ESCUELA POLITÉCNICA DEL EJÉRCITO

EXTENSIÓN LATACUNGA

CARRERA DE TECNOLOGÍA EN COMPUTACIÓN

Diseño e implementación de un sistema de control de rancho mediante lector de código de barras para la ESPE extensión Latacunga, utilizando software libre.

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN COMPUTACIÓN

CBOP. DE COM. PILCO MARTINEZ VICTOR JOFFRE

CBOS. DE I.M. CRUZ ALMEIDA FIDEL OSWALDO

Latacunga, Marzo 2011

ESCUELA POLITÉCNICA DEL EJÉRCITO

EXTENSIÓN LATACUNGA

CERTIFICADO

ING. SANTIAGO JÁCOME (DIRECTOR)

ING. MARCELO ÁLVAREZ (CODIRECTOR)

CERTIFICAN:

Que el trabajo titulado "DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL DE RANCHO MEDIANTE LECTOR DE CÓDIGO DE BARRAS PARA LA ESCUELA POLITÉCNICA DEL EJÉRCITO EXTENSIÓN LATACUNGA, UTILIZANDO SOFTWARE LIBRE." Realizado por los señores Víctor Joffre Pilco Martínez y Fidel Oswaldo Cruz Almeida, ha sido guiado y revisado periódicamente y cumple las normas estatutarias establecidas por la ESPE, en el reglamento de estudiantes de la Escuela Politécnica del Ejército.

Debido a que constituye un trabajo de excelente contenido científico, coadyuvara a la aplicación de conocimientos y al desarrollo profesional, SI recomiendan su publicación.

Latacunga, Marzo del 2011

Ing. Santiago Jácome
DIRECTOR

Ing. Marcelo Álvarez
CODIRECTOR

ESCUELA POLITÉCNICA DEL EJÉRCITO

EXTENSIÓN LATACUNGA

AUTORIZACIÓN

Nosotros, Víctor Joffre Pilco Martínez y Fidel Oswaldo Cruz Almeida.

Autorizamos a la Escuela Politécnica del Ejército la publicación, en la biblioteca virtual de la institución del trabajo "DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL DE RANCHO MEDIANTE LECTOR DE CÓDIGO DE BARRAS PARA LA ESCUELA POLITÉCNICA DEL EJÉRCITO EXTENSIÓN LATACUNGA, UTILIZANDO SOFTWARE LIBRE." Cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y autoría.

Latacunga, Marzo del 2011

Pilco Martínez Víctor Joffre

C.I. 1716376015

Fidel Oswaldo Cruz Almeida

C.I. 0604148965

ESCUELA POLITÉCNICA DEL EJÉRCITO

EXTENSIÓN LATACUNGA

DECLARACIÓN DE RESPONSABILIDAD

Nosotros, Víctor Joffre Pilco Martínez y Fidel Oswaldo Cruz Almeida.

DECLARAMOS QUE:

El proyecto de grado denominado "DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL DE RANCHO MEDIANTE LECTOR DE CÓDIGO DE BARRAS PARA LA ESCUELA POLITÉCNICA DEL EJÉRCITO EXTENSIÓN LATACUNGA, UTILIZANDO SOFTWARE LIBRE." Ha sido desarrollado con base a una investigación exhaustiva, respetando derechos intelectuales de terceros, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de nuestra autoría.

En virtud de esta declaración, nos responsabilizamos del contenido, veracidad y alcance científico del proyecto de grado en mención.

Latacunga, Marzo del 2011

Pilco Martínez Víctor Joffre

C.I. 1716376015

Fidel Oswaldo Cruz Almeida

C.I. 0604148965

AGRADECIMIENTOS

Expreso mis más sinceros agradecimientos a todas las personas que hicieron posible la realización de este proyecto de tesis en especial:

A mis padres y hermanos que me han apoyado desde el inicio de mis estudios y estuvieron pendientes por mi bienestar.

A mi esposa y a mis hijas las cuales son lo más importante y valioso de mi vida, por quienes va todo mi esfuerzo y trabajo.

Al Director de tesis, Ingeniero Santiago Jácome por su valiosa orientación y disponibilidad desde el inicio de este proyecto.

Al Codirector de tesis, Ingeniero Marcelo Álvarez por su aporte e interés para el desarrollo del proyecto de tesis.

A todos mis maestros de la Carrera de Sistemas e Informática, por los conocimientos impartidos durante la permanencia en la Escuela Politécnica del Ejército.

Víctor Joffre

AGRADECIMIENTOS

Agradezco en primer lugar a Dios y a mi esposa, que gracias a su gran apoyo he logrado culminar una meta más en mi vida profesional.

A mis padres y hermanos que han sido el pilar fundamental en el transcurso de mi vida.

A mis abuelos Delia Castelo y Pedro Almeida y a mi tío Eudoro Almeida, quienes me inculcaron los valores y principios que rigen mi vida y me han hecho de mí, un hombre de bien.

