UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA

UNAN-León

FACULTAD DE CIENCIAS

DEPARTAMENTO DE COMPUTACIÓN

TESIS PARA OPTAR AL TÍTULO DE INGENIERO EN SISTEMAS DE INFORMACIÓN

TEMA:

"AUTOMATIZACIÓN PARA EL CONTROL DE INVENTARIO EN EL ÁREA DE BODEGA DEL LABORATORIO CLÍNICO DEL HEODRA."

Presentado por:

- Douglas Iván Martínez Hernández.
- William Alberto Leyton Zapata.

Tutor

Lic. Rina del Pilar Arauz.

Diciembre 2006.

León, Nicaragua.

AGRADECIMIENTO

Queremos expresar nuestro agradecimiento:

*A **Dios** por habernos dado sabiduría y fortaleza para superar los momentos más difíciles de nuestra vida y lograr alcanzar nuestras metas.*

A nuestros padres por su apoyo incondicional que nos brindaron para poder culminar nuestra meta.

A todas aquellas personas que colaboraron de una u otra forma en la realización de esta investigación.

*A nuestra tutora, **Lic. Rina del Pilar Arauz** quien nos dirigió en nuestro trabajo investigativo.*

*A la **Lic. Rosa Adelina Alonso** responsable del laboratorio clínico del HEODRA, quien nos facilito la información necesaria para la elaboración de este proyecto.*

William Leyton Zapata

Douglas Iván Martínez

DEDICATORIA

Dedico este Trabajo, fruto de una ardua labor:

A Dios Todopoderoso por brindarme la vida, la sabiduría y el conocimiento necesario para preparar mi misión en la vida y alcanzar la meta de culminar mi carrera.

A mi madre Carmen Hernández Hurtado, regalo de Dios para mi vida y quien me ha brindado todo su amor y apoyo incondicional en mis triunfos y dificultades que he tenido a lo largo de mi vida.

A mis hermanas Fanny, Yessenia y Martha Lorena; quienes me motivaron para alcanzar mis metas y sueños.

Douglas Iván Martínez Hernández

DEDICATORIA

Dedico este mi tesis promeramenter:

A Dios, el señor Jesucristo por brindarme la sabiduría y el conocimiento necesario para alcanzar la meta de culminar mi carrera.

A mis padres, Felipe Leyton y Corina Zapata quienes me han dado su amor y apoyo incondicional en las dificultades que he tenido a lo largo de mi vida.

A mis hermanos Samuel y Yorlene; por su apoyo y comprensión en mi vida.

A mi abuelita Mesalina Leyton y a mi tio Pedro Leyton por darme su apoyo incondicional.

William Alberto Leyton Zapata.

INDICE

Contenido **Pagina**

1. INTRODUCCION

En la actualidad son muchas las necesidades de automatización de sistemas para el control de la información, como estudiantes de la carrera de Ingeniería en Sistemas de Información nos hemos propuesto desarrollar un Sistema donde pongamos en práctica los conocimientos adquiridos en los años de estudio de la carrera.

Los sistemas informáticos han transformado el mundo. Hoy todo gira en torno a los sistemas computarizados, agilizando y reduciendo el tiempo empleado en la realización de una tarea determinada.

Existe una gran variedad de sistemas. Desde sistemas contables, no contables, administrativos, gerenciales, de toma de decisiones, etc., utilizados por la industria y en las diferentes empresas.

Con lo dicho anteriormente, desarrollaremos un Sistema informático para Automatizar **EL CONTROL DE LA BODEGA DEL LABORATORIO CLINICO DEL HEODRA,** para tal efecto es necesario establecer los requisitos del software que debe cumplir el sistema, este llevara el control de la información referente a la bodega del laboratorio, los tipos de productos que están almacenados, un registro de las salidas y entradas de los productos. Así también facilitara las operaciones de solicitud a los proveedores de la bodega al proporcionar información sobre las existencias de insumos en bodega.

Una característica a destacar en nuestro proyecto es la utilización de Software Libre para el desarrollo del sistema ya que la necesidad de automatizar procesos, la existencia de una ley que protege los derechos de autor aunado con las condiciones económicas en la que nos encontramos actualmente, nos ha motivado de forma significativa a la utilización de Software libre como una alternativa eficiente que permitirá disminuir los costos de mantenimiento por el pago de licencia de software propietario (comerciales).

2. ANTECEDENTES

Actualmente el Laboratorio Clínico del **HEODRA** no cuenta con un sistema automatizado que le permita la facilitar el control de inventario en la bodega que surte a las diferentes secciones que forman el laboratorio clínico, información de vital importancia para el área administrativa para la toma de decisiones, ya que actualmente este proceso se realiza a mano.

Todo esto conlleva un gran esfuerzo y consumo de tiempo y el riesgo de cometer errores al momento de realizarlo.

Nuestro esfuerzo se enfoca en el desarrollo de una aplicación que contribuya en la agilización de los diferentes movimientos que se efectúan en el área de bodega del laboratorio clínico.

3. JUSTIFICACIÓN

En la actualidad las **PC's** han pasado a ser una herramienta necesaria y útil para las diferentes organizaciones tanto estatales como privadas y mas aun para las personas en el procesamiento de datos en cualquier área. La mayoría de las computadoras están dotadas de una serie de programas que le permiten al usuario el fácil manejo de la información, esto da como resultado un rotundo cambio en la forma de utilizar las **PC's**, ya que estas se pueden modelar o adecuar de acuerdo a nuestras necesidades. El proyecto que llevamos acabo servirá para adquirir nuevos conocimientos, poner en practica los conocimientos adquiridos durante estos 4 años y facilitar el manejo de la información.

4. OBJETIVOS

OBJETIVO GENERAL:

- Desarrollar una aplicación para automatizar el control de inventario de la bodega del laboratorio clínico **HEODRA**, para alcanzar nuestro fin haremos uso de las siguientes herramientas PostgreSQL como gestor de Base de Datos y como lenguaje de programación Java.

OBJETIVOS ESPECIFICOS:

- Actualización del inventario de Bodega de forma permanente.
- Agilizar el proceso de solicitud de insumos a los proveedores de la bodega.
- Proporcionar una interfaz grafica para el registro de las entradas y salidas que se realice en la Bodega.
- Generara una Requisa por cada petición de productos que se realice a la Bodega.
- Generara reportes de las entradas y salidas que se efectúen.

5. MARCO TEORICO.

5.1. Inventario: Algunos empresarios se dejan llevar por un único criterio, que en ocasiones funciona perfectamente, pero en otras nos lleva a cometer costosos errores; las personas que llevan tiempo manejando empresas saben que la realidad no es blanca ni negra, sino que, tiene diferentes matices de gris, por lo que en este escrito no se pretende dar la última palabra de nada, ya que esta no existe, lo que se busca es brindar instrumentos que aunados a su buen juicio le permitan tomar mejores decisiones, recalcando que un instrumento es útil dependiendo de la mano que lo maneja y que la toma de decisiones es aún la responsabilidad gerencial más importante. Sin embargo, a medida que logremos conocer y manejar un mayor número de instrumentos la probabilidad de éxito en nuestras empresas se incrementa, ya que la suerte esta del lado de la mente mejor preparada (esta frase se le atribuye a diversos personajes, pero independientemente de quien la haya dicho es perfectamente valida).

En el caso de los inventarios, se han escrito enciclopedias completas sobre su manejo, por lo que resumir en tan breve espacio, algunos de los criterios fundamentales para su manejo, equivale a lo que los expertos en mercadeo llaman dar una degustación del producto, buscando de esta manera, que los interesados indaguen más adelante, formas de adquirir una mayor cantidad de este. Los modelos que se van a mencionar son relativamente sencillos y se recomienda que en la medida de lo factible se "juegue" con ellos, en una hoja de calculo, creando diferentes escenarios, con el objeto de ver las consecuencias de algunos posibles errores en el papel, sin necesidad de sufrirlos en la realidad.

EJEMPLO MOTIVADOR

Hace unos años era frecuente encontrar en las tiendas, la caricatura del "flaco arruinado por vender a crédito, al lado del gordo opulento que vendía de contado"; este manejo maniqueo de la venta, algunas personas lo trasladan a los inventarios, donde pueden existir múltiples posiciones y es frecuente observar como algunos empresarios defienden los extremos, siendo estos en ocasiones contraproducentes a los intereses de la organización.

Así como en las ventas, los empresarios y aun las tiendas y los supermercados, han debido incluir diferentes esquemas de comercialización, donde se mezclan posibilidades de crédito, con descuentos por pago de contado, como única opción de supervivencia; los empresarios en general, deben buscar herramientas para la administración de los inventarios, que les permitan manejar estos desde diversos ángulos y estar preparados a reconsiderar en cualquier momento su uso, ya que, pretender que existe hoy, una única y mejor forma de resolver algo, es síntoma de terquedad y puede llevar a la empresa a complicaciones importantes y aun a la quiebra, en el mejor de los casos puede dar la apariencia de un "falso triunfo", que con seguridad más adelante se convertirá en una derrota no percibida.

En la actualidad no es extraño encontrar personas, que afirman que: "se deben manejar grandes cantidades de inventarios, ya que esto significa riqueza y que es preferible guardar en insumos o productos la plata, ya que con las alzas ocasionadas por la inflación y/o la devaluación, se consigue una mayor rentabilidad que en otras opciones". En el otro extremo se observan las personas que afirman que: "los inventarios se deben reducir a cero y manejar una política de justo a tiempo, donde los inventarios son un problema que se genera por ineficiencia gerencial"; lo que nos hace recordar el ejemplo inicial, con un agravante, en esta situación en los dos extremos se puede estar mal, ya que alguien puede tener inventarios en exceso y no tener como pagar la nómina, mientras que en el otro extremo estaría quien ha tenido que retrasar su proceso de producción por no contar en forma oportuna con los elementos requeridos para el efecto.

MODELOS PARA LA TOMA DE DECISIONES

Existen diferentes esquemas en la creación de modelos para la toma de decisiones, en este caso mencionaremos uno de los esquemas que se aplica en la ciencia de la administración, el cual, no pretende ser la verdad revelada, esto es válido para problemas de inventarios o para cualquier situación empresarial. Es fundamental enfatizar en que no debe verse como un conjunto rígido de procedimientos, en algunos casos se podrán saltar escalones, en otros tocará crear variantes, sin embargo, es importante manejar una guía que nos va a facilitar los procesos y coadyuvará a una gestión más eficiente, los pasos a seguir son los siguientes:

- Aceptar o reconocer la existencia de una(s) situación(es) susceptible(s) a mejorar, lo que es diferente a un problema manifiesto, lo importante es reconocer que de actuar diferente se podrían obtener mejores resultados; en este paso se debe ser muy cuidadoso, ya que, una persona puede tener diversos objetivos; también se presenta que diferentes áreas de la organización tengan objetivos contradictorios, por ejemplo el departamento comercial puede pretender resultados que entren en contradicción con los objetivos de quien maneja el departamento financiero, en el caso de los inventarios esto se presenta con frecuencia, ya que, para un gerente financiero los inventarios pueden representar lucro cesante, mientras que para el comercial pueden representar la forma de brindar un mejor servicio, por lo que lo más importante es fijar adecuadamente el(los) objetivo(s) a conseguir.
- Formular el problema, plantear la situación(es) ya definida(s), jerarquizándolas en caso de ser necesario en forma de problema, buscando solucionar primero las más críticas y/o las más viables a juicio de la gerencia. Este paso es delicado ya que un alto porcentaje de la solución, esta dado por un buen planteamiento, un planteamiento erróneo puede llevar a consecuencias desastrosas para la organización; en el ejemplo anterior, si yo pienso en ahorrar unos pesos, en esta difícil época para los negocios, lo puedo hacer en un momento determinado sacrificando servicio, si embargo, esto se puede revertir en contra de mi empresa en el mediano o largo plazo.
- Construir el modelo, representar de alguna manera el problema, para nuestro caso, en forma matemática, siendo consientes de que el mejor modelo no necesariamente es el más novedoso, ni el más complicado, la bondad del modelo se mide por la forma en que este interpreta la realidad y por las ayudas que pueda brindar al tomador de decisiones.
- Recolectar los datos, esta es una de las partes más dispendiosas y en donde es necesario ayuda de todas las áreas de la organización, en el caso de los inventarios debemos conocer los costos de mantenimiento y los de pedido, la demanda, el tiempo de entrega de nuestros proveedores, en fin la información que necesitamos es muy amplia y de su calidad depende la validez del modelo, ya que parodiando a

los ingenieros de sistemas, si basura entra, basura sale. Si lo anterior es cierto cuando se "alimenta" un computador con mayor razón se cumple en los modelos cuantitativos.

- Resolver el modelo, esta es realmente la parte más sencilla, si los pasos anteriores son correctamente efectuados, consiste en operar con los datos recolectados en el modelo previamente construido, en esta parte algunos autores recomiendan manejar escenarios, es decir hacer fluctuar los datos recopilados para analizar las posibles influencias de estos cambios y estar pendientes en caso que se presenten, otros recomiendan, de ser posible ver como se comporta el modelo en situaciones que ya sucedieron, con el objeto de verificar hasta que punto nos hubiesen ayudado a tomar la mejor decisión de acuerdo a las circunstancias.
- Interpretación de los resultados, en esta parte el criterio gerencial es fundamental, ya que, **NADA** puede remplazar totalmente el juicio de quien toma la decisión, la idea del modelo es ayudar, no remplazar al tomador de decisiones, aquí es indispensable que quien toma las decisiones se despoje de pre-juicios y vea la solución de la forma más objetiva posible, que no piense que al proponer el modelo soluciones diferentes a las que ha venido implementando, éste o las decisiones anteriores no sirven. El verdadero profesional, puede darse a partir de la consecución de un titulo universitario y/o por la experiencia y el estudio relacionados en el campo objeto de su labor, lo que realmente lo distingue es su capacidad de análisis y de evolucionar en y con el tiempo.
- Implementar el modelo, si los pasos anteriores fueron manejados correctamente, este debería ser el más sencillo, sin embargo, en la practica presenta ocasionalmente problemas, por el rechazo de algunas partes de la organización a los cambios, debido a que en todos los niveles de la organización es frecuente observar personas que por no salirse de rutinas preestablecidas, se oponen y retardan la puesta en practica de nuevas ideas.
- Retroalimentar y recrear el modelo, teniendo en cuenta que las circunstancias del entorno o de la empresa pueden cambiar, es indispensable estar permanentemente verificando el modelo, lo cual no le quita validez y muestra seriedad y objetividad en la toma de decisiones, recuerde que ni aun "a los paquetes computacionales más

complejos se les puede dejar decidir", que siempre el que toma las decisiones es el gestor, esto es válido aun más en lo referente a modelos de gestión.

En esta parte se sugirió crear el modelo en el sentido amplio de la palabra, no obstante, es importante recalcar, que en lo referente a inventarios existen gran cantidad de éstos, por lo que generalmente no es necesario "descubrir el agua tibia", lo importante es saber cuando y donde usarla. Los modelos deben permitirnos cometer errores en el papel, disminuyendo la probabilidad de cometerlos en la vida real, las aplicaciones en computador nos permiten ver rápidamente muchas opciones simplemente cambiando números de acuerdo a los supuestos razonados que usemos dentro de las situaciones que se pueden presentar en la empresa.

En el libro **INVESTIGACIÓN DE OPERACIONES EN LA CIENCIA ADMINISTRATIVA** de Eppen se afirma: "Ni las dimensiones ni el refinamiento matemático del modelo implican necesariamente, por sí mismos, que éste será útil. Los modelos sencillos y pequeños también pueden tener gran utilidad si brindan ayuda para la toma de decisiones. Más aún, los grandes y sofisticados modelos que tienen éxito han evolucionado casi siempre a partir de un precursor rudimentario, merced al éxito en la experiencia administrativa"; lo cual es absolutamente valido para este tema.

ADMINISTRACIÓN DE INVENTARIOS

En los negocios existe una realidad reconocida por muchos, pero desafortunadamente racionalizada e implementada por pocos "quien compra bien, vende o produce bien". El tener una buena política de compras, le va a permitir un manejo fluido a la empresa y disminuir sus costos, lo que obviamente mejorará su rentabilidad. Debido a lo anterior es necesario estudiar los inventarios desde el momento en que se proyecta la compra, es decir involucrarlos en los procesos de planeación de la compañía y en su contrapartida obligatoria, el control.

En la acepción más amplia de la palabra, los inventarios son recursos utilizables que se encuentran almacenados para su uso posterior en un momento determinado.

Algunos autores los definen simplemente como bienes ociosos almacenados en espera de ser utilizados. Otros autores los definen como un activo corriente de vital importancia para el funcionamiento de la empresa. Existen múltiples argumentos para justificar la tenencia o no de inventarios, de los cuales mencionaremos tan solo unos pocos.

Argumentos a favor

- Prever escasez.
- Es preferible ahorrar productos que plata
- Permiten obtener ganancias adicionales cuando hay alzas
- Facilitan desfasar (separar) los diferentes procesos de la empresa.

Argumentos en contra:

- Inmovilizan recursos que podrían usarse mejor
- Esconden los problemas de la empresa
- Disimulan la ineptitud del tomador de decisiones
- Facilitan esconder los problemas de calidad.

Los argumentos esgrimidos por los "partidarios" de cada corriente tienen validez relativa, esto es lo que los hace tan peligrosos, ya que al tener indiscutiblemente una parte de realidad son aun más difíciles de rebatir que las verdades verdaderas como diría Fuentes, debido a lo anterior es que debemos ser objetivos en la posición a asumir y no ser maniqueos, es decir no debemos creer que nuestro argumento es acertado y que todos los demás están equivocados.

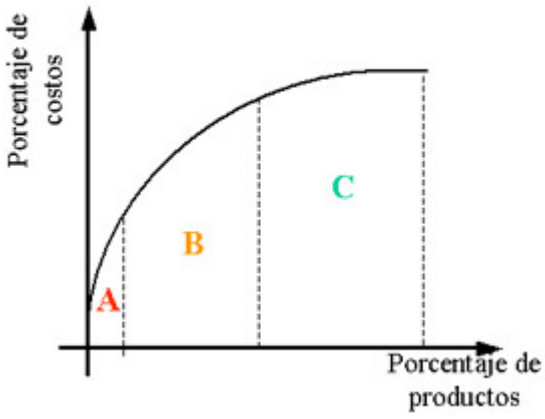
Lo que es indiscutible, es que los inventarios representan un alto porcentaje de los activos en el balance y a las compras les sucede lo mismo con respecto a las utilidades en los estados de resultados, entonces si desde el punto de vista financiero reconocemos esta realidad y no hacemos nada con el objeto de mejorar su manejo estamos siendo irresponsables con nuestra empresa.

CLASIFICACIÓN ABC

En cada empresa se utilizan diferentes productos, cada uno de ellos con sus propias características, por lo tanto, cada uno de ellos necesita de un manejo particular, dependiendo de su importancia en los procesos de la compañía y de las posibilidades de adquisición. El pensar que todos los productos se deben controlar de la misma manera, es una visión limitada de la realidad, que implica desgaste y sobrecostos innecesarios.

El análisis ABC es una manera de clasificar los productos de acuerdo a criterios preestablecidos, la mayor parte de los textos que manejan este tema, toman como criterio el valor de los inventarios y dan porcentajes relativamente arbitrarios para hacer esta clasificación. Por ejemplo, el 10% de los productos representan el 60% de las compras de la empresa por lo tanto esta es la zona A, un 40% de los productos el 30%, que serian los que están ubicados en la zona B, el resto (50% de los productos y 10% de las compras) son productos C.

Los valores anteriores son arbitrarios, cada empresa tiene sus particularidades, si alguien decide utilizar este criterio debe ser consciente de las realidades de su empresa. Se debe pensar no solo en los costos, es importante ver otros criterios, lo que es sin duda la principal dificultad en este tipo de análisis. Es innegable, sin embargo que un pequeño porcentaje de productos, desde cualquier criterio, es indispensable para el funcionamiento de la empresa y/o para mejorar su rentabilidad, estos serian clasificados como productos A típicos, y de acuerdo a este punto de vista se van seleccionando los productos de las demás zonas; si uno considera oportuno podría pensarse en la posibilidad de agregar una zona D, para productos realmente intrascendentes y de costo muy bajo.



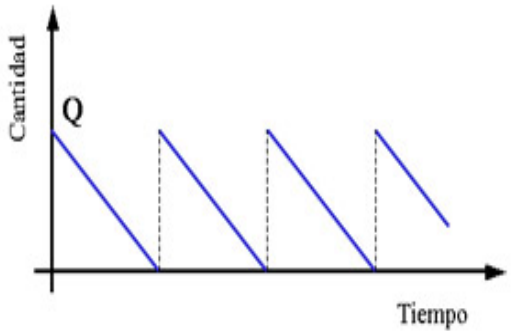
La siguiente gráfica nos da una visión de la clasificación ABC, no se utilizaron porcentajes en forma explícita, para no caer en la tentación de

dogmatizar sobre un valor en particular, la idea es que a los productos de la zona A se le busquen modelos que permitan un control muy fuerte sobre el criterio clave que se esté manejando y a medida que se alejen los productos de esta zona, los modelos puedan ser más flexibles; esto no quiere decir que se descuide el control físico de los inventarios, ya que como se mencionó en la introducción ese no es el objetivo del presente fascículo.

MODELO DE CANTIDAD ECONÓMICA DE PEDIDO.

Este modelo parte de una serie de supuestos fuertes, los cuales se van suavizando a medida que se avanza en la teoría, sin embargo sus aplicaciones y utilidad son importantes y los desarrollos posteriores que ha permitido, lo hacen un punto de referencia obligado en todos los campos donde se hable de inventarios. Por eso no es extraño encontrar menciones a este modelo en múltiples libros de costos, de administración de operaciones, de logística, de cálculo y de otros temas. Los supuestos sobre los que este modelo se construye son:

1. La demanda se conoce con certidumbre y es constante.
2. Los costos relacionados con el modelo permanecen constantes.
3. La cantidad de pedido por orden es la misma.
4. El pedido se recibe en el momento que se ordena.
5. El inventario se restablece en el momento en que se agota.
6. El proveedor nos surte las cantidades solicitadas en un solo lote.
7. Se considera un horizonte infinito y continuo en el tiempo.



El comportamiento de este modelo se aprecia fácilmente en la siguiente gráfica

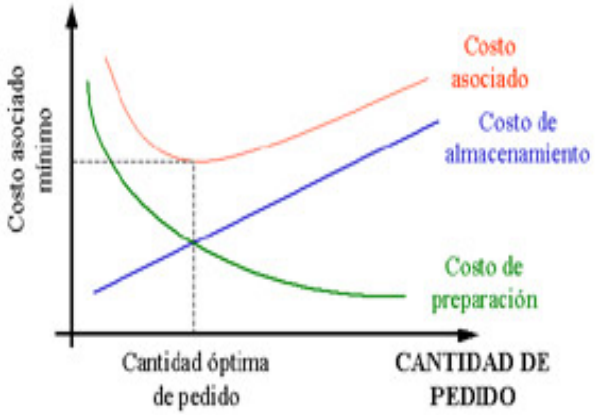
Para poder tomar una decisión sobre: la altura del triángulo (cantidad de pedido), el número de triángulos (números de pedidos en el periodo), la base del triángulo (tiempo entre pedidos) y conocer el valor asociado con estas decisiones es necesarios conocer los siguientes datos:

Demanda, normalmente se trabaja anual, aunque el modelo permite otros manejos, se calcula a partir de los presupuestos de la empresa.

Costo de pedido, este se genera cada vez que la compañía efectúa una compra, en su calculo debe involucrarse desde el tiempo que se toma para efectuar el pedido, hasta los gastos de transporte y recepción de la mercancía, sin olvidar incluir los gastos administrativos pertinentes al pago de la factura.

