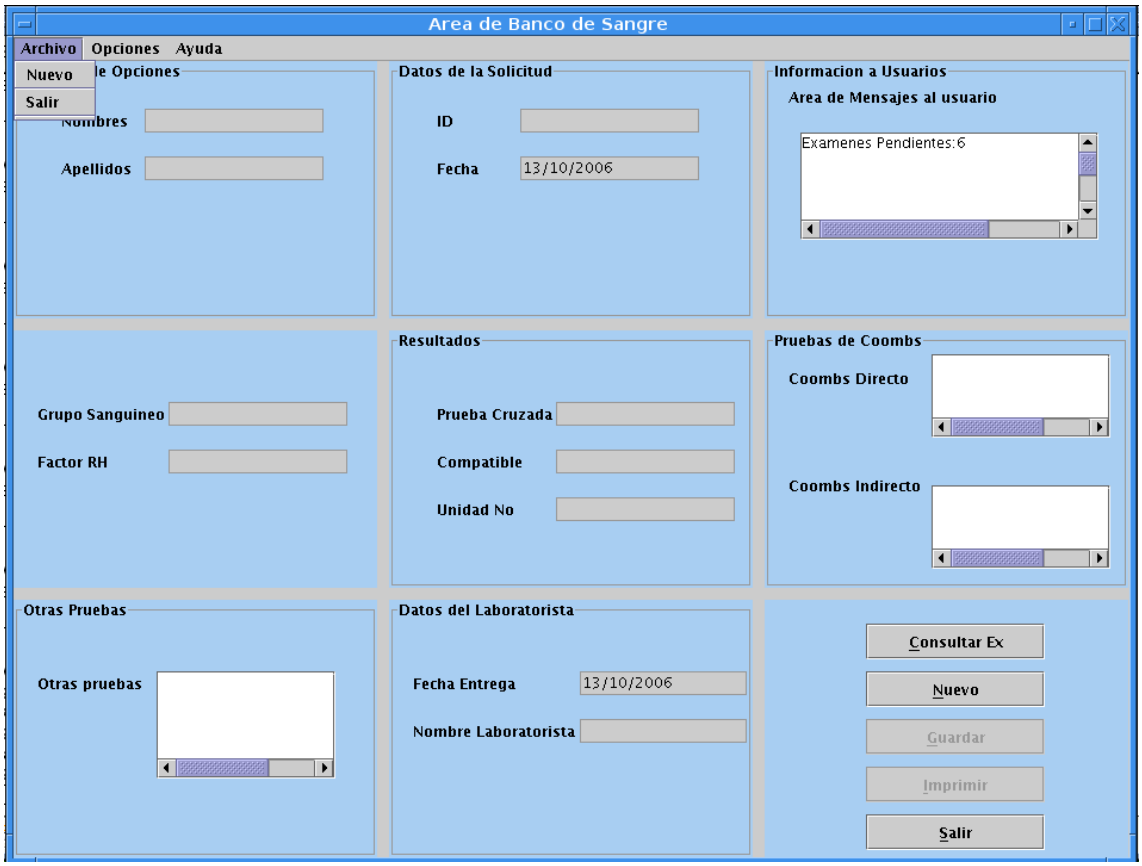
Menus de la Aplicación principal



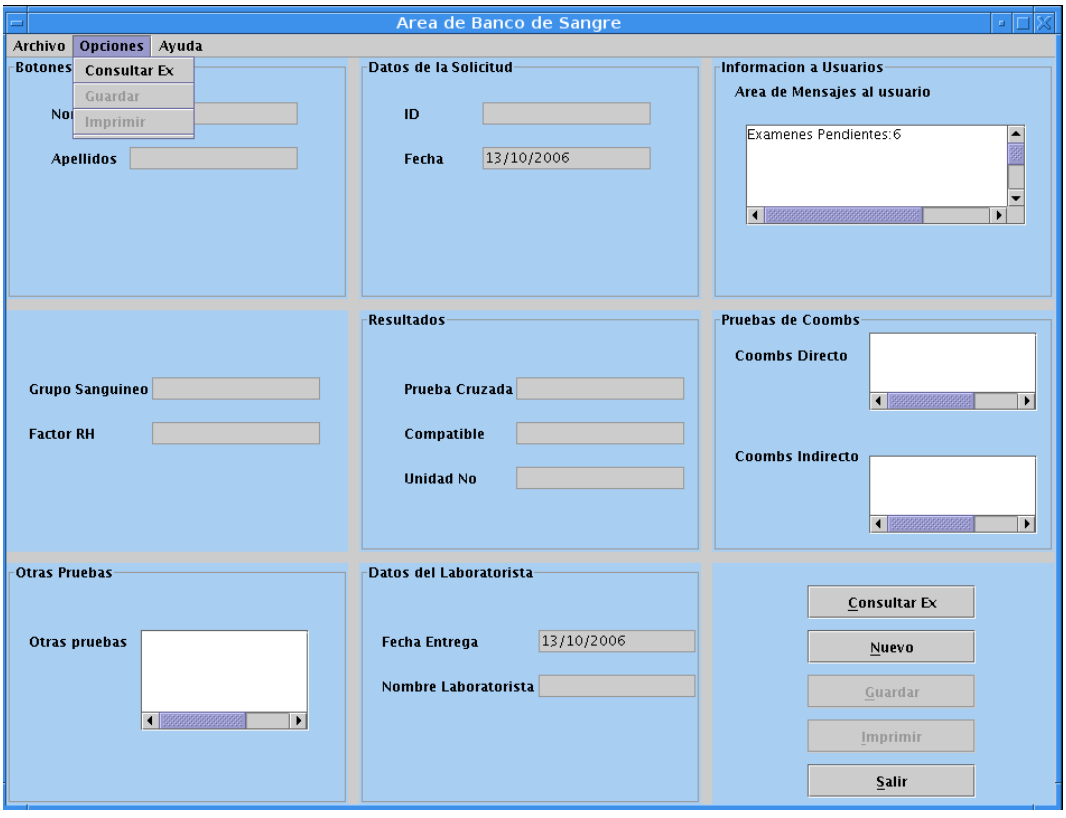


Salir de la Aplicación

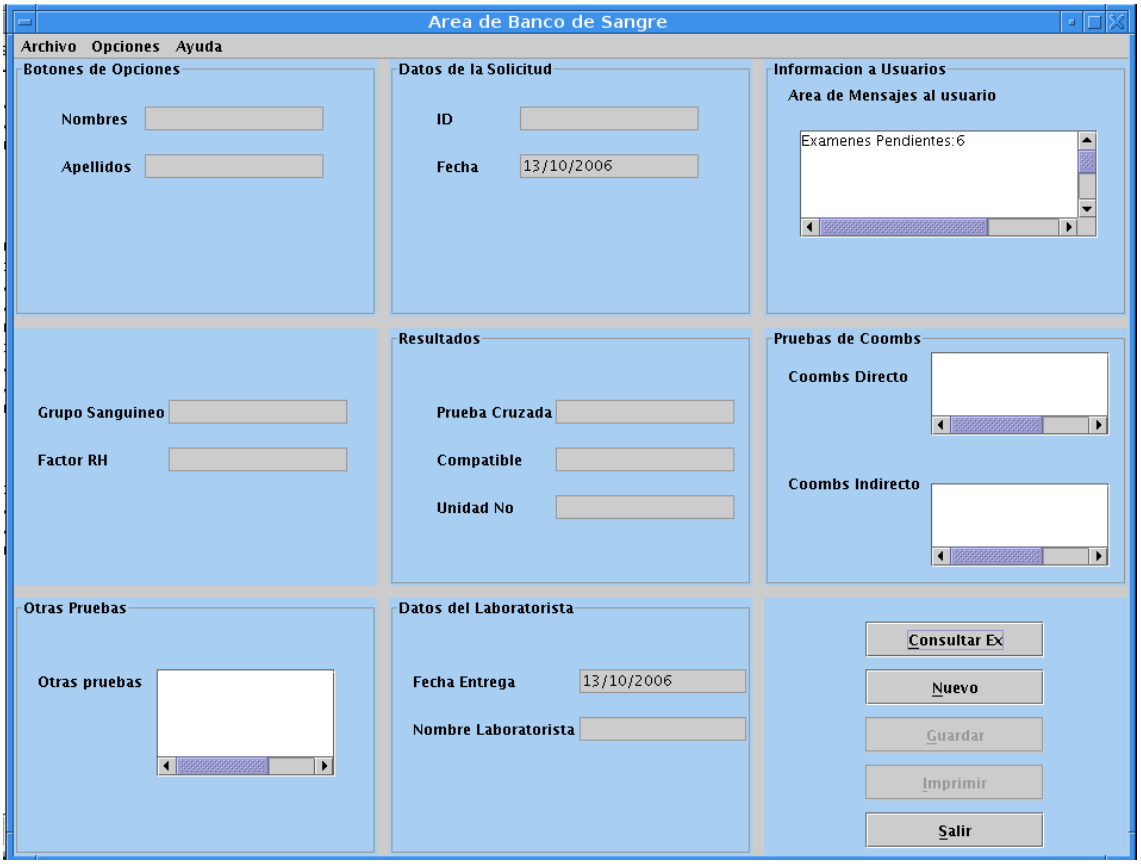
Pantalla de Banco de Sangre



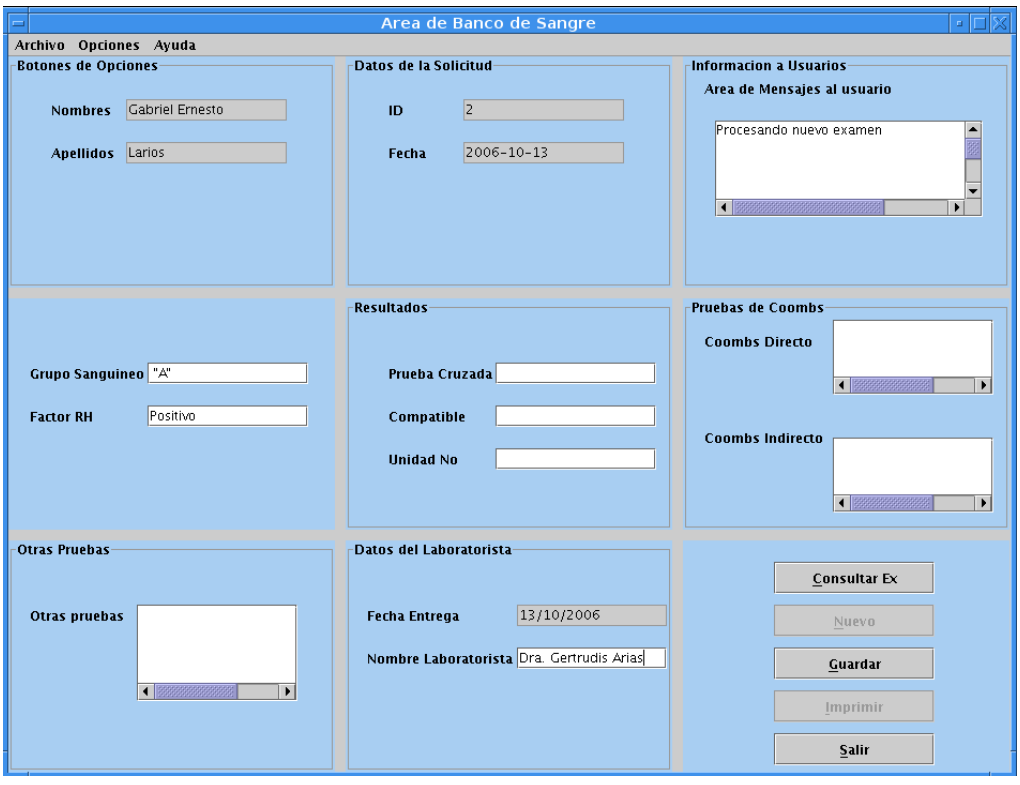
Opciones de Banco de Sangre



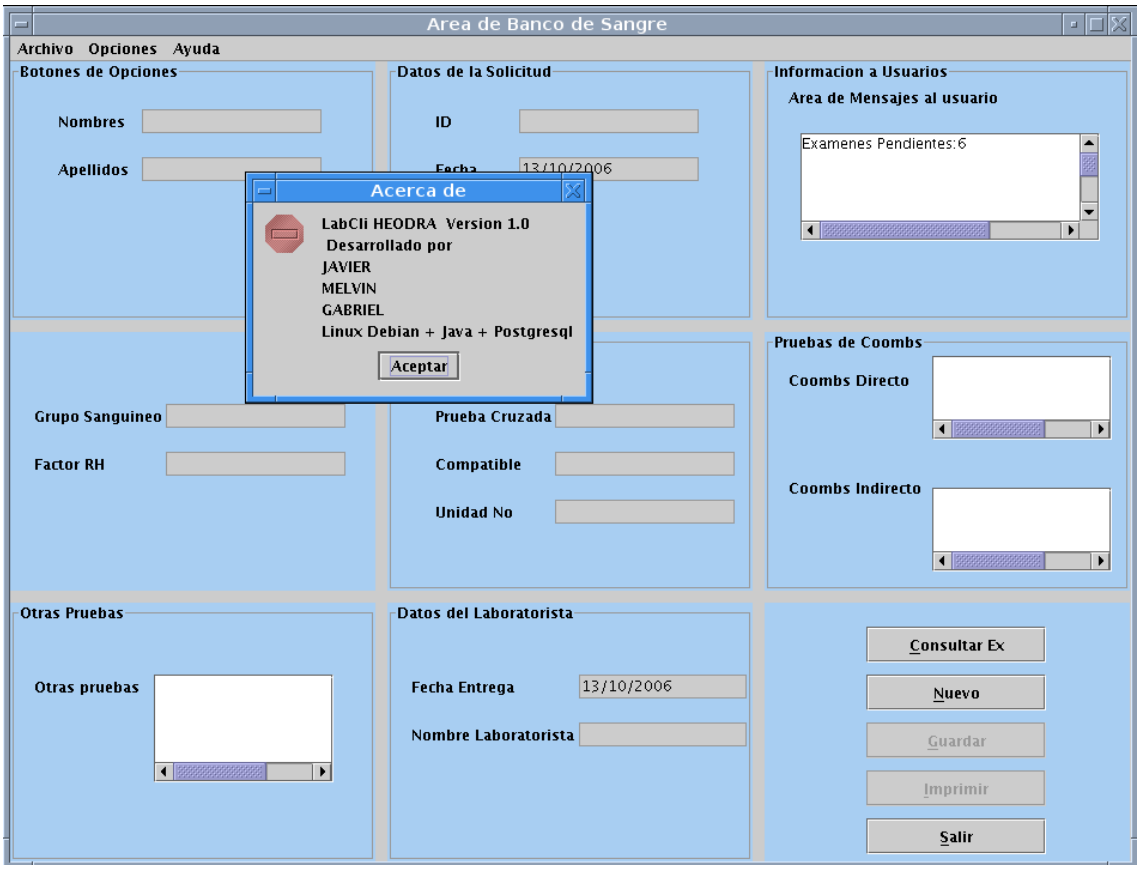
Consulta a la Base de datos de la aplicación principal



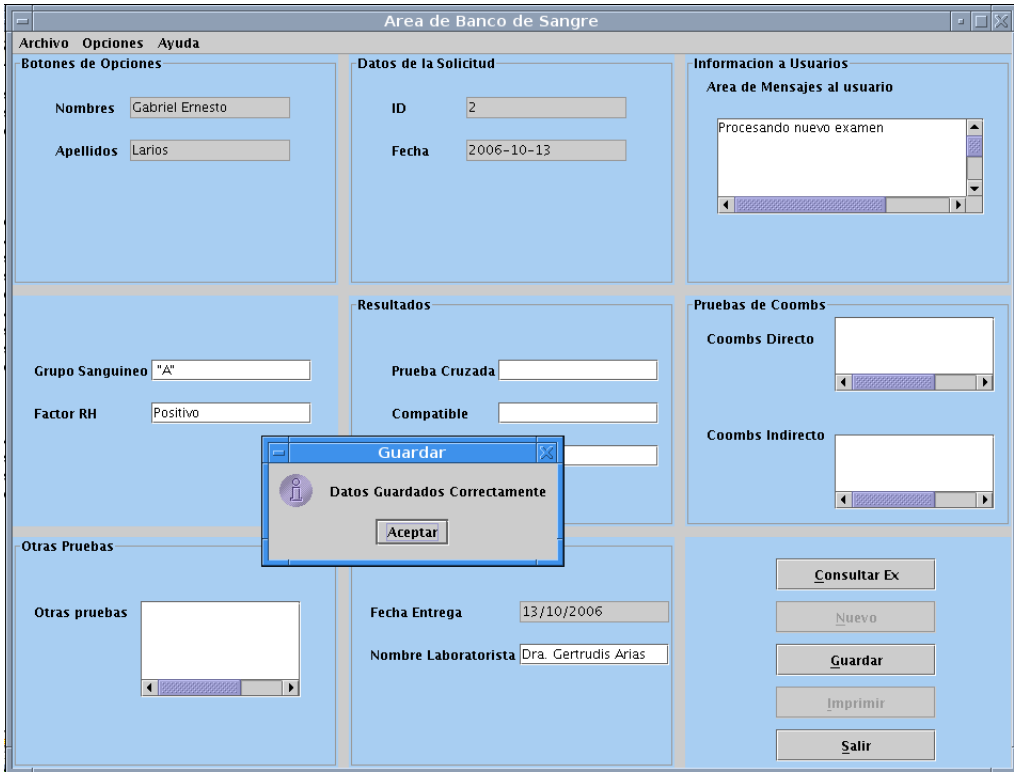
Datos de examenes de la aplicación principal