Y sobre todo agradezco a mi hija Valeria quien es la inspiración de mi vida y por ella lucho día a día venciendo cada obstáculo que se presenta en este camino... el camino de la vida.

“El conocimiento es la materia prima para la grandeza”.

Oswaldo Cruz

DEDICATORIA

Dedico este presente Proyecto de Tesis, mis estudios y todo lo que hago a Dios, a mis padres y hermanos por su apoyo incondicional.

A mi esposa y a mis hijas, quienes son la inspiración para superarme día a día, por amarme sin condiciones, aceptando mis errores, sufriendo mis decepciones, festejando mis triunfos.

Víctor Joffre

DEDICATORIA

Dedico este proyecto y toda mi carrera profesional a Dios, a mi esposa, a mis padres, hermanos y de forma muy especial a mi hija Valeria y mis Abuelitos Delia y Pedro Almeida por ser quienes han estado a mi lado en todo momento dándome las fuerzas necesarias para continuar luchando día tras día y seguir adelante rompiendo todas las barreras que se me han presentado.

Oswaldo Cruz

E S P E
ESCUELA POLITECNICA DEL EJERCITO
CAMINO A LA EXCELENCIA

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL DE RANCHO MEDIANTE LECTOR DE CÓDIGO DE BARRAS PARA LA ESPE EXTENSIÓN LATACUNGA, UTILIZANDO SOFTWARE LIBRE.

S I S R A N V E R. 1.0

PRÓLOGO

A raíz que el Gobierno Ecuatoriano estableció la utilización de Software Libre como una política de Gobierno y de Estado en el año 2007, es de gran importancia que los profesionales en computación seamos los impulsadores y desarrolladores, a fin que en mas hogares del país puedan disfrutar de los beneficios que brinda el software libre.

Contenido

Índice de Figuras

Figura

Página

Índice de Diagramas

Diagrama

Página

Índice de Tablas

Tabla ***Página***

CAPÍTULO 1: RESUMEN E INTRODUCCIÓN DEL SISTEMA DE CONTROL DE RANCHO SISRAN 1.0

1.1 Presentación

1.1.1 Resumen de la Tesis

El presente proyecto de investigación está enfocado a diseñar e implementar un sistema de control de Rancho mediante lector de código de barras para la ESPE extensión Latacunga, empleando la tecnología Java y utilizando Software de código abierto con el fin de facilitar la lectura de datos a través del lector de código de barras y a su vez la captura rápida de información, así mismo servirá para la elaboración de reportes diarios y mensuales de los consumos del Rancho y el posterior cobro al personal que hace uso de este servicio.

La presente aplicación será desarrollada en Ubuntu como sistema Operativo base, PostgreSQL como gestor de base de datos y NetBeans como lenguaje de programación Java.

1.1.2 Abstract

The present investigation project is focused to design and to implement a system of Rancho control by means of reader of code of bars for ESPE extension Latacunga, using the technology Java and using Software of open sourse with the purpose of facilitating the reading of data through the reader of code of bars and in turn the quick capture of information, likewise will be good for the elaboration of reports daily and monthly of the consumptions of the Rancho and the later collection to the personnel that makes use of this service.

The present application will be developed in Ubuntu like operating system bases, PostgreSQL like database agent and NetBeans like programming language Java.

1.1.1 Introducción

En la ESPE extensión Latacunga y a nivel Ejército Ecuatoriano se tiene el servicio de Rancho, el problema principal radica en la gran cantidad de personal que consume el mismo sin un control eficiente ocasionando inconvenientes a la hora de realizar los reportes de consumo individual, ya que es indispensable cuadrar el valor total de descuentos con la suma de todos los confrontas entregadas al encargado del rancho durante el transcurso del mes.

En el proceso de realización de estos reportes de descuento es muy importante tomar en cuenta que existen muchas personas que no han consumido el rancho todos los días, y que en algunos casos han estado en una situación especial como por ejemplo Licencias Anuales Planificadas (L.A.P), y a estas personas no se les debería cobrar en absoluto el mes de consumo de rancho, en otras puede darse otra situación que es muy común como es el permiso de 15 días que tienen derecho por alumbramiento, en otras ocasiones por comisiones, cursos y pases especiales que deben ser registrados oportunamente para evitar cobros erróneos y valores excesivos en los descuentos individuales.

Por otra parte en la directiva de cobro de rancho vigente esta la disposición que se tiene que cobrar todos los almuerzos del mes, y desayunos de martes y jueves además de las guardias que se cobra día completo. Esto está justificado ya que existen personas que por alguna razón no prevista no consumen el rancho del día y la confronta una vez pasada al encargado del rancho esa comida simplemente se perderá, por ese motivo es indispensable cobrar esos días a fin de evitar pérdidas para los encargados del rancho.

Todo debe estar estrictamente planificado y en concordancia con las confrontas que se emiten a diario, y todo esto esta estipulado detalladamente en la directiva de rancho de la unidad en vigencia.