Costo de mantenimiento (conservación), este nos indica cuanto vale tener la unidad de inventario en bodega, debe tenerse en cuenta desde el costo del dinero, hasta los seguros en caso de tenerlos, el de la bodega y el del personal que maneja los inventarios, este costo se debe dar en la misma unidad de tiempo en que se estima la demanda.

La parte compleja del modelo es precisamente la definición de los costos anteriores, si se calculan objetivamente el modelo da unos resultados válidos así no sean absolutamente exactos, el objetivo del modelo no es minimizar uno de estos costos, ya que su comportamiento es inverso y en caso de minimizar uno solo de ellos, el otro se dispara por lo que los costos asociados serán más altos, lo importante es minimizar la suma de los costos de pedir y de mantener , lo que se conoce con el nombre de costo asociado, en la siguiente gráfica observamos como dicho costo en los valores cercanos al mínimo, no cambia considerablemente, sin embargo si nos alejamos de este los costos pueden incrementarse de forma importante, por lo que la idea consiste en pedir un valor muy cercano a la cantidad económica de pedido.



La simbología que se va a utilizar es una de las tantas existentes, en caso de que se consulte a alguno de los autores citados o a otros es posible encontrar símbolos diferentes, esto no es problema lo importante es tener claros los elementos conceptuales.

D: Demanda

Co: Costo de pedido

Cc: Costo de conservación

Q*: Cantidad económica de pedido

N : Número de pedidos

Tc: Tiempo entre pedidos

CA: Costo asociado a la política de inventarios

CT: Costo total, involucra valor de los artículos y el costo asociado.

Calculando las primeras tres variables los demás valores quedan automáticamente dados, la demostración del porque se utilizan las formulas siguientes proviene del cálculo diferencial:

$$Q^* = \sqrt{\frac{2 \times D \times C_0}{C_C}}$$

$$N = \sqrt{\frac{D \times C_C}{2 \times C_O}} = \frac{D}{Q}$$

$$T_C = \frac{1}{N} \times \text{Número de días hábiles del periodo}$$

$$CA = \sqrt{2 \times D \times C_0 \times C_C}$$

$$CT = D \times C + \frac{D}{Q} \times C_0 + \frac{Q}{2} \times C_C$$

Ejemplo

Un impresor que en la actualidad esta haciendo una compra mensual, estudio el comportamiento del papel libro de 70 gr. en los últimos doce meses, encontró que su demanda fue de: 10, 11, 10, 9, 10, 11, 9, 10.5, 10, 9, 9 y 11.5 toneladas por mes, estima el precio de compra se va a mantener en $2.300.000 por tonelada, su costo de pedido en $500.000 y por política carga un 15% del costo unitario al manejo de los inventarios mas $55.000 por concepto de bodegaje, calcular:

1. El modelo a manejar en estas condiciones.
2. Si el proveedor ofrece darnos un descuento del 10% por compras superiores a 30 toneladas y uno del 11% por compras de 60 toneladas, como cambiaría mi política.
3. Si adicional al descuento logramos obtener un plazo que hace que nuestro costo de conservación se reduzca solamente al de bodegaje como cambiaría mi política.

Lo primero que debemos observar es el comportamiento de la demanda el cual vemos que es relativamente constante, por lo que podemos asumir que nuestro modelo se comporta de acuerdo a los parámetros de un modelo de cantidad económica de pedido con los siguientes datos de entrada:

D = 120 toneladas año
Co = $500.000
C = $2.300.000 tonelada
Cc = $400.000 tonelada/año
Por tanto

$$Q = \sqrt{\frac{2 \times 120 \times 500.000}{400.000}} = 17.32 \text{ Toneladas por pedido}$$

$$N = \frac{120}{17.32} = 6.93 \text{ Pedidos en el año}$$

$$Tc = \frac{1}{6.93} = 51.95 \text{ Días entre pedidos (asumiendo año de 360 días)}$$

$$CA = \sqrt{2 \times 120 \times 500.000 \times 400.000} = \$6.928.203$$

$$CT = 120 \times 2.300.000 + 6.928.203 = \$282.928.203$$

Como podemos observar en esta política de compra de inventarios, la empresa ahorra más de un 20% en el costo asociado a los inventarios que tendría si efectuase una compra mensual (CA = 12*500.000 + [12/2]*400.000 = $8.500.000), lo que sumado al ahorro que

se lograría con los diferentes productos que maneja la compañía permitirá mejoras importantes en la rentabilidad al final del ejercicio.

En las decisiones administrativas el criterio del experto es insustituible, sin embargo un buen manejo de los instrumentos cuantitativos facilita de manera considerable su labor, permitiéndole cometer errores en el papel, con lo que la rentabilidad de la compañía debe mejorar considerablemente, en el ejemplo del fascículo jugamos tan solo con dos posibles variaciones, se pueden manejar diferentes opciones entre las que podrían estar, como afirma BONINI afiliarse a clubes de compradores con el fin de obtener mejores condiciones de negociación.

5.2. Sistema: Es una combinación de medios (como personas, materiales, equipos, software, instalaciones, datos, etc.), integrados de forma tal que puedan desarrollar una determinada función en respuesta a una necesidad concreta". También se puede definir como un "Conjunto de elementos interrelacionados entre sí, de forma tal que, un elemento afecta el comportamiento de todo el conjunto".

5.2.1. 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**.

5.2.2. 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**.

5.3. Interfaces: Son el resultado de la programación que nos permite definir un conjunto de miembros que describen una determinada funcionalidad. Lo importante es que tan sólo la describen, pero no proporcionan implementación alguna de los mismos ya que ésta es una tarea de las clases de objetos, que son las que pueden contener código. Las interfaces se dividen en estáticas que no tienen cambios y son difíciles de modificar y las interfaces dinámicas que cambian de acuerdo a los requerimientos establecidos. Normalmente las interfaces de usuarios son utilizadas para proporcionar información o resultados que genera un sistema de cómputo.

5.4. Interfaz de usuario: Es uno de los componentes más importantes de cualquier sistema computacional, pues funciona como el vínculo entre el humano y la máquina, también es un conjunto de protocolos y técnicas para el intercambio de información entre

una aplicación computacional y el usuario, es responsable de solicitar comandos al usuario, y de desplegar los resultados de la aplicación de una manera comprensible.

5.5. 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.

5.6. 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.

5.7. Elementos de las Interfaces del sistema: En un interfaz intervienen 3 elementos importantes de los que al interaccionarse entre ellos se crea una comunicación:

- ✓ Humano
- ✓ Computadora
- ✓ Interacción

5.8. Base de Datos: es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su uso posterior. En informática existen los sistemas gestores de bases de datos (SGBD), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada.

5.9. Software Libre: se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. De modo más preciso, se refiere a cuatro libertades de los usuarios del software:

- ✓ La libertad de usar el programa, con cualquier propósito (libertad 0).
- ✓ La libertad de estudiar cómo funciona el programa, y adaptarlo a tus necesidades (libertad 1). El acceso al código fuente es una condición previa para esto.
- ✓ La libertad de distribuir copias (libertad 2).
- ✓ La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. (libertad 3). El acceso al código fuente es un requisito previo para esto.

Un programa es software libre si los usuarios tienen todas estas libertades. Así pues, deberías tener la libertad de distribuir copias, con o sin modificaciones, sea gratis o cobrando una cantidad por la distribución, a cualquiera y a cualquier lugar. El ser libre de hacer esto significa (entre otras cosas) que no tienes que pedir o pagar permisos.

5.10. Software Y Herramientas Para La Implementación Del Sistema

5.10.1. 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 hache, 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.

5.10.2. 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.

5.10.3. Características 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 diseñada 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.

5.10.4. Máquina 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.

Portabilidad

5.11. 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 aplicación 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 obtenido 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 escribiendo "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:

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 -version

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 deberán revisar cada una de las instrucciones y vuelves a probar, es necesario que este paso esté correcto para poder comenzar a compilar y ejecutar programas java.

5.12. 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, es el driver JDBC oficial**.

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), teléfono 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 sección de descargas escogemos el que mas se adecue a nuestra versión de jdk (este archivo tendrá una extensión .jar) Es necesario que este archivo este incluido en la variable de ambiente CLASSPATH.
export CLASSPATH=/home/usuario/driver/postgresql-8.1-407.jdbc2.jar:., esta línea puede ser agregada al archivo .bash_rc para que se reconozca la ruta del driver a través de la consola de Linux.

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();
```

5.13. 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();
```

5.14. 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 commando Del sistema operative y los datos obtenidos son mostrados en la consola Del sistema**********/

```
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);
    }
  }
}
```

5.15. PostgreSQL.

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, funciones, 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.

5.16. 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 derivada 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 lenguaje de consultas Postqre 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.

5.17. 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 cadenas 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/**.

5.18. 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 “usario” 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 están 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

5.19. Autentificación del cliente

Es una característica central para postgres, sin ella podría sacrificar la conectividad remota o por el contrario podría permitir que cualquiera pudiera conectar a su base de datos y recibir, incluso modificar sus datos. Postgres tiene varios tipos de autentificación de cliente a su disposición, los accesos basados en maquina (host) se encuentran especificados en el archivo pg_hba.conf, este tipo de autentificación es flexible, proporcionando una amplia variedad de opciones de configuración, 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 petición especificara un nombre de usuario postgres y la base de datos a la cual intenta conectar. Opcionalmente, puede ser proporcionada una contraseña.

5.19. ¿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.

DATA VISIÓN

Es una herramienta generador de reporte de **Open Source** silimar a **Crystal Reports**. Este puede correr, mostrarse y imprimirse en la aplicación o exportarse con diferentes tipos de formatos**: HTML, XML, PDF, LaTeX2e, Doc**.

DataVision esta escrito en JAVA, este generador de reportes de Base de Datos trabaja perfectamente con JDBC y se conecta fácilmente gestores de base de datos como: Oracle, PostgreSQL, MySQL, Informix, hsqldb, Microsoft Access, Progress, entre otros.

- Las nuevas versiones de DataVision requieren ahora de JAVA 1.4.x.

6. DISEÑO METODOLOGICO

La ingeniería del software se centra en el hecho, establecimiento y uso de los principios robustos de la ingeniería. Orientados a obtener software económicos y fiables para que funcionen de manera eficiente en máquinas reales. Este proyecto consta de herramientas de alta calidad permiten mayor rapidez en el procesamiento de los datos.

La ingeniería del software está formada por 3 elementos claves:

- ✓ Métodos.
- ✓ Herramientas.
- ✓ Procedimientos.

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 encargado del manejo del área de bodega.
- ✓ 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

6.1 Materiales

a. Hardware: Para la realización de este proyecto se hizo uso de las siguientes herramientas:

- 1 PC con las siguientes características:
  - ✓ Memoria RAM de 256 MB
  - ✓ 15 GB en Disco Duro
  - ✓ Procesador AMD K6
  - ✓ Velocidad de procesamiento 500Mhz

b. Software: Las herramientas software que emplearemos son:

- ✓ Open Office
- ✓ Easy Case Proffesional Versión 4.21.016
- ✓ Linux Debian (Sistema Operativo)
- ✓ PostgreSQL 8.1.3 (Sistema Gestor de Base de Datos)
- ✓ Kwrite (Editor de código)
- ✓ JVM 1.4.2_11
- ✓ DataVision 1.0.0
- ✓ Driver Postgresql -8.0-315.jdbc2.jar

6.2 Métodos

Indican cómo construir técnicamente el software con la ayuda de diferentes tipos de modelos (estructurado, cascada, espiral, etc.) los cuales proporcionan una serie de pasos que se deben seguir durante el proceso de desarrollo.

Los modelos del ciclo de vida realizan actividades comunes como:

- ✓ Gestión del proyecto por medio de la descomposición del mismo en etapas.
- ✓ Uso de alguna metodología de trabajo en cada etapa.
- ✓ Verificación y validación de cada fase de desarrollo.

6.3. Descripción del ciclo de vida

El ciclo de vida exige un enfoque sistemático y secuencial acerca del desarrollo del software que indica con el nivel del sistema y progresa a través del análisis, diseño, codificación, prueba y mantenimiento.

Los modelos de desarrollo abordan el estudio de las actividades del sistema en las siguientes etapas:

- Planificación.
- Análisis.
- Diseño del sistema.
- Construcción y elaboración del sistema.

El modelo de desarrollo de sistemas que será implementado en nuestro sistema es el **MODELO EN CASCADA**.

6.4. Modelo en Cascada

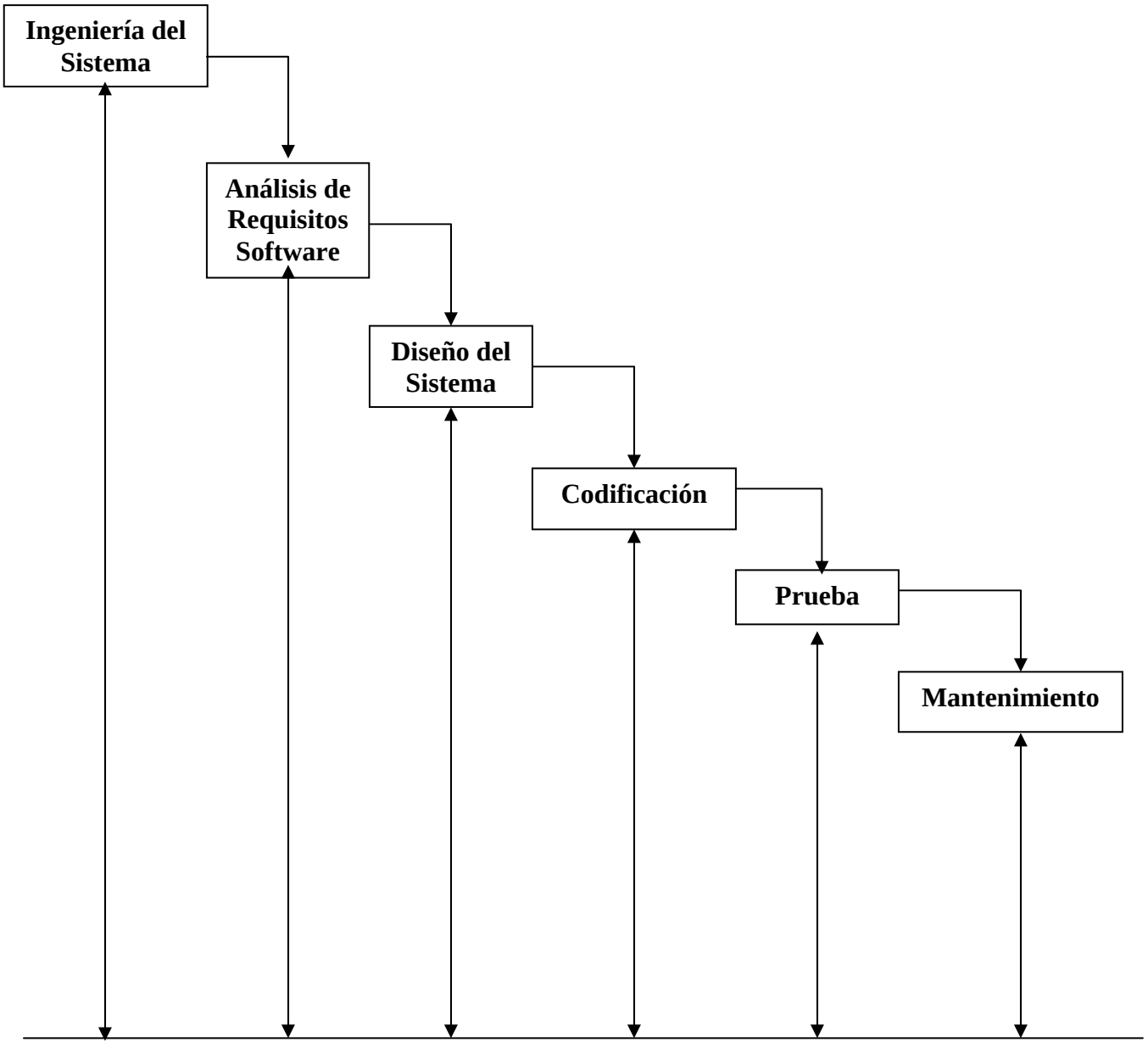
Descompone el proceso de desarrollo en diferentes fases, constituyendo la salida de cada una de ellas la entrada requerida por la siguiente. En este modelo se supone que todos los requisitos son conocidos y comprendidos perfectamente al iniciar el desarrollo del software.

6.5. Actividades del Ciclo de Vida en Cascada

- ✓ **Ingeniería del sistema:** estudio de sistemas para obtener una convicción del análisis y diseño. Abarca el cómo trabaja un sistema existente y preparar modificaciones al sistema para cambiar el comportamiento del mismo.
- ✓ **Análisis:** Análisis del ERS, especificación de tareas.
- ✓ **Diseño:** Definición de las estructuras de datos, arquitectura e interfaces.
- ✓ **Codificación:** Conversión del diseño a un lenguaje de programación.
- ✓ **Prueba:** El software debe ser probado para descubrir los defectos que puedan existir en la función, en la lógica y en la implementación.

- ✓ **Mantenimiento:** La fase de mantenimiento se centra en el cambio. Esta fase aplica los pasos de las fases de definición y de desarrollo, pero en el contexto del software ya existe. Durante la fase d mantenimiento se centran tres tipos de cambios: corrección, adaptación y mejora.

Método de Ciclo de Vida Clásico (Grafica)



7. ANÁLISIS

ESPECIFICACIÓN DE REQUISITOS SOFTWARE (ERS)

Introducción:

El software a desarrollar es de gestión orientada a resolver un problema en particular relacionado con el inventario. Para la solución de dicho problema realizamos una serie de entrevistas con el administrador, quien nos brinda la información sobre como realiza el inventario en dicho Hospital.

Nos informo de la necesidad de un sistema que fuese rápido, eficaz, confiable y fácil de usar para las operaciones del mismo; de esta manera nos dimos una idea de cómo debía operar el sistema y las interfaces requeridas, así como la información que pretende manejar y los resultados que debe proporcionar.

Propósito:

Definición del conjunto de especificaciones de requisitos software que debe cumplir la aplicación **"Automatización para el Control de Inventario en el área de Bodega para el Laboratorio Clínico del HEODRA."** Que consiste en la creación de los procesos de inventarios de la Bodega del Laboratorio Clínico como son: el ingreso de nuevos insumos, reabastecimiento de la bodega, Salida de insumo y requisas por salida.

Definiciones, acrónimos y abreviaturas:

- **ListaBasica:** Entidad Fuerte donde se almacenan los diferentes grupos de insumos que requiere el laboratorio clínico.
- **Insumos:** Entidad fuerte donde se almacenan los datos referentes a los insumos que forman los grupos de la Lista básica.
- **Entrada:** Entidad fuerte donde se almacenan cada una de la entradas de insumos a la bodega del laboratorio clínico.

- **Salida:** Entidad fuerte donde se almacena cada una de las salidas de insumos.
- **Detalle-insumo:** Entidad débil relacionada con la entidad insumo hace referencia detallada de cada uno de los insumos que forman parte de la lista Básica.
- **Detalle-entrada:** Entidad débil relacionada con la entidad Entrada que detalla cada una de las entrada de insumos que ingresan a la bodega.
- **Detalle-salida:** Entidad débil relacionada con la entidad Salida que detalla cada una de las salida de insumos.
- **Id-entrada:** Atributo principal de la entidad entrada que hace referencia al número de entrada que se le asigna, además es el atributo que relaciona la entidad Entrada y Detalle_entrada.
- **Id-salida:** Atributo principal de la entidad salida que relaciona a dicha entidad con la entidad débil Detalle_salida.
- **Acceso:** Entidad que contendrá información descriptiva referente ha cada uno de los usuarios del sistema.
- **Tipo-usuario:** Atributo de la entidad Acceso que contiene al tipo de usuario que acepta el sistema.
- **Clasificacion:** Atributo de la entidad ListaBasica que hace referencia al tipo de insumo ya sea este medico o no medico.
- **Codigo:** Atributo principal de la entidad Insumos que hace referencia al código de identificación de cada insumo.

Alcance:

La aplicación será conocida con el nombre de **"SILAC-HEODRA"** que es la abreviatura de Sistema de inventario del Laboratorio Clínico del HEODRA, permitiendo realizar las siguientes funciones:

- Captura de ingreso de insumos en Bodega.
- Captura de salida de insumos a las diferentes secciones que conforman al laboratorio.

- Captura datos de nuevo grupo insumos no existente en la lista básica.
- Eliminación de grupo insumos no requerido en a lista básica.
- Generación de un Listado o informe de entradas y salidas de insumos.
- Generación de requisas por salida de insumos a las secciones del laboratorio.

Descripción General:

Referente al Producto:

La aplicación interactuara con una base de datos contenida en el computador. El equipo en el que se implementara el producto final es:

- 40 GB. De Disco Duro.
- 128 MB. De RAM.
- Intel(R) Celeron(TM) CPU 1200MHz.
- Sistema Linux Debían.

Referente al Software:

- Sistema Operativo Linux Debían.
- Lenguaje de Programación JAVA
- Gestor de Base de Datos PostgreSQL.

Características del usuario:

El usuario final de la aplicación ya cuenta con conocimientos básico de un sistema computarizado.

Restricciones Generales:

El laboratorio Clínico del Hospital HEODRA la 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.

Funciones del Sistema

La aplicación contiene las siguientes tareas que se llevan a cabo manualmente en dicha institución de forma diaria.

Estas son:

1. Registrar la información de las entradas de insumos a la Bodega.
2. Registrar la información de salida de insumos de la bodega.
3. Generara los siguientes reportes:
   - Reportes de Entradas de insumos por periodo.
   - Reportes de Salida de Insumos por periodo.
   - Reporte de las existencias por Grupo de insumo.
4. Crear nuevo de Usuario.
5. Modificar Clave de Acceso.
6. Agregar nuevo grupo de insumo.
7. Agregar nuevo Insumos que no se encuentran en la lista básica.
8. Eliminar grupo de insumos que ya no son requeridos de la lista básica.
9. Generar requisas correspondientes a la salida de insumos hechas por alguna de las secciones del laboratorio.
10. Generar alertas de Caducidad de insumos al iniciar la aplicación.

Requisitos Específicos.

Requisitos Funcionales.

I. Entrada de Insumo.

Introducción:

Este proceso captura los datos correspondientes al ingreso de insumos a la bodega del laboratorio clínico.

Entradas:

Entradas por pantalla: Datos para codificar las existencias de cada insumo.

- Codigo.: Dato Obligatorio. Se creara un nuevo registro en caso de que se trate de un código nuevo. Los códigos existentes los proporciona el sistema.
- Fuente-financiera: Este dato hace referencia a los recursos que se utilizaron para conseguir el insumo.
- Ubicacion: dato no obligatorio, este hace referencia a la ubicación física del producto.
- Empaque: Presentación en la que se encuentra el insumo.
- Lote: Numero de lote correspondiente a cada insumo
- Marca: Marca del insumo si la tiene.
- Fecha-vence: Fecha de vencimiento del insumo.
- Cantidad: Cantidad insumos solicitada al proveedor y que ingresa a la bodega este actualizara las existencias del insumo en la base de datos del sistema.
- Costo-unitario: Precio asignado al insumo por unidad requerido para generar la requisa de solicitud.
- Costo-total: Campo calculado al multiplicar Cantidad por Costo-unitario.

Los siguientes datos los proporciona el sistema:

- Hora-entrada.
- Fecha-entrada.
- Id-entrada.
- Nombre-generico
- Codigo.