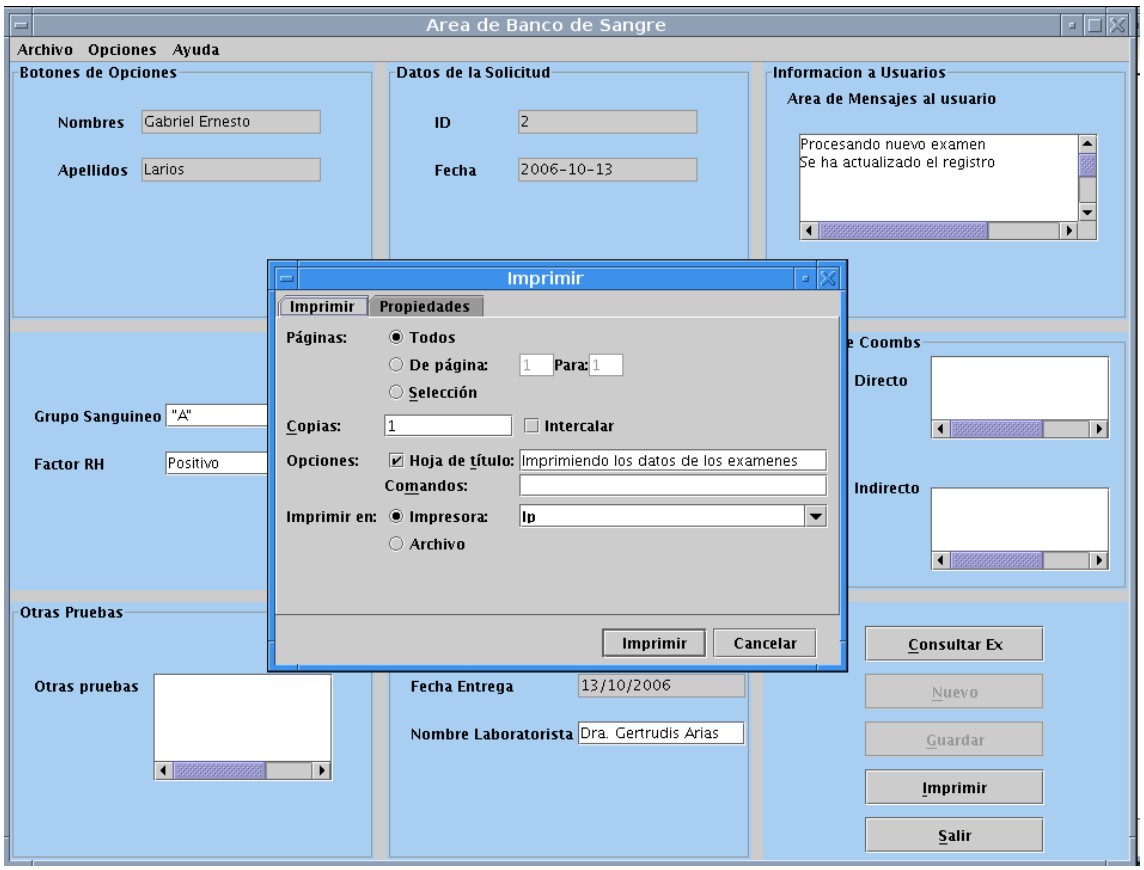
Créditos



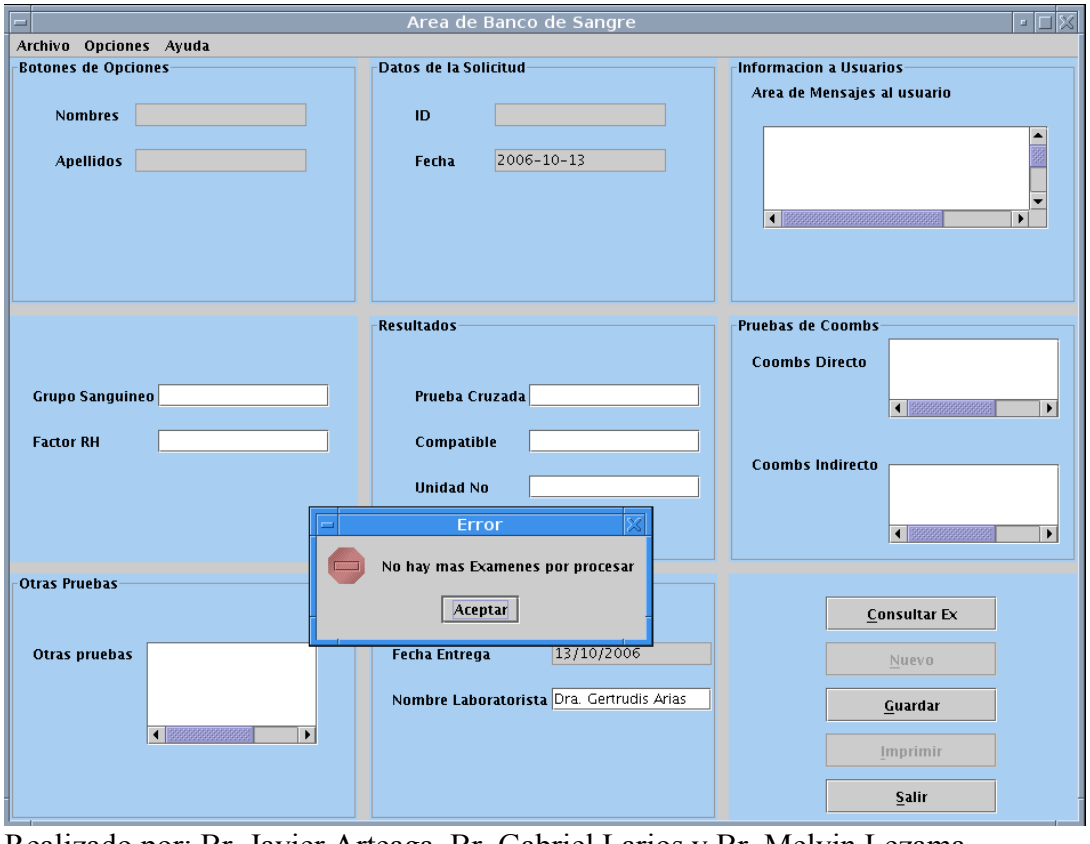
Datos Guardados



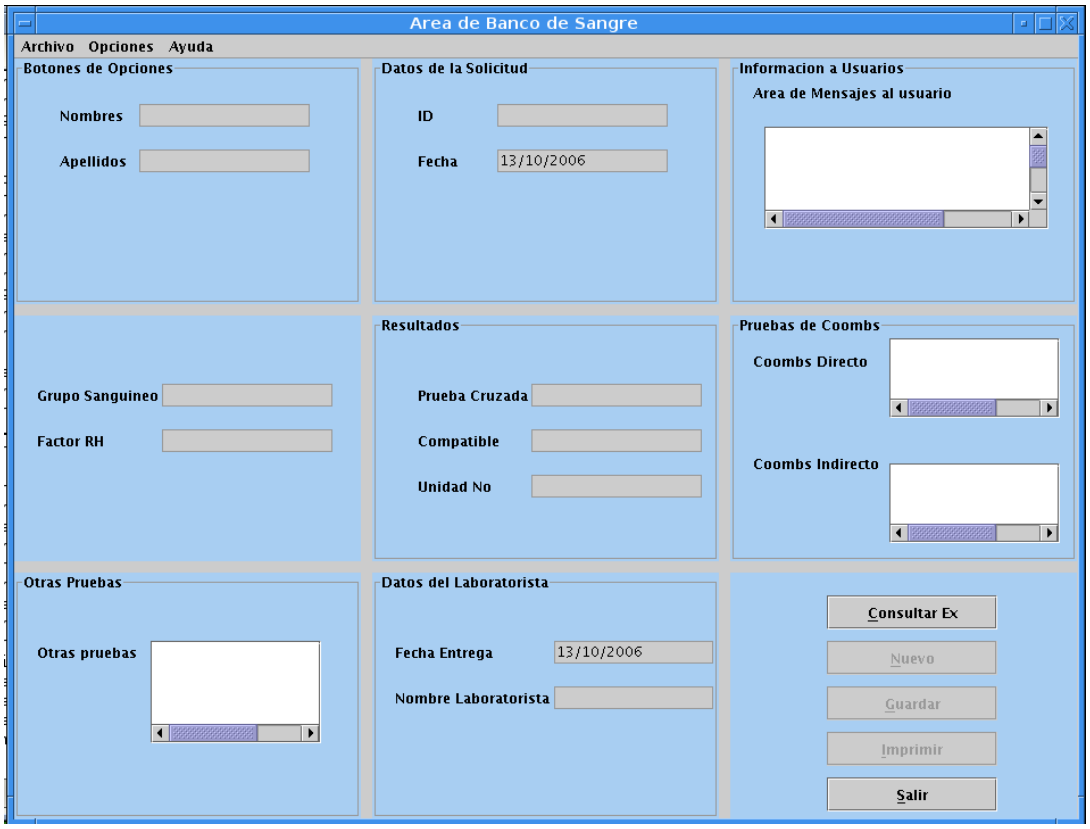
Imprimir Resultado



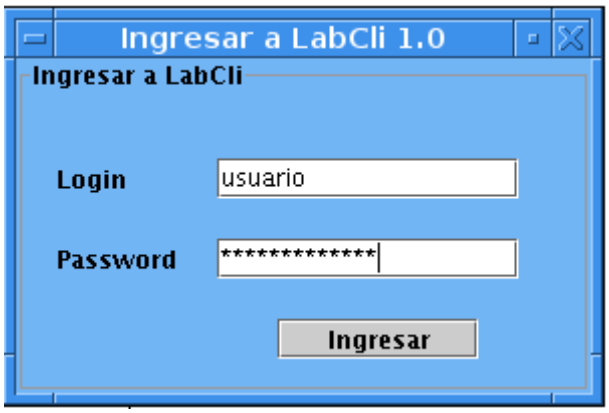
Validación de Datos



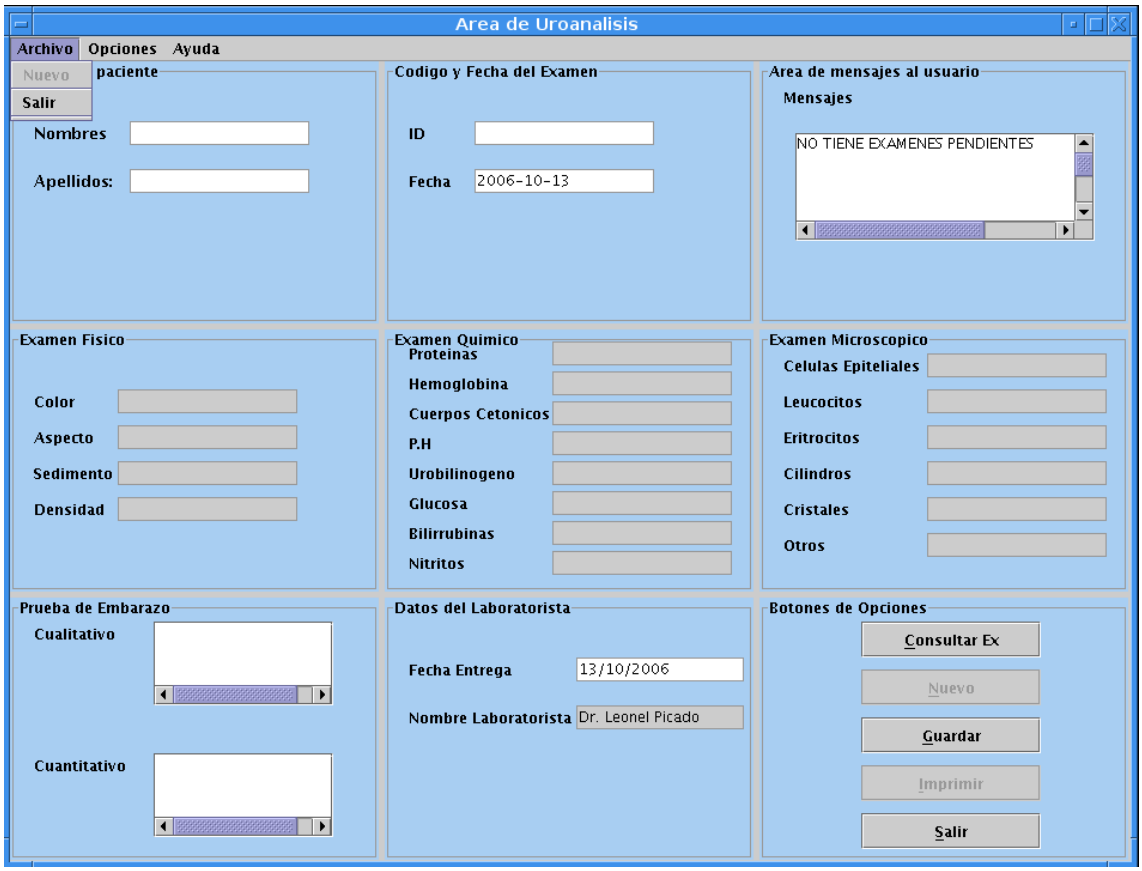
Aplicación vacía



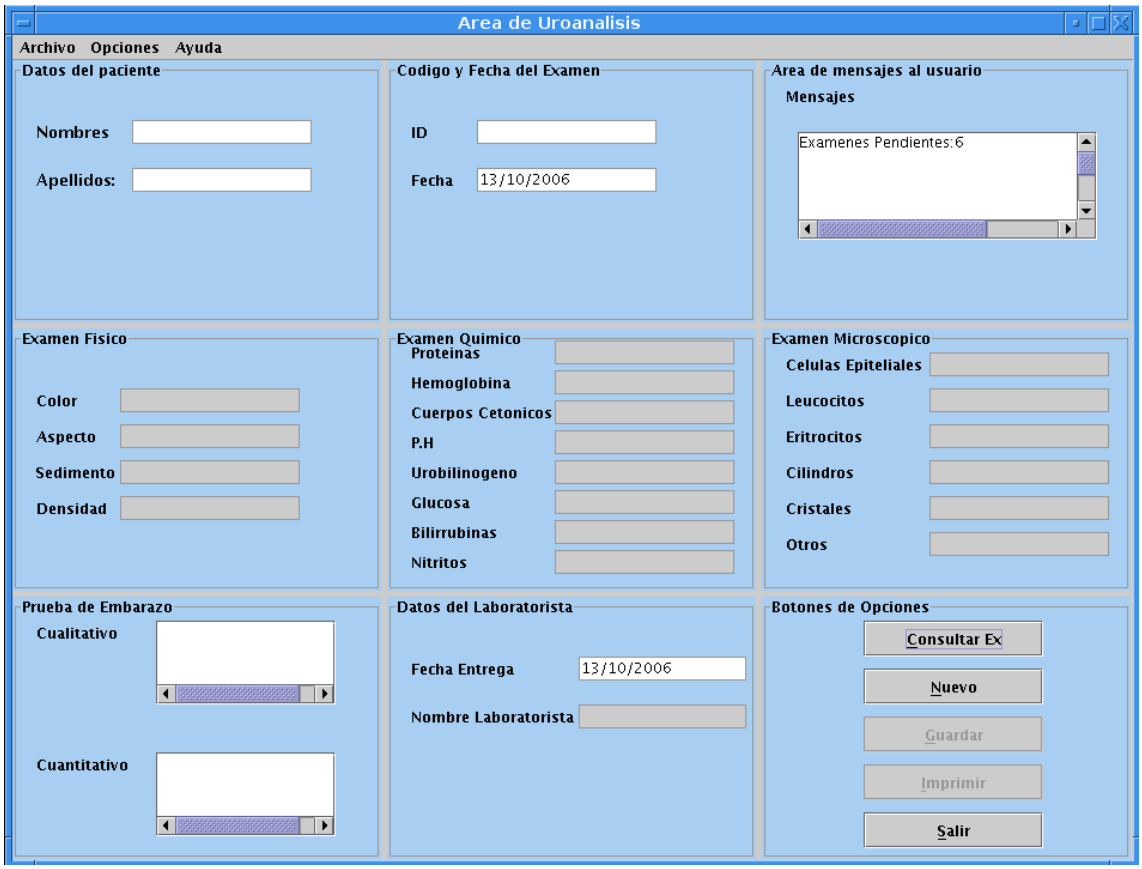
Ingreso de Usuario en Uroanalisis



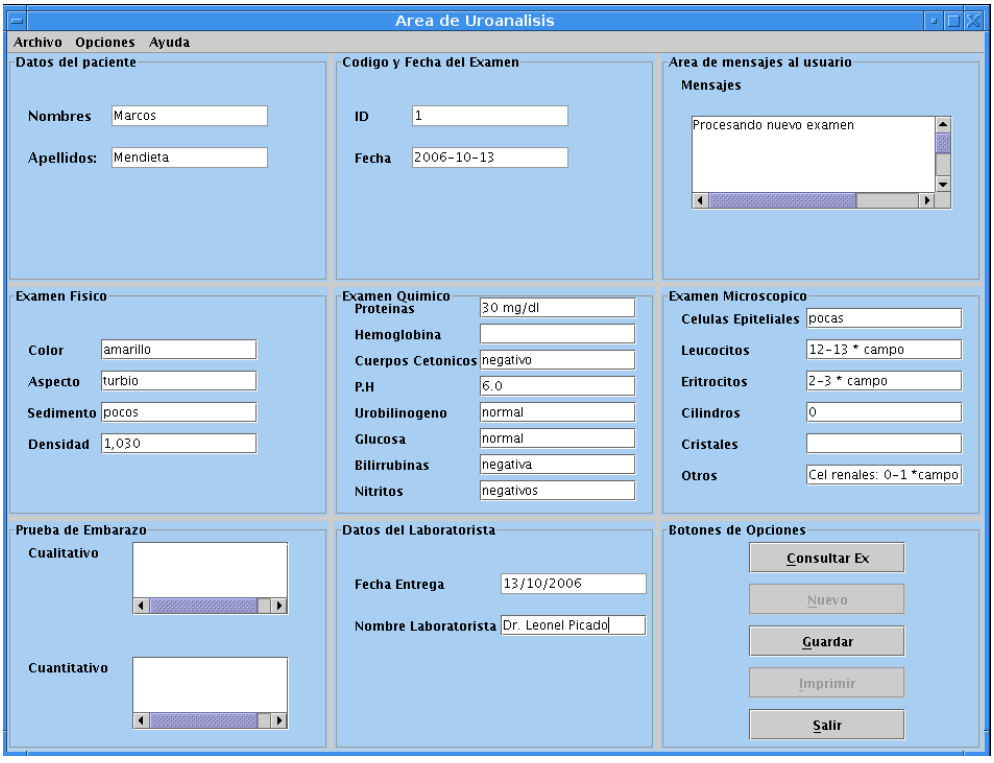
Menús de Uroanalisis



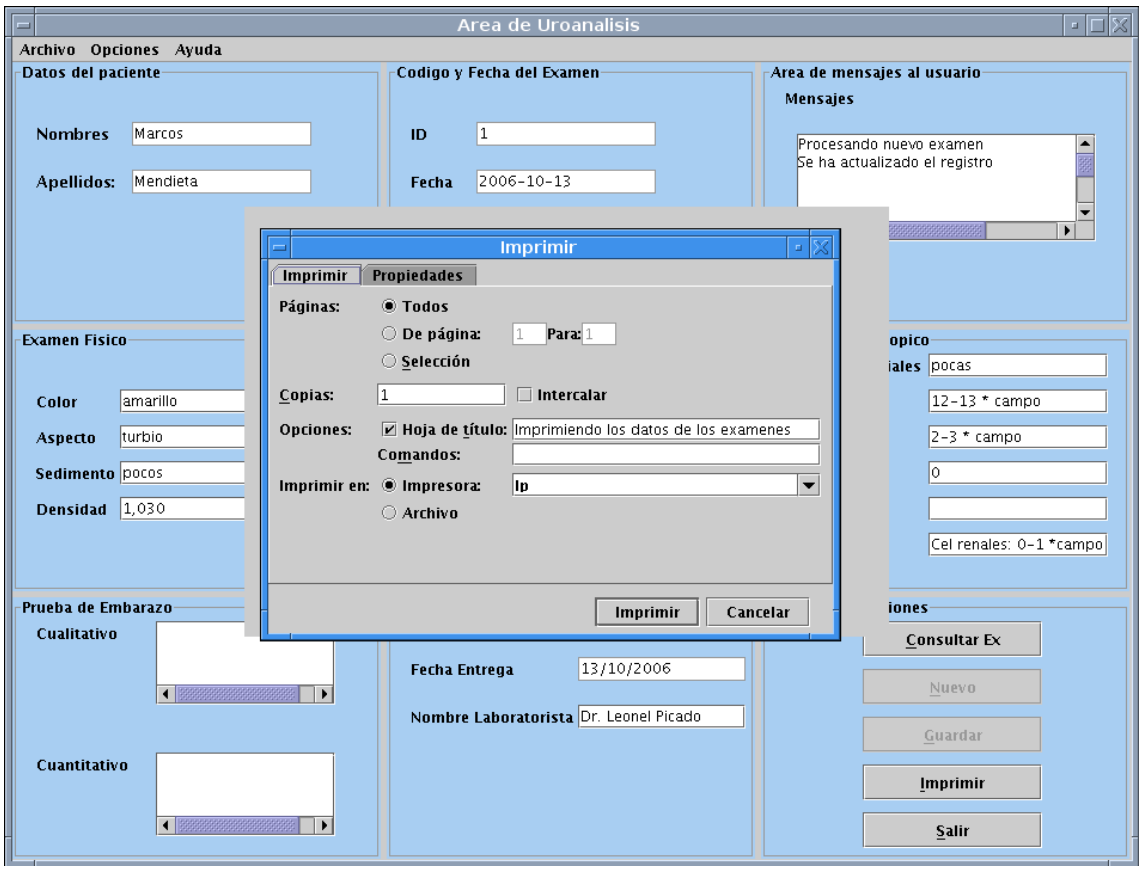
Aplicación consultando a la Base de Datos