Por lo cual es necesario crear un sistema de control de rancho que sea eficiente y flexible a la hora de realizar cambios importantes pero justificados para evitar problemas a la hora de realizar los reportes de descuentos de rancho y confrontas diarias para el encargado del rancho.

Al final del mes el sistema debe emitir reportes de descuento de rancho que estén en concordancia con las confrontas de rancho emitidas a diario.

Si por alguno motivo en especial se necesitan realizar cambios el sistema debe pedir una justificación que al final se incluirá en el reporte además del registro de los usuarios que realizan dichos reajustes, como por ejemplo en el caso de los ranchos especiales que por lo general son más costosos que los ranchos comunes.

El sistema ha realizar se llamara SISRAN 1.0 el cual deberá solventar a cabalidad todos los puntos expuestos anteriormente, para lo cual será necesario crear políticas de cobro que permitan al sistema automatizar eficientemente todos los procesos necesarios para realizar los reportes mensuales de cobro de rancho y emisión diario de confrontas.

1.2 Objetivos

1.1.2 Objetivo General

Diseñar e implementar un sistema de control de rancho mediante lector de código de barras para la ESPE extensión Latacunga, que sea flexible y efectivo.

1.1.3 Objetivos Específicos

- Diseñar e implementar una base de datos en Postgresql que contenga la información necesaria de todo el personal que consuma el rancho en la ESPE-L.
- Diseñar e implementar el una interfaz amigable y eficiente, que sea fácil de usar por todo el personal involucrado con la administración del rancho.
- Producir un manual de usuario en el cual se especifiquen todos los ítems necesarios para el manejo y administración del sistema de control de rancho.
- Automatizar todos los procesos usados para realizar los descuentos, cobros y realización de confrontas de rancho.
- Efectivizar el control de consumo de rancho mediante el lector de código de barras.

1.3 Justificación

1.1.4 Justificación Teórica

El sistema de control de rancho que desde este momento lo llamaremos SISRAN 1.0 será un programa que manejará una base de datos en la cual podremos almacenar toda la información concerniente al consumo de rancho del personal, y también el optimizar el control en las confrontas de rancho.

Además de esto el programa contará con un sistema de control mediante reportes diarios que podrán ser consultados tanto por el encargado de realizar las confrontas y por el encargado del rancho, debiendo ser flexible a la hora de realizar cambios para evitar pérdidas económicas a la vez que deberá ser justo a la hora de realizar los reportes de cobro de rancho al personal, justificando a cabalidad el costo total del rancho consumido en el transcurso del mes.

Este sistema respetará a cabalidad la directiva de cobro de rancho que se encuentra en vigencia.

1.1.5 Justificación Metodológica

Para elaborar el presente proyecto se procederá a utilizar el modelo entidad-relación mediante el cual podremos representar fácilmente un esquema de la base de datos mediante entidades y sus respectivas asociaciones.

Para automatizar y optimizar el registro de consumo de racho se procederá a realizarlo con el apoyo de lector de código de barras el cual simplemente leerá un código impreso en una tarjeta la cual será entregada a cada consumidor es decir tanto a personal civil como militar.

Este código será usado por el programa para registrar el consumo de acuerdo al horario y al precio de cada una de las comidas.

Para desarrollar el programa SISRAN 1.0 se utiliza software libre en este caso el UBUNTU 10.10 y NETBEANS IDE 6.9.1 como plataforma de programación y el motor de base de datos que utilizaremos será el POSTGRESQL 8.4 que también es software gratuito y de código abierto.

1.1.6 Alcance

El diseño e implementación del sistema de control de rancho SISRAN 1.0 se enfocará esencialmente en el desarrollo de una aplicación en java que se conectara con PostgreSQL como el motor de la base de datos que contendrá la información del personal que consume el rancho en la ESPE extensión Latacunga, tomando en cuenta que todos los procesos del sistema se basarán solo en las especificaciones que se dan en la directiva de rancho en vigencia.

1.1.7 Marco Teórico

Para la el diseño de la base de datos del proyecto utilizaremos el modelo entidad-relación el cual es el modelo más utilizado para el diseño de bases de datos. El modelo entidad-relación está formado por un conjunto de conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas estos conceptos se los detalla más adelante dentro del Marco Conceptual.

Y java será el lenguaje que se usará en la programación del proyecto el cual es básicamente un lenguaje de programación orientado a objetos. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

i. PostgreSQL

Para nuestro sistema de control de rancho vamos a emplear PostgreSQL 8.4, el cual es un sistema de gestión de bases de datos de código abierto bajo licencia GLP.

ii. Java Y Netbeans

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un *bytecode*, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el *bytecode* es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución

directa por hardware del *bytecode* por un procesador Java también es posible.

NetBeans IDE (Integrated Develoment Enviroment o en español, Entorno de Desarrollo Integrado) es un proyecto de código abierto de gran éxito, fundado y patrocinado hasta la actualidad por la empresa Sun MicroSystems en Junio del 2000.