Proceso:

Este proceso presentara al usuario una interfaz donde podrá agregar y almacenar los datos de las entradas de insumos a la bodega, informara al usuario a través de un dialogo si el proceso se almaceno, hubo algún error o faltan datos.

Salidas:

Se almacenara el registro en la entidad DetalleEntrada.

II. Salida de Insumo.

Introducción:

Este proceso almacena las salidas de insumos en la base de datos para futuras consultas.

Entrada:

Por pantalla: Datos para registrar las salidas de insumos.

- Salida-por: Campo no obligatorio que hace referencia al tipo de salida que se le da al insumo.
- Cantidad: Campo que hace referencia a la cantidad de producto al que se le da salida.

Los siguientes campos son proporcionados por la aplicación:

- Id-salida.
- Destinacion.
- Grupo.
- Codigo.
- Fecha-salida.
- Hora-salida.
- Nombre-generico.
- Presentacion.
- Existencias

Proceso:

Presentara una interfaz donde seleccionará el producto que desea dar salida y la cantidad solicitada, en caso de que ocurra un error se visualizara un dialogo informando del error, en caso de que todo este correcto se informara en un dialogo que la salida se a almacenado; a demás de proporcionar la opción de imprimir la requisa correspondiente a esa salida.

Salida:

Almacenara el registro correspondiente a la salida de insumos en la entidad DetalleSalida y se generara la respectiva requiza.

III. Agregar Grupo.

Introducción:

Este proceso agrega un nuevo registro que hace referencia a un nuevo grupo de insumos médicos.

Entradas:

=>Botón Nuevo Grupo: prepara la interfaz para que el usuario ingrese los datos necesarios para registrar el nuevo grupo.

Por pantalla: Datos necesarios para registrar el nuevo grupo a la lista.

- Nombre-grupo: Nombre que se le asigna a la categoría de insumos.
- Clasificacion: hace referencia al tipo de insumos a la que corresponde la categoría.

Proceso:

Este proceso solicita al usuario que ingrese el nombre del nuevo grupo que pasara a formar parte de la lista básica de insumos, además se le solicitara que ingrese los datos correspondientes a los insumos que conforman esta nueva categoría.

Salidas:

Se almacena el registro en la entidad ListaBasica

IV. Borrar Grupo.

Introducción:

Esta función le permite al usuario borrar de su lista básica uno de los grupos insumos que la conforman

Entradas:

Menú Borrar Grupo: para ingresar a la función borrar grupo de la aplicación SILAC-HEODRA.

Proceso:

Se le muestra al usuario una pequeña interfaz donde se le permite seleccionar el nombre del grupo que desea borrar de la lista básica de insumos de la base de datos de la aplicación SILAC-HEODRA.

Salidas:

Se eliminara el registro seleccionado de la entidad ListaBasica.

V. Modificar Código.

Introducción:

Esta función le permite al usuario modificar el código de un determinado insumo en caso de haber ingresado el código incorrectamente o en caso de un nuevo cambio de código por parte del ministerio de salud MINSA**.**

Entradas:

=> Boton Modificar Código: Este componente representa la función Modificar Código.

Por pantalla solicitara:

- Codigo Anterior: hace referencia al código que se desea borrar.
- Codigo Nuevo: se refiere al nuevo código que ha de sustituir al código anterior.

Proceso:

Este proceso presenta al usuario una pequeña interfaz don de le permite ingresar el código que desea sustituir y el nuevo proporciona también informa a través de caja de dialogo si el proceso se a concretado o hubo algún error.

Salidas:

Se actualizará el registro modificado en las entidades que contengan este campo.

VI. Existencias.

Introducción:

Esta función le permite al usuario un informe sobre las existencias por categoría o grupo de insumos que se encuentran en el inventario de la bodega del laboratorio clínico

Entradas:

=>Botón Existencias: para poder acceder a la función existencias hay que pulsar este botón de la aplicación SILAC-HEODRA.

Proceso:

Se le presentara al usuario una interfaz con la opción de seleccionar el grupo del cual quiera información de las existencias o podrá ingresar el código en el caso de un producto en especifico; también se le brinda la opción de imprimir el resultado de la consulta.

Salidas:

Se muestra en la tabla de la interfaz los datos correspondientes a la consulta hecha por el usuario.

VII. Informe de Entradas.

Introducción:

Esta función le brinda al usuario un informe de todas las entradas que ha habido de acuerdo al periodo que el usuario defina.

Entradas:

Por Pantalla: seleccionara el mes inicial, el mes final y el año que comprende el periodo de interés.

Proceso:

Se le presenta al usuario una interfaz donde se le permite seleccionar el mes en que iniciara la aplicación a generar el reporte a si como el mes en que finalizara y el año corres pendiente y le brinda la opción de imprimir el resultado.

Salidas:

Se muestra por pantalla una hoja con el resultado de la consulta.

VIII. Informe de Salidas.

Introducción:

Esta función le brinda al usuario un informe de todas las salidas que ha habido de acuerdo al periodo que este defina.

Entradas:

Por Pantalla: seleccionara el mes inicial, el mes final y el año que comprende el periodo de interés.

Proceso:

Se le presenta al usuario una interfaz donde se le permite seleccionar el mes en que iniciara la aplicación a generar el reporte a si como el mes en que finalizara y el año corres pendiente y le brinda la opción de imprimir el resultado.

Salidas:

Se muestra por pantalla una hoja con el resultado de la consulta.

IX. Crear Nuevo Usuario.

Introducción:

Este proceso es exclusivo del usuario administrador, crea un usuario ya sea administrador o invitado.

Entradas:

Por pantalla: campos para almacenar el nuevo usuario.

- Usuario: Nombre que se le designa al usuario.
- Contraseña: Clave por la cual el usuario tendrá acceso a la aplicación.
- Tipo-usuario: identificador necesario para los permisos que tendrá el usuario, este es proporcionado por el sistema.

Proceso:

Presenta al usuario una interfaz donde podrá ingresar el nombre del usuario que el desee, seleccionar el tipo de usuario para los privilegios e ingresar la contraseña correspondiente al usuario que se esta creando, se informara a través de diálogos si hubo algún error o el proceso se concreto correctamente.

Salida:

Registrar el nuevo usuario e informar al usuario de que el proceso se efectuó correctamente o hubo algún error a través de mensajes del sistema.

X. Modificar Clave de Acceso.

Introducción:

Esta función le brinda al usuario la opción de modificar su clave de acceso a la aplicación SILAC-HEODRA.

Entradas:

Por pantalla: el usuario ingresa la clave de acceso anterior y la nueva clave de acceso.

Proceso:

Se le presenta al usuario una interfaz que le permitirá ingresar la clave anterior y la nueva clave para que se modifique el registro, se le informara al usuario a través de diálogos si se concreto correctamente o ocurrió algún error al modificar la clave.

Salidas:

Se modifica el registro actual de la clave de acceso en la entidad Acceso.

XI. Salir de la Aplicación.

Introducción:

Esta función le permite al usuario salir del sistema una ves que haya terminado todas las tareas asignadas.

Entradas:

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

Proceso:

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

Salidas:

Ninguna.

XII. Acerca de.

Introducción:

Esta función permitirá a los usuarios obtener información sobre los Elaboradores de la aplicación.

Entradas:

⇒ Menú Acerca de: Para poder acceder a la función Acerca de SILAC-HEODRA el usuario deberá dar click en el menú Acerca de.

Proceso:

Al usuario se le mostrara un Frame donde estará la información referente a los elaboradores tal como los nombres, correos electrónicos y números telefónicos para poder contactarlos en caso de algún inconveniente con la aplicación.

Salidas:

Ninguna.

XIII. Acceso a la Aplicación.

Introducción:

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

Entradas:

- ⇒ 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:

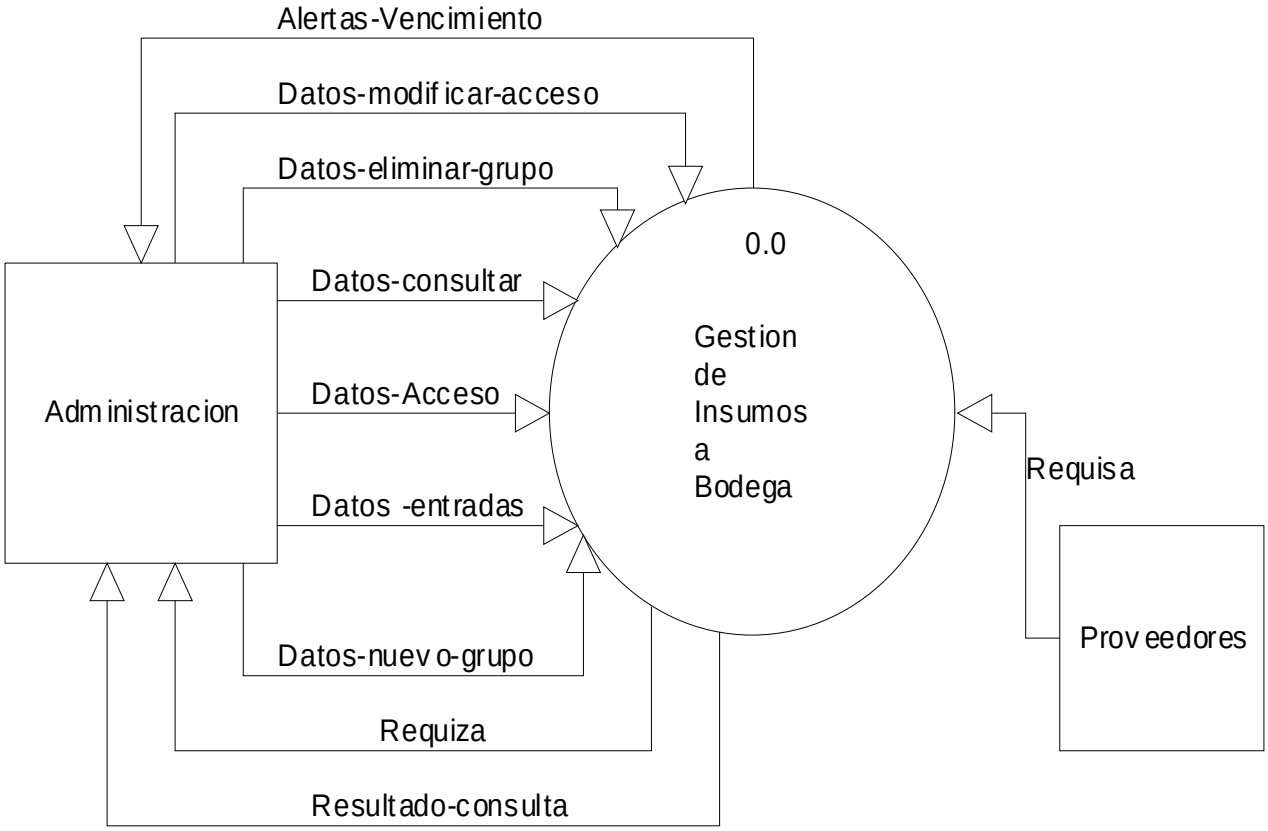
Se mostrará al usuario el Frame principal de acceso al sistema SILAC-HEODRA, donde deberá introducir el nombre de usuario y su contraseña, en la cual el sistema verificará si los datos introducidos son válidos, de ser así el usuario podrá ingresar, de lo contrario se presentará un mensaje indicándole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

Salidas:

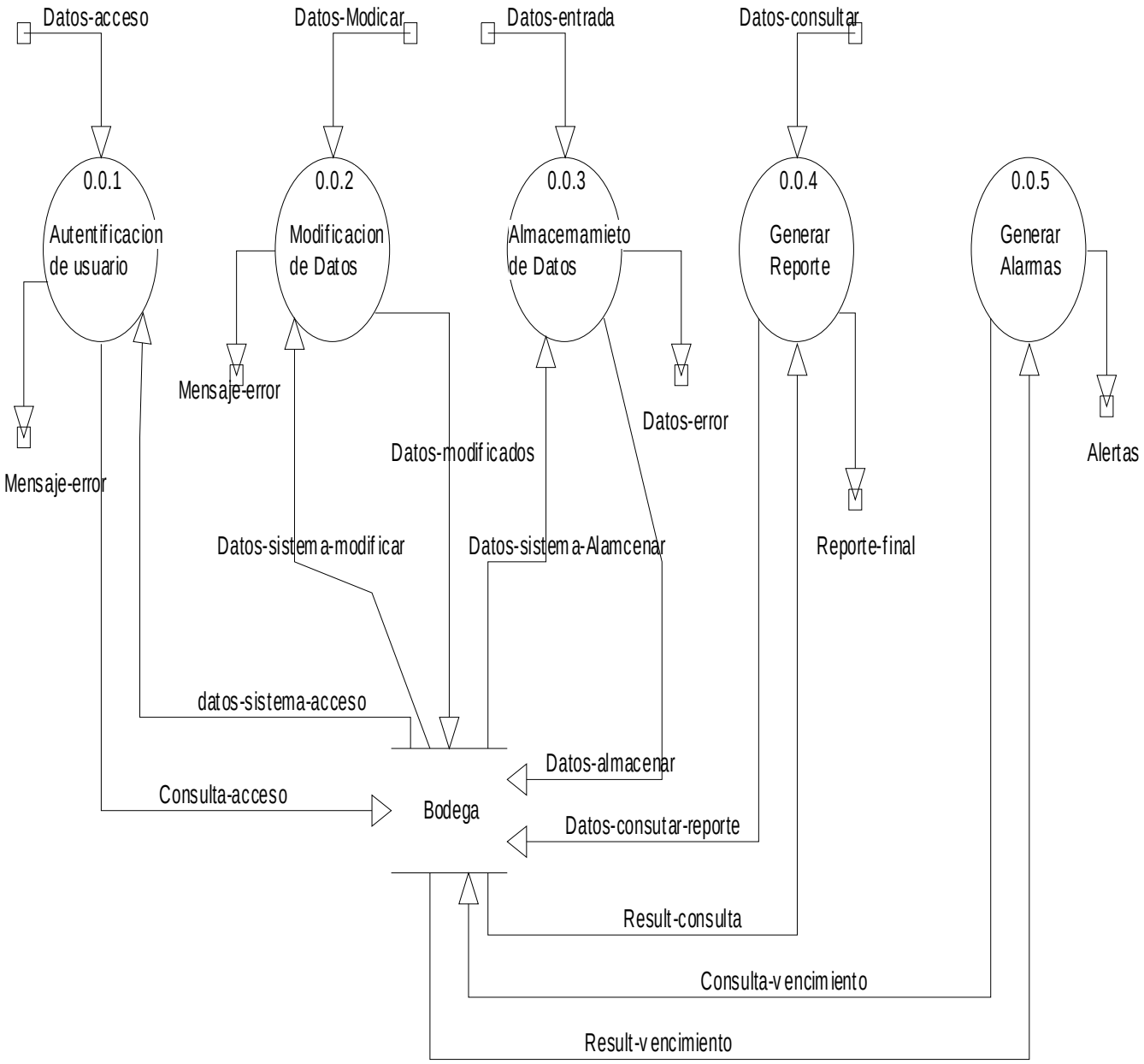
Ninguna.

7.1 Diagrama de Flujo de Datos (DFD)

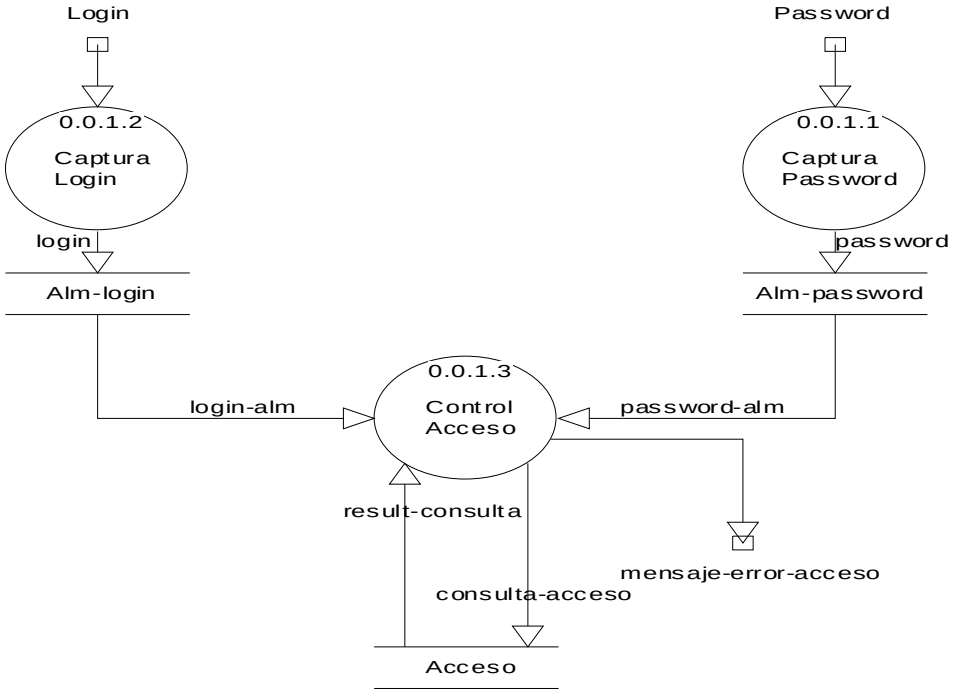
Nivel 0.



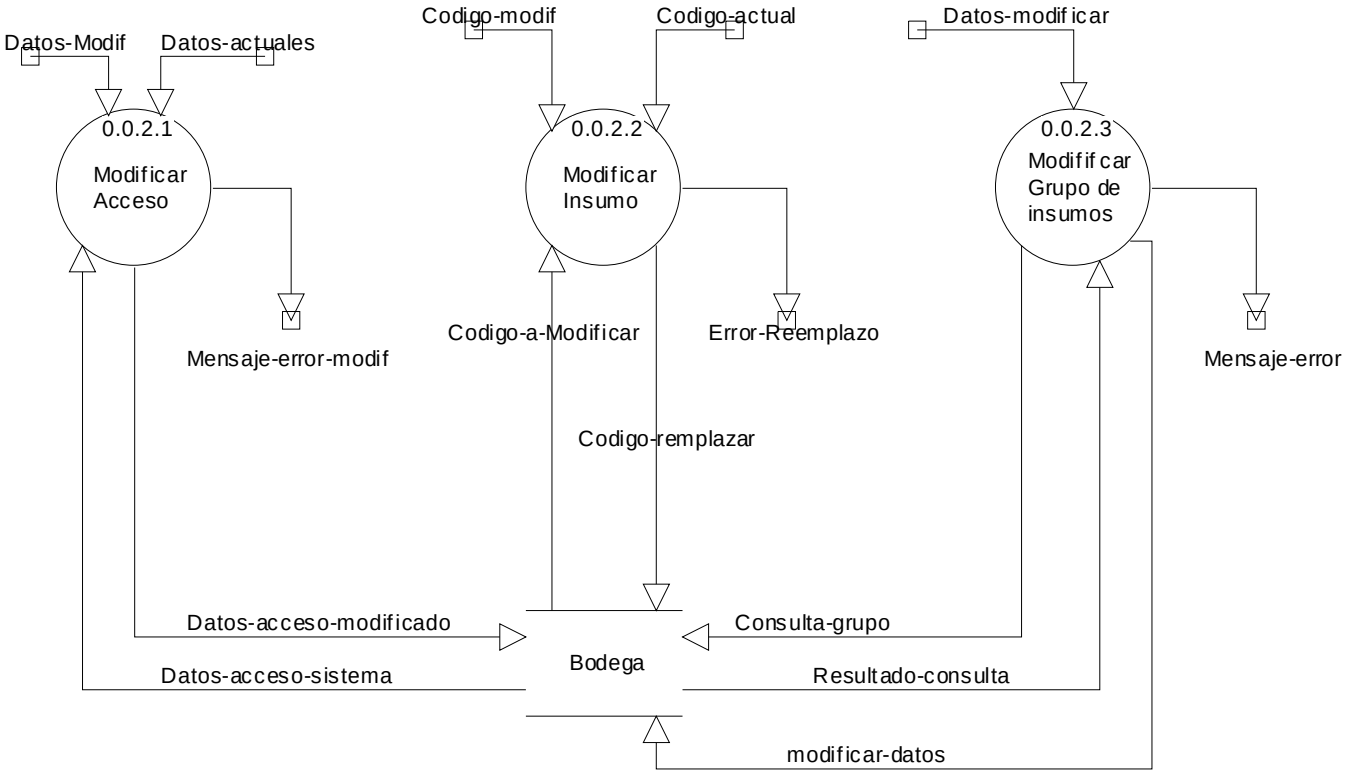
Nivel 1 del Proceso: Gestión de Insumos a Bodega



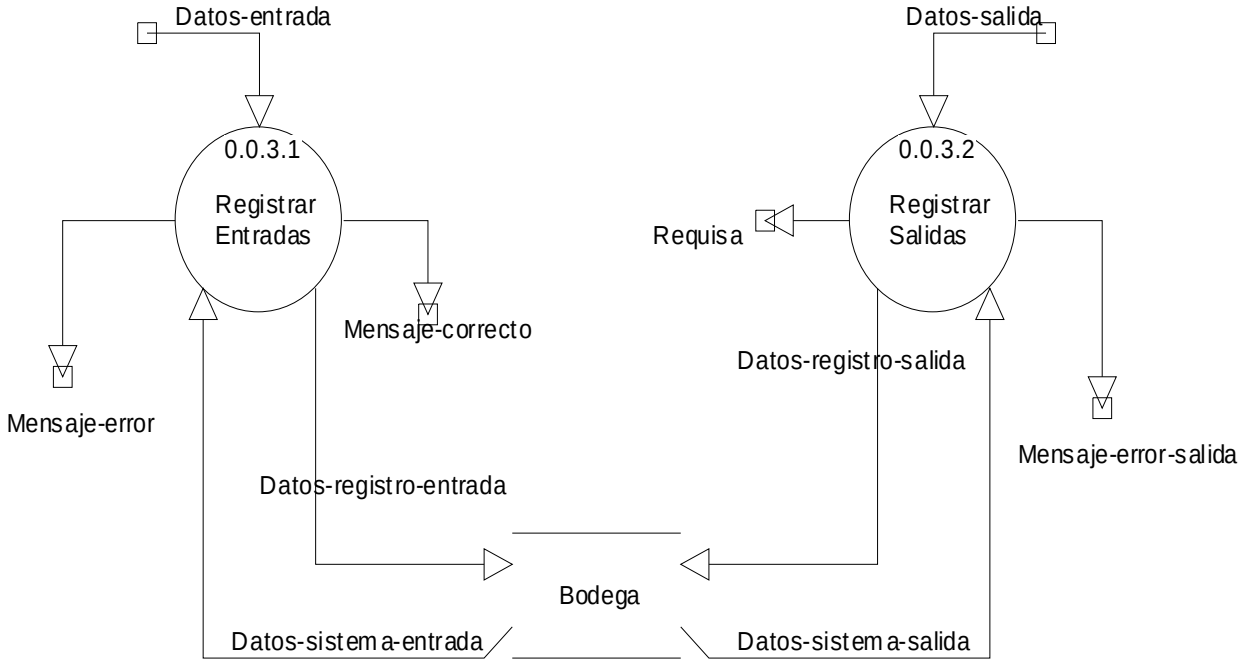
Nivel 2 del Proceso 1: Autentificación de Usuario.