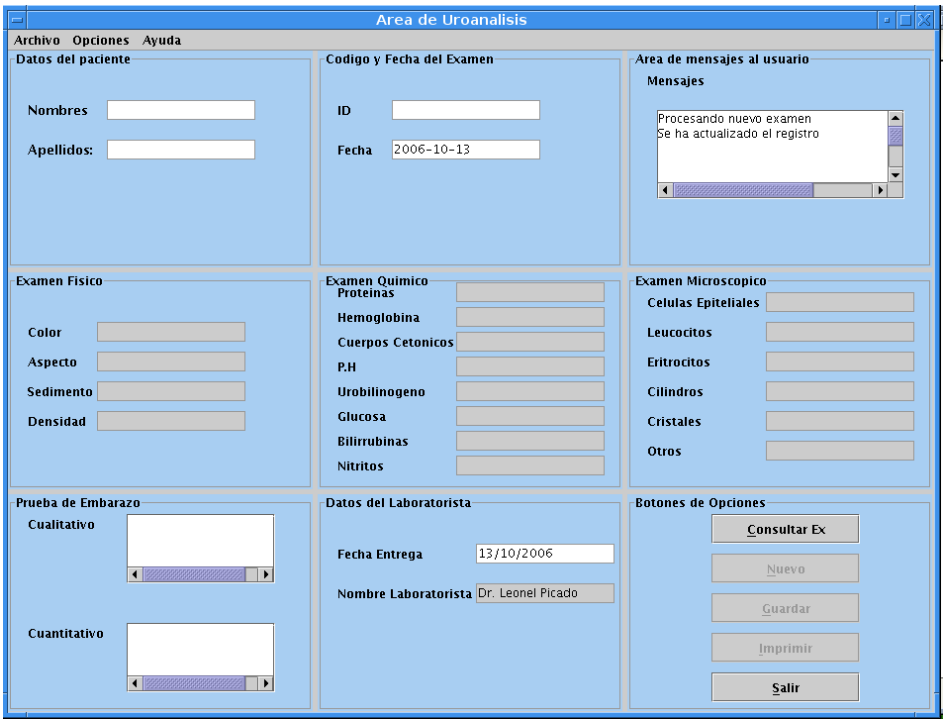
Procesando resultados



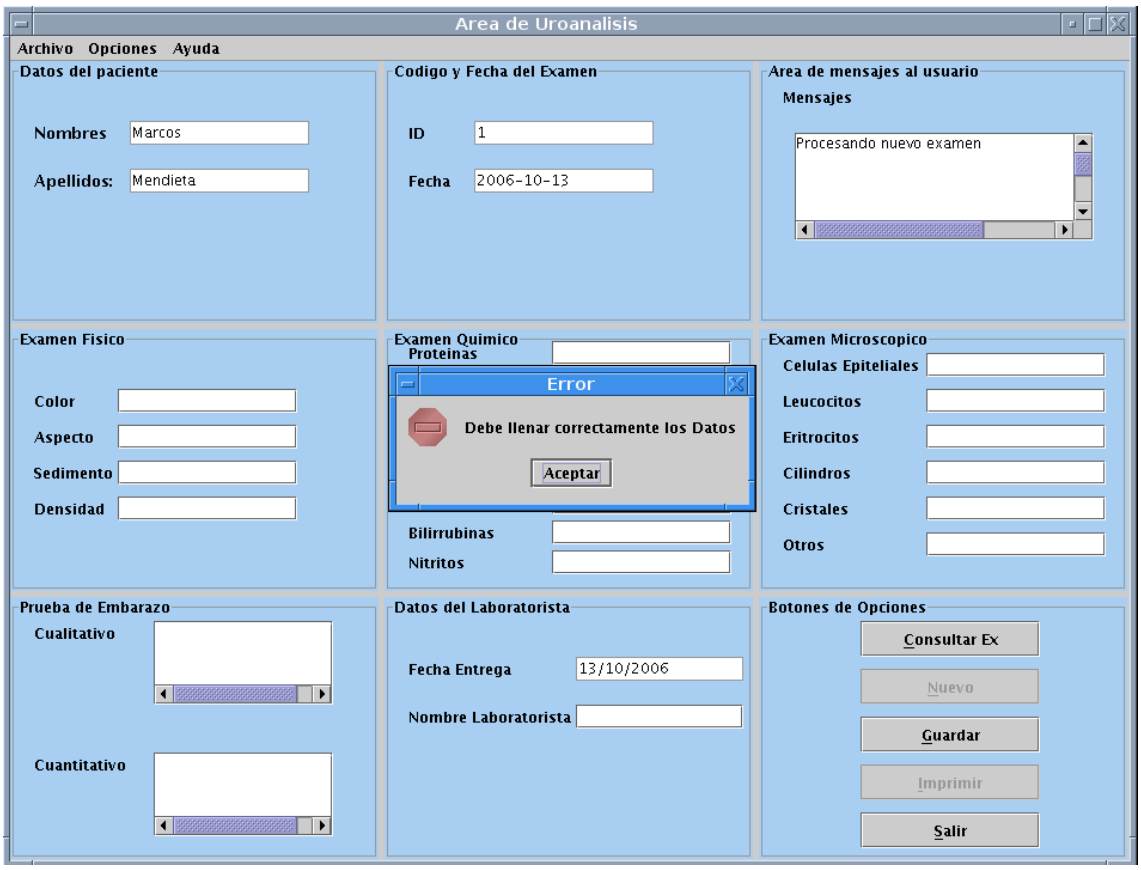
Imprimir resultado



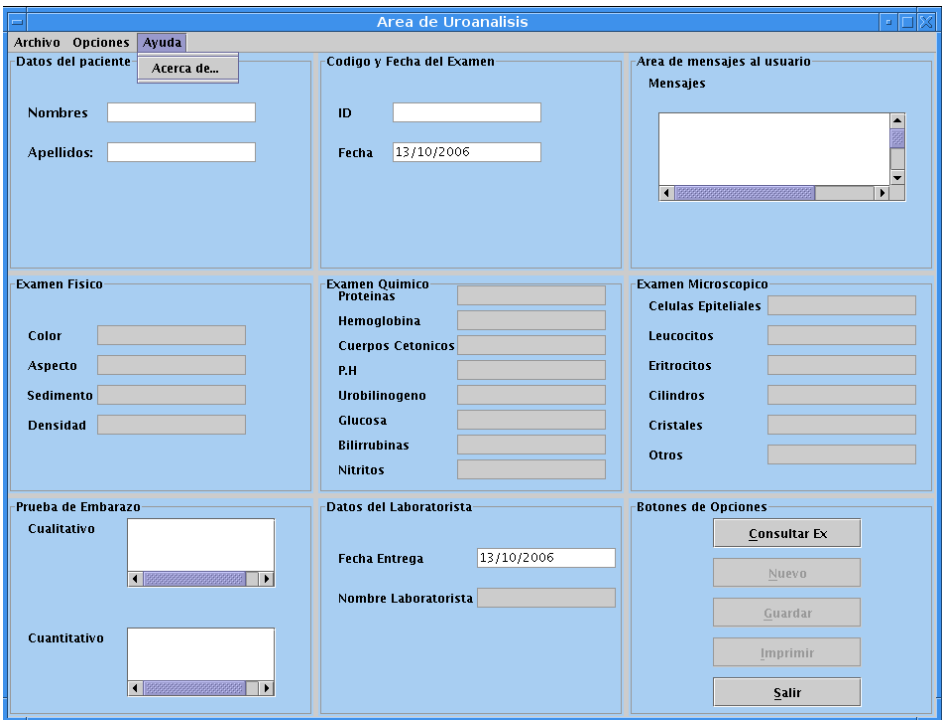
Aplicación después de la Impresión



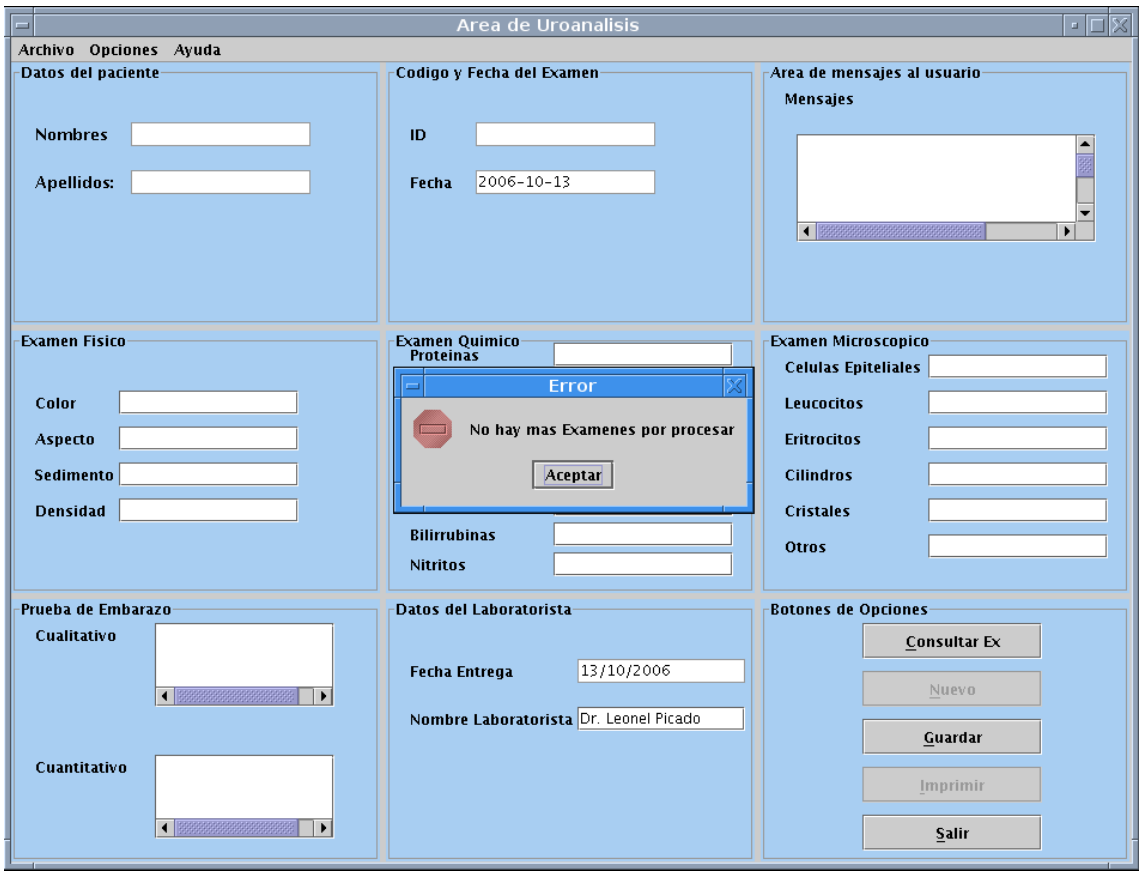
Error al llenar datos



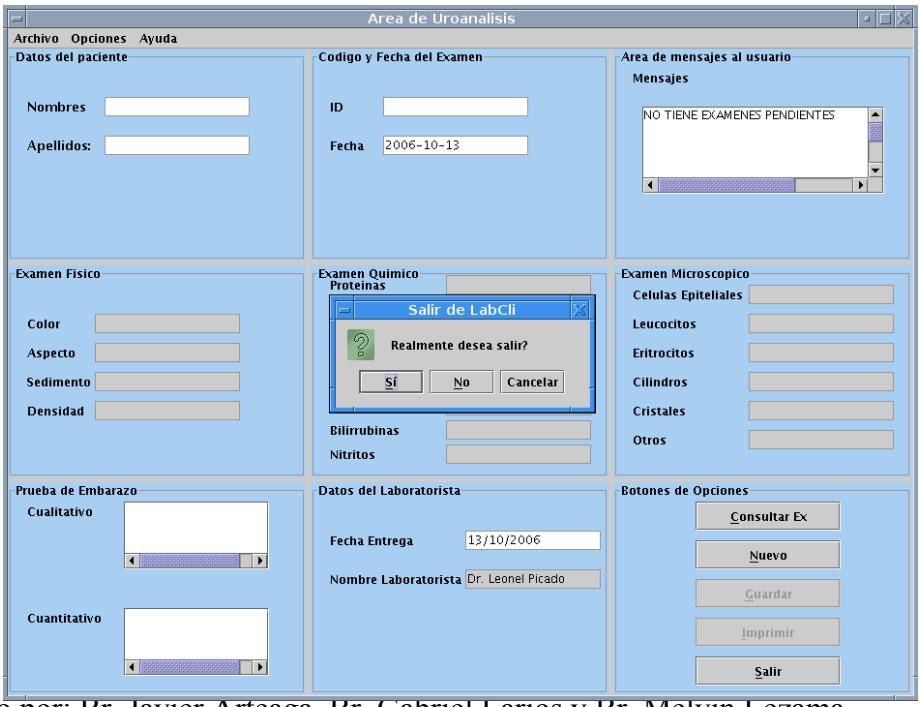
Menú Ayuda



Validación de datos



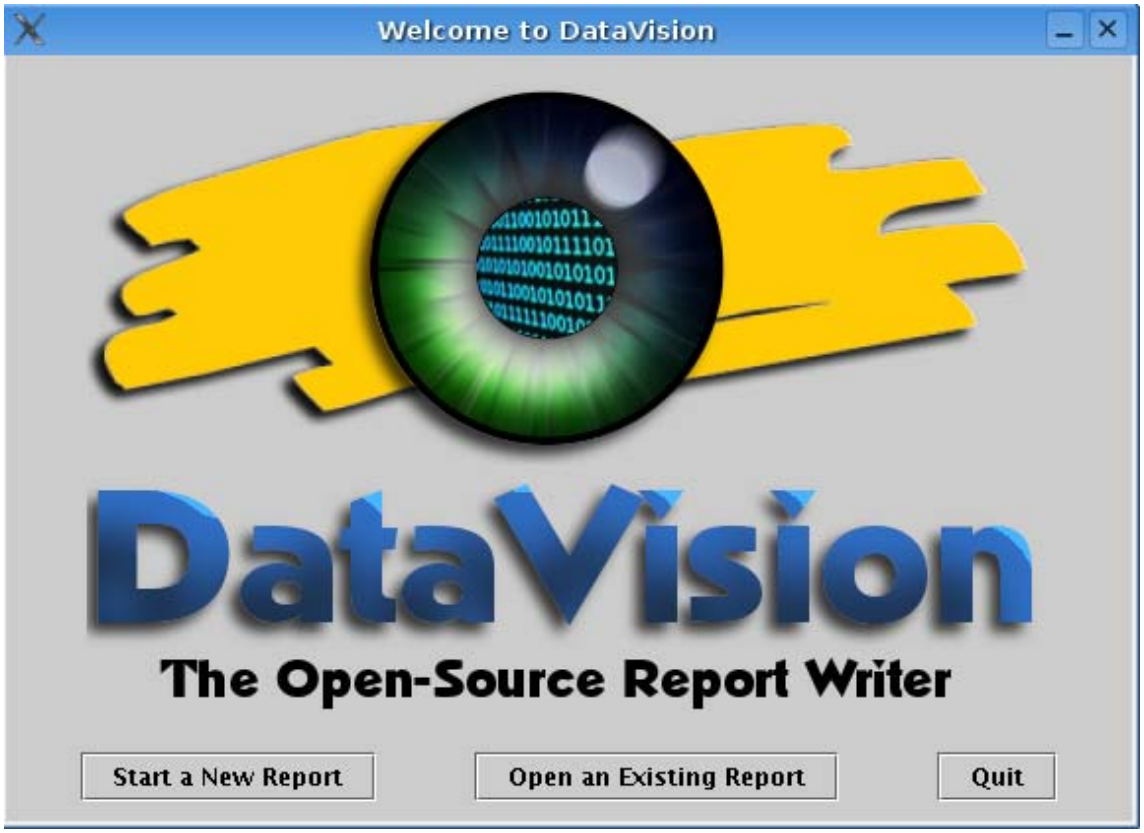
Salir de la Aplicación



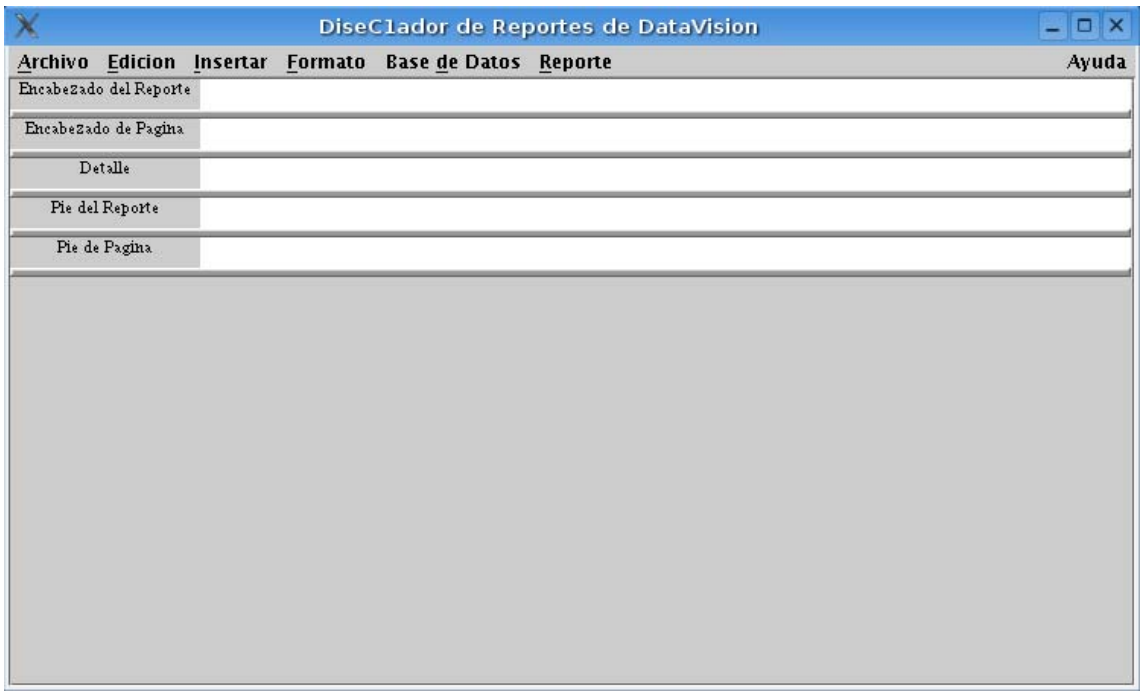
Pantallas de DataVision

Autentificación de la Aplicación

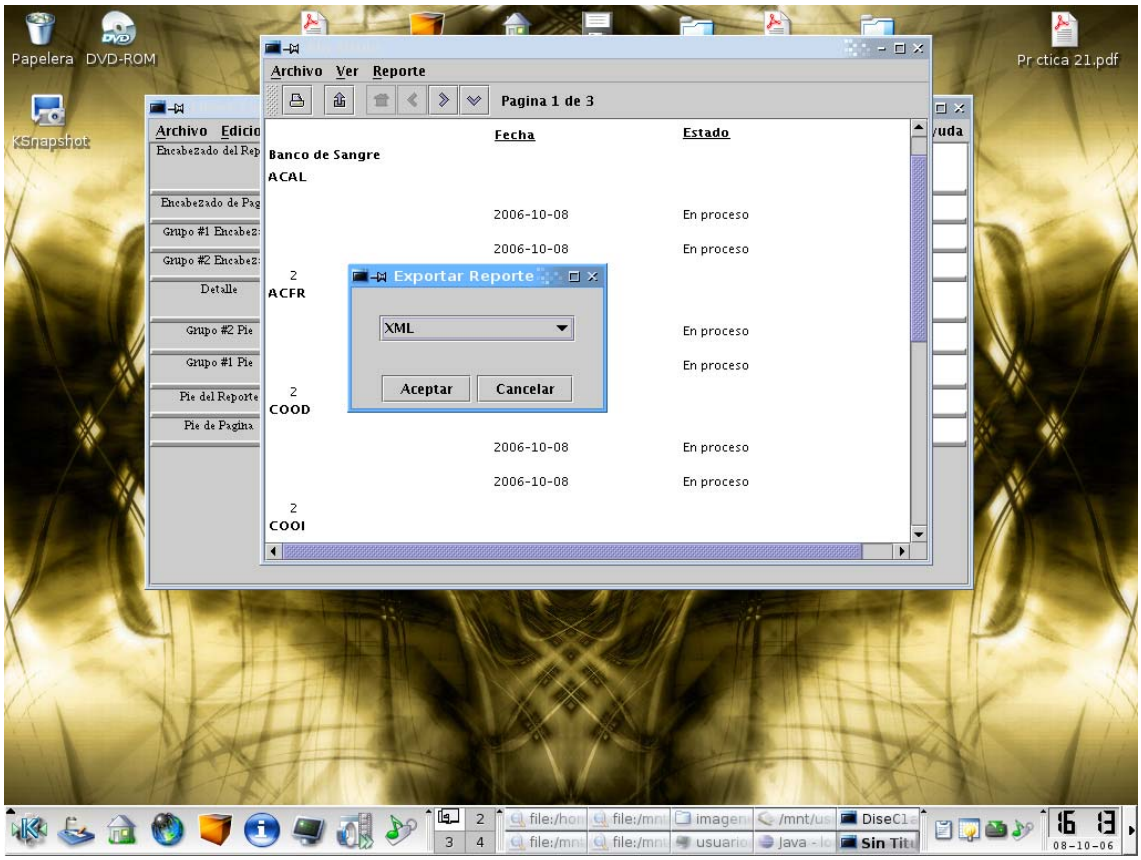
Pantalla Principal de la Aplicación



Diseño del Reporte



Generación de Reporte



Codigo de la Aplicación principal

```
import javax.swing.*;
import java.sql.*;
import java.awt.event.*;
import java.awt.*;
import javax.swing.JPanel;
import javax.swing.border.*;
import java.util.*;
import java.lang.String;
import java.sql.*;
import java.text.SimpleDateFormat;
import java.util.Date;

class principal

{
      public static void main(String []args)
      {


            try
                        {
                              Class.forName("org.postgresql.Driver");
                        }

                        catch(ClassNotFoundException ex)
                        {
                              System.out.println(ex);
                              System.exit(-1);
                        }


            Interfaz t=new Interfaz();
            t.setVisible(true);



      }
} // fin de la clase principal


class Interfaz extends JFrame implements ActionListener{

      JLabel lbl_login=new JLabel("Login");
      JLabel lbl_password=new JLabel("Password");

      JTextField txt_login=new JTextField(30);
      JPasswordField palabraclave=new JPasswordField(20);

      JButton btn_ingresar=new JButton("Ingresar");

      Container contenedor;
      Color fondopanel = new Color(20, 204, 204);


public Interfaz()
      {
                  //inicio del constructor
```

```
            setTitle("Ingresar a LabCli 1.0");
            setSize(300,200);

            contenedor=this.getContentPane();
            addWindowListener(new Manejador());
            setResizable(false);
            palabraclave.addActionListener(alp);
            btn_ingresar.addActionListener(this);

            JPanel panel=new JPanel();

            panel.setLayout(null);
            lbl_login.setBounds(20,50,70,20);
            panel.setToolTipText("Seguridad LabCli");
            panel.add(lbl_login);
            lbl_password.setBounds(20,90,70,20);
            panel.add(lbl_password);

            txt_login.setBounds(100,50,150,20);
            panel.add(txt_login);
            txt_login.setToolTipText("Introduce el nombre de usuario");
            palabraclave.setBounds(100,90,150,20);
            panel.add(palabraclave);
            palabraclave.setToolTipText("Introduce tu contrasena");


            btn_ingresar.setBounds(130,130,100,20);
            panel.add(btn_ingresar);
            btn_ingresar.setToolTipText("Click para ingresar a
LabCli");
            panel.setBackground(fondopanel);

      panel.setBorder(javax.swing.BorderFactory.createTitledBorder("In
gresar a LabCli"));

            contenedor.add(panel);
      }//fin del constructor de la aplicacion


public void actionPerformed(ActionEvent e)
      {     //inicio del actionPerformed de los botones

      if(e.getSource()==btn_ingresar)
            {
               comprobarpassword();
            }


       }


  ActionListener alp = new ActionListener(){

      public void actionPerformed(ActionEvent evt){

      if(evt.getSource()==palabraclave){
            comprobarpassword();
```

```
            }
      }

      };

      public void comprobarpassword()
      {

      char[]  passwordcorrecto={'g','a','b','r','i','e','l'};
                  String nombre_usuario = "usuario";
                  String obtiene_nombre = new String();

               String aux = new String (passwordcorrecto);
               char [] passwordelegido=palabraclave.getPassword();
               String var = new String(passwordelegido);

                  obtiene_nombre = txt_login.getText();

                  if(aux.equals(var) &&
obtiene_nombre.equals(nombre_usuario) )  {

                        frame miframe = new frame();
                        miframe.setVisible(true);
                         setVisible(false);


                        System.out.println("Permiso concedido al
sistema ");


                              }
                  else {
                        dialogo_error();
                        System.out.println("Permiso Denegado al
sistema");
                        }
}

        public void dialogo_error()
            {
JOptionPane.showMessageDialog( this,"Introduce bien tus datos
","Acceso denegado",JOptionPane.ERROR_MESSAGE);
}




      class Manejador extends WindowAdapter{
      public void windowClosing(WindowEvent e){
      System.out.println("saliendo...");
            System.exit(0);
                  }

            }

      }


class frame extends JFrame
```

```
{
      JButton btn_nuevo, btn_guardar, btn_limpiar, btn_salir;

      JLabel nombres, apellidos, inss, exp, id, fecha, estado,
procedencia,cod_area;

      JTextField txt_nombres, txt_apellidos, txt_inss, txt_exp,
txt_id;
      JTextField txt_estado;

      JFormattedTextField txt_fecha = new JFormattedTextField( new
java.util.Date());

      JComboBox cmb_areas,cmb_procedencias;

      JMenuBar barramenu;
      JMenu menu_archivo,menu_ayuda;
      JMenuItem menu_nuevo, menu_salir,
menu_guardar,menu_borrardatos,acerca_de;

      JCheckBox ch_ACAL = new JCheckBox("ACAL");
      JCheckBox ch_ACFR = new JCheckBox("ACFR");
      JCheckBox ch_COOD = new JCheckBox("COOD");
      JCheckBox ch_COOI = new JCheckBox("COOI");
      JCheckBox ch_FDu = new JCheckBox("FDu");
      JCheckBox ch_HBS = new JCheckBox("HBS");
      JCheckBox ch_PCRUZ = new JCheckBox("PCRUZ");
      JCheckBox ch_VIH = new JCheckBox("VIH");
      JCheckBox ch_TRh = new JCheckBox("TRh");



      JCheckBox ch_ESP = new JCheckBox("ESP");
      JCheckBox ch_EGO = new JCheckBox("EGO");
      JCheckBox ch_PBJ = new JCheckBox("PBJ");
      JCheckBox ch_GCH = new JCheckBox("GCH");
      JCheckBox ch_R_ADD = new JCheckBox("R-ADD");
      JCheckBox ch_TIT_GCH = new JCheckBox("TIT-GCH");

      JPanel panel1 = new JPanel();
      JPanel panel2 = new JPanel();
      JPanel panel;


      java.util.Date fechasol=new java.util.Date();
      SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");//Le
especifico el formato en que quiero que aparezca la facha

      Color fondopanel = new Color(20,200,200);
      Color fondospanel = new Color(51,153,255);
      Container contenedor = getContentPane();

      public frame()
      {
            setSize(800,500);
            setTitle("INGRESOS");
            addWindowListener(new manejador());

            panel = new JPanel();
            panel.setLayout(null);
```

```
setResizable(false);


nombres = new JLabel("Nombres");
nombres.setBounds(20,30,90,20);
panel.add(nombres);

txt_nombres = new JTextField(20);
txt_nombres.setBounds(80,30,150,20);
panel.add(txt_nombres);

apellidos = new JLabel("Apellidos");
apellidos.setBounds(20,60,150,20);
panel.add(apellidos);

txt_apellidos = new JTextField(20);
txt_apellidos.setBounds(80,60,150,20);
panel.add(txt_apellidos);

inss = new JLabel("INSS");
inss.setBounds(20,90,150,20);
panel.add(inss);

txt_inss = new JTextField(20);
txt_inss.setBounds(50,90,150,20);
panel.add(txt_inss);

exp = new JLabel("Expediente");
exp.setBounds(20,120,150,20);
panel.add(exp);

txt_exp = new JTextField(20);
txt_exp.setBounds(90,120,150,20);
panel.add(txt_exp);

id = new JLabel ("ID");
id.setBounds(400,30,150,20);
panel.add(id);

txt_id = new JTextField(20);
txt_id.setToolTipText("Indica la secuencia del examen");
txt_id.setEditable(false);
txt_id.setBounds(430,30,150,20);
panel.add(txt_id);

fecha = new JLabel("Fecha");
fecha.setBounds(400,60,150,20);
panel.add(fecha);


txt_fecha.setBounds(440,60,150,20);
txt_fecha.setText(sdf.format(fechasol));
panel.add(txt_fecha);

estado = new JLabel("Estado");
estado.setBounds(390,90,150,20);
panel.add(estado);

txt_estado = new JTextField(20);
txt_estado.setBounds(440,90,150,20);
```

```
            txt_estado.setText("En proceso");
            panel.add(txt_estado);

            procedencia = new JLabel("Procedencia");
            procedencia.setBounds(20,180,150,20);
            panel.add(procedencia);


      ////////////////////////////////////////////////////////////
////
            ////CREACION DE LISTAS DESPLEGABLES
            String [] procedencias = {"Seleccionar","La
Fraternidad","AMOCSA","BEMENIC","SUMEDICO"};
            cmb_procedencias = new JComboBox(procedencias);
            cmb_procedencias.setToolTipText("Lugar de donde Procede el
Paciente");
            cmb_procedencias.setSelectedIndex(0);
            cmb_procedencias.setBounds(100,180,150,20);
            panel.add(cmb_procedencias);

            cod_area = new JLabel("Codigo de Area");
            cod_area.setBounds(20,210,150,20);
            panel.add(cod_area);

            manejador_areas evento_areas = new manejador_areas();
            String [] areas = {"Seleccionar","Uroanalisis","Banco de
Sangre","Hematologia","Bactereologia"};
            cmb_areas = new JComboBox(areas);
            cmb_areas.setToolTipText("Areas para realizar examenes");
            cmb_areas.setSelectedIndex(0);
            cmb_areas.addActionListener(evento_areas);
            cmb_areas.setBounds(120,210,150,20);
            panel.add(cmb_areas);

      ////////////////////////////////////////////////////////////
////
            ///////////FIN DE LA CREACION DE LISTAS DESPLEGABLES/////
            manejador_nuevo nuevo = new manejador_nuevo();
            btn_nuevo = new JButton("Nuevo");
            btn_nuevo.setToolTipText("Añade un registro nuevo");
            btn_nuevo.setBounds(300,180,100,20);
            btn_nuevo.setMnemonic('N');
            btn_nuevo.addActionListener(nuevo);
            panel.add(btn_nuevo);

            manejador_guardar guardar = new manejador_guardar();
            btn_guardar = new JButton("Guardar");
            btn_guardar.setToolTipText("Almacena un registro");
            btn_guardar.setBounds(300,210,100,20);
            btn_guardar.setMnemonic('G');
            btn_guardar.addActionListener(guardar);
            panel.add(btn_guardar);

            manejador_limpiar limpiar = new manejador_limpiar();
            btn_limpiar = new JButton("Limpiar");
            btn_limpiar.setToolTipText("Limpiar cajas");
            btn_limpiar.setBounds(420,180,100,20);
            btn_limpiar.setMnemonic('L');
            btn_limpiar.addActionListener(limpiar);
            panel.add(btn_limpiar);
```

```
            manejador_salir salir = new manejador_salir();
            btn_salir = new JButton("Salir");
            btn_salir.setToolTipText("Terminar la Aplicacion");
            btn_salir.setBounds(420,210,100,20);
            btn_salir.setMnemonic('S');
            btn_salir.addActionListener(salir);
            panel.add(btn_salir);

            panel.setBackground(fondopanel);

            ////////////////////////añadimos los check
            panel1.setLayout(null);
            panel1.setBounds(540, 180, 200, 200);
            //manejador_check check = new manejador_check();

            //ch_ACAL.addItemListener(check);
            ch_ACAL.setBounds(20, 20, 80, 15);
            panel1.add(ch_ACAL);

            //ch_ACFR.addItemListener(check);
            ch_ACFR.setBounds(20, 40, 80, 15);
            panel1.add(ch_ACFR);

            //ch_COOD.addItemListener(check);
            ch_COOD.setBounds(20, 60, 80, 15);
            panel1.add(ch_COOD);

            //ch_COOI.addItemListener(check);
            ch_COOI.setBounds(20, 80, 80, 15);
            panel1.add(ch_COOI);

            //ch_FDu.addItemListener(check);
            ch_FDu.setBounds(20, 100, 80, 15);
            panel1.add(ch_FDu);

            //ch_HBS.addItemListener(check);
            ch_HBS.setBounds(20, 120, 80, 15);
            panel1.add(ch_HBS);

            //ch_PCRUZ.addItemListener(check);
            ch_PCRUZ.setBounds(20, 140, 80, 15);
            panel1.add(ch_PCRUZ);

            //ch_VIH.addItemListener(check);
            ch_VIH.setBounds(20, 160, 80, 15);
            panel1.add(ch_VIH);

            //ch_TRh.addItemListener(check);
            ch_TRh.setBounds(20, 180, 80, 15);
            panel1.add(ch_TRh);

            panel1.setBorder(new TitledBorder("Examenes BANCO
SANGRE"));
            panel1.setToolTipText("Examenes a realiar en Banco de
Sangre");
            //panel1.setBackground(fondospanel);
            panel.add(panel1);
```

```
            panel2.setLayout(null);
            panel2.setBounds(540, 180, 200, 200);

            //ch_ESP.addItemListener(check);
            ch_ESP.setBounds(20, 20, 80, 15);
            panel2.add(ch_ESP);

            //ch_EGO.addItemListener(check);
            ch_EGO.setBounds(20, 40, 80, 15);
            panel2.add(ch_EGO);

            //ch_PBJ.addItemListener(check);
            ch_PBJ.setBounds(20, 60, 80, 15);
            panel2.add(ch_PBJ);

            //ch_GCH.addItemListener(check);
            ch_GCH.setBounds(20, 80, 80, 15);
            panel2.add(ch_GCH);

            //ch_R_ADD.addItemListener(check);
            ch_R_ADD.setBounds(20, 100, 80, 15);
            panel2.add(ch_R_ADD);

            //ch_TIT_GCH.addItemListener(check);
            ch_TIT_GCH.setBounds(20, 120, 80, 15);
            panel2.add(ch_TIT_GCH);

            panel2.setBorder(new TitledBorder("Examenes UROANALISIS"));
            panel2.setToolTipText("Examenes a realizar en
Uroanalisis");
            //panel2.setBackground(fondospanel);

            panel.add(panel2);



            ////////////////////////////creacion de los
menus//////////////////////

             barramenu = new JMenuBar();
             menu_archivo = new JMenu("Archivo");
             menu_archivo.setMnemonic('A');

             menu_ayuda = new JMenu("Ayuda");

             menu_nuevo = new JMenuItem("Nuevo");
             menu_nuevo.setMnemonic('N');
             menu_nuevo.setToolTipText("Nuevo Registro");


             menu_salir = new JMenuItem("Salir");

             acerca_de = new JMenuItem("Acerca de..");


             menu_guardar = new JMenuItem("Guardar");

             menu_borrardatos = new JMenuItem("Borrar Datos");
```

```
            menu_archivo.add(menu_nuevo);
            menu_nuevo.addActionListener(nuevo);


            evento_menu_borrardatos evento_borrar = new
evento_menu_borrardatos();
            menu_borrardatos.addActionListener(evento_borrar);
            menu_borrardatos.setEnabled(true);
            menu_archivo.add(menu_borrardatos);

            menu_guardar.addActionListener(guardar);
            menu_guardar.setToolTipText("Almacenar registro");
            menu_guardar.setMnemonic('G');
            menu_guardar.setEnabled(false);
            menu_archivo.add(menu_guardar);

            menu_archivo.addSeparator();
            menu_archivo.add(menu_salir);
            menu_salir.setToolTipText("Terminar la Aplicacion");
            menu_salir.setMnemonic('S');
            menu_salir.addActionListener(salir);

            menu_ayuda.add(acerca_de);
            evento_menu_acerca evento_acerca = new
evento_menu_acerca();
            acerca_de.addActionListener(evento_acerca);
            barramenu.add(menu_archivo);

            barramenu.add(menu_ayuda);
            setJMenuBar(barramenu);

            invalidar();
            ocultarpanel1();
            ocultarpanel2();

            contenedor.add(panel);
      }/////////fin del constructor



                  //////////agregado interfaz para controlar eventos
del menu
                  class evento_menu_acerca implements ActionListener
                  {
                        public void actionPerformed(ActionEvent origen)
                        {


      JOptionPane.showMessageDialog(panel,"Version 0.0.0.1 LAB CLI
HEODRA \n Desarrollado por \nJAVIER \nMELVIN \nGABRIEL","Acerca
de",JOptionPane.OK_OPTION);

                              }
                        }



      class manejador extends WindowAdapter
      {
      public void windowClosing(WindowEvent e)
```

```
            {
                  System.out.println("Saliendo");
                  System.exit(0);
            }
      }/////fin windowAdapter

      class manejador_areas implements ActionListener
      {
            public void actionPerformed(ActionEvent evt)
            {
                  String compara = new String();
                  compara = (String)cmb_areas.getSelectedItem();

                  if(compara.equals("Banco de Sangre"))
                  {
                        ocultarpanel2();
                        mostrarpanel1();

                  }

                  if(compara.equals("Uroanalisis"))
                  {
                        ocultarpanel1();
                        mostrarpanel2();

                  }

            }
      }////////////fin de manejador_areas


      class evento_menu_borrardatos implements ActionListener
      {
            public void actionPerformed(ActionEvent evt)
            {

                  int opcion =
JOptionPane.showConfirmDialog(panel,"¿Seguro que deseas borrar los
registros?","Edicion",JOptionPane.YES_NO_OPTION);
                  if(opcion == JOptionPane.YES_OPTION)
                  {

                        //validar();
                        seguridad secret = new seguridad();
                        secret.setVisible(true);
                        //borrardatosBD();
                        menu_borrardatos.setEnabled(false);
                  }
            }
      }

      class manejador_salir implements ActionListener
      {
            public void actionPerformed(ActionEvent evt)
            {
                  int respuesta =
JOptionPane.showConfirmDialog(panel,"¿Deseas salir del
programa?","Saliendo",JOptionPane.OK_CANCEL_OPTION);
                  if(respuesta == JOptionPane.OK_OPTION)
                        System.exit(0);
```

```
		}
	}

	class manejador_limpiar implements ActionListener
	{
		public void actionPerformed(ActionEvent evt)
		{
			Limpiar();
		}
	}

	class manejador_nuevo implements ActionListener
	{
		public void actionPerformed(ActionEvent evt)
		{

			Connection conexion = null;

			Statement sentencia = null;
			Statement sentencia2 = null;

			ResultSet resultado = null;
			ResultSet resultado2 = null;

			String consulta = "select max(id), id  from solicitud
group by id";
			String consulta2 = "select nombres from solicitud";

			try
			{
				conexion =
DriverManager.getConnection("jdbc:postgresql://localhost/monografia","
usuario","usuario123");

				sentencia = conexion.createStatement();
				sentencia2 = conexion.createStatement();

				resultado = sentencia.executeQuery(consulta);
				resultado2 =
sentencia2.executeQuery(consulta2);

				int aumento = 0;

				if(!resultado2.next() )
				{
					txt_id.setText("1");

					resultado2.close();
					sentencia2.close();
					conexion.close();
				}

				else
				while(resultado.next())
				{
					aumento = resultado.getInt("id");
					aumento++;

					txt_id.setText(String.valueOf(aumento));
```

```
                        resultado.close();
                    }

                }//////////fin del bloque try

                catch(SQLException ex)
                {
                    //JOptionPane.showMessageDialog(panel,"Error al
acceder a la base de datos.","Error en
datos",JOptionPane.INFORMATION_MESSAGE);
                    System.out.println(ex);
                }

                finally
                {
                    try
                    {
                        if(resultado != null)
                            resultado.close();
                    }

                    catch(SQLException ex)
                    {

    //JOptionPane.showMessageDialog(panel,"Error al acceder a la
base de datos.","Error en datos",JOptionPane.INFORMATION_MESSAGE);
                    }

                    try
                    {
                        if(sentencia != null)
                            sentencia.close();