NetBeans está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans, por ejemplo se puede trabajar con C, C++, Ruby, Phyton, PHP y diseño de aplicaciones web en general.

NetBeans es un entorno de desarrollo, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados *módulos*. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

iii. JasperReports

En NetBeans se puede realizar el diseño de Reportes muy fácilmente gracias al componente llamado JasperReports, el cual es un framework bastante completo para desarrollar reportes tanto web como desktop en Java. Este componente puede ser utilizado dentro de NetBeans o de forma externa como lo utilizaremos en nuestro proyecto.

iv. JPA

Es una técnica de programación de bases de datos más potente y útil a la hora de realizar programas de gran complejidad, JPA significa Java Persistence API (Aplication Programing Interface, en español esto significa que es una Interfaz de Programación de Aplicaciones.) Esto quiere decir que es la API de persistencia desarrollada para la plataforma Java EE e incluida en el estándar EJB3. Esta API busca unificar la manera en que funcionan las utilidades que proveen un mapeo objeto-relacional. El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos, como sí pasaba con EJB2, y permitir usar objetos regulares (conocidos como POJOs). La gestión de bases de datos se basan principalmente en cuatro cosas: altas, bajas, cambios y consultas, este último es un tema aun mucho más extenso ya que existen consultas anidadas y algunas funciones como por ejemplo SUM() que permite sumar el contenido de algunos registros, pero esto se facilita con JPA ya que nos permite manejar las Tablas que se encuentran en la base de datos como Objetos. Para ello debemos tomar en cuenta la declaración de dichos objetos y las posibles operaciones que no permite realizar una librería llamada EclipseLink (JPA 2.0) que es la versión libre de TopLink Essentials de Oracle, aunque no es tan buena como Hibernate que es la mejor biblioteca de persistencia que existe en el mercado, EclipseLink es de código abierto, y esperamos que en futuras versiones ya pueda igualar y tal vez superar a Hibernate.

v. Lector Código de Barras

Lector Láser de Código de Barras Metrologic MS9520 Voyager.

Consiste en el escáner propiamente dicho, un decodificador y un cable que actúa como interfaz entre el decodificador y el terminal o la computadora.

La función del escáner es leer el símbolo del código de barras y proporcionar una salida eléctrica a la computadora, correspondiente a las barras y espacios del código de barras. Sin embargo, es el decodificador el que reconoce la simbología del código de barras, analiza el contenido del código de barras leído y transmite dichos datos a la computadora en un formato de datos tradicional.

Un escáner puede tener el decodificador incorporado en el mango o puede tratarse de un escáner sin decodificador que requiere una caja separada, llamada interfaz o emulador.

vi. Algoritmos de encriptación

Un algoritmo criptográfico, o cifrador, es una función matemática usada en los procesos de encriptación y desencriptación. Trabaja en combinación con una llave (un número, palabra, frase, o contraseña) para encriptar y desencriptar datos.

Para encriptar, el algoritmo combina matemáticamente la información a proteger con una llave provista. El resultado de este cálculo son los datos encriptados.

Para desencriptar, el algoritmo hace un cálculo combinando los datos encriptados con una llave provista, siendo el resultado de esta combinación los datos desencriptados (exactamente igual a como estaban antes de ser encriptados si se usó la misma llave).

Si la llave o los datos son modificados el algoritmo produce un resultado diferente. El objetivo de un algoritmo criptográfico es hacer tan difícil como sea posible desencriptar los datos sin utilizar la llave. Si se usa un algoritmo de encriptación realmente bueno, entonces no hay ninguna técnica significativamente mejor que intentar metódicamente con cada llave posible.

CAPÍTULO 2: ADMINISTRAR BASES DE DATOS CON POSTGRESQL

2.1 Introducción a PostgreSQL

Para nuestro sistema de control de rancho vamos a emplear PostgreSQL 8.4, el cual es un sistema de gestión de bases de datos de código abierto bajo licencia GLP.

PostgreSQL nace como un proyecto de graduación de la Universidad de Berkeley (EE.UU), Michael Stonebraker, líder de este proyecto logro implementar un motor de base de datos relacional llamado Ingres.

Después de haber trabajado un largo tiempo en Ingres y de haber tenido una experiencia comercial con él mismo, Michael decidió volver a la Universidad en 1985 para trabajar en un nuevo proyecto sobre la experiencia de Ingres, dicho proyecto fue llamado post-ingres o simplemente POSTGRES.

2.2 PostgreSQL en Ubuntu

Como se menciono con anterioridad una de las características de PostgreSQL es la de ser de código abierto u Open Source como común mente se lo llama en inglés, por lo tanto la comunidad de Ubuntu decidió incluirlo en sus repositorios a fin permitir al usuario desarrollador tener una herramienta muy eficiente para realizar la gestión de bases de datos en Ubuntu.

2.3 Instalación y primeros pasos para manejo de PostgreSQL

Para realizar la instalación de PostgreSQL en Ubuntu existen varias alternativas, pero las más comunes son la instalación desde la consola y desde el gestor de paquetes Synaptic.