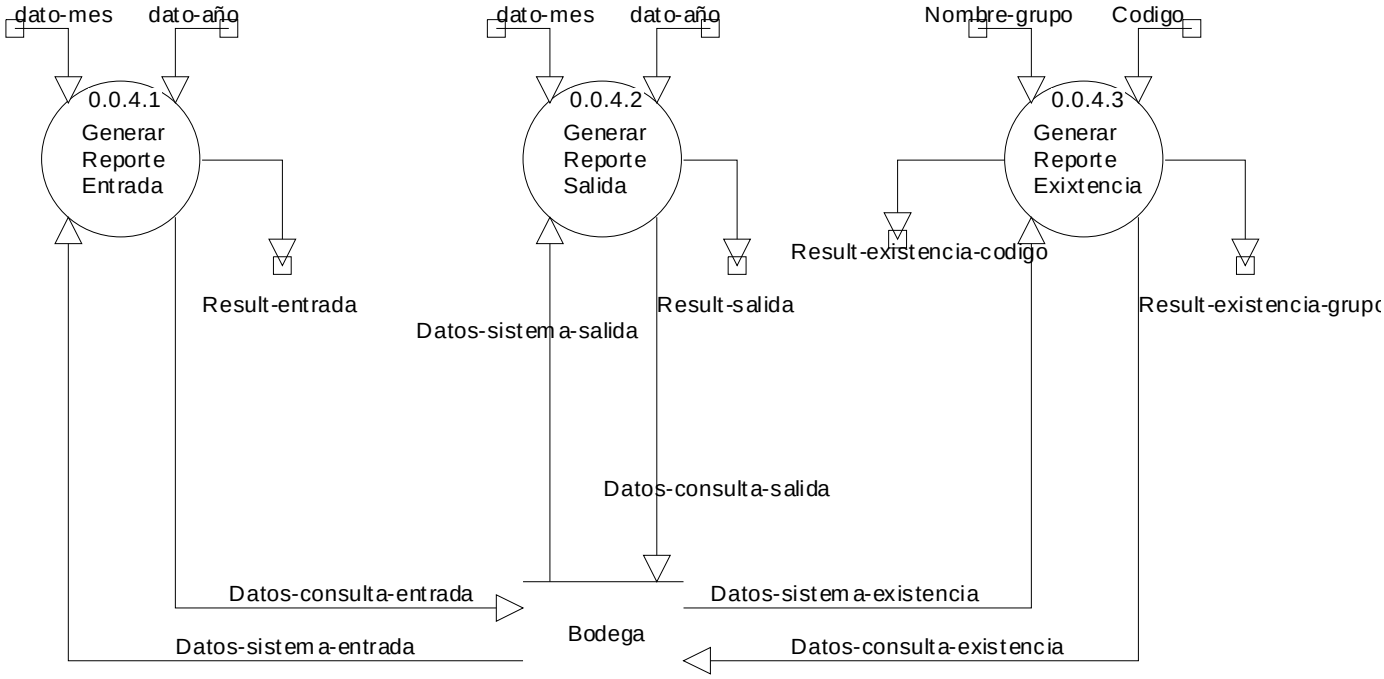
Nivel 2 del Proceso 2: Modificación de Datos



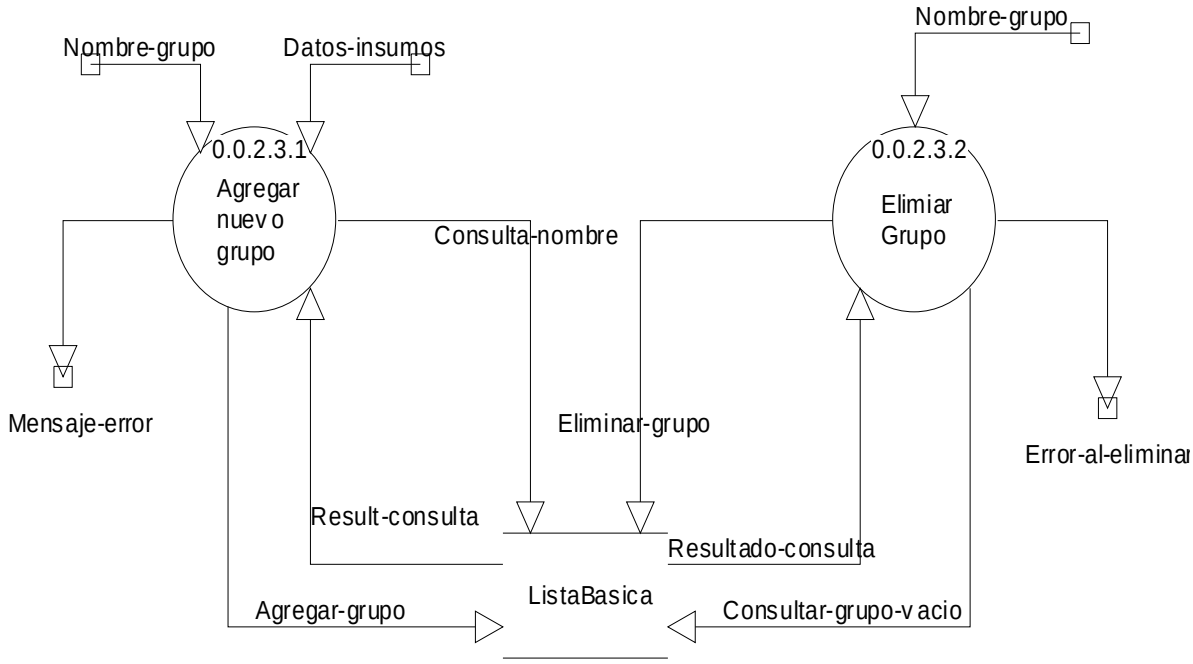
Nivel 2 Proceso 3: Almacenamiento de Datos.



Nivel 2 Proceso 4: Generar Reportes



Nivel 3 del Proceso 2.3. Modificar Grupo de Insumo.



7.1.1. Diccionario de Datos.

Datos_sistema_entrada: Nombre_generico + Presentacion + Nombre_grupo + Codigo + Id_entrada + Fecha_entrtada + Hora_entrada.

Datos_entrada: Marca + Empaque + Fuente_financiera + Lote + Ubicación + Fecha_vence + Cantidad + Costo_unitario + Costo_total.

Costo_total= Cantida * Costo_unitario.

Datos_sistema_salida: Id_salida + Destinacion + Fecha_salida + Hora_salida + Nombre_generico + codigo + Nombre_grupo + Presentacion + Existencias.

Datos_salida: Salida_por + Cantidad.

Datos_Acceso: Usuario + Contrasena.

Datos_requiza: Id_requiza + Salida_por + Destinacion + codigo + Nombre_generico + Marca + Empaque + Fuente_financiera + ubicacion + Lote + Fecha_vence + Cantidad + Costo_unitario + Costo_total.

Costo_total= Costo_unitario * Cantidad.

Datos_almacenar_salida: codigo + Marca + Empaque + Fuente_fianancira + Lote + Fecha_vence + Cantidad + Costo_unitario + id_salida + destinacion + fecha_salida + hora + mes + anyo.

Datos_almacenar_entrada: codigo + Marca + Empaque + Fuente_fianancira + Lote + Fecha_vence + Cantidad + Costo_unitario + id_entrtada + Fecha_entrada + Hora_entrada + mes + anyo.

Datos_consulta_entrada: mes + anyo.

Datos_consulta_salida: mes + anyo.

Datos_consulta_existencias: Nombre_grupo + codigo.

Result_existencias_grupo: Codigo + Nombre_generico + Presentacion + Existencia + uso + indicador_insumo.

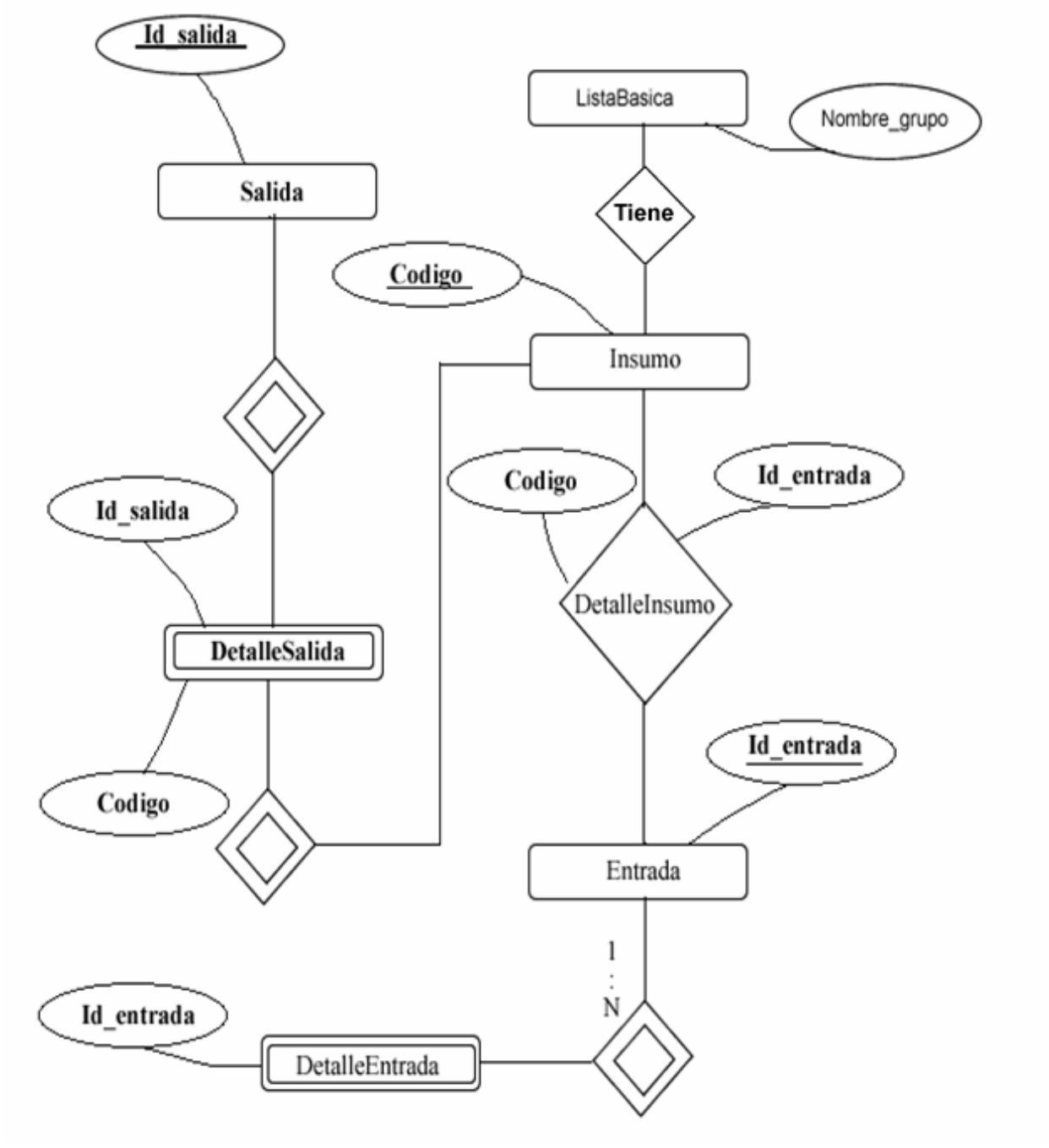
Result_existencias_codigo: Codigo + Marca + Empaque + Fuente_finanaciera + Ubicacion + Lote + Fecha_vence + Cantidad + Costo_unitario + Costo_total.

Costo_total = Costo_unitario + Cantidad.

Reset_entrada: Id_entrada + Fecha_entrada + Hora_entrada + Codigo + Marca + Empaque + Fuente_financiera + Fecha_vence + Cantidad.

Result_salida: Id_salida + Fecha_salida + Hora_salida + Salida_por + Destinacion + Codigo + Marca + Empaque + Fuente_financiera + Fecha_vence + Cantidad.

7.2. Diagrama Entidad-Relación.



7.3. Diseño.

DISEÑO DE DATOS

Tabla No. 1. ACCESO		
Nombre del Atributo	**Tipo**	**Tamaño**
Usuario	Varchar	30
Contrasena	Varchar	20
Tipo_usuario	Varchar	20

Tabla No. 2. LISTA BASICA		
Nombre del Atributo	**Tipo**	**Tamaño**
Nombre_grupo	Varchar	30
Clasificacion	Varchar	30

Tabla No. 3. INSUMO		
Nombre del Atributo	**Tipo**	**Tamaño**
Codigo	Varchar	10
Nombre_generico	Varchar	30
Presentacion	Varchar	5
NivelUso	Varchar	5
Uso	Varchar	30
Indicador_insumo	Varchar	50
Nombre_grupo	Varchar	30

Tabla No. 4. DETALLE INSUMO		
Nombre del Atributo	**Tipo**	**Tamaño**
Codigo	Varchar	10
Marca	Varchar	20
Empaque	Varchar	20
Fuente_financiera	Varchar	30
Ubicacion	Varchar	20
Lote	Varchar	20
Fecha_vence	Varchar	20
Cantidad	Integer	Autonumerico
Costo_unitario	Float8	Autonumerico
Costo_total	Float8	Autonumerico
Id_entrada	Float8	Autonumérico

Tabla No. 5. DETALLE SALIDA		
Nombre del Atributo	**Tipo**	**Tamaño**
Codigo	Varchar	10
Marca	Varchar	20
Empaque	Varchar	20
Fuente_financiera	Varchar	30
Lote	Varchar	20
Fecha_vence	Varchar	20
Cantidad	Integer	Autonumerico
Costo_unitario	Float8	Autonumerico
Costo_total	Float8	Autonumerico
Id_salida	Float8	Autonumerico

Tabla No. 5. DETALLE ENTRADA		
Nombre del Atributo	**Tipo**	**Tamaño**
Codigo	Varchar	10
Marca	Varchar	20
Empaque	Varchar	20
Fuente_financiera	Varchar	30
Lote	Varchar	20
Fecha_vence	Varchar	20
Cantidad	Integer	Autonumerico
Costo_unitario	Float8	Autonumerico
Costo_total	Float8	Autonumerico
Id_entrada	Float8	Autonumerico

Tabla No. 6. ENTRADA		
Nombre del Atributo	**Tipo**	**Tamaño**
Id_entrada	Float8	Autonumerico
Fecha_entrada	Date	
HoraEntrada	Time	
Mes	Integer	Autonumerico
Anyo	Integer	Autonumerico

Tabla No. 7. SALIDA

Nombre del Atributo	Tipo	Tamaño
Id_salida	Float8	Autonumerico
Salida_por	Varchar	50
Destinacion	Varchar	50
Fecha_salida	Date	
Hora	Time	
Mes	Integer	Autonumerico
Anyo	Integer	Autonumerico

8. CONCLUCION.

Al concluir nuestro trabajo monográfico, consideramos que hemos cumplido con los objetivos planteados, desarrollando nuestro Sistema SILAC-HEODRA y haber utilizado **Linux Debian** (Sistema Operativo), **JAVA** (Lenguaje de Programación), **PostgreSQL** (Sistema Gestor de Base de Datos), **Kwrite** (Editor de código), **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 para el Control de Inventario en el Área de Bodega del Laboratorio Clínico del HEODRA**", propuesta como objetivo fundamental de nuestro trabajo monográfico.

9. Recomendaciones.

- Implementar protocolo de transacción para garantizar la fiabilidad de los datos en caso de falta de fluido eléctrico al momento de una transacción de datos.

- Implementar método para el respaldo de los datos de forma grafica en caso de pérdida o eliminación de estos del computador.

- Promover entre los estudiantes del Departamento de computación el desarrollo de sistema haciendo uso de Software Libre.

10. Bibliografía.

- ✓ **http://es.wikipedia.org/wiki/PostgreSQL**
- ✓ **http://es.tldp.org/Postgresql-es/web/navegable/tutorial/x56.html**
- ✓ **www.gsl.unt.edu.ar/desc/Festival/diapositivas/PostgreSQL.pdf**
- ✓ **www.postgres.org**
- ✓ **http://www.postgresql.org/docs/8.0/static/index.html**
- ✓ **www.progaramacionfacil.com/cursojava.html**
- ✓ **www.jformdesigner.com**
- ✓ **www.webzip.com/itapicazo.html**
- ✓ **www.linuxfocus.org/castellano/**
- ✓ **www.nvu.com**

AnexOS

INICIO DE LA APLICACION

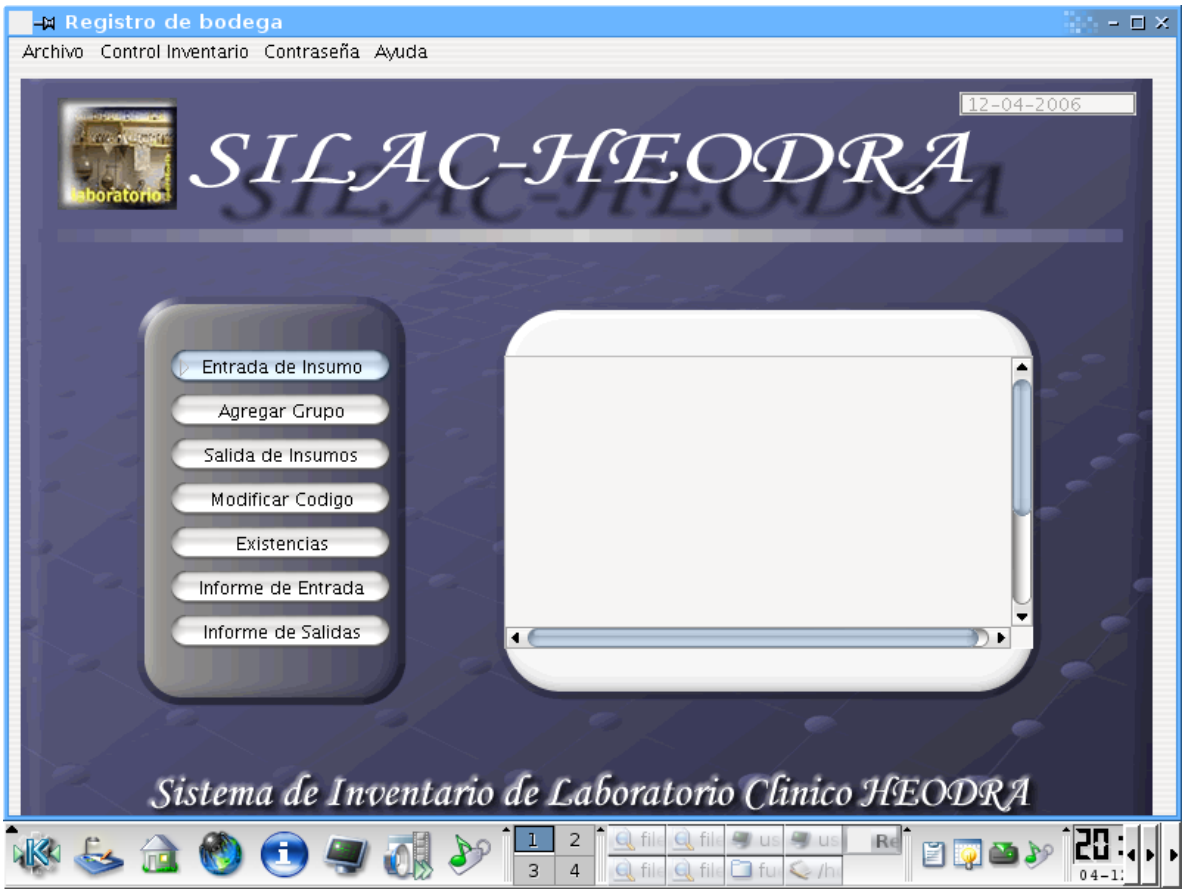
INTERFAZ DE CONTRASEÑA ACCESO

Codificacion de la Interfaz Acceso

```
public void verificarUsuario()
{
  char [] passwordelegido=palabraclave.getPassword();
  String var = new String(passwordelegido);
  boolean prob;
  String sUsuario=String.valueOf(tx_usuario.getText());
  Connection conexion;
  PreparedStatement  obtener_codigo;
   try
   {
    conexion =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
    obtener_codigo=conexion.prepareStatement("SELECT tipo_usuario FROM
Acceso WHERE usuario=? and contrasena = ?");
    obtener_codigo.setString(1,sUsuario);
    obtener_codigo.setString(2,var);
    ResultSet resul= obtener_codigo.executeQuery();
   if (conexion != null)
      {

     if(resul.next()==true)
        {
       String tipoUs = resul.getString("tipo_usuario");
```

```
        if(tipoUs.equals("Administrador"))
         {
              int i=0;
          restringir.setIconImage (new
ImageIcon("iconoprincipal.gif").getImage());
          restringir.setVisible(true);
              setVisible(false);
         }
      else
      {
      restringir.btEntrada.setEnabled(false);
            restringir.btModificarCodigo.setEnabled(false);
            restringir.mIetn_modificarClave.setEnabled(false);
            restringir.mIten_nuevoUsuario.setEnabled(false);
            restringir.mItem_modificaCodigo.setEnabled(false);
            restringir.mItem_borraGrupo.setEnabled(false);
        restringir.setVisible(true);
      }
     }
     else
     {
     JOptionPane.showMessageDialog(null, "Contraseña Invalida",
"Mensaje", JOptionPane.INFORMATION_MESSAGE);
     }
    }
      resul.close();
    obtener_codigo.close();
    conexion.close();
  }
 catch(SQLException ex)
  {
   ex.printStackTrace( );
  }

}
////// FIN DE LA ClASE
```