                    }

                    catch(SQLException ex)
                    {

    //JOptionPane.showMessageDialog(panel,"Error al acceder a la
base de datos.","Error en datos",JOptionPane.INFORMATION_MESSAGE);
                    }

                    try
                    {
                        if(conexion != null)
                            conexion.close();
                    }

                    catch(SQLException ex)
                    {

    //JOptionPane.showMessageDialog(panel,"Error al acceder a la
base de datos.","Error en datos",JOptionPane.INFORMATION_MESSAGE);
                    }
                }//////////////fin de finally

                btn_nuevo.setEnabled(false);
                menu_borrardatos.setEnabled(false);
                menu_nuevo.setEnabled(false);
                menu_guardar.setEnabled(true);
```

```
                validar();
                Limpiar();

                ocultarpanel1();
                ocultarpanel2();
            }/////////fin de action performed
      }////////////fin de manejador_nuevo


      class manejador_guardar implements ActionListener
      {
            public void actionPerformed(ActionEvent evt)
            {
                  Connection conexion = null;
                  PreparedStatement sentencia = null;
                  PreparedStatement sentencia2 = null;

                  String procedencias =
String.valueOf(cmb_procedencias.getSelectedItem());
                  String areas =
String.valueOf(cmb_areas.getSelectedItem());
                  String nombres = txt_nombres.getText();
                  String apellidos = txt_apellidos.getText();


                  if(nombres.equals("") && apellidos.equals(""))
                        JOptionPane.showMessageDialog(panel,"No hay
datos a almacenar","Datos invalidos",JOptionPane.OK_OPTION);


                  else if(cmb_areas.getSelectedIndex() == 0)
                        JOptionPane.showMessageDialog(panel,"No ha
seleccionado el Area","Error",JOptionPane.OK_OPTION);


                  else
                  try
                  {

                        java.sql.Date mi_fecha;
                        Calendar tiempo_actual =
Calendar.getInstance();
                        java.util.Date fecha_actual =
tiempo_actual.getTime();
                        mi_fecha = new
java.sql.Date(fecha_actual.getTime());

                        String mi_estado = "en proceso";
                        conexion =
DriverManager.getConnection("jdbc:postgresql://localhost/monografia","
usuario","usuario123");



                        conexion.setAutoCommit(false);
                        System.out.println("Iniciamos la transaccion");
```

```
                        sentencia = conexion.prepareStatement("INSERT
INTO solicitud VALUES (?,?,?,?,?,?,?,?,?)");

      sentencia.setInt(1,Integer.parseInt(txt_id.getText()));
                        sentencia.setString(2,txt_nombres.getText());
                        sentencia.setString(3,txt_apellidos.getText());
                        sentencia.setString(4,txt_inss.getText());
                        sentencia.setString(5,txt_exp.getText());
                        sentencia.setString(6,procedencias);
                        sentencia.setString(7,mi_estado);
                        sentencia.setString(8,areas);
                        sentencia.setDate(9,mi_fecha);
                        sentencia.executeUpdate();

                        //////
                        if(ch_ACAL.isSelected())
                        {
                              sentencia2 =
conexion.prepareStatement("INSERT INTO examen VALUES (?,?)");

      sentencia2.setInt(1,Integer.parseInt(txt_id.getText()));

      sentencia2.setString(2,ch_ACAL.getText());
                              sentencia2.executeUpdate();
                        }

                         if(ch_ACFR.isSelected())
                        {
                              sentencia2 =
conexion.prepareStatement("INSERT INTO examen VALUES (?,?)");

      sentencia2.setInt(1,Integer.parseInt(txt_id.getText()));

      sentencia2.setString(2,ch_ACFR.getText());
                              sentencia2.executeUpdate();
                        }

                         if(ch_COOD.isSelected())
                        {
                              sentencia2 =
conexion.prepareStatement("INSERT INTO examen VALUES (?,?)");

      sentencia2.setInt(1,Integer.parseInt(txt_id.getText()));

      sentencia2.setString(2,ch_COOD.getText());
                              sentencia2.executeUpdate();
                        }

                         if(ch_COOI.isSelected())
                        {
                              sentencia2 =
conexion.prepareStatement("INSERT INTO examen VALUES (?,?)");

      sentencia2.setInt(1,Integer.parseInt(txt_id.getText()));

      sentencia2.setString(2,ch_COOI.getText());
                              sentencia2.executeUpdate();
                        }
```

```
                        if(ch_FDu.isSelected())
                       {
                            sentencia2 =
conexion.prepareStatement("INSERT INTO examen VALUES (?,?)");

      sentencia2.setInt(1,Integer.parseInt(txt_id.getText()));
                            sentencia2.setString(2,ch_FDu.getText());
                            sentencia2.executeUpdate();
                       }

                        if(ch_HBS.isSelected())
                       {
                            sentencia2 =
conexion.prepareStatement("INSERT INTO examen VALUES (?,?)");

      sentencia2.setInt(1,Integer.parseInt(txt_id.getText()));
                            sentencia2.setString(2,ch_HBS.getText());
                            sentencia2.executeUpdate();
                       }

                        if(ch_PCRUZ.isSelected())
                       {
                            sentencia2 =
conexion.prepareStatement("INSERT INTO examen VALUES (?,?)");

      sentencia2.setInt(1,Integer.parseInt(txt_id.getText()));

      sentencia2.setString(2,ch_PCRUZ.getText());
                            sentencia2.executeUpdate();
                       }

                        if(ch_VIH.isSelected())
                       {
                            sentencia2 =
conexion.prepareStatement("INSERT INTO examen VALUES (?,?)");

      sentencia2.setInt(1,Integer.parseInt(txt_id.getText()));
                            sentencia2.setString(2,ch_VIH.getText());
                            sentencia2.executeUpdate();
                       }

                        if(ch_TRh.isSelected())
                       {
                            sentencia2 =
conexion.prepareStatement("INSERT INTO examen VALUES (?,?)");

      sentencia2.setInt(1,Integer.parseInt(txt_id.getText()));
                            sentencia2.setString(2,ch_TRh.getText());
                            sentencia2.executeUpdate();
                       }

                        if(ch_ESP.isSelected())
                       {
                            sentencia2 =
conexion.prepareStatement("INSERT INTO examen VALUES (?,?)");

      sentencia2.setInt(1,Integer.parseInt(txt_id.getText()));
                            sentencia2.setString(2,ch_ESP.getText());
                            sentencia2.executeUpdate();
                       }
```

```
                        if(ch_EGO.isSelected())
                        {
                              sentencia2 =
conexion.prepareStatement("INSERT INTO examen VALUES (?,?)");

      sentencia2.setInt(1,Integer.parseInt(txt_id.getText()));
                              sentencia2.setString(2,ch_EGO.getText());
                              sentencia2.executeUpdate();
                        }

                        if(ch_PBJ.isSelected())
                        {
                              sentencia2 =
conexion.prepareStatement("INSERT INTO examen VALUES (?,?)");

      sentencia2.setInt(1,Integer.parseInt(txt_id.getText()));
                              sentencia2.setString(2,ch_PBJ.getText());
                              sentencia2.executeUpdate();
                        }

                        if(ch_GCH.isSelected())
                        {
                              sentencia2 =
conexion.prepareStatement("INSERT INTO examen VALUES (?,?)");

      sentencia2.setInt(1,Integer.parseInt(txt_id.getText()));
                              sentencia2.setString(2,ch_GCH.getText());
                              sentencia2.executeUpdate();
                        }

                        if(ch_R_ADD.isSelected())
                        {
                              sentencia2 =
conexion.prepareStatement("INSERT INTO examen VALUES (?,?)");

      sentencia2.setInt(1,Integer.parseInt(txt_id.getText()));

      sentencia2.setString(2,ch_R_ADD.getText());
                              sentencia2.executeUpdate();
                        }

                        if(ch_TIT_GCH.isSelected())
                        {
                              sentencia2 =
conexion.prepareStatement("INSERT INTO examen VALUES (?,?)");

      sentencia2.setInt(1,Integer.parseInt(txt_id.getText()));

      sentencia2.setString(2,ch_TIT_GCH.getText());
                              sentencia2.executeUpdate();
                        }



                        conexion.commit();
                        conexion.setAutoCommit(true);

                        invalidar();
```

```
                        cmb_areas.setSelectedIndex(0);
                        cmb_procedencias.setSelectedIndex(0);
                        btn_nuevo.setEnabled(true);
                        menu_nuevo.setEnabled(true);
                        menu_guardar.setEnabled(false);
                        menu_borrardatos.setEnabled(true);
                        limpiar_checkBox_panelSangre();
                        limpiar_checkBox_panelUroanlisis();
                        ocultarpanel1();
                        ocultarpanel2();
                        datosguardados();

                  }////////////////fin del bloque try



                  catch(SQLException ex)
                  {
                        System.out.println(ex);
                        conexion.rollback();
                        System.out.println("La transaccion ha
fallado");
                        conexion.setAutoCommit(true);
                  }

                  finally
                  {
                        try
                        {
                              if(sentencia != null)
                                    sentencia.close();
                        }

                        catch(SQLException ex)
                        {
                              System.out.println(ex);
                        }

                        try
                        {
                              if(sentencia2 != null)
                                    sentencia2.close();
                        }

                        catch(SQLException ex)
                        {
                              System.out.println(ex);
                        }

                        try
                        {
                              if(conexion != null)
                                    conexion.close();
                        }

                        catch(SQLException ex)
                        {
                              System.out.println(ex);
                        }
```

```
                }////fin del finally

            }
      }/////////////////fin de manejador guardar

//////////////////////////Limpiar cajas
      public void Limpiar()
    {
      txt_nombres.setText("");
      txt_apellidos.setText("");
      txt_inss.setText("");
      txt_exp.setText("");
      txt_nombres.requestFocus();

      cmb_procedencias.setSelectedIndex(0);
      cmb_areas.setSelectedIndex(0);

      ocultarpanel1();
      ocultarpanel2();

      }//////////////fin de limpiar

      public void limpiar_checkBox_panelSangre()

      {
            ch_ACAL.setSelected(false);
            ch_ACFR.setSelected(false);
            ch_COOD.setSelected(false);
            ch_COOI.setSelected(false);
            ch_FDu.setSelected(false);
            ch_HBS.setSelected(false);
            ch_PCRUZ.setSelected(false);
            ch_VIH.setSelected(false);
            ch_TRh.setSelected(false);

      }



      public void limpiar_checkBox_panelUroanlisis()
      {
            ch_ESP.setSelected(false);
            ch_EGO.setSelected(false);
            ch_PBJ.setSelected(false);
            ch_GCH.setSelected(false);
            ch_R_ADD.setSelected(false);
            ch_TIT_GCH.setSelected(false);

      }




      public void borrardatosexamen()
            {


            String query =  "delete from examen";
```

```
            Connection conn = null;



                        try
                          {


                  conn =
DriverManager.getConnection("jdbc:postgresql://localhost/monografia","
usuario", "usuario123");

                        if (conn != null)
                              {


                        Statement sentencia, sentencia1;

                        sentencia = conn.createStatement();


                        ResultSet res = sentencia.executeQuery(query);
                        System.out.println("Datos borrados Examenes");



                        res.close();

                              sentencia.close();



                  }

                        conn.close();



            }
         catch(SQLException ex)
         {
            System.out.println(ex);



         }

            }




      public void borrardatosolicitud()
            {



            String borrarsol =  "delete from solicitud";
            Connection conn = null;
```

```
                        try
                          {


                  conn =
DriverManager.getConnection("jdbc:postgresql://localhost/monografia","
usuario", "usuario123");

                        if (conn != null)
                              {


                        Statement  sentencia1;

                        //sentencia = conn.createStatement();

                        sentencia1 = conn.createStatement();



                        ResultSet res =
sentencia1.executeQuery(borrarsol);

                        System.out.println("Datos borrados Solicitud");




                        res.close();

                              sentencia1.close();



                  }



                        conn.close();

            }
       catch(SQLException ex)
       {
          System.out.println(ex);



       }

            }


//////////////////////////////Invalidar cajas y botones
      public void invalidar()
      {
            txt_nombres.setEditable(false);
```

```
            txt_apellidos.setEditable(false);
            txt_inss.setEditable(false);
            txt_exp.setEditable(false);
            txt_fecha.setEditable(false);
            txt_estado.setEditable(false);
            btn_guardar.setEnabled(false);
            btn_limpiar.setEnabled(false);

            //mostrarpanel1();
      }//////////////fin de invalidar


      public void datosguardados()
      {
      Limpiar();
      JOptionPane.showMessageDialog( this,"Datos Guardados
Correctamente","Guardar",JOptionPane.INFORMATION_MESSAGE);
      }



      public void validar()
      {
            txt_nombres.setEditable(true);
            txt_apellidos.setEditable(true);
            txt_inss.setEditable(true);
            txt_exp.setEditable(true);
            btn_guardar.setEnabled(true);
            btn_limpiar.setEnabled(true);
            txt_nombres.requestFocus();

      }////////////fin de validar


      public void mostrarpanel1()
      {
            panel1.setVisible(true);
            ch_ACAL.setEnabled(true);
            ch_ACFR.setEnabled(true);
            ch_COOD.setEnabled(true);
            ch_COOI.setEnabled(true);
            ch_FDu.setEnabled(true);
            ch_HBS.setEnabled(true);
            ch_PCRUZ.setEnabled(true);
            ch_VIH.setEnabled(true);
            ch_TRh.setEnabled(true);
      }///////////fin de mostrarpanel1

      public void mostrarpanel2()
      {
            panel2.setVisible(true);
            ch_ESP.setEnabled(true);
            ch_EGO.setEnabled(true);
            ch_PBJ.setEnabled(true);
            ch_GCH.setEnabled(true);
            ch_R_ADD.setEnabled(true);
            ch_TIT_GCH.setEnabled(true);


      }////////////////////fin de mostrarpanel2
```

```
	public void ocultarpanel1()
	{
		panel1.setVisible(false);

	}

	public void ocultarpanel2()
	{
		panel2.setVisible(false);

	}



class seguridad extends JFrame implements ActionListener{

	JLabel lbl_login=new JLabel("Login");
	JLabel lbl_password=new JLabel("Password");

	JTextField txt_login=new JTextField(30);
	JPasswordField palabraclave=new JPasswordField(20);

	JButton btn_ingresar=new JButton("Borrar ?");

	Container contenedor;
	Color fondopanel = new Color(255, 101, 205);


public seguridad()
	{
			//inicio del constructor



		setTitle("Borrar Registros");
		setSize(300,200);

		contenedor=this.getContentPane();
		addWindowListener(new Manejador());
		setResizable(false);
		palabraclave.addActionListener(alp);
		btn_ingresar.addActionListener(this);

		JPanel panel=new JPanel();

		panel.setLayout(null);
		lbl_login.setBounds(20,50,70,20);
		panel.setToolTipText("Seguridad LabCli");
		panel.add(lbl_login);
		lbl_password.setBounds(20,90,70,20);
		panel.add(lbl_password);

		txt_login.setBounds(100,50,150,20);
		panel.add(txt_login);
		txt_login.setToolTipText("Introduce el nombre de usuario");
		palabraclave.setBounds(100,90,150,20);
		panel.add(palabraclave);
```

```
            palabraclave.setToolTipText("Introduce tu contrasena");


            btn_ingresar.setBounds(130,130,100,20);
            panel.add(btn_ingresar);
            btn_ingresar.setToolTipText("Click para ingresar a
LabCli");
            panel.setBackground(fondopanel);

      panel.setBorder(javax.swing.BorderFactory.createTitledBorder("Bo
rrar datos de las solicitud"));

            contenedor.add(panel);
      }//fin del constructor de la aplicacion


public void actionPerformed(ActionEvent e)
      {     //inicio del actionPerformed de los botones

      if(e.getSource()==btn_ingresar)
            {
               comprobarpassword();
            }


       }


  ActionListener alp = new ActionListener(){

      public void actionPerformed(ActionEvent evt){

      if(evt.getSource()==palabraclave){
            comprobarpassword();

            }
      }

      };

      public void comprobarpassword()
      {

      char[]  passwordcorrecto={'g','a','b','r','i','e','l'};
                  String nombre_usuario = "usuario";
                  String obtiene_nombre = new String();

               String aux = new String (passwordcorrecto);
               char [] passwordelegido=palabraclave.getPassword();
               String var = new String(passwordelegido);

                  obtiene_nombre = txt_login.getText();

                  if(aux.equals(var) &&
obtiene_nombre.equals(nombre_usuario) )  {


                        setVisible(false);
                        borrardatosexamen();
                        borrardatosolicitud();
```

```
                        dialogoDatosBorrados();
                        //System.out.println("Permiso concedido al
sistema ");


                                }
                else {
                        dialogo_error();
                        System.out.println("Permiso Denegado al
sistema");
                        }
}

        public void dialogo_error()
            {
JOptionPane.showMessageDialog( this,"Introduce bien tus datos
","Acceso denegado",JOptionPane.ERROR_MESSAGE);
}

      public void dialogoDatosBorrados()
      {
      JOptionPane.showMessageDialog( this,"Registros Borrados de las
tablas","Borrado",JOptionPane.INFORMATION_MESSAGE);
      }



      class Manejador extends WindowAdapter{
      public void windowClosing(WindowEvent e){
      System.out.println("saliendo...");
            System.exit(0);
                  }

            }

      }


}/////////final de la clase frame
/////////para la construccion de la ventana
```