2.3.1 Instalación desde la Consola

Bueno en realidad este es el paso más sencillo que se puede realizar, primero que nada debemos estar conectados a internet, si lo estamos nos dirigimos a Aplicaciones/Accesorios/Terminal, en el cual escribimos la siguiente línea de comandos.

```
$ sudo aptitude install postgresql-8.4 postgresql-client-8.4
pgadmin3
```

Quedándonos de la siguiente forma:

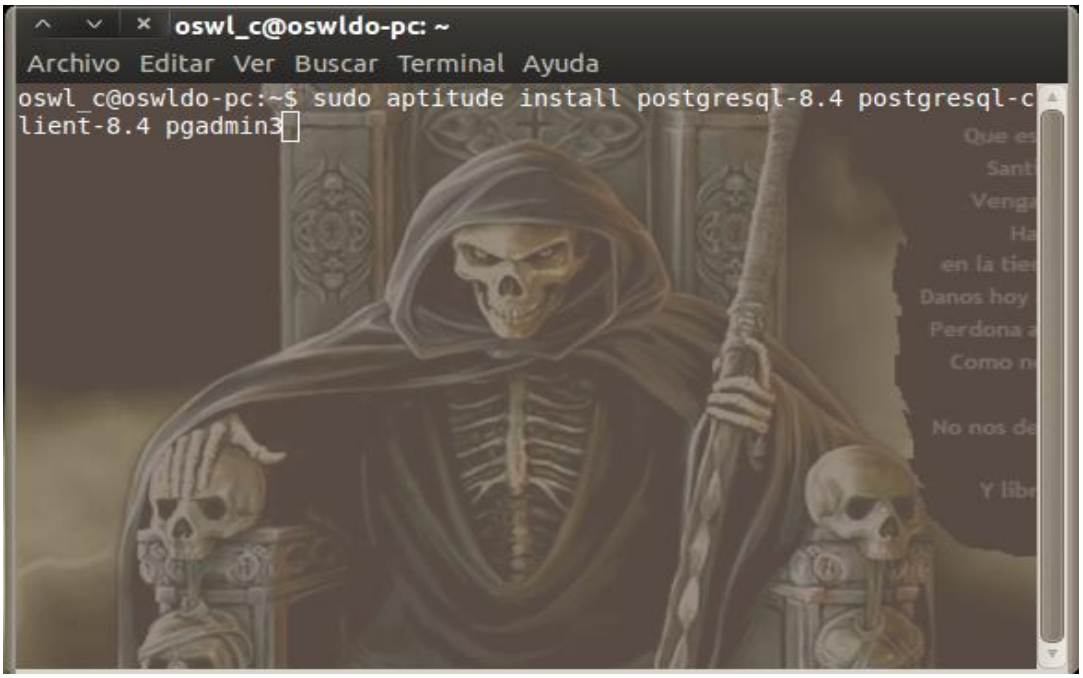


Fig. 1 Instalación de Postgresql desde la consola

Una vez instalado procedemos a realizar la respectiva configuración la cual se explicara más adelante.

2.3.2 Instalación desde el gestor de paquetes Synaptic

Es muy común realizar instalaciones desde el gestor de paquetes Synaptic ya que es un método grafico que tiene una

ventaja la cual es que todos los paquetes de los repositorios están enlistados y tienen un pequeño resumen de lo que hace cada uno de ellos. Por tanto es un método muy eficaz a la hora de realizar cualquier tipo de instalación.

Para lo cual vamos a Sistema/Administración/Gestor de Paquetes Synaptic.

Vamos a la barra de búsqueda, escribimos Postgresql, y marcamos los siguientes resultados de la búsqueda:

- Postgresql
- postgresql-client
- pgadmin

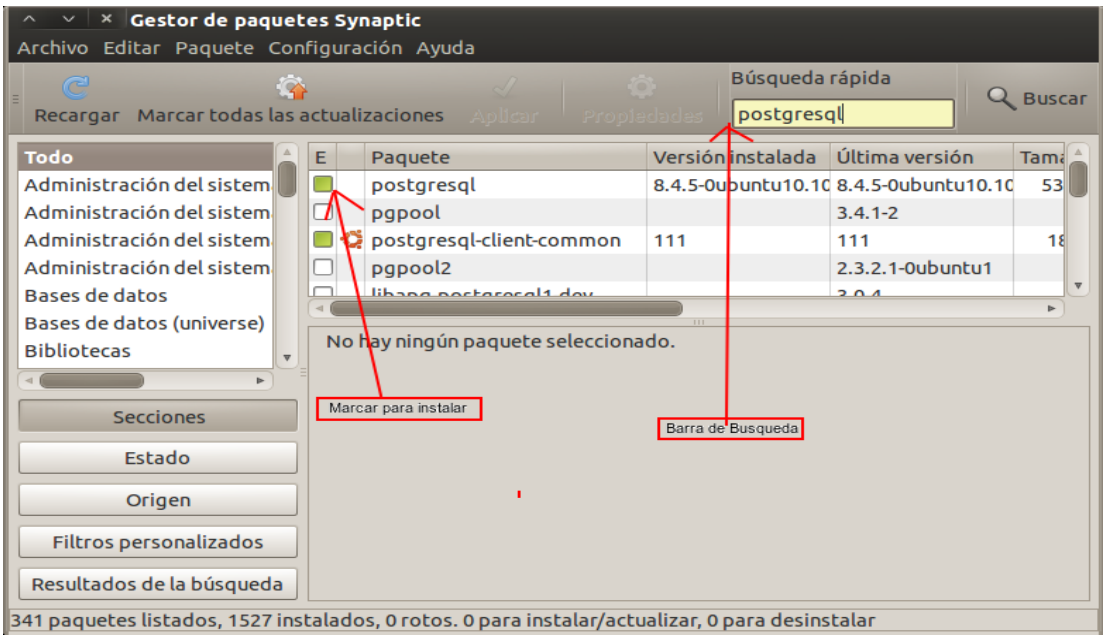


Fig. 2 Instalación desde el gestor de paquetes Synaptic

Finalmente hacemos clic en aplicar y comenzara la descarga de los paquetes para posteriormente realizar la instalación, una vez instalado nos saldrá el siguiente mensaje:

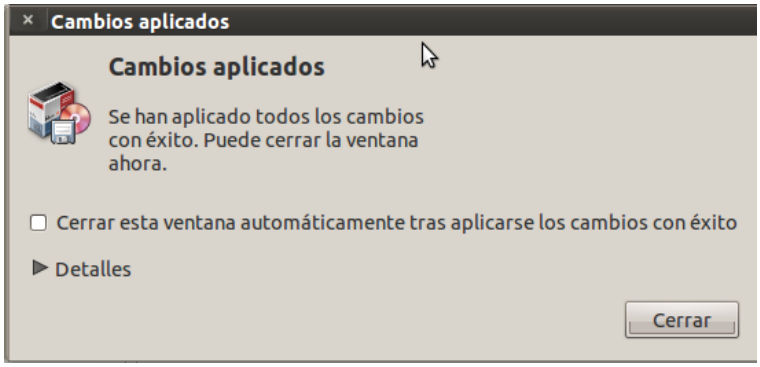


Fig. 3 Ventana de cambios aplicados

2.3.3 Instalación desde un archivo ejecutable .bin

Otra forma de instalar PostgreSQL es descargando directamente desde la pagina de PostgreSQL *www.postgresql.org/download/* los paquetes para ser instalados gráficamente.

Paso 1.

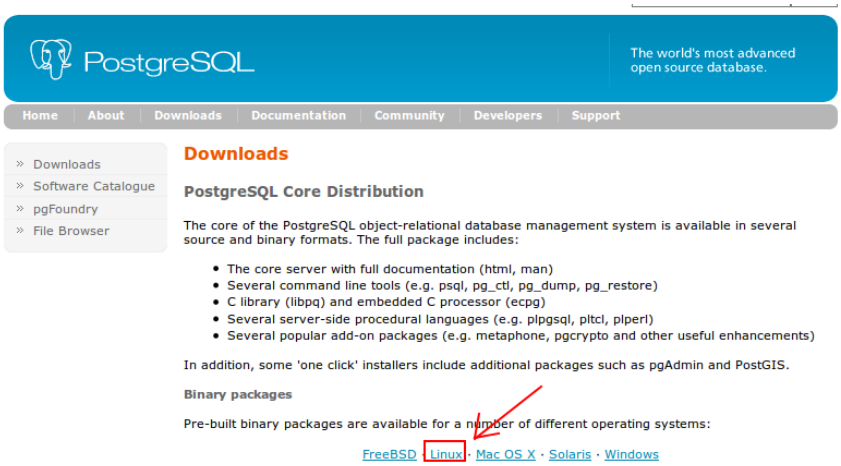


Fig. 4 Instalación desde un archivo ejecutable .bin

Paso 2.

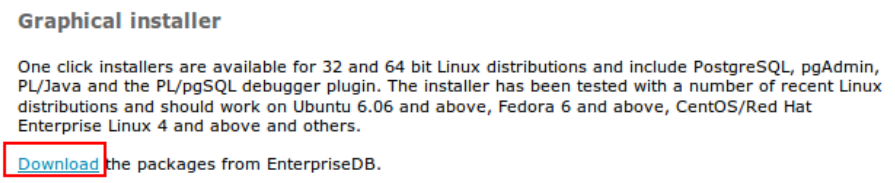


Fig. 5 Instalador Gráfico

Paso 3.

Fig. 6 Descarga de PostgreSQL

Como pudimos observar en la imagen anterior Postgresql ya cuenta con la versión 9.0.1-1 pero como dijimos que vamos a trabajas con la versión 8.4 procedemos a descargar esta versión para 32 bits y también para 64 bits ya que vamos a trabajar con los dos tipos de sistema operativo.