INTERFAZ SIBLAC-HEODRA



Codificación de la Interfaz Principal.

```
import javax.swing.*;
import java.sql.*;
import java.awt.*;
import javax.swing.JPanel;
import java.lang.String.*;
import java.io.*;
import java.lang.*;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JComboBox;
import javax.swing.table.*;
import javax.swing.JTable;
import java.util.Vector;
import java.util.Date;
import java.text.DateFormat;
import java.awt.event.*;
import java.text.*;
import java.util.*;
import javax.swing.Timer;
```

```
public class siblab extends JFrame implements ActionListener
 {
   public JButton  btEntrada= new JButton( " Entrada de Insumo " );
   JButton btSalida= new JButton( " Salida de Insumos " );
   JButton btnuevoGrupo=new JButton("Nuevo Grupo");
   JButton btgrupo,btborrar_C,btModificarCodigo,btExistencias;
   JButton
btInformeEntrada,btinforme_salida_3,btinforme_entrada_A,btinforme_salida_
A;
   JMenuBar barramenu;
   JMenu menu_archivo,menuP_control,menu_Contrasena,menu_ayuda;
   JMenuItem mItem_InsumoVencido;
   JMenuItem
mIetn_modificarClave,mIten_nuevoUsuario,mItem_modificaCodigo,mItem_borraG
rupo;
   JMenuItem m_existencia,m_mdificarCodigo, menu_salir,acerca_de;
   JMenuItem mItem_Entrada,mItem_Salida;
      JLabel lbLogo;
      Container c;
      int count;
      JLabel principal=new JLabel(new ImageIcon("principal.gif"));
      JPanel panel1,panel3, panel,panel4,panel5,panel6,panel7;
      JScrollPane areaScroll = new JScrollPane();
    JTextArea jtxareaMensaje=new JTextArea(20,50);
    String acercaDe;
    JFormattedTextField sFechaActual= new JFormattedTextField(new
java.util.Date( ));
  public  siblab()
  {
          setTitle("Registro de bodega");
            setSize(800,600);
        setResizable(true);
            getContentPane().setLayout(null);
            addWindowListener(new Manejador());
            btEntrada.addActionListener(this);
        btSalida.addActionListener(this);
            btnuevoGrupo.addActionListener(this);
        panel=new JPanel(true);
        panel4=new JPanel();
        panel5=new JPanel();
        Color color=new Color(155, 155, 155);
        panel.setLayout(null);
        panel1=new JPanel();
        panel1.setLayout(null);
        getContentPane().add(panel1);
        btEntrada.setBounds(20,30,150,23);
        panel1.add(btEntrada);
        btgrupo=new JButton(" Agregar Grupo ");
        btgrupo.setBounds(20,60,150,23);
        panel1.add(btgrupo);
        btgrupo.addActionListener(this);
        btSalida.setBounds(20,90,150,23);
        panel1.add(btSalida);
        btModificarCodigo=new JButton("Modificar Codigo");
```

```
        btModificarCodigo.setBounds(20,120,150,23);
        panel1.add(btModificarCodigo);
        btModificarCodigo.addActionListener(this);
        btExistencias=new JButton("Existencias");
        btExistencias.setBounds(20,150,150,23);
        panel1.add(btExistencias);
        btExistencias.addActionListener(this);
        btInformeEntrada=new JButton("Informe de Entrada");
        btInformeEntrada.setBounds(20,180,150,23);
        panel1.add(btInformeEntrada);
        btInformeEntrada.addActionListener(this);
        btinforme_salida_3=new JButton("Informe de Salidas");
        btinforme_salida_3.setBounds(20,210,150,23);
        panel1.add(btinforme_salida_3);
        btinforme_salida_3.addActionListener(this);
        panel1.setBounds(90,160,180,260);
        getContentPane().add(panel1);
        panel3=new JPanel(); // panel3 para ubicar los botones de informe
        panel3.setLayout(null);
        areaScroll.setViewportView(jtxareaMensaje);
            areaScroll.setBounds(1,1,360,200);
            panel3.add(areaScroll);
            panel3.setBounds(335,193,360,200);
        getContentPane().add(panel3);//fin del panel de informes
        sFechaActual.setBounds(645,10,120,16);
            panel5.add(sFechaActual);
            sFechaActual.setEnabled(false);
        panel5.setLayout(null);
        lbLogo=new JLabel(new ImageIcon("portada.gif"));
        lbLogo.setBounds(5,1,775,540);
        panel5.add(lbLogo);
        panel5.setBounds(0,5,785,600);
        getContentPane().add(panel5);
        panel.setBounds(5,10,770,500);// panel principal de control de
Insumos
        getContentPane().add(panel);
        /***** menus***/
        barramenu = new JMenuBar();
            menu_archivo = new JMenu("Archivo");
            menu_ayuda = new JMenu("Ayuda");
            menu_salir = new JMenuItem("Salir");
            acerca_de = new JMenuItem("Acerca de..");
            menuP_control=new JMenu("Control Inventario");
            mItem_Entrada=new JMenuItem("Entradas");
            mItem_Salida=new JMenuItem("Salidas");
            mItem_InsumoVencido=new JMenuItem("Insumos Vencidos");
            mItem_modificaCodigo=new JMenuItem("Modificar Codigo");
            mItem_borraGrupo=new JMenuItem("Borrar Grupo");
            menuP_control.addSeparator();
          menuP_control.add(mItem_Entrada);
          mItem_Entrada.addActionListener(evento_menu);
        menuP_control.add(mItem_Salida);
        mItem_Salida.addActionListener(evento_menu);
        menuP_control.add(mItem_InsumoVencido);
        mItem_InsumoVencido.addActionListener(evento_menu);

          menuP_control.add(mItem_modificaCodigo);
```

```
        menuP_control.add(mItem_borraGrupo);
        mItem_modificaCodigo.addActionListener(evento_menu);
        mItem_borraGrupo.addActionListener(evento_menu);
  ////////////////añadiendo un evento al listener///////
          menu_archivo.addSeparator();
          menu_archivo.add(menu_salir);
          menu_salir.addActionListener(evento_menu);
          menu_Contrasena=new JMenu("Contraseña");
          mIetn_modificarClave=new JMenuItem("Modificar Contraseña");
          menu_Contrasena.add(mIetn_modificarClave);
          mIetn_modificarClave.addActionListener(evento_menu);
          mIten_nuevoUsuario=new JMenuItem("Crear Usuario");
          menu_Contrasena.add(mIten_nuevoUsuario);
          mIten_nuevoUsuario.addActionListener(evento_menu);
          menu_ayuda.add(acerca_de);
          acerca_de.addActionListener(evento_menu);
          barramenu.add(menu_archivo);
          barramenu.add(menuP_control);
          barramenu.add(menu_Contrasena);
          barramenu.add(menu_ayuda);
          setJMenuBar(barramenu);
      String fVencer=String.valueOf(sFechaActual.getText());
      enviaMensajeAtesVencer(fVencer);
      jtxareaMensaje.setEditable(false);
  }

  public void actionPerformed(ActionEvent e)
     {
         if(e.getSource()==btEntrada)
                {

                EntradaInsumo entrada=new EntradaInsumo(this);
                 entrada.setVisible(true);
                }
           if(e.getSource()==btSalida)
                {
                new salidas(this).setVisible(true);
                }


           if(e.getSource()==btExistencias)
                {
                 new Existencias(this,true).setVisible(true);

                 }

                 if(e.getSource()==btInformeEntrada)
                {
                 new InformeEntrada().setVisible(true);

                 }
                 if(e.getSource()==btinforme_salida_3)
                {
                 new InformeSalida().setVisible(true);

                 }
```

```
                if(e.getSource() == btModificarCodigo)
                                {
                                      new
ModificarCodigo(this,true).setVisible(true);

                                }


            if(e.getSource()==btgrupo)
                  {
                    new GrupoPrincipal(this,true).setVisible(true);
                  }
          }


        ActionListener evento_menu = new ActionListener()
                  {
                        public void actionPerformed(ActionEvent origen)
                        {
                              siblab obvisible =new siblab();
                              if(origen.getSource() == menu_salir)
                              {
                        int n = JOptionPane.showConfirmDialog(null, "Estas
seguro que Deseas Salir?","Atencion",JOptionPane.YES_NO_OPTION);
                  if (n == JOptionPane.YES_OPTION)
                     {
                      System.exit(0);
                     }

                        }
                            if(origen.getSource() ==  mItem_Entrada)
                              {
                                      new
EntradaInsumo(siblab.this).setVisible(true);

                                }

                                if(origen.getSource() ==  mItem_Salida)
                                {
                                      new
salidas(siblab.this).setVisible(true);

                                }

                                if(origen.getSource() ==
mItem_InsumoVencido)
                                {
                                      new
InsumoVencido(siblab.this,true).setVisible(true);

                                }

                                if(origen.getSource() ==
mItem_modificaCodigo)
                                {
                                      new
ModificarCodigo(obvisible,true).setVisible(true);
```

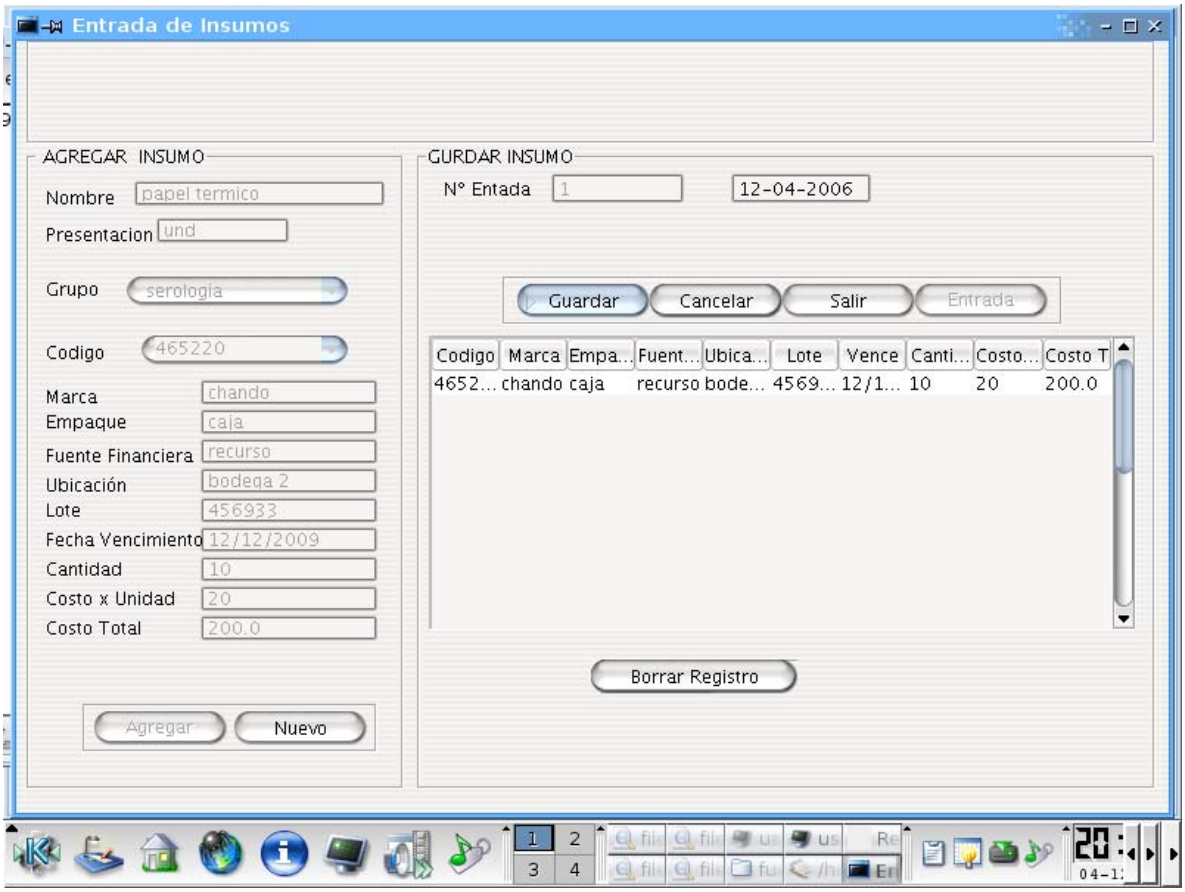
```
                                }
                                if(origen.getSource() == mItem_borraGrupo)
                                {
                                      new
BorrarGrupo(siblab.this,true).setVisible(true);
                                }

                                if(origen.getSource() ==
mIetn_modificarClave)
                                {
                                      new
ModificaUsuario(obvisible,true).setVisible(true);
                                }
                                if(origen.getSource() == mIten_nuevoUsuario)
                                {
                                      new
CrearUsuario(obvisible,true).setVisible(true);
                                }
                                if(origen.getSource() == acerca_de)
                                {
                                      acercaDe="Aplicacion Realizada en el
año 2006 "+"\n"
                                      +"\t Colaboradores \n\n"+" William
Leyton Zapata \n"
                                      +"\t Telefono: 6346413\n Correo:
williamlz2912@yahoo.es \n\n"
                                      +" Douglas Martinez\n"+"\t Telefono:
6309797\n Correo :douglasim6309797@yahoo.es";
                                JOptionPane.showMessageDialog(null,acercaDe
, "Creadores", JOptionPane.INFORMATION_MESSAGE);
                                }
                          }
                    };///hasta aqui ha sido agregado
class Manejador extends WindowAdapter
{public void windowClosing(WindowEvent e)
      {       System.exit(0);
      }}
public void enviaMensajeAtesVencer(String fechaActual)
   {
     String fecha2=fechaActual;
       String diaV="";
     String dia2=fecha2.substring(0,2);
     String mes2=fecha2.substring(3,6);
     String anyo2=fecha2.substring(7,11);
     int dia=0;
     int diaActual=0;
     if(mes2.equals("ene")){mes2="01";dia=31;}
     if(mes2.equals("feb")){mes2="02";dia=28;}
     if(mes2.equals("mar")){mes2="03";dia=31;}
     if(mes2.equals("abr")){mes2="04";dia=30;}
     if(mes2.equals("may")){mes2="05";dia=31;}
     if(mes2.equals("jun")){mes2="06";dia=30;}
     if(mes2.equals("jul")){mes2="07";dia=31;}
     if(mes2.equals("ago")){mes2="08";dia=31;}
     if(mes2.equals("sep")){mes2="09";dia=30;}
     if(mes2.equals("oct")){mes2="10";dia=31;}
```

```
     if(mes2.equals("nov")){mes2="11";dia=30;}
     if(mes2.equals("dic")){mes2="12";dia=30;}
     int diaDecremento=0;
     int mesDecremento=0;
     diaActual=Integer.parseInt(dia2);
     int m2=Integer.parseInt(mes2);
     int a2=Integer.parseInt(anyo2);
     int diaA=dia;
     String mesV="";
     diaActual=diaActual+10;
     if(diaActual>diaA)
      {
            m2=m2+1;
            diaActual=diaActual-diaA;
            String sdia=String.valueOf(diaActual);
             String cero=new String("0");
             diaV=cero.concat(sdia);
             diaDecremento=diaActual;
             diaActual=33;
             mesDecremento=m2;
        }
     if(m2<10)
       {
             String smes=String.valueOf(m2);
             String cero=new String("0");
             mesV=cero.concat(smes);
             mesDecremento=m2;
        }
           if(m2>10)
           {
              mesV=String.valueOf(m2);
              mesDecremento=m2;
           }
     if(diaActual<diaA)
        {
         diaV=String.valueOf(diaActual);
         diaDecremento=diaActual;
         }
          String barra="/";
          String
sFecha=(((diaV.concat(barra)).concat(mesV)).concat(barra)).concat(anyo2);
  Connection conexion,conex;
  PreparedStatement  obtener_grupo;
  PreparedStatement  obtener_existencia; String sD="";
  String mD="";
   for(int i=0;i<10;i++)
    {
      try
      {
        //conexionn =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega","usuario
", "usuario123");
        conexion =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
        obtener_grupo=conexion.prepareStatement("SELECT
Codigo,Ubicacion,fecha_vence FROM detalle_insumo WHERE fecha_vence = ?");
```

```
            if (conexion != null)
           {
                   obtener_grupo.setString(1,sFecha);
                   ResultSet resul= obtener_grupo.executeQuery();
                 if(resul.next()==true)
                  {
                       jtxareaMensaje.append( "Atencion El Insumo Esta
Proximo A vencer"+   "\n\n");
                       boolean seguir=true;
                       while(seguir)
                       {
                        String codigo = resul.getString("Codigo");
                String sUbicacion = resul.getString("Ubicacion");
                String sfecha = resul.getString("fecha_vence");
                  jtxareaMensaje.append("  Codigo= "+
codigo+"\t"+"Ubicacion "+sUbicacion+"\t"+"Fecha a vencer
"+sfecha+"\n\n");
                   seguir=resul.next();
               }
 }
    resul.close();
    obtener_grupo.close();
    conexion.close();
      }
   }
   catch(SQLException ex)
    {
     ex.printStackTrace( );
    }
      diaDecremento=diaDecremento-1;
    if(diaDecremento==0)
      {
      mesDecremento=mesDecremento-1;
        diaDecremento=dia;
      }
    if(diaDecremento<10 && diaDecremento >=1)
    {
     String c0=new String("0");
     String df=String.valueOf(diaDecremento);
     sD=c0.concat(df);
    }
    if(diaDecremento>=10)
    {
      sD=String.valueOf(diaDecremento);
    }
    if(mesDecremento<10 && mesDecremento >=1)
    {
     String c10=new String("0");
     String mf=String.valueOf(mesDecremento);
     mD=c10.concat(mf);
     }
     if(mesDecremento>=10)
     {
      mD=String.valueOf(mesDecremento);
     }
```

```
    String
sFechaDecremen=(((sD.concat(barra)).concat(mD)).concat(barra)).concat(any
o2);
    sFecha=sFechaDecremen;}
  }
  }//FIN DE LA CLASE SIBLAC
 }//Fin de la interfaz siblac
```

INTERFAZ PARA LA ENTRADA DE INSUMOS



Principales Funciones de Entrada de Insumos:

```
public void inabilitarTodo()
{
      combo_grupo.setEnabled(false);
      tx_nombre_gen.setEnabled(false);
      tx_presentacion.setEnabled(false);
      combo_cod.setEnabled(false);
      combo_nombre_gen.setEnabled(false);
```

```
      tx_um.setEnabled(false);
      tx_marca.setEnabled(false);
      tx_empaque.setEnabled(false);
      tx_fuente_financ.setEnabled(false);
      tx_fecha_ven.setEnabled(false);
      tx_ubicacion.setEnabled(false);
      tx_lote.setEnabled(false);
      tx_cantidad.setEnabled(false);
      tx_cost_un.setEnabled(false);
      btagregar.setEnabled(false);
      btguardar.setEnabled(false);
      btcancelar.setEnabled(false);
      btborrar.setEnabled(false);
      btnuevo.setEnabled(false);
      tx_nro_entrada.setEnabled(false);
      tx_cost_t.setEditable(false);
      combo_nombre_gen.setEditable(true);

}



public void agregarDatosTabla()
{
      String vence1=String.valueOf(tx_fecha_ven.getText());
      if( tx_marca.getText().length()==0 ||
tx_empaque.getText().length()==0 || tx_ubicacion.getText().length()==0 ||
tx_lote.getText().length()==0 || tx_cantidad.getText().length()==0 ||
vence1.compareTo("__/__/____")==0 )
      {
       JOptionPane.showMessageDialog(null, "Favor llenar todos los
campos...", "Error", JOptionPane.ERROR_MESSAGE);
         btguardar.setEnabled(false);
      }
       else
       {
        int fila=modelo.getRowCount();
        int col=modelo.getColumnCount();
        int estado=1;
      for(int i=0;i<fila;i++)
      {
      for(int j=0;j<col;j++)
      {
          Object datoCelda1=modelo.getValueAt(i,j);
           if(j==0)
              {
               String Seleccion = (String)combo_cod.getSelectedItem();
               String codigo=String.valueOf(datoCelda1);
               if(codigo.equals(Seleccion))
                {
                 JOptionPane.showMessageDialog(null, "Ese Insumo ya fue
agregado", "Error", JOptionPane.ERROR_MESSAGE);
                     estado=0;
                   }
              }
        }
      }
```

```
          if(estado==1)
           {
            double resul;
            String temp;
            int cant=Integer.parseInt(tx_cantidad.getText());
            resul=
Integer.parseInt(tx_cantidad.getText())*Double.parseDouble(tx_cost_un.get
Text());
                temp=Double.toString(resul);
                tx_cost_t.setText(temp);
              String codig = String.valueOf(combo_cod.getSelectedItem());

            String marca=String.valueOf(tx_marca.getText());
              String empaque=String.valueOf(tx_empaque.getText());
            String fuenteFin=String.valueOf(tx_fuente_financ.getText());
              String ubicacion=String.valueOf(tx_ubicacion.getText());
              String lote=String.valueOf(tx_lote.getText());
              String vence=String.valueOf(tx_fecha_ven.getText());
              String cantidad=String.valueOf(tx_cantidad.getText());
              String costoUnit=String.valueOf(tx_cost_un.getText());
              String
costoT=String.valueOf(tx_cost_t.getText());//Double.parseDouble(tx_cost_t
.getText());
              String id="";
            ///añadimos los datos alas tablas

control.anhadeFila(codig,marca,empaque,fuenteFin,ubicacion,lote,vence,can
tidad,costoUnit,costoT);
            tx_um.setEnabled(false);
              tx_marca.setEnabled(false);
              tx_empaque.setEnabled(false);
              tx_fuente_financ.setEnabled(false);
              tx_fecha_ven.setEnabled(false);
              tx_ubicacion.setEnabled(false);
              tx_lote.setEnabled(false);
              tx_fecha.setEnabled(false);
              tx_hora.setEnabled(false);
              tx_cantidad.setEnabled(false);
              tx_cost_un.setEnabled(false);
              tx_cost_t.setEnabled(false);
              btguardar.setEnabled(true);
              btcancelar.setEnabled(true);
              btnuevo.setEnabled(true);
              btcancelar.setEnabled(true);
              btagregar.setEnabled(false);
              btborrar.setEnabled(true);
              combo_cod.setEnabled(false);
              combo_grupo.setEnabled(false);
            }
     }
}

public void guardarInsumosBD()
  {
        int fila=modelo.getRowCount();
          int col=modelo.getColumnCount();
        String codigo=null;
```

```
        String marca=null;
        String empaque=null;
        String fuenteFin=null;
        String Ubicacion=null;
        String lote=null;
        String vencimiento=null;
        int cantidadInt=0;
        double costoUnitDoub=0;
        double costoTotDoub=0;
        int Id_entadaInt=0;
        int ExistenciaEntrada=0;
        String sId=String.valueOf(tx_nro_entrada.getText());
        double Id=Double.valueOf(sId).doubleValue();
        Connection conexion = null;
          PreparedStatement sentencia0 = null;
          PreparedStatement sentencia = null;
          PreparedStatement sentencia2 = null;
          PreparedStatement sentencia3 = null;
             ////////////// Para la Fecha SQL ///////
             java.sql.Date fechaSql;
             Calendar fechaSistema=Calendar.getInstance();
             java.util.Date fechaJava=fechaSistema.getTime();
             fechaSql=new java.sql.Date(fechaJava.getTime());
             /////////// Para la hora sql //////////////
             Calendar hora=Calendar.getInstance();
         java.sql.Time horaSql=new
java.sql.Time((hora.getTime()).getTime());
          mesAnyo();
       try
       {

       //conexion=DriverManager.getConnection("jdbc:postgresql://localhost
/bodega1","usuario","usuario123");
             conexion =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
             sentencia0 = conexion.prepareStatement("INSERT INTO Entrada
VALUES (?,?,?,?,?)");
           sentencia0.setDouble(1,Id);
           sentencia0.setDate(2,fechaSql);
           sentencia0.setTime(3,horaSql);
           sentencia0.setInt(4,me);
           sentencia0.setInt(5,ae);
           sentencia0.executeUpdate();
           for(int i=0;i<fila;i++)
          {
             for(int j=0;j<col;j++)
            {
             Object datoCelda=modelo.getValueAt(i,j);
               if(j==0)
                {
                 codigo=String.valueOf(datoCelda);
                }
               if(j==1)
                 {
                  marca=String.valueOf(datoCelda);
                 }
```