Codigo de la Aplicación Banco de Sangre

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.lang.*;
import java.lang.String.*;
import java.util.*;
import java.io.*;
import java.net.*;
import javax.swing.JScrollPane;
import java.lang.*;
import java.lang.Character;
import java.lang.String;
import java.sql.*;
import java.io.PrintStream;
import javax.swing.border.TitledBorder;
import java.text.SimpleDateFormat;
import java.util.Date;


class sangre

{
		public static void main(String []args)
		{


		  try
				{
					Class.forName("org.postgresql.Driver");
				}

				catch(ClassNotFoundException ex)
				{
					System.out.println(ex);
					System.exit(-1);
				}


		  Interfaz t=new Interfaz();
		  t.setVisible(true);



		}
} // fin de la clase principal
```

```
class Interfaz extends JFrame implements ActionListener{

        JLabel lbl_login=new JLabel("Login");
        JLabel lbl_password=new JLabel("Password");

        JTextField txt_login=new JTextField(30);
        JPasswordField palabraclave=new JPasswordField(20);

        JButton btn_ingresar=new JButton("Ingresar");

        Container contenedor;
        Color fondopanel = new Color(204,204,255);


public Interfaz()
        {
                //inicio del constructor



          setTitle("Ingresar a LabCli 1.0");
          setSize(300,200);

          contenedor=this.getContentPane();
          addWindowListener(new Manejador());
          setResizable(false);
          palabraclave.addActionListener(alp);
          btn_ingresar.addActionListener(this);

          JPanel panel=new JPanel();

          panel.setLayout(null);
          lbl_login.setBounds(20,50,70,20);
          panel.setToolTipText("Seguridad LabCli");
          panel.add(lbl_login);
          lbl_password.setBounds(20,90,70,20);
          panel.add(lbl_password);

          txt_login.setBounds(100,50,150,20);
          panel.add(txt_login);
          txt_login.setToolTipText("Introduce el nombre de usuario");
          palabraclave.setBounds(100,90,150,20);
          panel.add(palabraclave);
          palabraclave.setToolTipText("Introduce tu contrasena");


          btn_ingresar.setBounds(130,130,100,20);
          panel.add(btn_ingresar);
          btn_ingresar.setToolTipText("Click para ingresar a LabCli");
```

```
                panel.setBackground(fondopanel);
                panel.setBorder(javax.swing.BorderFactory.createTitledBorder("Ingresar
a LabCli"));

                contenedor.add(panel);
            }//fin del constructor de la aplicacion principal de Acceso


public void actionPerformed(ActionEvent e)

            //inicio del actionPerformed de los botones
            {

            if(e.getSource()==btn_ingresar)
               {
                 comprobarpassword();
               }


             }


            ActionListener alp = new ActionListener(){

            public void actionPerformed(ActionEvent evt){

            if(evt.getSource()==palabraclave){
               comprobarpassword();

               }
            }

            };

            public void comprobarpassword()
            {

            char[]  passwordcorrecto={'m','e','l','v','i','n'};
                        String nombre_usuario = "usuario";
                        String obtiene_nombre = new String();

                String aux = new String (passwordcorrecto);
                char [] passwordelegido=palabraclave.getPassword();
                String var = new String(passwordelegido);

                        obtiene_nombre = txt_login.getText();

                        if(aux.equals(var) && obtiene_nombre.equals(nombre_usuario) )
            {
```

```
                        Bancosangre B = new Bancosangre();
                        B.setVisible(true);
                         setVisible(false);


                        System.out.println("Permiso concedido al sistema ");


                                }
                else {
                        dialogo_error();
                        System.out.println("Permiso Denegado al sistema");
                        }
            }// fin de la funcion comprueba Password

    public void dialogo_error()
            {
        JOptionPane.showMessageDialog( this,"Introduce bien tus datos ","Acceso
                                denegado",JOptionPane.ERROR_MESSAGE);

            }



        class Manejador extends WindowAdapter{
        public void windowClosing(WindowEvent e){
        System.out.println("saliendo...");
          System.exit(0);
                }

          }

        }//fin de la interfaz principal


class Bancosangre extends JFrame implements ActionListener{


        //Creacion de barra de menus
        JMenuBar barramenu;
        JMenu archivomenu;
        JMenu opcionesmenu;
        JMenu ayudamenu;


        JMenuItem nuevomenu;
```

```
        JMenuItem salirmenu;
        JMenuItem guardarmenu;
        JMenuItem imprimirmenu;
        JMenuItem consultar_Ex;
        JMenuItem acercademenu;



        JLabel lbl_nombres=new JLabel("Nombres");
        JLabel lbl_apellidos=new JLabel("Apellidos");

        JTextField txt_nombres=new JTextField(30);
        JTextField txt_apellidos=new JTextField(30);


        JLabel lbl_id=new JLabel("ID");
        JLabel lbl_fecha=new JLabel("Fecha");

        JTextField txt_id=new JTextField(30);


        JFormattedTextField ftf= new JFormattedTextField(new java.util.Date( ));

//Datos del primer panel
        JLabel lbl_gruposanguineo=new JLabel("Grupo Sanguineo");
        JLabel lbl_factorrh=new JLabel("Factor RH");


        JTextField txt_gruposanguineo=new JTextField(30);
        JTextField txt_factorrh=new JTextField(30);


//Datos del segundo panel

        JLabel lbl_pruebacruzada=new JLabel("Prueba Cruzada");
        JLabel lbl_compatible=new JLabel("Compatible");
        JLabel lbl_unidadno=new JLabel("Unidad No");


        JTextField txt_pruebacruzada=new JTextField(20);
        JTextField txt_compatible=new JTextField(30);
        JTextField txt_unidadno=new JTextField(30);



//Datos del tercer panel

        JLabel lbl_coombsdirecto=new JLabel("Coombs Directo");
        JLabel lbl_coombsindirecto=new JLabel("Coombs Indirecto");
```

```
JScrollPane js_coombsdirecto = new JScrollPane();
JScrollPane js_coombsindirecto = new JScrollPane();


JTextArea txtarea_coombsdirecto=new JTextArea(3,20);
JTextArea txtarea_coombsindirecto=new JTextArea(3,20);


// Datos de otras pruebas

JLabel lbl_otraspruebas=new JLabel("Otras pruebas");
JTextArea txtarea_otraspruebas=new JTextArea(3,20);
JScrollPane js_otraspruebas = new JScrollPane();



//Informacion a mostrar a los usuarios

JLabel lbl_mensajes = new JLabel("Area de Mensajes al usuario");
JScrollPane js_mensajes = new JScrollPane();
JTextArea txtarea_mensajes=new JTextArea(10,30);



//Datos del laboratorista

JLabel lbl_fechaentrega=new JLabel("Fecha Entrega");
JLabel lbl_nombrelaboratorista=new JLabel("Nombre Laboratorista");

//JTextField txt_fechaentrega=new JTextField(20);
JFormattedTextField ftf_entrega= new JFormattedTextField(new
java.util.Date( ));
JTextField txt_nombrelaboratorista=new JTextField(30);


//Datos de botones de Opciones

JButton btn_consultar=new JButton("Consultar Ex");
JButton btn_nuevo=new JButton("Nuevo");
JButton btn_guardar=new JButton("Guardar");
JButton btn_imprimir=new JButton("Imprimir");
JButton btn_salir=new JButton("Salir");


JPanel panel =new JPanel();
JPanel panel1=new JPanel();
JPanel panel2=new JPanel();
JPanel panel3=new JPanel();
```

```
            JPanel panel4=new JPanel();
            JPanel panel5=new JPanel();
            JPanel panel6=new JPanel();
            JPanel panel7=new JPanel();
            JPanel panel8=new JPanel();

// fuentes y colores de fondo a utilizar
            Color fondopanel = new Color(168, 206, 242);
            Color fondo = new Color(255, 255, 255);


            //Elementos del panel

            java.util.Date fecha=new java.util.Date();
            SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");//Le
especifico el formato en que quiero que aparezca la facha

            Container contenedor;


            class solicitud{
               int sid;
               String snombres;
               String sapellidos;
               java.sql.Date fechasol;

               }

            public Bancosangre()
            {

               setTitle("Area de Banco de Sangre");
               setSize(950,800);
               contenedor=this.getContentPane();
               addWindowListener(new Manejador());
               //b1.addActionListener(this);
               //b2.addActionListener(this);
               btn_consultar.addActionListener(this);
               btn_nuevo.addActionListener(this);
               btn_guardar.addActionListener(this);
               btn_imprimir.addActionListener(this);
               btn_salir.addActionListener(this);
               contenedor.setLayout(new GridLayout(3,3,10,10));
               barramenu = new JMenuBar();
               archivomenu = new JMenu("Archivo");
               opcionesmenu = new JMenu("Opciones");
               ayudamenu = new JMenu("Ayuda");
               nuevomenu = new      JMenuItem("Nuevo");
               salirmenu =  new JMenuItem("Salir");
```