Estos instaladores tienen la extensión .bin, este tipo de archivos deben ser ejecutados de la siguiente forma:

Primero debemos ubicarnos en el sitio en donde se descargaron el archivo, procedemos a darle permisos de ejecución, y finalmente a ejecutar el archivo, para ello abrimos la consola y ejecutamos las siguientes líneas de comandos:

```
$ cd Descargas

~/Descargas$ sudo chmod +x postgresql-8.4.5-1-linux.bin

~/Descargas$ sudo ./postgresql-8.4.5-1-linux.bin
```

Ahora explicando cada línea de comandos el cd Descargas es simplemente para ubicarnos en la carpeta descargas, el sudo chmod +x es para darle permisos de ejecución y finalmente sudo ./ es para ejecutar el programa.

En la instalación va a aparecer la siguiente ventana:

Fig. 7 Instalador de PostgreSQL

Damos clic en siguiente dejando las configuraciones determinadas por defecto.

En una de esas ventanas nos pedirá que ingresemos la contraseña para postgresql.

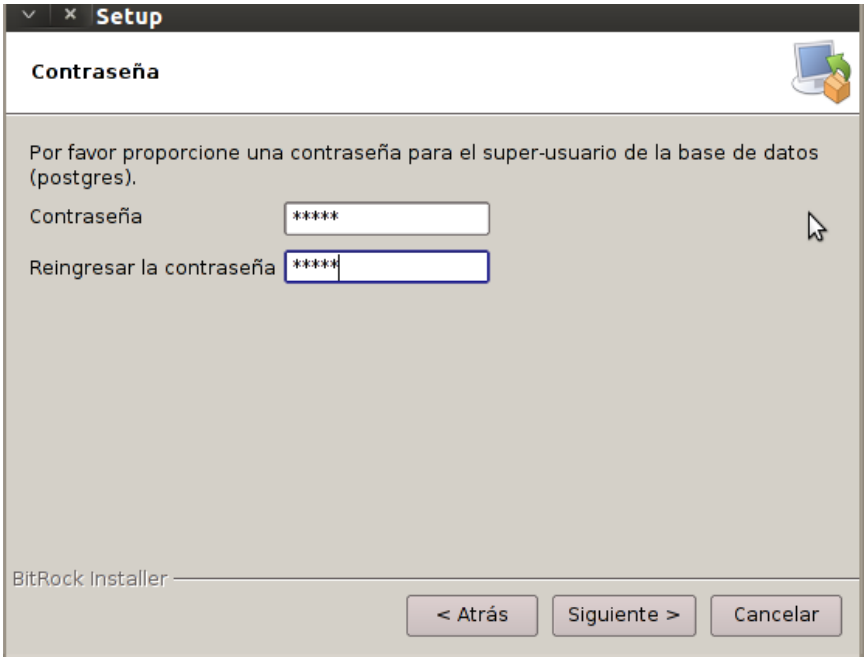


Fig. 8 Contraseña para instalación

Luego nos preguntara que puerto vamos a utilizar, dejamos el puerto por defecto 5432.

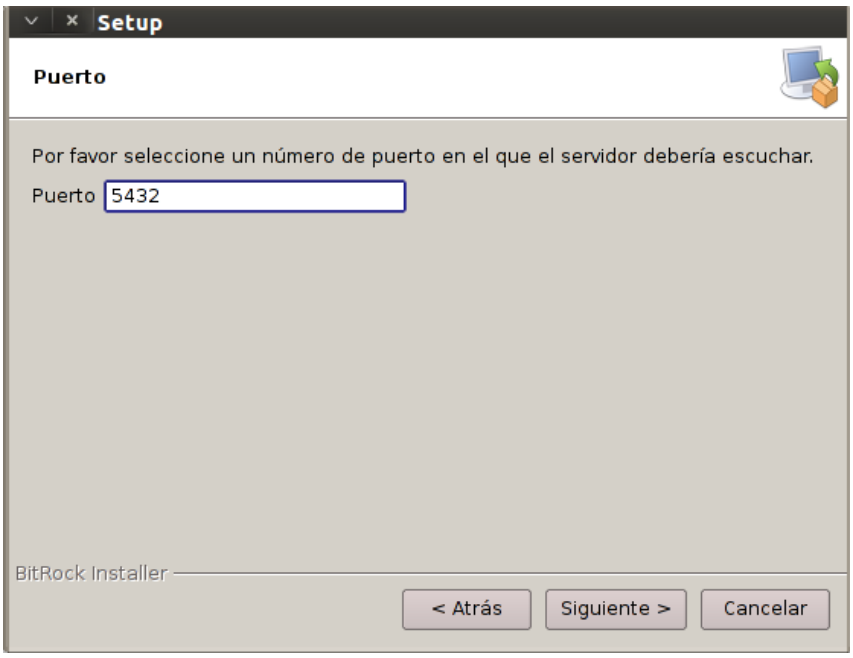


Fig. 9 Configuración del puerto de servidor

Al finalizar no preguntara si deseamos instalar complementos que pueden ser descargados desde el internet.