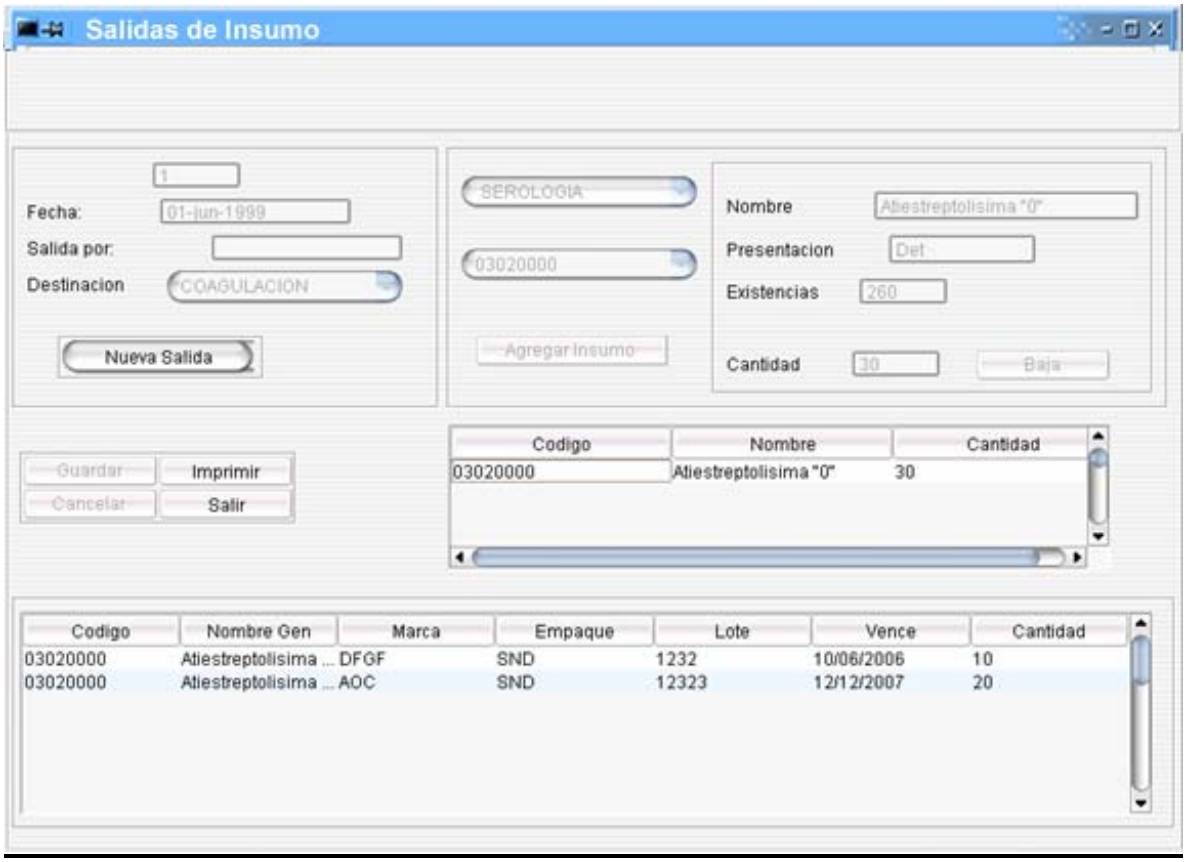
```
            if(j==2)
             {
              empaque=String.valueOf(datoCelda);
             }
            if(j==3)
             {
              fuenteFin=String.valueOf(datoCelda);
             }
            if(j==4)
             {
              Ubicacion=String.valueOf(datoCelda);
             }
            if(j==5)
             {
              lote=String.valueOf(datoCelda);
             }
            if(j==6)
             {
              vencimiento=String.valueOf(datoCelda);
              divideFechaVence(vencimiento);
             }
            if(j==7)
             {
                String cantidad=String.valueOf(datoCelda);
              cantidadInt=Integer.parseInt(cantidad.trim());
               }
            if(j==8)
             {
              String costoUnit=String.valueOf(datoCelda);
              costoUnitDoub=Double.valueOf(costoUnit).doubleValue();
             }
            if(j==9)
             {
              String costoTot=String.valueOf(datoCelda);
              costoTotDoub=Double.valueOf(costoTot).doubleValue();
              String sfecha="";

     if(vence==0)
       {
            vence=Id;
            FechaVence="";
            vencimiento=new String("");
       }
    else{
      //sfecha=;
      }
      sentencia = conexion.prepareStatement("INSERT INTO detalle_insumo
VALUES (?,?,?,?,?,?,?,?,?,?,?,?)");
            sentencia.setString(1,codigo);
            sentencia.setString(2,marca);
            sentencia.setString(3,empaque);
            sentencia.setString(4,fuenteFin);
            sentencia.setString(5,Ubicacion);
            sentencia.setString(6,lote);
            sentencia.setString(7,vencimiento);
            sentencia.setDouble(8,vence);
            sentencia.setInt(9,cantidadInt);
```

```
            sentencia.setDouble(10,costoUnitDoub);
            sentencia.setDouble(11,costoTotDoub);
            sentencia.setDouble(12,Id);
            ////////////ejecutamos  la consulta////
            sentencia.executeUpdate();
//////////Los datos los guardamo eb detalle entrada ///////
        sentencia2 = conexion.prepareStatement("INSERT INTO
detalle_entrada VALUES (?,?,?,?,?,?,?,?,?,?,?,?)");
        sentencia2.setString(1,codigo);
            sentencia2.setString(2,marca);
            sentencia2.setString(3,empaque);
            sentencia2.setString(4,fuenteFin);
            sentencia2.setString(5,Ubicacion);
            sentencia2.setString(6,lote);
            sentencia2.setString(7,vencimiento);
            sentencia2.setDouble(8,vence);
            sentencia2.setInt(9,cantidadInt);
            sentencia2.setDouble(10,costoUnitDoub);
            sentencia2.setDouble(11,costoTotDoub);
            sentencia2.setDouble(12,Id);
            sentencia2.executeUpdate();

     }////////fin del ultimo if
   } //// fin del segundo for
 }/// fin del primer for
       }
       catch(SQLException ex)
        {
            System.out.println(ex);
        }
       finally
         {
          try
              {
               if(sentencia0 != null)
                  sentencia0.close();
              }

              catch(SQLException ex)
                {
                  System.out.println(ex);
                }
              try
                {
                 if(conexion != null)
                   conexion.close();
                }
                  catch(SQLException ex)
                    {
                     System.out.println(ex);
                    }
       }//fin de finally
 JOptionPane.showMessageDialog(null, "Los Datos Se han Guardado",
"Mensaje", JOptionPane.INFORMATION_MESSAGE);
} //// fin dela funcion

////////////// funcion para comprobar la fecha ////////////////////
```

```
public void compruebaFechaVence(String fechaVence ,String fechaActual)
{
  String fecha1=fechaVence;
  String fecha2=fechaActual;
  String dia1=fecha1.substring(0,2);
  String mes1=fecha1.substring(3,5);
  String anyo1=fecha1.substring(6,10);
  String dia2=fecha2.substring(0,2);String mes2=fecha2.substring(3,6);
  String anyo2=fecha2.substring(7,11);
 ///le asigna el valor correspondiente acada mes
  if(mes2.equals("ene")){mes2=new String("01");}
  if(mes2.equals("feb")){mes2=new String("02");}
  if(mes2.equals("mar")){mes2=new String("03");}
  if(mes2.equals("abr")){mes2=new String("04");}
  if(mes2.equals("may")){mes2=new String("05");}
  if(mes2.equals("jun")){mes2=new String("06");}
  if(mes2.equals("jul")){mes2=new String("07");}
  if(mes2.equals("ago")){mes2=new String("08");}
  if(mes2.equals("sep")){mes2=new String("09");}
  if(mes2.equals("oct")){mes2=new String("10");}
  if(mes2.equals("nov")){mes2=new String("11");}
  if(mes2.equals("dic")){mes2=new String("12");}
  String cadena=(anyo2.concat(mes2)).concat(dia2);
  int fechaEntrada=Integer.parseInt(cadena);
  int d1=Integer.parseInt(dia1);
  int d2=Integer.parseInt(dia2);
  int m1=Integer.parseInt(mes1);
  int m2=Integer.parseInt(mes2);
  int a1=Integer.parseInt(anyo1);
  int a2=Integer.parseInt(anyo2);
  de=d2;
  me=m2;
  ae=a2;
  // se realiza la comparacion para determinr que  la fecha cumpla 6
meses para vencer
 if(a1<a2)//si el anño es menor alertamos
 {
  JOptionPane.showMessageDialog(null, "Ese Año es menor que el año actual
", "Error", JOptionPane.ERROR_MESSAGE);
 }
 if(a1==a2)///si los años son iguales comparamos el dia
 {
  int maux=m2;
  maux=m1-m2;
  int diaaux=d1-d2;
  if(maux<5 || d1<=d2)
   {
    JOptionPane.showMessageDialog(null, "La fecha no cumple con los 6
meses  ", "Error", JOptionPane.ERROR_MESSAGE);
   }
 }
 if(a1>a2)  //si el año vece es mayor se comparan los dias
  {
   int suma=m2+6;
   m2=suma-12;
   int anyo;
   anyo=a1-a2;
```

```
    if(anyo<=1)
     {
      if(m2==m1)
      {
       if(d1<=d2)
        {
              JOptionPane.showMessageDialog(null, "La fecha no cumple con
los 6 meses  ", "Error", JOptionPane.ERROR_MESSAGE);
       }
        }
      if(m1<m2)
       {
            JOptionPane.showMessageDialog(null, "La fecha no cumple con
los 6 meses  ", "Error", JOptionPane.ERROR_MESSAGE);
        }
      }
    }
 }
```

INTERFAZ PARA LA SALIDA



Funciones Principales de Salida de Insimos

```
//////////////////// FUNCIOM PARA DAR DE ALTA SEGUN FECHA SE AP A vecer
///
 public void darDeAlta(String selecionado,int sCantidad)
      {
     String buscar=selecionado;
     int mes=0;
     double fechamenor=0;
     int ExistenciaC=0;
     double idEntrada=0;
     Connection conexion,conexion1,conexion2,conexion3;
     PreparedStatement  obtener_nombre;
     PreparedStatement  obtener_cantidad;
     PreparedStatement  obtener_fechaMenor;
     PreparedStatement  saldo_0Eliminar;
     PreparedStatement  saldo_actulisar;
    // PreparedStatement  saldo_actulisar;
    double fechaCompara=0;
    int cont=0;
     boolean seguir=true;
     boolean seguir2=true;
     try
      {
            conexion =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
obtener_fechaMenor=conexion.prepareStatement("select
id_entrada,vencimiento,Cantidad FROM detalle_insumo where Codigo=?");
        obtener_fechaMenor.setString(1,buscar);
        ResultSet resulFecha= obtener_fechaMenor.executeQuery();
         if (conexion != null)
            {
               if(resulFecha.next()==true)
               {
                 while(seguir2)
                 {
               cont=cont+1;
              fechamenor= resulFecha.getDouble("vencimiento");
               if(cont<=1)
               {
                   fechaCompara=fechamenor;
                   idEntrada= resulFecha.getDouble("id_entrada");
               }
               if(fechamenor<fechaCompara)
               {
                   fechaCompara=fechamenor;
                   idEntrada= resulFecha.getDouble("id_entrada");
               }
               seguir2=resulFecha.next();
              }//fin del while
              cont=0;
              }//fin del if
```

```
            }
          resulFecha.close();
        obtener_fechaMenor.close();
        conexion.close();
        }
        catch(SQLException ex)
            {
            JOptionPane.showMessageDialog(null, "Error al selce facha menor",
"Error", JOptionPane.ERROR_MESSAGE);
            ex.printStackTrace( );
          }
          System.out.println(fechaCompara+"  aqui es otro \n");
       try
          {
               conexion1 =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
            obtener_nombre=conexion1.prepareStatement("SELECT Cantidad FROM
detalle_insumo WHERE Codigo=? and vencimiento =?");
            obtener_nombre.setString(1,buscar);
            obtener_nombre.setDouble(2,fechaCompara);

            ResultSet resul= obtener_nombre.executeQuery();
             if (conexion1 != null)
                {
             while(resul.next())
           {
             ExistenciaC= resul.getInt("Cantidad");
            int saldo=ExistenciaC - sCantidad;
            if(saldo<=0)
            {
                mostrarRequisaTabla(buscar,fechaCompara,ExistenciaC);
            try
             {
              conexion2=
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
              saldo_0Eliminar=conexion2.prepareStatement("DELETE FROM
detalle_insumo WHERE vencimiento=? and Codigo=? and id_entrada=?");
              saldo_0Eliminar.setDouble(1,fechaCompara);
              saldo_0Eliminar.setString(2,buscar);
              saldo_0Eliminar.setDouble(3,idEntrada);
              saldo_0Eliminar.executeUpdate();
              saldo_0Eliminar.close();
              conexion2.close();
             }
           catch(SQLException ex1)
                {
                JOptionPane.showMessageDialog(null, "error al eliminar",
"Error", JOptionPane.ERROR_MESSAGE);
               ex1.printStackTrace( );
              }
              if(saldo<0)
              {
               saldo=saldo * -1;
               darDeAlta(buscar,saldo);
              }
```

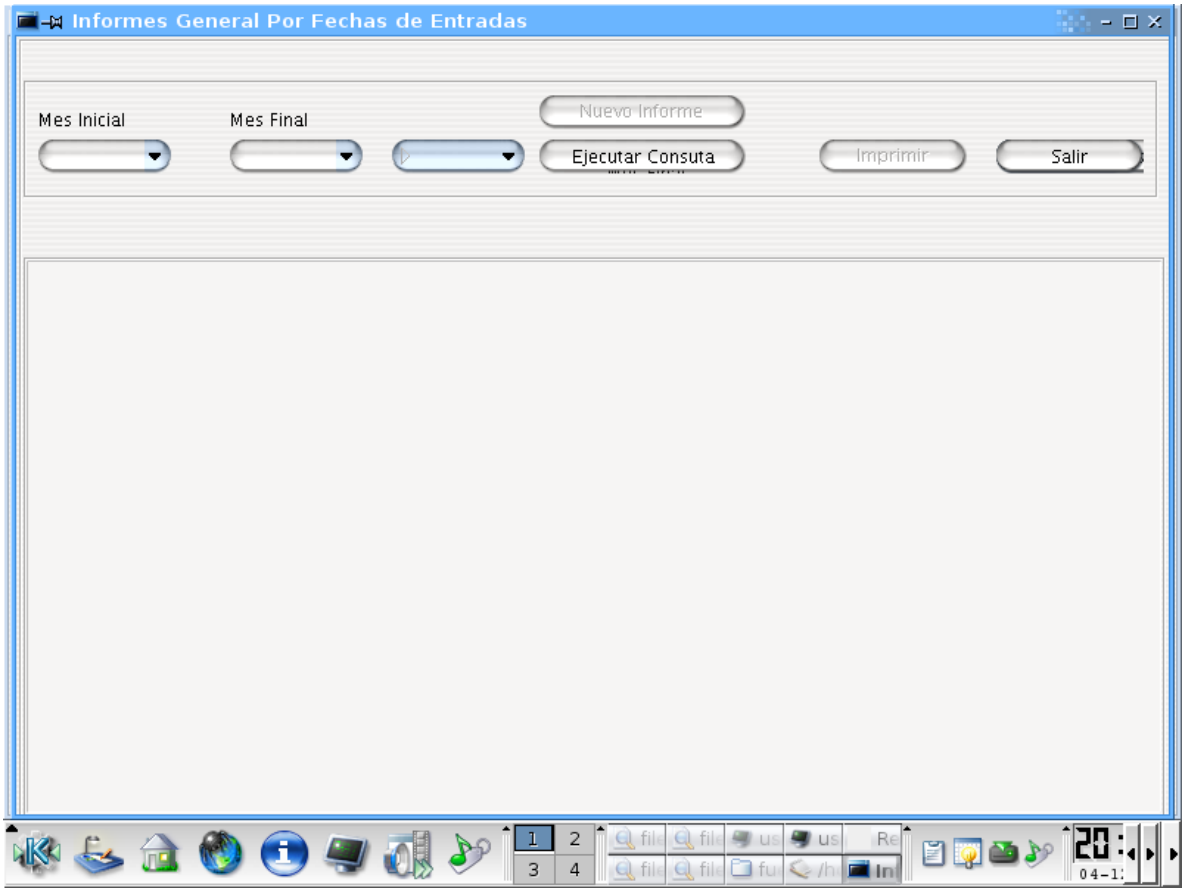
```
          } // fin del if para eliminar registro
          if(saldo>0)
          {
             mostrarRequisaTabla(buscar,fechaCompara,sCantidad);
          try
          {
           conexion3=
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
           saldo_actulisar=conexion3.prepareStatement("UPDATE
detalle_insumo SET Cantidad =? WHERE vencimiento=? and Codigo=?");
           saldo_actulisar.setInt(1,saldo);
           saldo_actulisar.setDouble(2,fechaCompara);
           saldo_actulisar.setString(3,buscar);
           saldo_actulisar.executeUpdate();
           saldo_actulisar.close();
           conexion3.close();
          }
        catch(SQLException ex2)
              {
               JOptionPane.showMessageDialog(null, "error al actualizar",
"Error", JOptionPane.ERROR_MESSAGE);
             ex2.printStackTrace( );
            }
          }
         }
        }
    resul.close();
    obtener_nombre.close();
    conexion1.close();
    }
    catch(SQLException ex)
        {
        JOptionPane.showMessageDialog(null, "error final", "Error",
JOptionPane.ERROR_MESSAGE);
      ex.printStackTrace( );
      }
      }
      ////////////////////// funcion muestra datos requisa

public void mostrarRequisaTabla(String codigo,double fecha,int cantidadE)
{

   String buscar_codigo=codigo;
   int estado=1;
   Connection conexion,conexion1;
   PreparedStatement  obtener_datos;
   PreparedStatement  obtener_nombre;
   String sVencimiento=null;
   String Nombre=null;
   try
      {
        conexion =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
        obtener_nombre=conexion.prepareStatement("SELECT Nombre_generico
FROM Insumos WHERE Codigo=?");
```

```
         obtener_nombre.setString(1,buscar_codigo);
         ResultSet resul1= obtener_nombre.executeQuery();
         if (conexion != null)
             {
              while(resul1.next())
                {
                  Nombre= resul1.getString("Nombre_generico");
                }
             }
             resul1.close();
             obtener_nombre.close();
             conexion.close();
            }
       catch(SQLException ex0)
           {
           ex0.printStackTrace( );
         }
       try
         {
          conexion1 =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
          obtener_datos=conexion1.prepareStatement("SELECT
Marca,Empaque,Fuente_finaciera,Lote,fecha_vence,Cantidad,Costo_unitario
FROM detalle_insumo WHERE Codigo=? and vencimiento=?");
          obtener_datos.setString(1,buscar_codigo);
          obtener_datos.setDouble(2,fecha);
          ResultSet resul= obtener_datos.executeQuery();
          int cont;
          int y=0;
              while(resul.next()) {
          String sMarca = resul.getString("Marca");
          String sEmpaque = resul.getString("Empaque");
          String sFuente_finaciera= resul.getString("Fuente_finaciera");
          String sLote = resul.getString("Lote");
          String  vencimiento=resul.getString("fecha_vence");
          int cantidad=resul.getInt("Cantidad");
          String sCantidad=String.valueOf(cantidadE);
          double Costo_unitario=resul.getDouble("Costo_unitario");
          double costoTotal=cantidadE * Costo_unitario;
          String sCosto_unitario = String.valueOf(Costo_unitario);
          String sCosto_total =String.valueOf(costoTotal);

guardarDetalleBaseDatos(codigo,sMarca,sEmpaque,sFuente_finaciera,sLote,ve
ncimiento,cantidadE,Costo_unitario,costoTotal);
          String a="1";
          if(estado==1)
          {
           control.agregarRegistroInsumo
(buscar_codigo,Nombre,sMarca,sEmpaque,sLote,vencimiento,sCantidad);
          }
        else
        {
           estado=0;
        }
       }
                  resul.close();
```

```
            obtener_datos.close();
            conexion1.close();
      }
       catch(SQLException ex)
          {
          ex.printStackTrace( );
        }

}
public void guardarSalida()
{
      String sId=String.valueOf(tx_numeroSalida.getText());
    double Id=Double.valueOf(sId).doubleValue();
    Connection conexion = null;
      PreparedStatement sentencia0 = null;
      java.sql.Date fechaSql;
      Calendar fechaSistema=Calendar.getInstance();
    java.util.Date fechaJava=fechaSistema.getTime();
    fechaSql=new java.sql.Date(fechaJava.getTime());
    Calendar hora=Calendar.getInstance();
    java.sql.Time horaSql=new java.sql.Time((hora.getTime()).getTime());
    String fecha2=String.valueOf(fecha_Salida.getText());
    String mes2=fecha2.substring(3,6);
    String anyo2=fecha2.substring(7,11);
    if(mes2.equals("ene")){mes2="01";}
    if(mes2.equals("feb")){mes2="02";}
    if(mes2.equals("mar")){mes2="03";}
    if(mes2.equals("abr")){mes2="04";}
    if(mes2.equals("may")){mes2="05";}
    if(mes2.equals("jun")){mes2="06";}
    if(mes2.equals("jul")){mes2="07";}
    if(mes2.equals("ago")){mes2="08";}
    if(mes2.equals("sep")){mes2="09";}
    if(mes2.equals("oct")){mes2="10";}
    if(mes2.equals("nov")){mes2="11";}
    if(mes2.equals("dic")){mes2="12";}
    int m2=Integer.parseInt(mes2);
    int a2=Integer.parseInt(anyo2);
    String salidaPor=String.valueOf(tx_salidaPor.getText());
    String sdestino= (String)comboDestino.getSelectedItem();
    try
      {
            conexion =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
      sentencia0 = conexion.prepareStatement("INSERT INTO Salida VALUES
(?,?,?,?,?,?,?)");
          sentencia0.setDouble(1,Id);
        sentencia0.setString(2, salidaPor);
          sentencia0.setString(3,sdestino);
          sentencia0.setDate(4,fechaSql);
          sentencia0.setTime(5,horaSql);
          sentencia0.setInt(6,m2);
          sentencia0.setInt(7,a2);
      sentencia0.executeUpdate();
      sentencia0.close();
      conexion.close();
```

```
         }
          catch(SQLException ex)
          {
          JOptionPane.showMessageDialog(null, "error al guardar entrada",
"Error", JOptionPane.ERROR_MESSAGE);
          ex.printStackTrace( );
        }

}
public void guardarDetalleBaseDatos(String codigo,String Marca, String
Empaque,String Fuente_finaciera,String Lote,String vencimiento,int
Cantidad,double Costo_unitario,double Costo_total)
{
      String id_salida=String.valueOf(tx_numeroSalida.getText());
      double Id=Double.valueOf(id_salida).doubleValue();
      Connection conexion = null;
    PreparedStatement sentencia = null;
      try
      {
            conexion =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
        sentencia = conexion.prepareStatement("INSERT INTO detalle_salida
VALUES (?,?,?,?,?,?,?,?,?,?)");
        sentencia.setString(1,codigo);
            sentencia.setString(2,Marca);
            sentencia.setString(3,Empaque);
            sentencia.setString(4,Fuente_finaciera);
            sentencia.setString(5,Lote);
            sentencia.setString(6,vencimiento);
            sentencia.setInt(7,Cantidad);
            sentencia.setDouble(8,Costo_unitario);
            sentencia.setDouble(9,Costo_total);
            sentencia.setDouble(10,Id);
            sentencia.executeUpdate();
            sentencia.close();
            conexion.close();
      }
       catch(SQLException ex)
        {
            System.out.println(ex);
        }
}
```

INTERFAZ DE INFORME ENTRADA(funciones principales)