```
                guardarmenu = new JMenuItem("Guardar");
                imprimirmenu = new JMenuItem("Imprimir");
                consultar_Ex = new JMenuItem("Consultar Ex");
                acercademenu = new JMenuItem(" Acerca de...");
                //salirmenu.setIcon(new
ImageIcon(getClass().getResource("/nuevo1.png")));
                nuevomenu.addActionListener( al );
                salirmenu.addActionListener( al );
                guardarmenu.addActionListener( al );
                imprimirmenu.addActionListener( al );
                consultar_Ex.addActionListener( al );
                acercademenu.addActionListener( al );

                archivomenu.add(nuevomenu);
                archivomenu.addSeparator();
                archivomenu.add(salirmenu);
                archivomenu.addSeparator();

                opcionesmenu.add(consultar_Ex);
                opcionesmenu.addSeparator();
                opcionesmenu.add(guardarmenu);
                opcionesmenu.addSeparator();
                opcionesmenu.add(imprimirmenu);
                opcionesmenu.addSeparator();

                ayudamenu.add(acercademenu);
                ayudamenu.addSeparator();

                barramenu.add(archivomenu);
                barramenu.add(opcionesmenu);
                barramenu.add(ayudamenu);
                this.setJMenuBar(barramenu);




                   panel.setLayout(null);
                lbl_nombres.setBounds(40,40,70,20);
                panel.add(lbl_nombres);
                lbl_apellidos.setBounds(40,80,70,20);
                panel.add(lbl_apellidos);


                txt_nombres.setBounds(110,40,150,20);
                panel.add(txt_nombres);
                txt_apellidos.setBounds(110,80,150,20);
                panel.add(txt_apellidos);
```

```
panel.setBackground(fondopanel);
panel.setBorder(new TitledBorder("Datos del Paciente"));



panel1.setLayout(null);
lbl_id.setBounds(40,40,70,20);
panel1.add(lbl_id);
lbl_fecha.setBounds(40,80,70,20);
panel1.add(lbl_fecha);

txt_id.setBounds(110,40,150,20);
panel1.add(txt_id);
ftf.setBounds(110,80,150,20);
ftf.setEditable(false);
ftf.setText(sdf.format(fecha));
panel1.add(ftf);

panel1.setBackground(fondopanel);
panel1.setBorder(new TitledBorder("Datos de la Solicitud"));



panel2.setLayout(null);
lbl_mensajes.setBounds(20,20,200,20);
panel2.add(lbl_mensajes);
txtarea_mensajes.setEditable(false);
js_mensajes.setViewportView(txtarea_mensajes);
js_mensajes.setBounds(30,60,250,90);
js_mensajes.setToolTipText("Area de mensajes para el usuario");
panel2.add(js_mensajes);

panel2.setBackground(fondopanel);
panel2.setBorder(new TitledBorder("Informacion a Usuarios"));



//anadimos los datos del primer panel


panel3.setLayout(null);

lbl_gruposanguineo.setBounds(20,60,150,20);
panel3.add(lbl_gruposanguineo);
lbl_factorrh.setBounds(20,100,70,20);
panel3.add(lbl_factorrh);
```

```
            txt_gruposanguineo.setBounds(130,60,150,20);
            panel3.add(txt_gruposanguineo);
            txt_factorrh.setBounds(130,100,150,20);
            panel3.add(txt_factorrh);

            panel3.setBackground(fondopanel);

            panel.setBorder(new TitledBorder("Datos Sangre"));



//anadimos los datos del segundo examen


            panel4.setLayout(null);

            lbl_pruebacruzada.setBounds(40,60,150,20);
            panel4.add(lbl_pruebacruzada);
            lbl_compatible.setBounds(40,100,150,20);
            panel4.add(lbl_compatible);
            lbl_unidadno.setBounds(40,140,150,20);
            panel4.add(lbl_unidadno);


            txt_pruebacruzada.setBounds(140,60,150,20);
            panel4.add(txt_pruebacruzada);
            txt_compatible.setBounds(140,100,150,20);
            panel4.add(txt_compatible);
            txt_unidadno.setBounds(140,140,150,20);
            panel4.add(txt_unidadno);

            panel4.setBackground(fondopanel);
            panel4.setBorder(new TitledBorder("Resultados"));



//anadimos los datos del tercer panel


            panel5.setLayout(null);

            lbl_coombsdirecto.setBounds(20,30,150,20);
            panel5.add(lbl_coombsdirecto);
            lbl_coombsindirecto.setBounds(20,120,150,20);
            panel5.add(lbl_coombsindirecto);
```

```
txtarea_coombsdirecto.setEditable(false);
js_coombsdirecto.setViewportView(txtarea_coombsdirecto);
js_coombsdirecto.setBounds(140,20,150,70);
panel5.add(js_coombsdirecto);
js_coombsdirecto.setToolTipText("Coombs Directo");
panel5.add(js_coombsdirecto);

txtarea_coombsindirecto.setEditable(false);
js_coombsindirecto.setViewportView(txtarea_coombsindirecto);
js_coombsindirecto.setBounds(140,130,150,70);
panel5.add(js_coombsindirecto);
js_coombsindirecto.setToolTipText("Datos Cuantitativos");


panel5.setBackground(fondopanel);
panel5.setBorder(new TitledBorder("Pruebas de Coombs"));



//anadimos los datos del cuarto panel


panel6.setLayout(null);

lbl_otraspruebas.setBounds(20,60,150,20);
panel6.add(lbl_otraspruebas);

txtarea_otraspruebas.setEditable(false);
js_otraspruebas.setViewportView(txtarea_otraspruebas);
js_otraspruebas.setBounds(120,60,150,90);
panel6.add(js_otraspruebas);
js_otraspruebas.setToolTipText("Otras Pruebas Realizadas");

panel6.add(js_otraspruebas);


panel6.setBackground(fondopanel);
panel6.setBorder(new TitledBorder("Otras Pruebas"));


//Datos del laboratorista


panel7.setLayout(null);

lbl_fechaentrega.setBounds(20,60,150,20);
panel7.add(lbl_fechaentrega);
lbl_nombrelaboratorista.setBounds(20,100,150,20);
```

```
panel7.add(lbl_nombrelaboratorista);


ftf_entrega.setBounds(160,60,140,20);
ftf_entrega.setEditable(false);
ftf_entrega.setText(sdf.format(fecha));
panel7.add(ftf_entrega);
txt_nombrelaboratorista.setBounds(160,100,140,20);
panel7.add(txt_nombrelaboratorista);

panel7.setBackground(fondopanel);
panel7.setBorder(new TitledBorder("Datos del Laboratorista"));




//anadimos los botones de opciones


panel8.setLayout(null);

btn_consultar.setBounds(85,20,150,30);
btn_consultar.setMnemonic('C');
btn_consultar.setToolTipText("Consultar si hay datos");
panel8.add(btn_consultar);
btn_nuevo.setBounds(85,60,150,30);
btn_nuevo.setMnemonic('N');
btn_nuevo.setToolTipText("Cargar Nuevo Examen");
panel8.add(btn_nuevo);
btn_guardar.setMnemonic('G');
btn_guardar.setToolTipText("Guardar los datos ");
btn_guardar.setBounds(85,100,150,30);
panel8.add(btn_guardar);
btn_imprimir.setMnemonic('I');
btn_imprimir.setToolTipText("Imprimir resultados Ex");
btn_imprimir.setBounds(85,140,150,30);
panel8.add(btn_imprimir);
btn_salir.setMnemonic('S');
btn_salir.setToolTipText("Salir de la aplicacion");
btn_salir.setBounds(85,180,150,30);
panel8.add(btn_salir);

//btn_salir.setIcon(new ImageIcon("/imagen.jpg"));


panel8.setBackground(fondopanel);
panel.setBorder(new TitledBorder("Botones de Opciones"));
```

```
        contenedor.add(panel);
        contenedor.add(panel1);
        contenedor.add(panel2);
        contenedor.add(panel3);
        contenedor.add(panel4);
        contenedor.add(panel5);
        contenedor.add(panel6);
        contenedor.add(panel7);
        contenedor.add(panel8);

        invalidar();


        }




        ActionListener al = new ActionListener() {
public void actionPerformed( ActionEvent evt ){
            if(evt.getSource()==nuevomenu)
                    {
                    validar();
                    conectarBD();
                    System.out.println("Se ha presionado el menu nuevo");
                    dialogoinformacion();
                    btn_nuevo.setEnabled(false);
                    btn_guardar.setEnabled(true);
                    nuevomenu.setEnabled(false);
                    guardarmenu.setEnabled(true);

                    }




            if(evt.getSource()==consultar_Ex)
                  {

                    consultarBD();
                    btn_nuevo.setEnabled(true);
                    nuevomenu.setEnabled(true);

                    System.out.println("Se ha escogido el menu Consultar EX");
```

```
        }



if(evt.getSource()==guardarmenu)
        {

        boolean estado;

        estado=comprobar_datos();
        if(estado){
        actualizar_estado();
        btn_imprimir.setEnabled(true);
        imprimirmenu.setEnabled(true);
        }
        System.out.println("Se ha escogido el menu guardar");
        }



        if(evt.getSource()==imprimirmenu)
        {

        System.out.println("Se ha presionado el menu imprimir");

        dialogoimprimir();
        limpiar();
        nuevomenu.setEnabled(false);
        btn_nuevo.setEnabled(false);
        guardarmenu.setEnabled(false);
        btn_guardar.setEnabled(false);
        imprimirmenu.setEnabled(false);
        btn_imprimir.setEnabled(false);
        invalidar();

        }



if(evt.getSource()==acercademenu)
        {

        AcercaDe();

        System.out.println("Se ha escogido el menu acerca de...");
        }
```

```
            if(evt.getSource()==salirmenu)
                    {
                    dialogosalir();


                    }

}
        };




        //Eventos de los menus
        public void actionPerformed(ActionEvent e)
        {

            if(e.getSource()==btn_consultar)
                    {
                    consultarBD();
                    System.out.println("Se ha presionado el boton consultar");
                    btn_nuevo.setEnabled(true);
                    nuevomenu.setEnabled(true);

                    }



            if(e.getSource()==btn_nuevo)
                    {
                    validar();
                    conectarBD();
                    System.out.println("Se ha presionado el boton nuevo");
                    dialogoinformacion();
                    btn_nuevo.setEnabled(false);
                    btn_guardar.setEnabled(true);
                    nuevomenu.setEnabled(false);
                    guardarmenu.setEnabled(true);
                    }
```

```
                if(e.getSource()==btn_guardar)
                        {
                if(txt_nombres.getText().length()==0 ||
txt_apellidos.getText().length()==0|| txt_id.getText().length()==0)
                {
                dialogofinExamen();
                invalidar();
                btn_guardar.setEnabled(false);
                guardarmenu.setEnabled(false);
                btn_nuevo.setEnabled(false);
                btn_consultar.setEnabled(true);
                //btn_salir.setEanbled(true);
                btn_imprimir.setEnabled(false);
                imprimirmenu.setEnabled(false);
                nuevomenu.setEnabled(false);

                 }

                        boolean estado;

                        estado=comprobar_datos();
                        if(estado){
                        actualizar_estado();
                        btn_imprimir.setEnabled(true);
                        }


                        }



                        if(e.getSource()==btn_imprimir)
                        {
                        System.out.println("Se ha presionado el boton imprimir");


                        dialogoimprimir();
                        limpiar();
                        nuevomenu.setEnabled(false);
                        btn_nuevo.setEnabled(false);
                        guardarmenu.setEnabled(false);
                        btn_guardar.setEnabled(false);
                        imprimirmenu.setEnabled(false);
                        btn_imprimir.setEnabled(false);
                        invalidar();
```

```
                    }


                if(e.getSource()==btn_salir)
                    {
                    dialogosalir();
                    }


                }




            class Manejador extends WindowAdapter{
            public void windowClosing(WindowEvent e){
               System.out.println("saliendo...");
               System.exit(0);
            }
}




            public void conectarBD(){

                    System.out.println("Entrando a la funcion conectarBD");


               String consultaid = "select  distinct min (id), id from solicitud where
estado='en proceso' and cod_area = 'Banco de Sangre' group by id";
               Connection conn = null;
               int var;
            try
              {


      conn =
DriverManager.getConnection("jdbc:postgresql://localhost/monografia","usuario",
"usuario123");
```

```
                System.out.println("Estableciendo la conexion con la base de
datos");

        if (conn != null)
          {
                System.out.println("Entrando en el if");

          Statement stmt = conn.createStatement();
          ResultSet res  = stmt.executeQuery(consultaid);


                while(res.next())

                {



                        System.out.println("Entrando en el while");

                        solicitud objeto = new solicitud();

                objeto.sid  = res.getInt("id");
                        int pid;
                        pid=objeto.sid;

                        System.out.println("id:"+pid);

                        compruebaRango(pid);

                        res.close();
                        }

          //res.close();
          stmt.close();
          conn.close();
          }


          else
                {
                String mensaje = "No hay datos en los examenes" +"\n";
                System.out.println(mensaje);
                txtarea_mensajes.append(mensaje);



                }
        }
    catch(SQLException ex)
```

```
    {
      //System.out.println(ex);
                System.out.println("Cerraste inadecuadamente el resultset del while");
    }

  }
//fin de la primera consulta conectarBD();*/




            public void consultarBD(){

                        System.out.println("Entrando a la funcion consultarBD");


              String consultaid = "select   id from solicitud where estado='en proceso'
and cod_area = 'Banco de Sangre'";
              Connection conn = null;
              int var = 0;
            try
              {

      conn =
DriverManager.getConnection("jdbc:postgresql://localhost/monografia","usuario",
"usuario123");
                        System.out.println("Estableciendo la conexion con la base de
datos");


                        if (conn != null)
              {
                        System.out.println("Entrando en el if");

              Statement stmt = conn.createStatement();
              ResultSet res  = stmt.executeQuery(consultaid);




                          while(res.next())
                        {
```

```
                                System.out.println("Entrando en el while");

                                solicitud objeto = new solicitud();

                        objeto.sid  = res.getInt("id");
                                int pid;
                                pid=objeto.sid;

                                System.out.println("id:"+pid);

                                var++;
                                System.out.println("Tienes:"+var+ "\t Examen
pendiente"+" \n");
                                String aux = new String();
                                aux = "Examenes Pendientes:";
                                String concatena = new String();
                                concatena = String.valueOf(var);
                                aux.concat(concatena);
                                System.out.println("Valor de concatena"+concatena);
                                txtarea_mensajes.setText(aux+concatena + "\n");

                                }

                        res.close();
                        stmt.close();
                        conn.close();
                        }

                    }
            catch(SQLException ex)
            {
              System.out.println(ex);
                    txtarea_mensajes.setText("No tiene examenes pendientes"+ "\n");
                      //System.out.println("Error de Excepcion");
            }

          }
//fin de la primera consulta conectarBD();




                        public void compruebaRango( int numero_id)
                        {
```

```
            System.out.println("Entrando a la funcion compruebaRango");

            String query = "select id, nombres, apellidos, fecha from solicitud where
estado='en proceso' and  id = ?";
            Connection conn = null;
            int numeroderegistro =0; //superior;
            numeroderegistro = numero_id;
            //superior= numeroderegistro + 2;

          try
            {

      conn =
DriverManager.getConnection("jdbc:postgresql://localhost/monografia","usuario",
"usuario123");

          if (conn != null)
            {

                  PreparedStatement sentencia;
                  sentencia = conn.prepareStatement(query);

                  sentencia.setInt(1, numeroderegistro);


                  ResultSet resulset = sentencia.executeQuery();


                  while(resulset.next())
                  {

                        System.out.println("Entrando al while de comprueba
rango");

                        solicitud objeto = new solicitud();

                  objeto.sid  = resulset.getInt("id");
                  objeto.snombres = resulset.getString("nombres");
                  objeto.sapellidos = resulset.getString("apellidos");
                        int pid;
                        pid=objeto.sid;
                        String pnombres = new String();
                        pnombres=objeto.snombres;
                        String papellidos = new String();
                        papellidos=objeto.sapellidos;
                        objeto.fechasol = resulset.getDate("fecha");

                        txt_nombres.setText(pnombres);
```

```
                                txt_apellidos.setText(papellidos);
                                txt_id.setText(String.valueOf(pid));
                                ftf.setText(String.valueOf(objeto.fechasol));
                                ftf.setEditable(false);

                        System.out.println(pid
+"\t\t"+pnombres+"\t\t"+papellidos+"\t\t"+objeto.fechasol);
                                //res.close();

                                System.out.println("Se esta recorriendo la Base de Datos
Laboratorio");
                        }
                                txtarea_mensajes.append("Procesando nuevo
examen"+"\n");

                                sentencia.executeUpdate();
                    resulset.close();
                    sentencia.close();
                    conn.close();
                    }

                }
        catch(SQLException ex)
        {
          // System.out.println(ex);

        }

                }









                public void actualizar_estado()
                {


                String query = "update solicitud set estado = 'finalizado' where id = ?";
                Connection conn = null;

                /*if(txt_nombres.getText().length()==0 ||
txt_apellidos.getText().length()==0|| txt_id.getText().length()==0)
                {
```

```
            dialogofinExamen();

             }*/


                    try
                      {


            conn =
DriverManager.getConnection("jdbc:postgresql://localhost/monografia","usuario",
"usuario123");

                    if (conn != null)
                            {

                            int  id_actualizar;


                            PreparedStatement sentencia;

                            sentencia = conn.prepareStatement(query);

                            id_actualizar= Integer.parseInt(txt_id.getText());

                            sentencia.setInt(1, id_actualizar);

                            ResultSet res = sentencia.executeQuery();


                            sentencia.executeUpdate();


                    res.close();
                    sentencia.close();




                    }

            conn.close();

                }
    catch(SQLException ex)
    {
      System.out.println(ex);

            String mensaje = "Se ha actualizado el registro"+ "\n";
```

```
            System.out.println(mensaje);
            txtarea_mensajes.append(mensaje);

}


        }









        public void invalidar()
        {

        txt_nombres.setEditable(false);
        txt_apellidos.setEditable(false);

        txt_id.setEditable(false);
        //js_mensajes.setVisible(false);

        txt_gruposanguineo.setEditable(false);
        txt_factorrh.setEditable(false);

        txt_pruebacruzada.setEditable(false);
        txt_compatible.setEditable(false);
        txt_unidadno.setEditable(false);


        txtarea_coombsdirecto.setEditable(false);
        txtarea_coombsindirecto.setEditable(false);



        txtarea_otraspruebas.setEditable(false);


        txt_nombrelaboratorista.setEditable(false);

        btn_nuevo.setEnabled(false);
        btn_guardar.setEnabled(false);
        btn_imprimir.setEnabled(false);

        nuevomenu.setEnabled(false);
```

```
guardarmenu.setEnabled(false);
imprimirmenu.setEnabled(false);


}






public void validar()
{
txt_nombres.setEditable(false);
txt_apellidos.setEditable(false);

txt_id.setEditable(false);


txt_gruposanguineo.setEditable(true);
txt_factorrh.setEditable(true);

txt_pruebacruzada.setEditable(true);
txt_compatible.setEditable(true);
txt_unidadno.setEditable(true);


txtarea_coombsdirecto.setEditable(true);
txtarea_coombsindirecto.setEditable(true);



txtarea_otraspruebas.setEditable(true);


txt_nombrelaboratorista.setEditable(true);

txtarea_mensajes.setText("");
btn_nuevo.setEnabled(true);
btn_guardar.setEnabled(false);
btn_imprimir.setEnabled(false);


}
```

```
public void limpiar(){

txt_nombres.setText("");
txt_apellidos.setText("");
txt_id.setText("");


txt_nombres.setEditable(false);
txt_apellidos.setEditable(false);

txt_id.setEditable(false);

txt_gruposanguineo.setText("");
txt_factorrh.setText("");

txt_pruebacruzada.setText("");
txt_compatible.setText("");
txt_unidadno.setText("");


txtarea_coombsdirecto.setText("");
txtarea_coombsindirecto.setText("");

txtarea_otraspruebas.setText("");

txt_nombrelaboratorista.setEditable(true);

txt_nombres.requestFocus();
}









public boolean comprobar_datos()
{


if(txt_gruposanguineo.getText().length()==0 || txt_factorrh.getText().length()==0 )
    {
    btn_guardar.setEnabled(true);
    guardarmenu.setEnabled(true);
```

```
            dialogoErrorDatos();
            return false;
            }

            else{
            dialogoDatosGuardados();
            btn_guardar.setEnabled(false);
            guardarmenu.setEnabled(false);
            return true;
               }

         }




         public void dialogosalir()
         {
         int res = JOptionPane.showConfirmDialog( this,"Realmente desea
salir?","Salir de LabCli",JOptionPane.INFORMATION_MESSAGE);
                  String respuesta = null;

                        if( res == JOptionPane.YES_OPTION ) {
                              respuesta = "Si";
                              System.out.println("Saliendo de los dialogos y de
la aplicacion");
                                    System.exit(0);

                                          }
                         else
                        respuesta = "No";
                              System.out.println( "Respuesta: "+respuesta );

                  System.out.println("Se ha presionado el boton salir");
                  System.out.println("saliendo...");
            //System.exit(0);

         }
```

```
public void dialogoimprimir()
{
Bancosangre miFrame = new Bancosangre();


                PrintJob miPrintJob = miFrame.getToolkit().getPrintJob(
miFrame,"Imprimiendo los datos de los examenes",null );

                        if( miPrintJob != null ) {
                        // Obtenemos  el objeto gráfico que va a imprimir
                        Graphics graficoImpresion = miPrintJob.getGraphics();