Función Principal:

```
public void mostrarInformeSalidas()
      {
            String sMesInic = (String)comboMesInicio.getSelectedItem();
            String sMesFin = (String)comboMesFinal.getSelectedItem();
            String sAnyo = (String)comboAnyo.getSelectedItem();
            int mesInicio=Integer.parseInt(sMesInic);
            int mesFin=Integer.parseInt(sMesFin);
            int Anyo=Integer.parseInt(sAnyo);
            double idSalida=0;
            int mes=0;
            Connection conexion=null,conexion1=null,conexion2=null;
            PreparedStatement  obtener_Informe;
            PreparedStatement  obtener_detalle;
            PreparedStatement  guardar;
            int c=0;
            double nroRequisa=0;
            int nregistroR=0;
          try
```

```
        {
                  conexion =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
            obtener_Informe=conexion.prepareStatement("SELECT
Salida_por,id_salida, Destinacion, fecha_salida, hora,mes FROM Salida
WHERE  anyo=?");
            obtener_Informe.setInt(1,Anyo);
             ResultSet resul= obtener_Informe.executeQuery();
            if(resul.next()==true)
            {
                  jtxarea_informe.append("\t\t\t INFORME DE LAS SALIDAS
DE INSUMOS POR FECHA"+"\n"+"\t\t\t\t Año  ("+Anyo+")."+"\n\n\n");
             boolean seguir=true;
             while(seguir)
            {
              mes=resul.getInt("mes");
                  if(mes <= mesFin && mes >= mesInicio)
                  {
              idSalida=resul.getDouble("id_salida");
              nregistroR=nregistroR+1;
              jtxarea_informe.append("   Registro No
"+"("+nregistroR+")"+"\tREQUISA N0   "+idSalida+"\tSAlida POR"+"\t"+
"DESTINACION"+"\t"+"FECHA"+"\t"
               +"HORA" + "\n");

jtxarea_informe.append("\t\t\t\t"+resul.getString("Salida_por")+"\t"+resu
l.getString("Destinacion")+"\t"
               +resul.getDate("fecha_salida")+"\t "
+resul.getTime("hora")+"\n\n");
         try
         {
          conexion1=
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
          obtener_detalle=conexion1.prepareStatement("SELECT
Codigo,Marca,Empaque,fuente_financiera,Lote,Vencimiento,Cantidad FROM
detalle_salida WHERE id_salida=?");
          obtener_detalle.setDouble(1,idSalida);
          ResultSet resul1= obtener_detalle.executeQuery();
          jtxarea_informe.append(" \t"+" Codigo "+"\t"+" Marca
"+"\t"+"Empaque"+"\t"+"fuente_financiera"+"
"+"vencimiento"+"\t"+"Cantidad"+"\n");
            while(resul1.next())
             {
                  c=c+1;
      jtxarea_informe.append("\t"+resul1.getString("Codigo")+"\t"+resul1.
getString("Marca")+"\t"+resul1.getString("Empaque")+"\t"+resul1.getString
("fuente_financiera")+"\t"+resul1.getString("vencimiento")+"\t"+resul1.ge
tInt("Cantidad")+"\n");
             }
            jtxarea_informe.append("\n");
           resul1.close();
           obtener_detalle.close();
           conexion1.close();
            }
    catch(SQLException ex1)
```

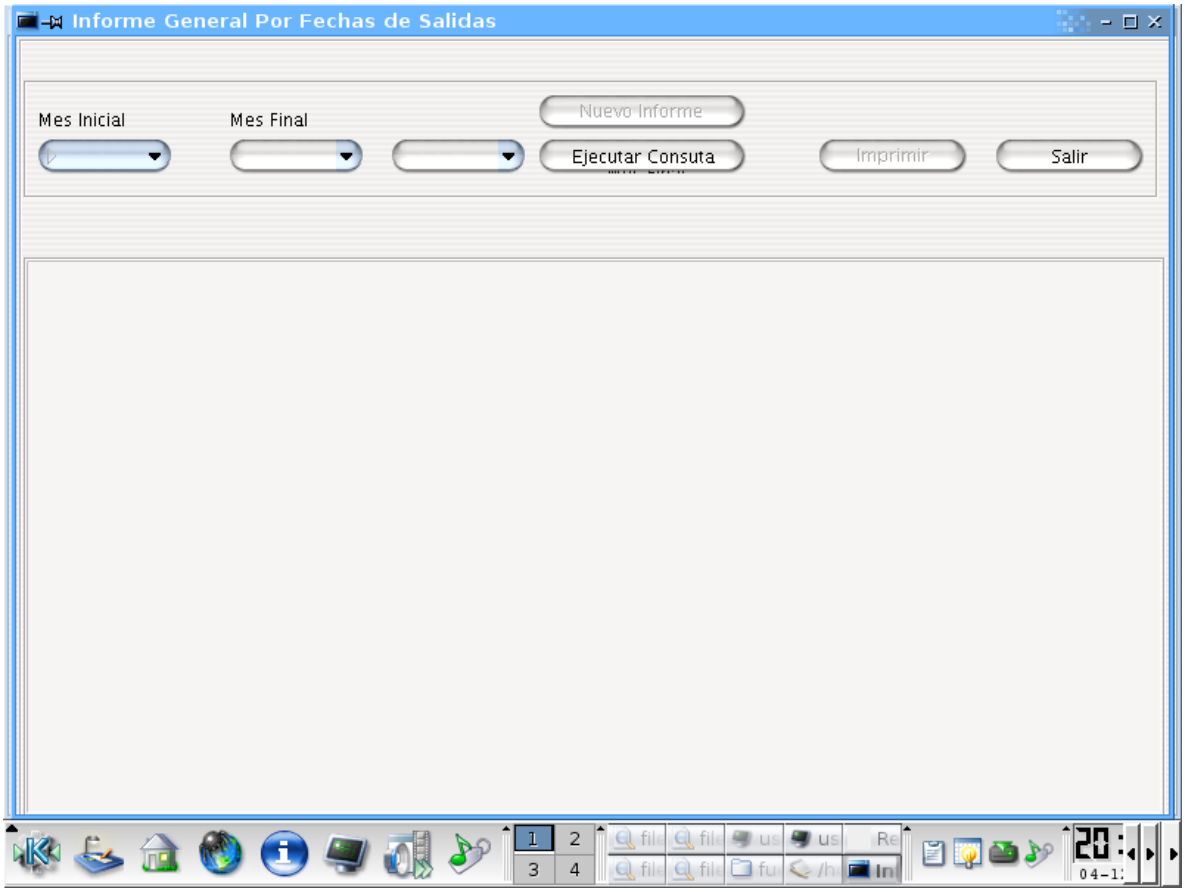
```
        {
        ex1.printStackTrace( );
      }
            }//fin del if
               seguir=resul.next();
            }//fin del while
    }
            else
            {
            JOptionPane.showMessageDialog(null, "No Exisren Datos  ",
"Error", JOptionPane.ERROR_MESSAGE);
      }
    resul.close();
    obtener_Informe.close();
    conexion.close();
    }//fin del try
    catch(SQLException ex)
        {
        ex.printStackTrace( );
      }}
public void ImprimirInforme()
    {
      Font fuente = new Font("Dialog", Font.PLAIN, 10);
        PrintJob pj;
        Graphics pagina;
        pj = Toolkit.getDefaultToolkit().getPrintJob(new Frame(), "SCAT",
null);
        try
            {
                  pagina = pj.getGraphics();
                  pagina.setFont(fuente);
                  pagina.setColor(Color.black);
                  JFormattedTextField fecha= new JFormattedTextField(new
java.util.Date( ));
            String sfecha=String.valueOf(fecha.getText());
            pagina.drawString(sfecha,30, 50);
                  String texto="";
                  String sMesInic =
(String)comboMesInicio.getSelectedItem();
            String sMesFin = (String)comboMesFinal.getSelectedItem();
                String sAnyo = (String)comboAnyo.getSelectedItem();
            int mesInicio=Integer.parseInt(sMesInic);
            int mesFin=Integer.parseInt(sMesFin);
            int Anyo=Integer.parseInt(sAnyo);
                double idEntrada=0;
                  int j=0;
                  Connection conex=null,conexion1=null,conexion2=null;
                  PreparedStatement  obtener_Informe;
                  PreparedStatement  obtener_detalle;
                  PreparedStatement  guardar;
                int c=0;
                int mes=0;
                  double nroRequisa=0;
                  int nregistroR=0;
                try
                  {
```

```
                conex =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
            obtener_Informe=conex.prepareStatement("SELECT
Salida_por,id_salida, Destinacion, fecha_salida, hora,mes FROM Salida
WHERE  anyo=?");
            obtener_Informe.setInt(1,Anyo);
            ResultSet resul= obtener_Informe.executeQuery();
            if(resul.next()==true)
            {
                  texto="INFORME DE LAS SALIDAS DE INSUMOS POR FECHA";
                  j=55;
                  pagina.drawString(texto, 180,j);j=j+20;
                  pagina.drawString("Año", 250,j);
                  pagina.drawString(sAnyo, 300,j); j=j+20;
               boolean seguir=true;
           while(seguir)
           {
            mes=resul.getInt("mes");
            if(mes <= mesFin && mes >= mesInicio)
            {
              idEntrada=resul.getDouble("id_salida");
              nregistroR=nregistroR+1;
              texto="SALIDA Nro";
              pagina.drawString(texto,30,j);
              texto="SALIDA POR";
              pagina.drawString(texto,100,j);
              texto="Destinacion";
              pagina.drawString(texto,300,j);
              texto="FECHA";
              pagina.drawString(texto,400,j);
              texto="HORA";
              pagina.drawString(texto,480,j); j=j+20;
              texto=String.valueOf(idEntrada);
              pagina.drawString(texto,30,j);
              texto=String.valueOf(resul.getString("Salida_por"));
              pagina.drawString(texto,100,j);
              texto=String.valueOf(resul.getString("Destinacion"));
              pagina.drawString(texto,300,j);
              texto=String.valueOf(resul.getDate("fecha_salida"));
              pagina.drawString(texto,400,j);
              texto=String.valueOf(resul.getTime("hora"));
              pagina.drawString(texto,480,j);j=j+20;
              try
                  {
                    conexion1=
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
                    obtener_detalle=conexion1.prepareStatement("SELECT
Codigo,Marca,Empaque,vencimiento,Cantidad FROM detalle_salida WHERE
id_salida=?");
                  obtener_detalle.setDouble(1,idEntrada);
                  ResultSet resul1= obtener_detalle.executeQuery();
                  texto="Codigo";
                  pagina.drawString(texto,30,j);//j=j+20;
                  texto="Marca";
                  pagina.drawString(texto,100,j);
```

```
                texto="Empaque";
                pagina.drawString(texto,170,j);
                texto="Vencimiento";
                  pagina.drawString(texto,240,j);
                texto="Cantidad";
                pagina.drawString(texto,310,j);
                j=j+20;
                while(resul1.next())
                {
                   c=c+1;
                 texto=resul1.getString("Codigo");
                   pagina.drawString(texto,30,j);//j=j+20;
                 texto=resul1.getString("Marca");
                 pagina.drawString(texto,100,j);//j=j+20;
                 texto=resul1.getString("Empaque");
                 pagina.drawString(texto,170,j);//j=j+20;
                 texto=resul1.getString("vencimiento");
                 pagina.drawString(texto,240,j);//j=j+20;
                 texto=String.valueOf(resul1.getInt("Cantidad"));
                 pagina.drawString(texto,310,j);j=j+30;
              }
            resul1.close();
            obtener_detalle.close();
            conexion1.close();
             }
    catch(SQLException ex1)
        {
        ex1.printStackTrace( );
      }
         }//fin del if
               seguir=resul.next();
            }
    }
            else
            {
            JOptionPane.showMessageDialog(null, "No Exisren Datos  ",
"Error", JOptionPane.ERROR_MESSAGE);
            }

      resul.close();
    obtener_Informe.close();
    conex.close();
   }//fin del try
    catch(SQLException ex)
        {
        ex.printStackTrace( );
      }
                if(j==700)
                {
                //    pj.next();
               //this.PAGE_EXISTS;
                }
                pagina.dispose();
                pj.end();
            }catch(Exception e)
            {
```

```
            JOptionPane.showMessageDialog(null, "La Impresion ha sido
cancelada ", "Error", JOptionPane.ERROR_MESSAGE);
      }
    }
```

INTERFAZ PARA LOS INFORMES DE SALIDA



Función Principal:

```
public void mostrarInformeSalidas()
      {
            String sMesInic = (String)comboMesInicio.getSelectedItem();
            String sMesFin = (String)comboMesFinal.getSelectedItem();
            String sAnyo = (String)comboAnyo.getSelectedItem();
            int mesInicio=Integer.parseInt(sMesInic);
            int mesFin=Integer.parseInt(sMesFin);
            int Anyo=Integer.parseInt(sAnyo);
            double idSalida=0;
            int mes=0;
            Connection conexion=null,conexion1=null,conexion2=null;
            PreparedStatement  obtener_Informe;
            PreparedStatement  obtener_detalle;
            PreparedStatement  guardar;
            int c=0;
```

```
            double nroRequisa=0;
            int nregistroR=0;
          try
         {
                  conexion =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
            obtener_Informe=conexion.prepareStatement("SELECT
Salida_por,id_salida, Destinacion, fecha_salida, hora,mes FROM Salida
WHERE  anyo=?");
            obtener_Informe.setInt(1,Anyo);
             ResultSet resul= obtener_Informe.executeQuery();
            if(resul.next()==true)
            {
                  jtxarea_informe.append("\t\t\t INFORME DE LAS SALIDAS
DE INSUMOS POR FECHA"+"\n"+"\t\t\t\t Año  ("+Anyo+")."+"\n\n\n");

             boolean seguir=true;
             while(seguir)
            {
              mes=resul.getInt("mes");
                  if(mes <= mesFin && mes >= mesInicio)
                  {
              idSalida=resul.getDouble("id_salida");
              nregistroR=nregistroR+1;
              String raya1="\t\t\t---------------------------------------
----------------";
              jtxarea_informe.append("   Registro No
"+"("+nregistroR+")"+"\tREQUISA N0   "+idSalida+"\tSAlida POR"+"\t"+
"DESTINACION"+"\t"+"FECHA"+"\t"
              +"HORA" + "\n");

jtxarea_informe.append("\t\t\t\t"+resul.getString("Salida_por")+"\t"+resu
l.getString("Destinacion")+"\t"
              +resul.getDate("fecha_salida")+"\t "
+resul.getTime("hora")+"\n\n");
         try
         {
          conexion1=
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
          obtener_detalle=conexion1.prepareStatement("SELECT
Codigo,Marca,Empaque,fuente_financiera,Lote,Vencimiento,Cantidad FROM
detalle_salida WHERE id_salida=?");
          obtener_detalle.setDouble(1,idSalida);
          ResultSet resul1= obtener_detalle.executeQuery();
          jtxarea_informe.append(" \t"+" Codigo "+"\t"+" Marca
"+"\t"+"Empaque"+"\t"+"fuente_financiera"+"
"+"vencimiento"+"\t"+"Cantidad"+"\n");
            while(resul1.next())
             {
                  c=c+1;

      jtxarea_informe.append("\t"+resul1.getString("Codigo")+"\t"+resul1.
getString("Marca")+"\t"+resul1.getString("Empaque")+"\t"+resul1.getString
("fuente_financiera")+"\t"+resul1.getString("vencimiento")+"\t"+resul1.ge
tInt("Cantidad")+"\n");
```

```
            }
            jtxarea_informe.append("\n");
           resul1.close();
           obtener_detalle.close();
           conexion1.close();
            }
    catch(SQLException ex1)
        {
        ex1.printStackTrace( );
      }
            }//fin del if
               seguir=resul.next();
            }//fin del while
    }
            else
            {
            JOptionPane.showMessageDialog(null, "No Exisren Datos  ",
"Error", JOptionPane.ERROR_MESSAGE);
      }
    resul.close();
    obtener_Informe.close();
    conexion.close();
    }//fin del try
    catch(SQLException ex)
        {
        ex.printStackTrace( );
      }

 }

public void cargarComboMes()
      {
      Connection conn= null;
      try
               {
                  String consulta= "SELECT DISTINCT Salida.mes FROM
Salida ORDER BY Salida.mes";
                  //conn =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega","usuario
", "usuario123");
                  conn =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
             if (conn != null)
                  {
                        Statement stmt = conn.createStatement();
                        ResultSet datcodigo  =
stmt.executeQuery(consulta);
                        comboMesInicio.setToolTipText("Selecione un Grupo
");
                              while (datcodigo.next())
                              {
                        int mes = datcodigo.getInt("mes");
                               String sMes=String.valueOf(mes);
                              comboMesInicio.addItem(sMes);
                              }
                       datcodigo.close();
```

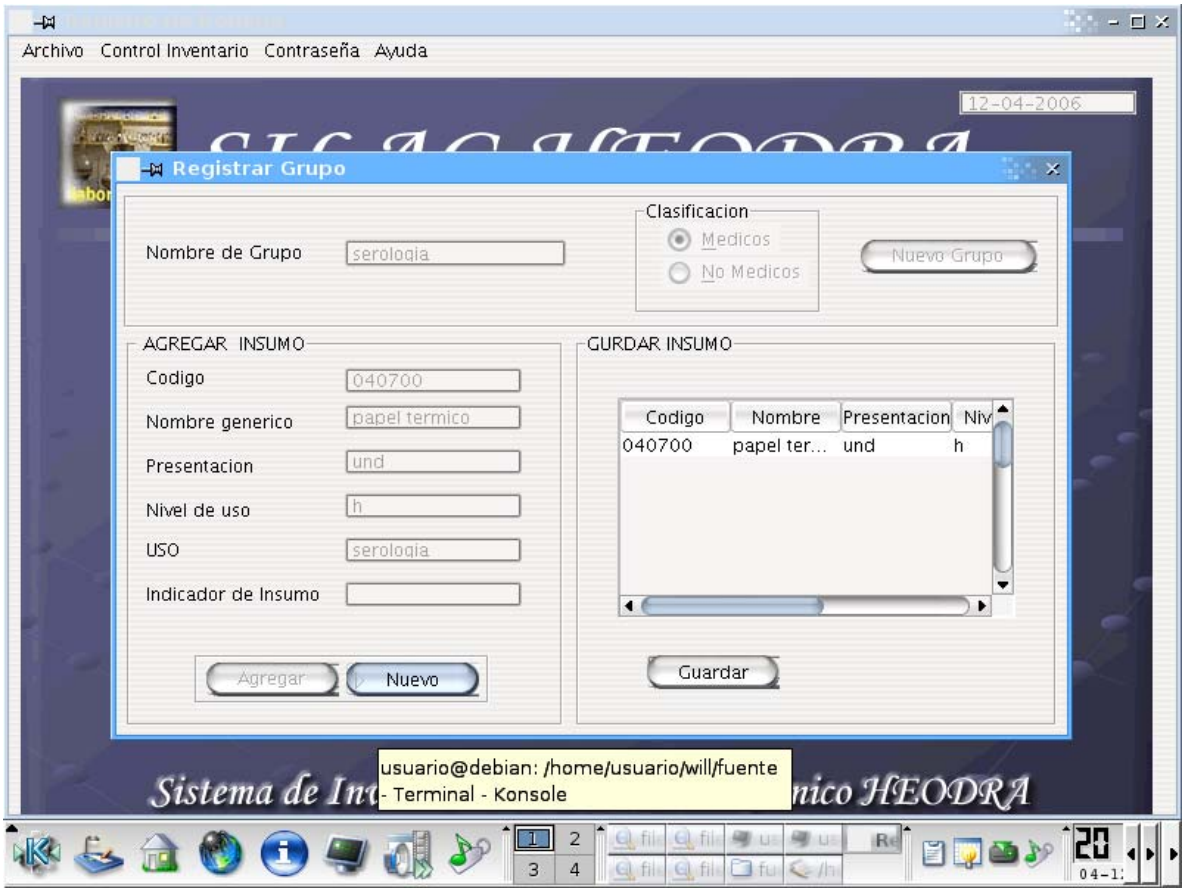
```
                stmt.close();
                  conn.close();
                }
              }
            catch(SQLException ex)
              {
                          ex.printStackTrace( );
            }
Connection conexion2= null;
      try
     {
       String consulta= "SELECT DISTINCT Salida.mes FROM Salida ORDER BY
Salida.mes desc";
       //conn =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega","usuario
", "usuario123");
       conexion2 =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
       if (conexion2 != null)
            {
            Statement stmt2 = conexion2.createStatement();
                  ResultSet result  = stmt2.executeQuery(consulta);
                  comboMesInicio.setToolTipText("Selecione un Grupo ");
                  while (result.next()) {
              int mes = result.getInt("mes");
              String sMesF=String.valueOf(mes);
              comboMesFinal.addItem(sMesF);
            }
                    result.close();
                  stmt2.close();
                  conexion2.close();
             }

      }
      catch(SQLException ex)
          {
           ex.printStackTrace( );
          }
   Connection conexion3= null;
      try
     {
       String consulta= "SELECT DISTINCT Salida.anyo FROM Salida ORDER BY
Salida.anyo desc";
       //conn =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega","usuario
", "usuario123");
       conexion3 =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
       if (conexion3 != null)
            {
            Statement stmt3 = conexion3.createStatement();
            ResultSet result  = stmt3.executeQuery(consulta);
            comboMesInicio.setToolTipText("Selecione un Grupo ");
                  while (result.next())
                  {
```

```
                int anyo = result.getInt("anyo");
             String sAnyo=String.valueOf(anyo);
                   comboAnyo.addItem(sAnyo);
               }
                           result.close();
                         stmt3.close();
                         conexion3.close();
               }
        }
        catch(SQLException ex)
            {
           ex.printStackTrace( );
          }
        }

        public void ImprimirInforme()
      {
        Font fuente = new Font("Dialog", Font.PLAIN, 10);
          PrintJob pj;
          Graphics pagina;
          pj = Toolkit.getDefaultToolkit().getPrintJob(new Frame(), "SCAT",
null);
          try
              {
                   pagina = pj.getGraphics();
                   pagina.setFont(fuente);
                   pagina.setColor(Color.black);
                   JFormattedTextField fecha= new JFormattedTextField(new
java.util.Date( ));
              String sfecha=String.valueOf(fecha.getText());
              pagina.drawString(sfecha,30, 50);
                   String texto="";
                   String sMesInic =
(String)comboMesInicio.getSelectedItem();
              String sMesFin = (String)comboMesFinal.getSelectedItem();
                  String sAnyo = (String)comboAnyo.getSelectedItem();
              int mesInicio=Integer.parseInt(sMesInic);
              int mesFin=Integer.parseInt(sMesFin);
              int Anyo=Integer.parseInt(sAnyo);
                  double idEntrada=0;
                    int j=0;
                    Connection conex=null,conexion1=null,conexion2=null;
                    PreparedStatement  obtener_Informe;
                    PreparedStatement  obtener_detalle;
                    PreparedStatement  guardar;
                  int c=0;
                  int mes=0;
                    double nroRequisa=0;
                    int nregistroR=0;
                  try
                    {
                    conex =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
              obtener_Informe=conex.prepareStatement("SELECT
Salida_por,id_salida, Destinacion, fecha_salida, hora,mes FROM Salida
WHERE  anyo=?");
```

```
            obtener_Informe.setInt(1,Anyo);
            ResultSet resul= obtener_Informe.executeQuery();
            if(resul.next()==true)
            {
                  texto="INFORME DE LAS SALIDAS DE INSUMOS POR FECHA";
                  j=55;
                  pagina.drawString(texto, 180,j);j=j+20;
                  pagina.drawString("Año", 250,j);
                  pagina.drawString(sAnyo, 300,j); j=j+20;
                boolean seguir=true;
           while(seguir)
           {
            mes=resul.getInt("mes");
            if(mes <= mesFin && mes >= mesInicio)
            {
              idEntrada=resul.getDouble("id_salida");
              nregistroR=nregistroR+1;
              texto="SALIDA Nro";
              pagina.drawString(texto,30,j);
              texto="SALIDA POR";
              pagina.drawString(texto,100,j);
              texto="Destinacion";
              pagina.drawString(texto,300,j);
              texto="FECHA";
              pagina.drawString(texto,400,j);
              texto="HORA";
              pagina.drawString(texto,480,j); j=j+20;
              texto=String.valueOf(idEntrada);
              pagina.drawString(texto,30,j);
              texto=String.valueOf(resul.getString("Salida_por"));
              pagina.drawString(texto,100,j);
              texto=String.valueOf(resul.getString("Destinacion"));
              pagina.drawString(texto,300,j);
              texto=String.valueOf(resul.getDate("fecha_salida"));
              pagina.drawString(texto,400,j);
              texto=String.valueOf(resul.getTime("hora"));
              pagina.drawString(texto,480,j);j=j+20;
              try
                  {
                    conexion1=
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
                    obtener_detalle=conexion1.prepareStatement("SELECT
Codigo,Marca,Empaque,vencimiento,Cantidad FROM detalle_salida WHERE
id_salida=?");
                  obtener_detalle.setDouble(1,idEntrada);
                  ResultSet resul1= obtener_detalle.executeQuery();
                  texto="Codigo";
                  pagina.drawString(texto,30,j);//j=j+20;
                  texto="Marca";
                  pagina.drawString(texto,100,j);
                  texto="Empaque";
                  pagina.drawString(texto,170,j);
                  texto="Vencimiento";
                    pagina.drawString(texto,240,j);
                  texto="Cantidad";
                  pagina.drawString(texto,310,j);
```

```
                j=j+20;
                while(resul1.next())
                {
                   c=c+1;
                 texto=resul1.getString("Codigo");
                   pagina.drawString(texto,30,j);//j=j+20;
                 texto=resul1.getString("Marca");
                 pagina.drawString(texto,100,j);//j=j+20;
                 texto=resul1.getString("Empaque");
                 pagina.drawString(texto,170,j);//j=j+20;
                 texto=resul1.getString("vencimiento");
                 pagina.drawString(texto,240,j);//j=j+20;
                 texto=String.valueOf(resul1.getInt("Cantidad"));
                 pagina.drawString(texto,310,j);j=j+30;
              }
            resul1.close();
            obtener_detalle.close();
            conexion1.close();
             }
    catch(SQLException ex1)
        {
        ex1.printStackTrace( );
      }
         }//fin del if
               seguir=resul.next();
            }
    }
            else
            {
            JOptionPane.showMessageDialog(null, "No Exisren Datos  ",
"Error", JOptionPane.ERROR_MESSAGE);
            }

      resul.close();
    obtener_Informe.close();
    conex.close();
   }//fin del try
    catch(SQLException ex)
        {
        ex.printStackTrace( );
      }
                if(j==700)
                {
                //    pj.next();
              //this.PAGE_EXISTS;
                }

                pagina.dispose();
                pj.end();
            }catch(Exception e)
            {
            JOptionPane.showMessageDialog(null, "La Impresion ha sido
cancelada ", "Error", JOptionPane.ERROR_MESSAGE);
      }
    }
```

INTERFAZ PARA AGREGAR UN NUEVO GRUPO A LA LISTA BASICA



Funcion Principal:

```
///// funcion para guardar//////
public void guardarGrupo()
{
 int fila=modelo.getRowCount();
 int col=modelo.getColumnCount();
 String codigo=null;
 String nombreGenerico=null;
 String presentacion=null;
 String nivelUso=null;
 String uso=null;
 String indicadorInsumo=null;
```

```
 int existencia=0;
 int compruebaGuardar=1;
 Connection conexion = null;
 PreparedStatement sentencia0 = null;
 PreparedStatement sentencia = null;
 PreparedStatement sentencia2 = null;
 // Para la Fecha SQL ///////
 java.sql.Date fechaSql;
 Calendar fechaSistema=Calendar.getInstance();
 java.util.Date fechaJava=fechaSistema.getTime();
 fechaSql=new java.sql.Date(fechaJava.getTime());
 /////////// Para la hora sql //////////////
 Calendar hora=Calendar.getInstance();
 java.sql.Time horaSql=new java.sql.Time((hora.getTime()).getTime());
try
 {

//conexion=DriverManager.getConnection("jdbc:postgresql://localhost/bodeg
a","usuario","usuario123");
  conexion =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
  sentencia0 = conexion.prepareStatement("INSERT INTO ListaBasica VALUES
(?,?)");
  sentencia0.setString(1,tx_grupo.getText());
  sentencia0.setString(2,clasificacion);
  sentencia0.executeUpdate();
 }
 catch(SQLException ex)
       {
          compruebaGuardar=0;
        inabilitarTodo();
            System.out.println(ex);
       }
 try
  {
      for(int i=0;i<fila;i++)
      {
      for(int j=0;j<col;j++)
      {
       Object datoCelda=modelo.getValueAt(i,j);
         if(j==0)
          {
           codigo=String.valueOf(datoCelda);
          }
         if(j==1)
          {
           nombreGenerico=String.valueOf(datoCelda);
          }
         if(j==2)
          {
           presentacion=String.valueOf(datoCelda);
          }
        if(j==3)
         {
          nivelUso=String.valueOf(datoCelda);
         }
```

```
        if(j==4)
         {
          uso=String.valueOf(datoCelda);
         }
        if(j==5)
         {
          indicadorInsumo=String.valueOf(datoCelda);
          sentencia = conexion.prepareStatement("INSERT INTO Insumos
VALUES (?,?,?,?,?,?,?)");
              sentencia.setString(1,codigo);
              sentencia.setString(2,nombreGenerico);
              sentencia.setString(3,presentacion);
              sentencia.setString(4,nivelUso);
              sentencia.setString(5,uso);
              sentencia.setString(6,indicadorInsumo);
            sentencia.setString(7,tx_grupo.getText());
              sentencia.executeUpdate();//ejecutamos la consulta

         }//fin del ultimo if
       } //// fin del segundo for
     }/// fin del primer for

   }
 catch(SQLException ex)
  {
   compruebaGuardar=0;
   try{
        conexion =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
      sentencia2 = conexion.prepareStatement("DELETE FROM ListaBasica
WHERE Nombre_grupo=?");
           sentencia2.setString(1,tx_grupo.getText());
        sentencia2.executeUpdate();
        sentencia2.close();
        inabilitarTodo();
       }
  catch(SQLException ex1)
       {
         compruebaGuardar=0;
           System.out.println(ex1);
         }
           System.out.println(ex);
       }
   finally
       {
        try
       {
        if(sentencia0 != null)
         sentencia0.close();
        }
       catch(SQLException ex)
        {
         System.out.println(ex);
        }
       try
        {
```

```
          if(conexion != null)
          conexion.close();
         }
       catch(SQLException ex)
        {
       System.out.println(ex);
        }
       if(compruebaGuardar==1)
        {
         JOptionPane.showMessageDialog(null, "Los datos han Sido
Guardados", "Mensaje", JOptionPane.INFORMATION_MESSAGE);
      }
      else
         {
             JOptionPane.showMessageDialog(null, "No se podra
guardar..Datos duplicados", "Mensaje", JOptionPane.INFORMATION_MESSAGE);
       }
   }//fin de finally
 }
}
```

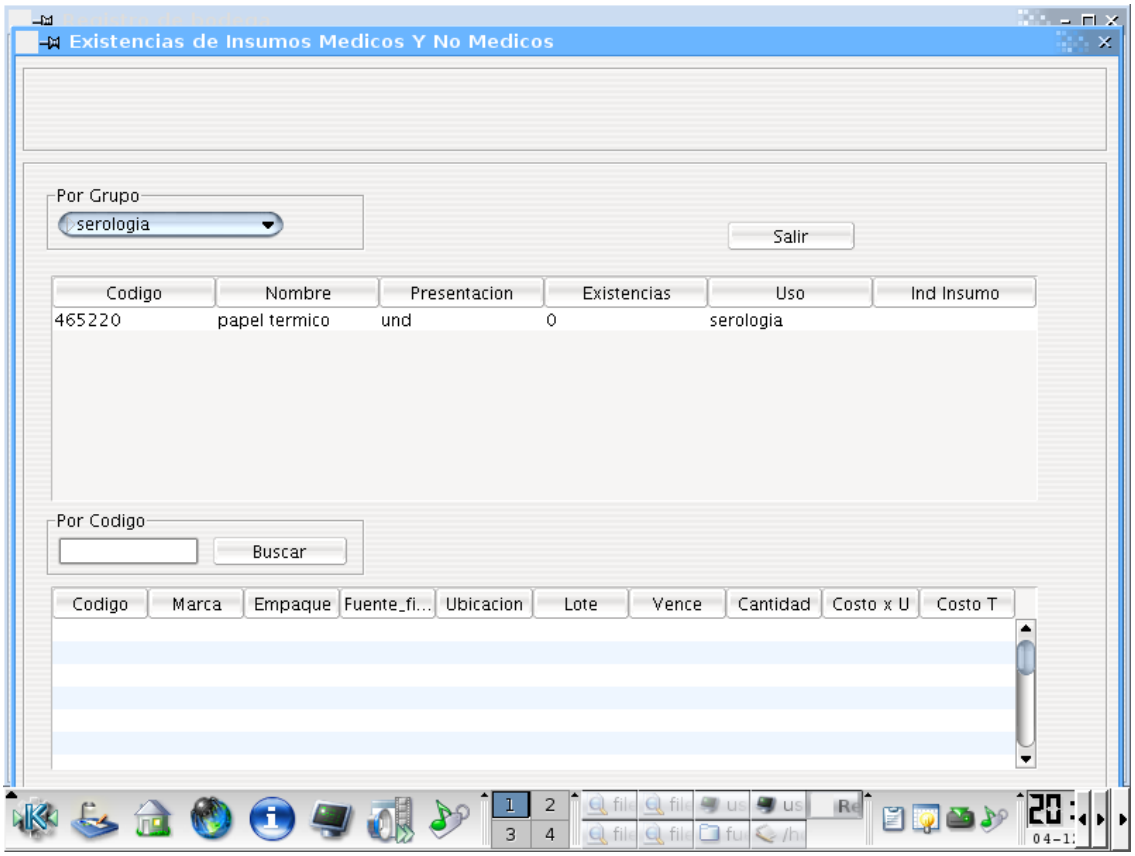
INTERFAZ DE CREAR UN NUEVO USUARIO

Función para Guardar un Nuevo Usuario.

```
public void guardarUsuario(String usuario,String Clave)
      {
       String claveAnterior=null;
       Connection conexion;
     PreparedStatement  ingresarUsuario;
     try
      {
```

```
            String tipoUsuario =
(String)comboTipoUsuario.getSelectedItem();
        conexion=
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
        ingresarUsuario=conexion.prepareStatement("INSERT INTO acceso
VALUES (?,?,?)");
        ingresarUsuario.setString(1,usuario);
        ingresarUsuario.setString(2,Clave);
        ingresarUsuario.setString(3,tipoUsuario);
        ingresarUsuario.executeUpdate();
        ingresarUsuario.close();
        conexion.close();
        JOptionPane.showMessageDialog(null, "El Usuario se ha Creado",
"Mensaje", JOptionPane.INFORMATION_MESSAGE);
          btIngresar.setEnabled(false);
       }
      catch(SQLException ex2)
           {
            ex2.printStackTrace( );
                  JOptionPane.showMessageDialog(null, "Existe una
contraseña igual que la Ingresada ", "Atencion",
JOptionPane.ERROR_MESSAGE);
           }
    }
```

INTERFAZ PARA MOSTRAR LAS EXISTENCIAS.

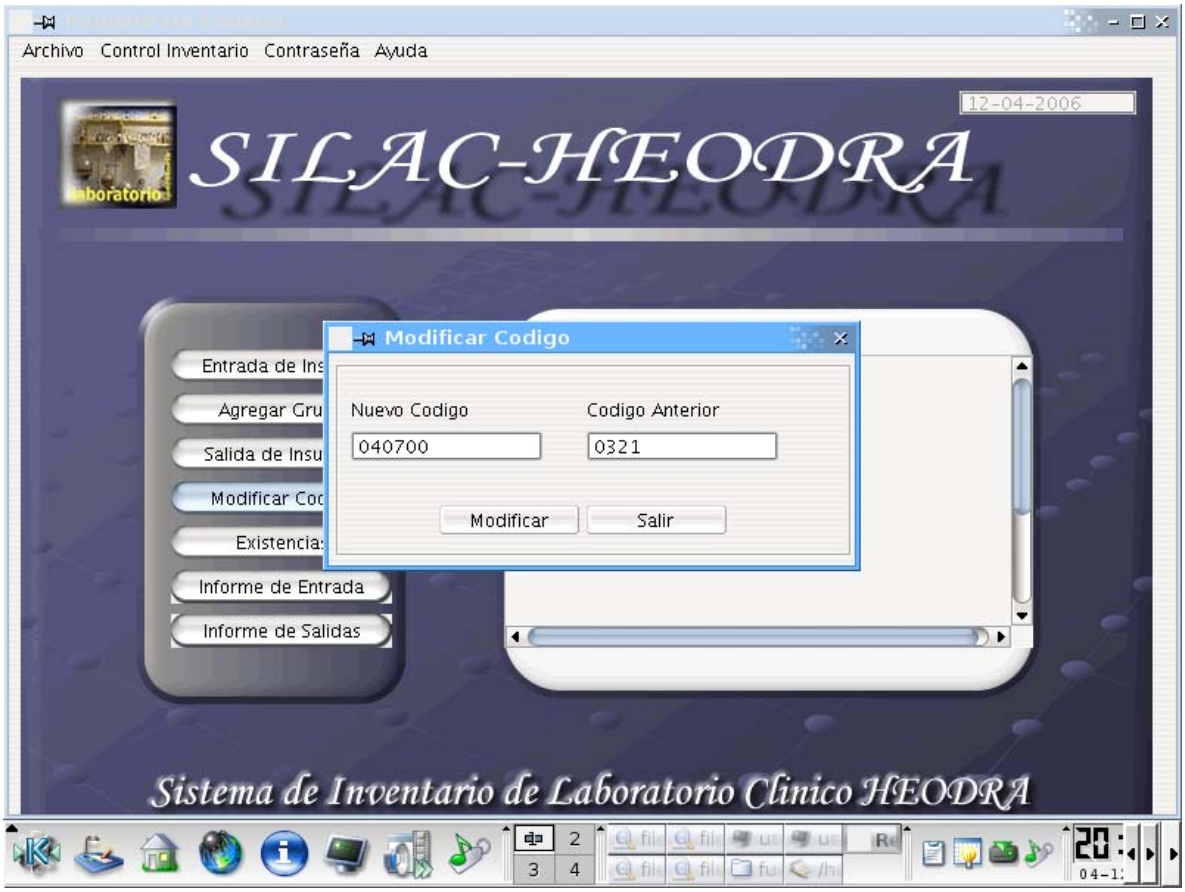


Función para mostrar las existencias

```
public void cargarTablaDatos(String grupo)
      {
     String buscar_grupo=grupo;
     Connection conexion,conex;
     PreparedStatement  obtener_grupo;
     PreparedStatement  obtener_existencia;
     String Existen=null;
     try
      {
        //conexionn =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega","usuario
", "usuario123");
        conexion =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
        obtener_grupo=conexion.prepareStatement("SELECT
Codigo,Nombre_generico,Presentacion,uso,indicador_insumo FROM Insumos
WHERE Nombre_Grupo = ?");
        if (conexion != null)
         {
          obtener_grupo.setString(1,buscar_grupo);
          ResultSet resul= obtener_grupo.executeQuery();
          control.borrarTodo();
          int cont;
          int y=0;
              while(resul.next())
              {
           String codigo = resul.getString("Codigo");
           String nombre = resul.getString("Nombre_generico");
           String presentacion = resul.getString("Presentacion");
           String uso=resul.getString("uso");
           String ind_insumo=resul.getString("indicador_insumo");;
           try
            {
               conex =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
             obtener_existencia=conexion.prepareStatement("SELECT
Sum(Cantidad) AS Existencia FROM detalle_insumo WHERE
(((detalle_insumo.Codigo)=?))");
             obtener_existencia.setString(1,codigo);
             ResultSet resul2= obtener_existencia.executeQuery();
             resul2.next();
             int existenciaE = resul2.getInt("Existencia");
             Existen=Integer.toString(existenciaE);
    resul2.close();
    conex.close();
    obtener_existencia.close();
             }
```

```
    catch(SQLException ex1)
          {
          ex1.printStackTrace( );
         }
control.agregarRegistroInsumo(codigo,nombre,presentacion,Existen,uso,ind_
insumo);
 }
    resul.close();
      obtener_grupo.close();
    conexion.close();
      }
   }
   catch(SQLException ex)
      {
      ex.printStackTrace( );
    }
```

INTERFAZ PARA MODIFICAR CODIGO

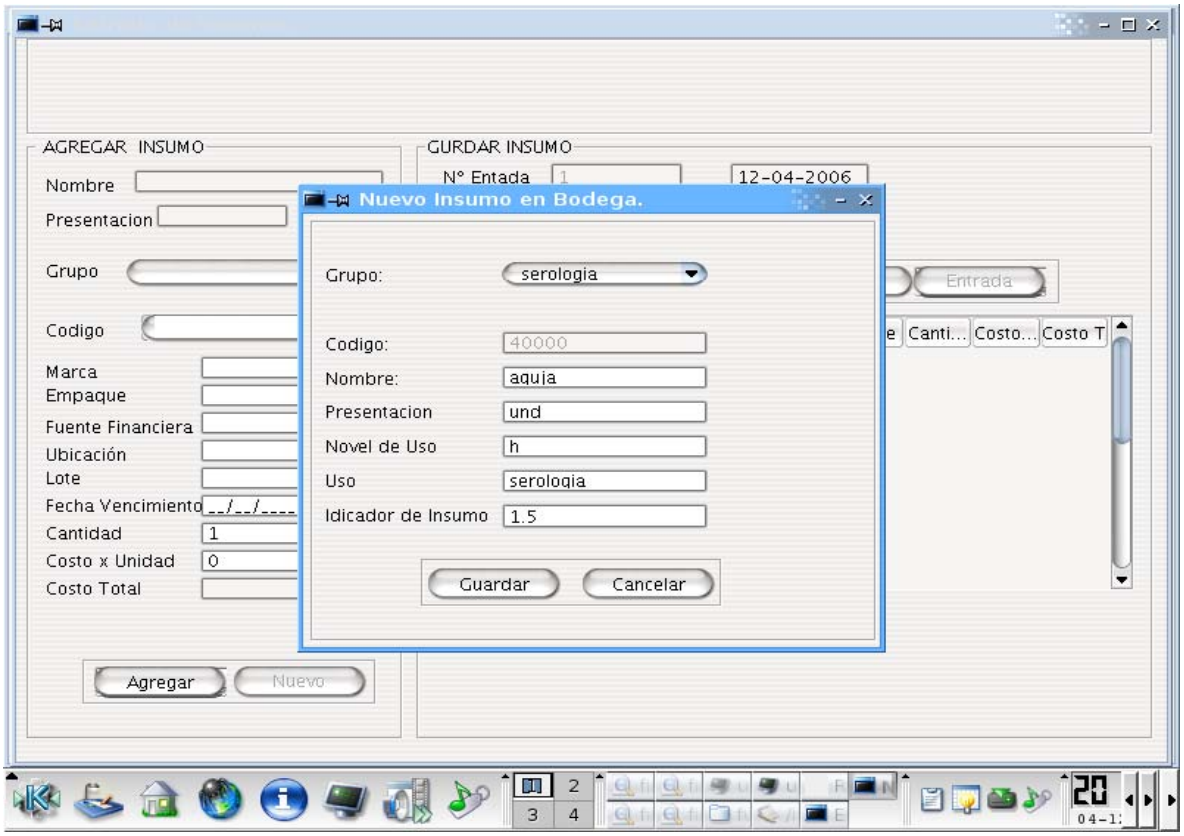


Función Principal:

```
public void modificarCodigo(String nuevoCodigo,String codigoAnterior)
      {

            String usuario=String.valueOf(tx_CodigoNuevo.getText());
            String claveAnterior=null;
                  Connection conexion,conexion1;



  PreparedStatement  obtener_codigo,actualizarClave;


      //    String claveAnterior=null;
      //          Connection conexion,conexion1;
  //PreparedStatement  obtener_codigo,actualizarClave;
      try
      {
        conexion =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
        obtener_codigo=conexion.prepareStatement("SELECT Codigo FROM
Insumos WHERE Codigo=?");

            obtener_codigo.setString(1,codigoAnterior);
            //obtener_codigo.setString(2,ClaveAnterior);
            ResultSet resul= obtener_codigo.executeQuery();

           if(resul.next()==true)
                 {

                try
                 {

          conexion1=
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
          actualizarClave=conexion1.prepareStatement("UPDATE Insumos SET
Codigo=? WHERE Codigo=?");
          actualizarClave.setString(1,nuevoCodigo);
          actualizarClave.setString(2,codigoAnterior);
          //actualizarClave.setString(3,claveAnterior);
          actualizarClave.executeUpdate();

          actualizarClave.close();
          conexion1.close();
          JOptionPane.showMessageDialog(null, "El codigo ha sido
Modificado", "Mensaje", JOptionPane.INFORMATION_MESSAGE);
           // btCambiar.setEnabled(false);

        }
       catch(SQLException ex2)
             {
            ex2.printStackTrace( );
            //JOptionPane.showMessageDialog(null, "Ese Codigo no se
encuentra en la base de datos", "Mensaje",
JOptionPane.INFORMATION_MESSAGE);
```

```
            }

            }

     else
     {
     JOptionPane.showMessageDialog(null, "Ese Codigo no se encuentra en
la base de datos", "Mensaje", JOptionPane.INFORMATION_MESSAGE);


     }

    resul.close();
    obtener_codigo.close();
    conexion.close();


   // }

    }
    catch(SQLException ex)
        {
        ex.printStackTrace( );
      }
       }
```

INTERFAZ DE NUEVO INSUMO



Function Principal:

```
public void guardarNuevoInsumo()
{
String sGrupo = String.valueOf(comboGrupo.getSelectedItem());
             if(sGrupo.compareTo("")==0 ||
tx_codigo.getText().length()==0 || tx_nombre.getText().length()==0)
              {
               JOptionPane.showMessageDialog(null, "Favor Llenar todos
los campos...", "Mensaje", JOptionPane.ERROR_MESSAGE);
              }
              else{
                  int existencias=0;
                  Connection conexion = null;
                  PreparedStatement sentencia = null;
                 try
                 {
                 String sGrupoN =
String.valueOf(comboGrupo.getSelectedItem());

//conexion=DriverManager.getConnection("jdbc:postgres1ql://localhost/bode
ga1","usuario","usuario123");
                     conexion =
DriverManager.getConnection("jdbc:postgresql://localhost/bodega1","postgr
es1","usuario123");
               sentencia = conexion.prepareStatement("INSERT INTO
Insumos VALUES (?,?,?,?,?,?,?)");
                      sentencia.setString(1,tx_codigo.getText());
                      sentencia.setString(2,tx_nombre.getText());
                      sentencia.setString(3,tx_presentacion.getText());
                      sentencia.setString(4,tx_nivelUso.getText());
                      sentencia.setString(5,tx_uso.getText());

sentencia.setString(6,tx_indicadorInsumo.getText());

                     sentencia.setString(7,sGrupoN);
                       sentencia.executeUpdate();
           }
                 catch(SQLException ex)
                      {
                            System.out.println(ex);
                      }
                      finally
                      {
                            try
                            {
                                  if(sentencia != null)
                                        sentencia.close();
                            }
                            catch(SQLException ex)
                            {
                                  System.out.println(ex);
                            }
                            try
```

```
                        {
                              if(conexion != null)
                                    conexion.close();
                        }
                        catch(SQLException ex)
                        {
                              System.out.println(ex);
                        }
                  }////fin del finally*/
            JOptionPane.showMessageDialog(null, "El Insumo a Sido
Guadado", "Mensaje", JOptionPane.INFORMATION_MESSAGE);
          }
```

MINISTERIO DE SALUD - SILAIS LEÓN

Hospital Escuela Oscar Danilo Rosales Arguello (HEODRA)

Departamento de Laboratorio Clinico y Banco de Sangre

REQUISA Nro : 1

Salida Por : consumo propio

Destinacion : Uroanalisis

Fecha de Salida : 14/01/07 0:00

Codigo	Nombre	U.M	Marca	Empaque	Lote	Vencimiento	Cantidad	Costo Und	Costo Tot
030100001	prueba	prueba	sin	sin	sin lote	15/02/2007	5	0.5	2.5
030100002	prueba	prueba	sin	sin	sin		10	1.5	15.0

Total de Bodega : 17.5

Elaborado | Autorizado | Atendido | Recibí Conforme