                                graficoImpresion.setFont( new Font(
"Arial",Font.BOLD,14 ) );

                                if( graficoImpresion != null ) {

                                panel.printAll(graficoImpresion);
                                        panel1.printAll(graficoImpresion);
                                        panel3.printAll(graficoImpresion);
                                        panel4.printAll(graficoImpresion);
                                        panel5.printAll(graficoImpresion);
                                        panel6.printAll(graficoImpresion);
                                        panel7.printAll(graficoImpresion);


                                //miFrame.printAll( graficoImpresion );
                                        // Hacemos que se libere el papel de la
impresora y los
                                // recursos del sistema que estaba utilizando el
                                // objeto gráfico
                                graficoImpresion.dispose();

                                }
                                        miPrintJob.end();

                                }
        }
```

```
        public void dialogoinformacion()
        {
        JOptionPane.showMessageDialog( this,"Nuevo dato
procesado","Agregando",JOptionPane.INFORMATION_MESSAGE);

        }



        public void dialogoerror()
        {
        JOptionPane.showMessageDialog( this,"Datos
Guardados","Guardar",JOptionPane.ERROR_MESSAGE);
        }



        public void dialogoErrorDatos()
        {
        JOptionPane.showMessageDialog( this,"Debe llenar correctamente los
Datos","Error",JOptionPane.ERROR_MESSAGE);
        }



        public void dialogofinExamen()
        {
        JOptionPane.showMessageDialog( this,"No hay mas Examenes por
procesar","Error",JOptionPane.ERROR_MESSAGE);
        txtarea_mensajes.setText("NO TIENE EXAMENES PENDIENTES");
        }



        public void dialogoDatosGuardados()
        {
        JOptionPane.showMessageDialog( this,"Datos Guardados
Correctamente","Guardar",JOptionPane.INFORMATION_MESSAGE);
        }



        public void AcercaDe()
```

```
            {
            JOptionPane.showMessageDialog(panel,"LabCli HEODRA  Version 1.0
\n Desarrollado por \nJAVIER \nMELVIN                    \nGABRIEL  \nLinux
Debian + Java + Postgresql","Acerca de",JOptionPane.OK_OPTION);
            }
}
```

Codigo de la Aplicación Banco de Orina

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.lang.*;
import java.lang.String.*;
import java.util.*;
import java.io.*;
import java.net.*;
import javax.swing.JScrollPane;
import java.lang.*;
import java.lang.Character;
import java.lang.String;
import java.sql.*;
import java.io.PrintStream;
import javax.swing.border.TitledBorder;
import java.text.SimpleDateFormat;
import java.util.Date;



class orina

{
      public static void main(String []args)
      {


            try
                        {
                              Class.forName("org.postgresql.Driver");
                        }

                        catch(ClassNotFoundException ex)
                        {
                              System.out.println(ex);
                              System.exit(-1);
                        }


            Interfaz t=new Interfaz();
            t.setVisible(true);



      }
} // fin de la clase principal


class Interfaz extends JFrame implements ActionListener{

      JLabel lbl_login=new JLabel("Login");
      JLabel lbl_password=new JLabel("Password");

      JTextField txt_login=new JTextField(30);
      JPasswordField palabraclave=new JPasswordField(20);
```

```
      JButton btn_ingresar=new JButton("Ingresar");

      Container contenedor;
      Color fondopanel = new Color(114, 181, 245);


public Interfaz()
      {
                  //inicio del constructor



            setTitle("Ingresar a LabCli 1.0");
            setSize(300,200);

            contenedor=this.getContentPane();
            addWindowListener(new Manejador());
            setResizable(false);
            palabraclave.addActionListener(alp);
            btn_ingresar.addActionListener(this);

            JPanel panel=new JPanel();

            panel.setLayout(null);
            lbl_login.setBounds(20,50,70,20);
            panel.setToolTipText("Seguridad LabCli");
            panel.add(lbl_login);
            lbl_password.setBounds(20,90,70,20);
            panel.add(lbl_password);

            txt_login.setBounds(100,50,150,20);
            panel.add(txt_login);
            txt_login.setToolTipText("Introduce el nombre de usuario");
            palabraclave.setBounds(100,90,150,20);
            panel.add(palabraclave);
            palabraclave.setToolTipText("Introduce tu contrasena");


            btn_ingresar.setBounds(130,130,100,20);
            panel.add(btn_ingresar);
            btn_ingresar.setToolTipText("Click para ingresar a
LabCli");
            panel.setBackground(fondopanel);

      panel.setBorder(javax.swing.BorderFactory.createTitledBorder("In
gresar a LabCli"));

            contenedor.add(panel);
      }//fin del constructor de la aplicacion


public void actionPerformed(ActionEvent e)
      {     //inicio del actionPerformed de los botones

      if(e.getSource()==btn_ingresar)
            {
               comprobarpassword();
            }
```

```
       }


  ActionListener alp = new ActionListener(){

      public void actionPerformed(ActionEvent evt){

      if(evt.getSource()==palabraclave){
            comprobarpassword();

            }
      }

      };

      public void comprobarpassword()
      {

      char[]  passwordcorrecto={'j','a','v','i','e','r'};
                  String nombre_usuario = "usuario";
                  String obtiene_nombre = new String();

                String aux = new String (passwordcorrecto);
                char [] passwordelegido=palabraclave.getPassword();
                String var = new String(passwordelegido);

                  obtiene_nombre = txt_login.getText();

                  if(aux.equals(var) &&
obtiene_nombre.equals(nombre_usuario) )  {

                        Uroanalisis U = new Uroanalisis();
                        U.setVisible(true);
                         setVisible(false);


                        System.out.println("Permiso concedido al
sistema ");


                              }
                  else {
                        dialogo_error();
                        System.out.println("Permiso Denegado al
sistema");
                        }
}

        public void dialogo_error()
            {
JOptionPane.showMessageDialog( this,"Introduce bien tus datos
","Acceso denegado",JOptionPane.ERROR_MESSAGE);
}



      class Manejador extends WindowAdapter{
      public void windowClosing(WindowEvent e){
      System.out.println("saliendo...");
```

```
            System.exit(0);
                  }

            }

      }

class Uroanalisis extends JFrame implements ActionListener{   //inicio
de la interfaz


      //Creacion de barra de menus
      JMenuBar barramenu;
      JMenu archivomenu;
      JMenu opcionesmenu;
      JMenu ayudamenu;


//lo que esta adentro de los menus
      JMenuItem nuevomenu;
      JMenuItem salirmenu;
      JMenuItem guardarmenu;
      JMenuItem imprimirmenu;
      JMenuItem consultar_Ex;
      JMenuItem acercademenu;

      JLabel lbl_nombres=new JLabel("Nombres");
      JLabel lbl_apellidos=new JLabel("Apellidos:");

      JTextField txt_nombres=new JTextField(30);
      JTextField txt_apellidos=new JTextField(30);


      JLabel lbl_id=new JLabel("ID");
      JLabel lbl_fecha=new JLabel("Fecha");

      JTextField txt_id=new JTextField(30);
      //JTextField txt_fecha=new JTextField(30);


      JFormattedTextField ftf= new JFormattedTextField(new
java.util.Date( ));


// Panel para area de mensajes a usuarios del sistema

      JScrollPane js_mensajes = new JScrollPane();
      JTextArea txtarea_mensajes=new JTextArea(10,30);

//Datos del examen fisico
      JLabel lbl_color=new JLabel("Color");
      JLabel lbl_aspecto=new JLabel("Aspecto");
      JLabel lbl_sedimento=new JLabel("Sedimento");
      JLabel lbl_densidad=new JLabel("Densidad");


      JTextField txt_color=new JTextField(30);
      JTextField txt_aspecto=new JTextField(30);
      JTextField txt_sedimento=new JTextField(30);
      JTextField txt_densidad=new JTextField(30);
```

```
//Datos del examen Quimico

      JLabel lbl_proteinas=new JLabel("Proteinas");
      JLabel lbl_hemoglobina=new JLabel("Hemoglobina");
      JLabel lbl_cuerposcetonicos=new JLabel("Cuerpos Cetonicos");
      JLabel lbl_ph=new JLabel("P.H");
      JLabel lbl_urobilinogeno=new JLabel("Urobilinogeno");
      JLabel lbl_glucosa=new JLabel("Glucosa");
      JLabel lbl_bilirrubinas=new JLabel("Bilirrubinas");
      JLabel lbl_nitritos=new JLabel("Nitritos");

      JTextField txt_proteinas=new JTextField(20);
      JTextField txt_hemoglobina=new JTextField(30);
      JTextField txt_cuerposcetonicos=new JTextField(30);
      JTextField txt_ph=new JTextField(20);
      JTextField txt_urobilinogeno=new JTextField(30);
      JTextField txt_glucosa=new JTextField(30);
      JTextField txt_bilirrubinas=new JTextField(20);
      JTextField txt_nitritos=new JTextField(30);


//Datos del examen microscopico

      JLabel lbl_celulasepiteliales=new JLabel("Celulas Epiteliales");
      JLabel lbl_leucocitos=new JLabel("Leucocitos");
      JLabel lbl_eritrocitos=new JLabel("Eritrocitos");
      JLabel lbl_cilindros=new JLabel("Cilindros");
      JLabel lbl_cristales=new JLabel("Cristales");
      JLabel lbl_otros=new JLabel("Otros");

      JTextField txt_celulasepiteliales=new JTextField(20);
      JTextField txt_leucocitos=new JTextField(30);
      JTextField txt_eritrocitos=new JTextField(30);
      JTextField txt_cilindros=new JTextField(20);
      JTextField txt_cristales=new JTextField(30);
      JTextField txt_otros=new JTextField(30);


//Datos del examen de prueba  de embarazo

      JLabel lbl_cualitativo=new JLabel("Cualitativo");
      JLabel lbl_cuantitativo=new JLabel("Cuantitativo");


      JScrollPane js_cualitativo = new JScrollPane();
      JScrollPane js_cuantitativo = new JScrollPane();

      JTextArea txtarea_cualitativo=new JTextArea(3,15);
      JTextArea txtarea_cuantitativo=new JTextArea(3,15);


//Datos de la cantidad de examenes a realizar
JLabel lbl_datosexamenes = new JLabel("Mensajes");


//Datos del laboratorista

      JLabel lbl_fechaentrega=new JLabel("Fecha Entrega");
```

```
      JLabel lbl_nombrelaboratorista=new JLabel("Nombre
Laboratorista");

      JFormattedTextField txt_fechaentrega=new JFormattedTextField(new
java.util.Date( ));
      JTextField txt_nombrelaboratorista=new JTextField(30);


//Datos de botones de Opciones

      JButton btn_nuevo=new JButton("Nuevo");
      JButton btn_guardar=new JButton("Guardar");
      JButton btn_imprimir=new JButton("Imprimir");
      JButton btn_consultar=new JButton("Consultar Ex");
      JButton btn_salir=new JButton("Salir");


      // creamos los paneles a utilizar
      JPanel panel =new JPanel();
      JPanel panel1=new JPanel();
      JPanel panel2=new JPanel();
      JPanel panel3=new JPanel();
      JPanel panel4=new JPanel();
      JPanel panel5=new JPanel();
      JPanel panel6=new JPanel();
      JPanel panel7=new JPanel();
      JPanel panel8=new JPanel();

      Color fondopanel = new Color(168, 206, 242);
      Color fondo = new Color(255, 255, 255);
      Font fuente = new Font("SansSerif",Font.BOLD,13 );
      java.util.Date fecha=new java.util.Date();
      SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");//Le
especifico el formato en que quiero que aparezca la facha



      Container contenedor;

      class solicitud{
            int sid;
            String snombres;
            String sapellidos;
            java.sql.Date fechasol;

            }


      public Uroanalisis()
      {
                  //inicio del constructor


            System.out.println("Entrando al constructor de
uroanalisis");
            setTitle("Area de Uroanalisis");
            setSize(950,800);

            contenedor=this.getContentPane();
            addWindowListener(new Manejador());
```

```
            btn_nuevo.addActionListener(this);
            btn_guardar.addActionListener(this);
            btn_imprimir.addActionListener(this);
            btn_consultar.addActionListener(this);
            btn_salir.addActionListener(this);
            contenedor.setLayout(new GridLayout(3,3,5,5));
            barramenu = new JMenuBar();
            archivomenu = new JMenu("Archivo");
            opcionesmenu = new JMenu("Opciones");
            ayudamenu = new JMenu("Ayuda");
            nuevomenu = new   JMenuItem("Nuevo");
            salirmenu =  new JMenuItem("Salir");
            guardarmenu = new JMenuItem("Guardar");
            imprimirmenu = new JMenuItem("Imprimir");
            consultar_Ex = new JMenuItem("Consultar Ex");
            acercademenu = new JMenuItem(" Acerca de...");
            //salirmenu.setIcon(new
ImageIcon(getClass().getResource("/nuevo1.png")));
            nuevomenu.addActionListener( al );
            salirmenu.addActionListener( al );
            guardarmenu.addActionListener( al );
            imprimirmenu.addActionListener( al );
            consultar_Ex.addActionListener( al );
            acercademenu.addActionListener( al );

            archivomenu.add(nuevomenu);
            archivomenu.addSeparator();
            archivomenu.add(salirmenu);
            archivomenu.addSeparator();



            opcionesmenu.add(consultar_Ex);
            opcionesmenu.addSeparator();
            opcionesmenu.add(guardarmenu);
            opcionesmenu.addSeparator();
            opcionesmenu.add(imprimirmenu);
            opcionesmenu.addSeparator();



            ayudamenu.addSeparator();
            ayudamenu.add(acercademenu);
            ayudamenu.addSeparator();

            barramenu.add(archivomenu);
            barramenu.add(opcionesmenu);
            barramenu.add(ayudamenu);
            this.setJMenuBar(barramenu);




            panel.setLayout(null);
            panel.setToolTipText("Datos Genarales Paciente");
            lbl_nombres.setFont(fuente);
            lbl_nombres.setBounds(20,50,70,20);
            panel.add(lbl_nombres);
            lbl_apellidos.setFont(fuente);
```

```
            lbl_apellidos.setBounds(20,90,70,20);
            panel.add(lbl_apellidos);


            txt_nombres.setBounds(100,50,150,20);
            txt_nombres.setBackground(fondo);
            panel.add(txt_nombres);
            txt_nombres.setToolTipText("Nombres Paciente");
            txt_apellidos.setBackground(fondo);
            txt_apellidos.setBounds(100,90,150,20);
            panel.add(txt_apellidos);
            txt_apellidos.setToolTipText("Apellidos Paciente");




            panel.setBackground(fondopanel);
            panel.setBorder(new TitledBorder("Datos del paciente"));


            panel1.setLayout(null);
            panel1.setToolTipText("Datos de Solicitud");
            lbl_id.setBounds(20,50,70,20);
            panel1.add(lbl_id);
            lbl_fecha.setBounds(20,90,70,20);
            panel1.add(lbl_fecha);


            txt_id.setToolTipText("ID Solicitud");
            txt_id.setBackground(fondo);
            txt_id.setBounds(75,50,150,20);
            panel1.add(txt_id);
            txt_id.setToolTipText("ID Solicitud");



            ftf.setBackground(fondo);
            ftf.setBounds(75,90,150,20);
            ftf.setEditable(false);
            ftf.setText(sdf.format(fecha));
            ftf.setToolTipText("Fecha de la solicitud");
            panel1.add(ftf);


            panel1.setBackground(fondopanel);
            panel1.setBorder(new TitledBorder("Codigo y Fecha del
Examen"));


            panel2.setLayout(null);

            panel2.setToolTipText("Muestra mensajes a usuario");
            lbl_datosexamenes.setBounds(20,20,150,20);
            panel2.add(lbl_datosexamenes);


            txtarea_mensajes.setEditable(false);
            js_mensajes.setViewportView(txtarea_mensajes);
            js_mensajes.setBounds(30,60,250,90);
```

```
            js_mensajes.setToolTipText("Area de mensajes para el
usuario");
            panel2.add(js_mensajes);


            panel2.setBackground(fondopanel);
            panel2.setBorder(new TitledBorder("Area de mensajes al
usuario"));



//anadimos los datos del examen fisico


            panel3.setLayout(null);
            panel3.setToolTipText("Datos del Examen Fisico");
            lbl_color.setBounds(20,50,70,20);
            panel3.add(lbl_color);
            lbl_aspecto.setBounds(20,80,70,20);
            panel3.add(lbl_aspecto);
            lbl_sedimento.setBounds(20,110,70,20);
            panel3.add(lbl_sedimento);
            lbl_densidad.setBounds(20,140,70,20);
            panel3.add(lbl_densidad);

            txt_color.setBounds(90,50,150,20);
            panel3.add(txt_color);
            txt_color.setToolTipText("Color Orina");
            txt_aspecto.setBounds(90,80,150,20);
            panel3.add(txt_aspecto);
            txt_aspecto.setToolTipText("Aspecto Orina");
            txt_sedimento.setBounds(90,110,150,20);
            panel3.add(txt_sedimento);
            txt_sedimento.setToolTipText("Sedimento orina");
            txt_densidad.setBounds(90,140,150,20);
            panel3.add(txt_densidad);
            txt_densidad.setToolTipText("Densidad Orina");


            panel3.setBackground(fondopanel);
            panel3.setBorder(new TitledBorder("Examen Fisico"));



//anadimos los datos del examen quimico


            panel4.setLayout(null);
            panel4.setToolTipText("Datos Quimicos");
            lbl_proteinas.setBounds(20,10,150,20);
            panel4.add(lbl_proteinas);
            lbl_hemoglobina.setBounds(20,35,150,20);
            panel4.add(lbl_hemoglobina);
            lbl_cuerposcetonicos.setBounds(20,60,150,20);
            panel4.add(lbl_cuerposcetonicos);
            lbl_ph.setBounds(20,85,150,20);
            panel4.add(lbl_ph);
            lbl_urobilinogeno.setBounds(20,110,150,20);
            panel4.add(lbl_urobilinogeno);
            lbl_glucosa.setBounds(20,135,150,20);
```

```
            panel4.add(lbl_glucosa);
            lbl_bilirrubinas.setBounds(20,160,150,20);
            panel4.add(lbl_bilirrubinas);
            lbl_nitritos.setBounds(20,185,150,20);
            panel4.add(lbl_nitritos);

            txt_proteinas.setBounds(140,10,150,20);
            panel4.add(txt_proteinas);
            txt_proteinas.setToolTipText("Datos Proteinas");
            txt_hemoglobina.setBounds(140,35,150,20);
            panel4.add(txt_hemoglobina);
            txt_hemoglobina.setToolTipText("Dato Hemoglobina");
            txt_cuerposcetonicos.setBounds(140,60,150,20);
            panel4.add(txt_cuerposcetonicos);
            txt_cuerposcetonicos.setToolTipText("Cuerpos Cetonicos");
            txt_ph.setBounds(140,85,150,20);
            panel4.add(txt_ph);
            txt_ph.setToolTipText("Ph Encontrado");
            txt_urobilinogeno.setBounds(140,110,150,20);
            panel4.add(txt_urobilinogeno);
            txt_urobilinogeno.setToolTipText("Urobilinogeno");
            txt_glucosa.setBounds(140,135,150,20);
            panel4.add(txt_glucosa);
            txt_glucosa.setToolTipText("Dato Glucosa");
            txt_bilirrubinas.setBounds(140,160,150,20);
            panel4.add(txt_bilirrubinas);
            txt_bilirrubinas.setToolTipText("Dato Bilirrubina");
            txt_nitritos.setBounds(140,185,150,20);
            panel4.add(txt_nitritos);
            txt_nitritos.setToolTipText("Datos Nitritos");


            panel4.setBackground(fondopanel);
            panel4.setBorder(new TitledBorder("Examen Quimico"));



//anadimos los datos del examen microscopico


            panel5.setLayout(null);
            panel5.setToolTipText("Datos Microscopicos");
            lbl_celulasepiteliales.setBounds(20,20,150,20);
            panel5.add(lbl_celulasepiteliales);
            lbl_leucocitos.setBounds(20,50,150,20);
            panel5.add(lbl_leucocitos);
            lbl_eritrocitos.setBounds(20,80,150,20);
            panel5.add(lbl_eritrocitos);
            lbl_cilindros.setBounds(20,110,150,20);
            panel5.add(lbl_cilindros);
            lbl_cristales.setBounds(20,140,150,20);
            panel5.add(lbl_cristales);
            lbl_otros.setBounds(20,170,150,20);
            panel5.add(lbl_otros);


            txt_celulasepiteliales.setBounds(140,20,150,20);
            panel5.add(txt_celulasepiteliales);
            txt_celulasepiteliales.setToolTipText("Celulas");
            txt_leucocitos.setBounds(140,50,150,20);
```

```
            panel5.add(txt_leucocitos);
            txt_leucocitos.setToolTipText("Datos Leucocitos");
            txt_eritrocitos.setBounds(140,80,150,20);
            panel5.add(txt_eritrocitos);
            txt_eritrocitos.setToolTipText("Datos Eritrocitos");
            txt_cilindros.setBounds(140,110,150,20);
            panel5.add(txt_cilindros);
            txt_cilindros.setToolTipText("Datos Cilindros");
            txt_cristales.setBounds(140,140,150,20);
            panel5.add(txt_cristales);
            txt_cristales.setToolTipText("Datos de Cristales");
            txt_otros.setBounds(140,170,150,20);
            panel5.add(txt_otros);
            txt_otros.setToolTipText("Otros datos encontrados");



            panel5.setBackground(fondopanel);
            panel5.setBorder(new TitledBorder("Examen Microscopico"));

//anadimos los datos del examen prueba de embarazo


            panel6.setLayout(null);
            panel6.setToolTipText("Datos Prueba de Embarazo");
            lbl_cualitativo.setBounds(20,20,150,20);
            panel6.add(lbl_cualitativo);
            lbl_cuantitativo.setBounds(20,130,150,20);
            panel6.add(lbl_cuantitativo);




            txtarea_cualitativo.setEditable(false);
            js_cualitativo.setViewportView(txtarea_cualitativo);
            js_cualitativo.setBounds(120,20,150,70);
            panel6.add(js_cualitativo);
            js_cualitativo.setToolTipText("Datos Cualitativos");


            txtarea_cuantitativo.setEditable(false);
            js_cuantitativo.setViewportView(txtarea_cuantitativo);
            js_cuantitativo.setBounds(120,130,150,70);
            panel6.add(js_cuantitativo);
            js_cuantitativo.setToolTipText("Datos Cuantitativos");




            panel6.setBackground(fondopanel);
            panel6.setBorder(new TitledBorder("Prueba de Embarazo"));


            //Datos del laboratorista

            panel7.setLayout(null);
            panel7.setToolTipText("Datos del Laboratorista");
            lbl_fechaentrega.setBounds(20,50,150,20);
            panel7.add(lbl_fechaentrega);
            lbl_nombrelaboratorista.setBounds(20,90,150,20);
```

```
            panel7.add(lbl_nombrelaboratorista);


            txt_fechaentrega.setBounds(160,50,140,20);
            txt_fechaentrega.setEditable(false);
            txt_fechaentrega.setBackground(fondo);
            txt_fechaentrega.setText(sdf.format(fecha));
            panel7.add(txt_fechaentrega);
            txt_fechaentrega.setToolTipText("Fecha entrega Ex");
            txt_nombrelaboratorista.setBounds(160,90,140,20);
            panel7.add(txt_nombrelaboratorista);
            txt_nombrelaboratorista.setToolTipText("Datos del
Tecnologo");


            panel7.setBackground(fondopanel);
            panel7.setBorder(new TitledBorder("Datos del
Laboratorista"));



//anadimos los botones de opciones

            //JPanel panel8=new JPanel();
            panel8.setLayout(null);
            panel8.setToolTipText("Area de Opciones");
            btn_consultar.setBounds(85,20,150,30);
            panel8.add(btn_consultar);
            btn_consultar.setMnemonic('C');
            btn_consultar.setToolTipText("Consultar si hay datos");
            btn_nuevo.setBounds(85,60,150,30);
            panel8.add(btn_nuevo);
            btn_nuevo.setMnemonic('N');
            btn_nuevo.setToolTipText("Cargar Nuevo Examen");
            btn_guardar.setBounds(85,100,150,30);
            panel8.add(btn_guardar);
            btn_guardar.setMnemonic('G');
            btn_guardar.setToolTipText("Guardar los datos ");
            btn_imprimir.setBounds(85,140,150,30);
            panel8.add(btn_imprimir);
            btn_imprimir.setMnemonic('I');
            btn_imprimir.setToolTipText("Imprimir resultados Ex");
            btn_salir.setBounds(85,180,150,30);
            panel8.add(btn_salir);
            btn_salir.setMnemonic('S');
            btn_salir.setToolTipText("Salir de la aplicacion");

            //btn_salir.setIcon(new ImageIcon("/imagen.jpg"));



            panel8.setBackground(fondopanel);
            panel8.setBorder(new TitledBorder("Botones de Opciones"));


      contenedor.add(panel);
      contenedor.add(panel1);
      contenedor.add(panel2);
      contenedor.add(panel3);
      contenedor.add(panel4);
```

```
contenedor.add(panel5);
contenedor.add(panel6);
contenedor.add(panel7);
contenedor.add(panel8);


invalidar();

}//fin del constructor de la aplicacion



ActionListener al = new ActionListener() {

//Escuchador de los eventos de los menus
public void actionPerformed(ActionEvent evt)
{

      if(evt.getSource()==nuevomenu)
            {

            validar();
            conectarBD();
            System.out.println("Se ha presionado el menu nuevo");
            dialogoinformacion();
            btn_nuevo.setEnabled(false);
            btn_guardar.setEnabled(true);
            nuevomenu.setEnabled(false);
            guardarmenu.setEnabled(true);

            }

      if(evt.getSource()==salirmenu)
            {
            dialogosalir();
            System.out.println("Se ha presionado el menu salir");
            System.out.println("saliendo...");

            }




            if(evt.getSource()==consultar_Ex)
            {
            consultarBD();
            btn_nuevo.setEnabled(true);
            nuevomenu.setEnabled(true);


            }

            if(evt.getSource()==imprimirmenu)
            {

            System.out.println("Se ha presionado el menu
imprimir");

            dialogoimprimir();
            limpiar();
```

```
                    nuevomenu.setEnabled(false);
                    btn_nuevo.setEnabled(false);
                    guardarmenu.setEnabled(false);
                    btn_guardar.setEnabled(false);
                    imprimirmenu.setEnabled(false);
                    btn_imprimir.setEnabled(false);
                    invalidar();

                    }


                    if(evt.getSource()==guardarmenu)
                    {



               if(txt_nombres.getText().length()==0 ||
txt_apellidos.getText().length()==0|| txt_id.getText().length()==0)
               {
               dialogofinExamen();
               invalidar();
               btn_guardar.setEnabled(false);
               guardarmenu.setEnabled(false);
               btn_nuevo.setEnabled(false);
               btn_consultar.setEnabled(true);

               btn_imprimir.setEnabled(false);
               imprimirmenu.setEnabled(false);
               nuevomenu.setEnabled(false);


                 }

                    boolean estado;

                    estado=comprobar_datos();
                    if(estado){
                    actualizar_estado();
                    btn_imprimir.setEnabled(true);
                    imprimirmenu.setEnabled(true);
                    }

                    }

                    if(evt.getSource()==acercademenu)
                    {

                    AcercaDe();


                    }
          }
};
 // fin de la funcion actionPerformed

 // fin del escuchador de los eventos de los menus

       public void actionPerformed(ActionEvent e)
       {                              //inicio del actionPerformed de
los botones
```

```
            if(e.getSource()==btn_nuevo)
                  {

                  validar();
                  conectarBD();
                  System.out.println("Se ha presionado el boton
nuevo");
                  dialogoinformacion();
                  btn_nuevo.setEnabled(false);
                  btn_guardar.setEnabled(true);
                  nuevomenu.setEnabled(false);
                  guardarmenu.setEnabled(true);
                  }


            if(e.getSource()==btn_salir)
                  {txtarea_mensajes.setText("NO TIENE EXAMENES
PENDIENTES");
                  dialogosalir();
                  }


            if(e.getSource()==btn_imprimir)
                  {
                  System.out.println("Se ha presionado el boton
imprimir");

                  dialogoimprimir();
                  limpiar();
                  nuevomenu.setEnabled(false);
                  btn_nuevo.setEnabled(false);
                  guardarmenu.setEnabled(false);
                  btn_guardar.setEnabled(false);
                  imprimirmenu.setEnabled(false);
                  btn_imprimir.setEnabled(false);
                  invalidar();


                  }


            if(e.getSource()==btn_guardar)
                  {


            if(txt_nombres.getText().length()==0 ||
txt_apellidos.getText().length()==0|| txt_id.getText().length()==0)
            {
            dialogofinExamen();
            invalidar();
            btn_guardar.setEnabled(false);
            guardarmenu.setEnabled(false);
            btn_nuevo.setEnabled(false);
            btn_consultar.setEnabled(true);
            btn_imprimir.setEnabled(false);
            imprimirmenu.setEnabled(false);
            nuevomenu.setEnabled(false);

              }
```

```
                boolean estado;
                estado=comprobar_datos();
                if(estado){
                actualizar_estado();
                btn_imprimir.setEnabled(true);
                }

                }

                if(e.getSource()==btn_consultar)
                {

                consultarBD();

                btn_nuevo.setEnabled(true);
                nuevomenu.setEnabled(true);

                }


          } // fin del action performed de los botones



     class Manejador extends WindowAdapter{
     public void windowClosing(WindowEvent e)
           {
                System.out.println("Saliendo");
                System.exit(0);
           }
     }/////fin windowAdapter




     public void conectarBD(){

           System.out.println("Entrando a la funcion conectarBD");

           String consultaid = "select  distinct min (id), id from
solicitud where estado='en proceso' and cod_area = 'Uroanalisis' group
by id";
           Connection conn = null;
           int var;
           try
             {


        conn =
DriverManager.getConnection("jdbc:postgresql://localhost/monografia","
usuario", "usuario123");
                System.out.println("Estableciendo la conexion con la
base de datos");

           if (conn != null)
                {
                System.out.println("Entrando en el if");
```

```
                        Statement stmt = conn.createStatement();
                        ResultSet res  = stmt.executeQuery(consultaid);

                  while(res.next())

                              {

                        System.out.println("Entrando en el while");

                        solicitud objeto = new solicitud();

                              objeto.sid  = res.getInt("id");
                        int pid;
                        pid=objeto.sid;

                        System.out.println("id:"+pid);


                        compruebaRango(pid);

                        res.close();
                        }

                        //res.close();
                        stmt.close();
                        conn.close();
                  }


            else
                  {
                  String mensaje = "No hay datos en los examenes"
+"\n";
                  System.out.println(mensaje);
                  txtarea_mensajes.append(mensaje);



                  }
            }
       catch(SQLException ex)
       {
          //System.out.println(ex);
            System.out.println("Cerraste inadecuadamente el resultset
del while");
       }

   }
//fin de la primera consulta conectarBD();




      public void consultarBD(){

            System.out.println("Entrando a la funcion consultarBD");
```

```
            String consultaid = "select   id from solicitud where
estado='en proceso' and cod_area = 'Uroanalisis'";
            Connection conn = null;
            int var = 0;
            try
              {

         conn =
DriverManager.getConnection("jdbc:postgresql://localhost/monografia","
usuario", "usuario123");
                  System.out.println("Estableciendo la conexion con la
base de datos");


                  if (conn != null)
                  {
                  System.out.println("Entrando en el if");

                        Statement stmt = conn.createStatement();
                        ResultSet res  = stmt.executeQuery(consultaid);

                     while(res.next())
                              {

                        System.out.println("Entrando en el while");

                        solicitud objeto = new solicitud();

                              objeto.sid  = res.getInt("id");
                        int pid;
                        pid=objeto.sid;

                        System.out.println("id:"+pid);

                        var++;
                        System.out.println("Tienes:"+var+ "\t Examen
pendiente"+" \n");
                        String aux = new String();
                        aux = "Examenes Pendientes:";
                        String concatena = new String();
                        concatena = String.valueOf(var);
                        aux.concat(concatena);
                        System.out.println("Valor de
concatena"+concatena);
                        txtarea_mensajes.setText(aux+concatena + "\n");

                        }

                        res.close();
                        stmt.close();
                        conn.close();
                  }

            }
       catch(SQLException ex)
       {
          System.out.println(ex);
      txtarea_mensajes.setText("No tiene examenes pendientes"+ "\n");
            //System.out.println("Error de Excepcion");
```

```
       }

   }
//fin de la primera consulta conectarBD();









            public void compruebaRango( int numero_id)
            {

            System.out.println("Entrando a la funcion compruebaRango");
            String query = "select id, nombres, apellidos, fecha from
solicitud where estado='en proceso' and  id = ?";
            Connection conn = null;
            int numeroderegistro =0;
            numeroderegistro = numero_id;


            try
              {


         conn =
DriverManager.getConnection("jdbc:postgresql://localhost/monografia","
usuario", "usuario123");

            if (conn != null)
                  {

                  PreparedStatement sentencia;
                  sentencia = conn.prepareStatement(query);

                  sentencia.setInt(1, numeroderegistro);


                  ResultSet resulset = sentencia.executeQuery();




                              while(resulset.next())
                              {

                        System.out.println("Entrando al while de
comprueba rango");
                        solicitud objeto = new solicitud();

                              objeto.sid  = resulset.getInt("id");
                              objeto.snombres =
resulset.getString("nombres");
                              objeto.sapellidos =
resulset.getString("apellidos");
                        int pid;
                        pid=objeto.sid;
                        String pnombres = new String();
```

```
                        pnombres=objeto.snombres;
                        String papellidos = new String();
                        papellidos=objeto.sapellidos;
                        objeto.fechasol = resulset.getDate("fecha");

                        txt_nombres.setText(pnombres);
                        txt_apellidos.setText(papellidos);
                        txt_id.setText(String.valueOf(pid));
                        ftf.setText(String.valueOf(objeto.fechasol));
                        ftf.setEditable(false);

                              System.out.println(pid
+"\t\t"+pnombres+"\t\t"+papellidos+"\t\t"+objeto.fechasol);

                        System.out.println("Se esta recorriendo la Base
de Datos Laboratorio");
                              }
                        txtarea_mensajes.append("Procesando nuevo
examen"+"\n");

                  sentencia.executeUpdate();
                        resulset.close();
                        sentencia.close();
                        conn.close();
                  }

            }
         catch(SQLException ex)
         {
           // System.out.println(ex);

         }

       }




      public void actualizar_estado()
            {


            String query = "update solicitud set estado = 'finalizado'
where id = ?";
            Connection conn = null;



                        try
                          {


                  conn =
DriverManager.getConnection("jdbc:postgresql://localhost/monografia","
usuario", "usuario123");

                        if (conn != null)
                              {
```

```
                        int  id_actualizar;


                        PreparedStatement sentencia;

                        sentencia = conn.prepareStatement(query);

                        id_actualizar=
Integer.parseInt(txt_id.getText());

                        sentencia.setInt(1, id_actualizar);

                        ResultSet res = sentencia.executeQuery();



                        sentencia.executeUpdate();


                        res.close();
                              sentencia.close();




                  }

                        conn.close();

             }
          catch(SQLException ex)
          {
             System.out.println(ex);

               String mensaje = "Se ha actualizado el registro"+ "\n";
               System.out.println(mensaje);
               txtarea_mensajes.append(mensaje);


          }

               }









        public void invalidar()
        {

        txt_nombres.setEditable(false);
        txt_apellidos.setEditable(false);

        txt_id.setEditable(false);

        txt_color.setEditable(false);
```

```
txt_aspecto.setEditable(false);
txt_sedimento.setEditable(false);
txt_densidad.setEditable(false);

txt_proteinas.setEditable(false);
txt_hemoglobina.setEditable(false);
txt_cuerposcetonicos.setEditable(false);
txt_ph.setEditable(false);
txt_urobilinogeno.setEditable(false);
txt_glucosa.setEditable(false);
txt_bilirrubinas.setEditable(false);
txt_nitritos.setEditable(false);

txt_celulasepiteliales.setEditable(false);
txt_leucocitos.setEditable(false);
txt_eritrocitos.setEditable(false);
txt_cilindros.setEditable(false);
txt_cristales.setEditable(false);
txt_otros.setEditable(false);

txtarea_cualitativo.setEditable(false);
txtarea_cuantitativo.setEditable(false);


txt_nombrelaboratorista.setEditable(false);

btn_nuevo.setEnabled(false);
btn_guardar.setEnabled(false);
btn_imprimir.setEnabled(false);

nuevomenu.setEnabled(false);
guardarmenu.setEnabled(false);
imprimirmenu.setEnabled(false);


      }



public void validar()
{
txt_nombres.setEditable(false);
txt_apellidos.setEditable(false);

txt_id.setEditable(false);

txt_color.setEditable(true);
txt_aspecto.setEditable(true);
txt_sedimento.setEditable(true);
txt_densidad.setEditable(true);

txt_proteinas.setEditable(true);
txt_hemoglobina.setEditable(true);
txt_cuerposcetonicos.setEditable(true);
txt_ph.setEditable(true);
txt_urobilinogeno.setEditable(true);
txt_glucosa.setEditable(true);
txt_bilirrubinas.setEditable(true);
txt_nitritos.setEditable(true);
```

```
txt_celulasepiteliales.setEditable(true);
txt_leucocitos.setEditable(true);
txt_eritrocitos.setEditable(true);
txt_cilindros.setEditable(true);
txt_cristales.setEditable(true);
txt_otros.setEditable(true);

txtarea_cualitativo.setEditable(true);
txtarea_cuantitativo.setEditable(true);


txt_nombrelaboratorista.setEditable(true);

txtarea_mensajes.setText("");
btn_nuevo.setEnabled(true);
btn_guardar.setEnabled(false);
btn_imprimir.setEnabled(false);

}


public void limpiar(){

txt_nombres.setText("");
txt_apellidos.setText("");
txt_id.setText("");


txt_nombres.setEditable(false);
txt_apellidos.setEditable(false);

txt_id.setEditable(false);


txt_color.setText("");
txt_aspecto.setText("");
txt_sedimento.setText("");
txt_densidad.setText("");

txt_proteinas.setText("");
txt_hemoglobina.setText("");
txt_cuerposcetonicos.setText("");
txt_ph.setText("");
txt_urobilinogeno.setText("");
txt_glucosa.setText("");
txt_bilirrubinas.setText("");
txt_nitritos.setText("");

txt_celulasepiteliales.setText("");
txt_leucocitos.setText("");
txt_eritrocitos.setText("");
txt_cilindros.setText("");
txt_cristales.setText("");
txt_otros.setText("");

txtarea_cualitativo.setText("");
txtarea_cuantitativo.setText("");

txt_nombrelaboratorista.setEditable(true);
```

```
      txt_nombres.requestFocus();
      }



      public boolean comprobar_datos()
      {


        if(txt_color.getText().length()==0 ||
txt_aspecto.getText().length()==0 ||
txt_sedimento.getText().length()==0 ||
txt_densidad.getText().length()==0)
            {
            btn_guardar.setEnabled(true);
            guardarmenu.setEnabled(true);

            dialogoErrorDatos();
            return false;
            }

            else{
            dialogoDatosGuardados();
            btn_guardar.setEnabled(false);
            guardarmenu.setEnabled(false);
            return true;
                }

      }


      public void dialogosalir()
      {
      int res = JOptionPane.showConfirmDialog( this,"Realmente desea
salir?","Salir de LabCli",JOptionPane.INFORMATION_MESSAGE);
                  String respuesta = null;

                        if( res == JOptionPane.YES_OPTION ) {
                                    respuesta = "Si";
                              System.out.println("Saliendo de los
dialogos y de la aplicacion");
                                    System.exit(0);

                                          }
                         else
                              respuesta = "No";
                              System.out.println( "Respuesta:
"+respuesta );

                  System.out.println("Se ha presionado el boton
salir");
                  System.out.println("saliendo...");


      }



      public void dialogoimprimir()
```

```
      {

            Uroanalisis miFrame = new Uroanalisis();



                  PrintJob miPrintJob =
miFrame.getToolkit().getPrintJob( miFrame,"Imprimiendo los datos de
los examenes",null );

                        if( miPrintJob != null ) {
                        // Obtenemos el objeto grÃ¡fico que va a
imprimir

                        Graphics graficoImpresion =
miPrintJob.getGraphics();

                              graficoImpresion.setFont( new Font(
"Arial",Font.BOLD,14 ) );

                             if( graficoImpresion != null ) {

                                   panel.printAll(graficoImpresion);
                                   panel1.printAll(graficoImpresion);
                                   panel3.printAll(graficoImpresion);
                                   panel4.printAll(graficoImpresion);
                                   panel5.printAll(graficoImpresion);
                                   panel6.printAll(graficoImpresion);

                                   panel7.printAll(graficoImpresion);


                                         //miFrame.printAll(
graficoImpresion );
                                   // Hacemos que se libere el papel
de la impresora y los
                                         // recursos del sistema que
estaba utilizando el
                                         // objeto grÃ¡fico
                                         graficoImpresion.dispose();

                                   }
                                   miPrintJob.end();

                             }
      }


      public void dialogoinformacion()
      {
      JOptionPane.showMessageDialog( this,"Nuevo dato
procesado","Agregando",JOptionPane.INFORMATION_MESSAGE);

      }




      public void dialogoerror()
      {
```

```
      JOptionPane.showMessageDialog( this,"Datos
Guardados","Guardar",JOptionPane.ERROR_MESSAGE);
      }









      public void dialogoErrorDatos()
      {
      JOptionPane.showMessageDialog( this,"Debe llenar correctamente
los Datos","Error",JOptionPane.ERROR_MESSAGE);
      }




      public void dialogoDatosGuardados()
      {
      JOptionPane.showMessageDialog( this,"Datos Guardados
Correctamente","Guardar",JOptionPane.INFORMATION_MESSAGE);
      }



      public void AcercaDe()
      {
      JOptionPane.showMessageDialog(panel,"LabCli HEODRA  Version 1.0
\n Desarrollado por \nJAVIER \nMELVIN                \nGABRIEL
\nLinux Debian + Java + Postgresql","Acerca
de",JOptionPane.OK_OPTION);
      }



      public void dialogofinExamen()
      {
      JOptionPane.showMessageDialog( this,"No hay mas Examenes por
procesar","Error",JOptionPane.ERROR_MESSAGE);
      txtarea_mensajes.setText("NO TIENE EXAMENES PENDIENTES");
      }



   }
// fin de la interfaz
```

ESTRUTURA FISICA DE LA RED EN EL LABORATORIO