Fig. 10 Terminación de la instalación

Este es el único método de instalación que no requiere una configuración desde la consola ya que el asistente grafico de instalación nos da la posibilidad de realizar dichas configuraciones sin problema.

2.4 Configuración de PostgreSQL y creación de una Base de Datos

Una vez instalado PostgreSQL procedemos a realizar las siguientes configuraciones:

2.4.1 Crear una contraseña de seguridad para el usuario postgresql.

Para crear una contraseña para el usuario postgresql en Ubuntu abrimos la consola y digitamos la siguiente línea de comando:

```
$ sudo passwd postgres
```

Posteriormente el sistema nos pedirá que nos validemos como súper usuario ingresando nuestra propia clave, luego nos pedirá la nueva clave para el usuario postgres y finalmente nos pedirá que ingresemos de nuevo la clave como un método de verificación.

2.4.2 Cambiar al usuario postgresql y crear una base de datos.

Una vez creada la nueva clave para el usuario postgres realizamos el cambio de usuario digitando la siguiente línea de comandos en la consola:

```
$ sudo su postgresql
```

Nos pedirá la clave anteriormente creada y una vez que nos validemos tendremos que crear la nueva base de datos para lo cual tecleamos la siguiente línea de comandos:

```
$ createdb sisrandb
```

2.4.3 Ingresar a la consola de PostgreSQL y cambiar la clave interna al usuario postgres.

Una vez creada la base de datos procedemos a ingresar a la consola
propia de PostgreSQL para lo cual digitamos lo siguiente:

```
$psql sisrandb
```

Y ahora procedemos a cambiar la clave interna del usuario postgresql con las siguientes líneas de comandos:

```
sisrandb=# ALTER USER postgres WITH PASSWORD
'espel';

ALTER ROLE

sisrandb=#\q
```

2.4.4 Configurar PostgreSQL para admitir conexiones remotas.

La configuración por defecto de PostgreSQL no admite conexiones externas. Para habilitarlas tenemos que editar el siguiente archivo:

```
$ sudo gedit /etc/postgresql/8.4/main/postgresql.conf
```

Ahora buscamos las siguientes líneas que se encuentran comentadas con el signo numeral al inicio:

```
#listen_addresses = 'localhost'
```

Y la substituimos por la siguiente línea:

```
listen_addresses = '*'
```

Buscamos la siguiente línea y la descomentamos:

```
#password_encryption = on
```

Y nos debe quedar así:

```
password_encryption = on
```

Guardamos los cambios y reiniciamos el servicio de PostgreSQL:

```
$ sudo /etc/init.d/postgresql-8.4 restart
```

2.4.5 Configurar la lista de acceso.

Para configurar la lista de acceso debemos editar el siguiente archivo:

```
$ sudo gedit /etc/postgresql/8.4/main/pg_hba.conf
```

Al final del archivo se encuentra una lista de acceso predeterminada, ahora, dependiendo de su necesidad puedes hacer lo siguiente:

* Si necesita que cualquier usuario se conecte por medio de una dirección IP en específico, agregue al final la siguiente línea:

```
host all all 192.168.1.4 255.255.255.0 md5
```

* Si necesita que cualquier usuario se conecte por medio de una IP determinada sin importar el password (confiamos en dicha IP), la línea es:

```
host all all 192.168.1.4 255.255.255.255 trust
```

* Si necesita que cualquier usuario (usuario de base de datos autentificándose, claro) se conecte por medio de cualquier dirección IP, agregue al final la siguiente línea:

```
host all all 0.0.0.0 0.0.0.0 md5
```

* Si necesita que un usuario determinado se conecte a una base de datos determinada por medio de una dirección IP en específico, agregue al final la siguiente línea:

```
host sisrandb postgres 192.168.1.4 255.255.255.0 md5
```

* Guarda los cambios realizados en el archivo y reinicia el servicio para que los cambios surtan efecto:

```
$ sudo /etc/init.d/postgresql-8.4 restart
```

2.5 Creación y manipulación de Tablas y Datos en Postgresql

Para la creación y manipulación de tablas y datos en PostgreSQL tenemos varias formas, en esta ocasión vamos a trabajar con la herramienta mas común para el manejo de datos en PostgreSQL.

2.5.1 Usando el programa de administración de datos pgAdmin III

PgAdmin III es una aplicación gráfica para gestionar el gestor de bases de datos Postgresql.

Con el pgAdmin III los usuarios pueden desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. Esta aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor y un agente para lanzar scripts programados.

i. Instalación

La instalación se la puede realizar desde la consola:

```
$ sudo apt-get install pgAdmin3
```

Desde el gestor de paquetes synaptic, desde archivos previamente descargados o desde el centro de software de Ubuntu el cual la opcional más fácil, simplemente ingresamos a aplicaciones/centro de software de Ubuntu, buscamos pgAdmin y damos clic en instalar.

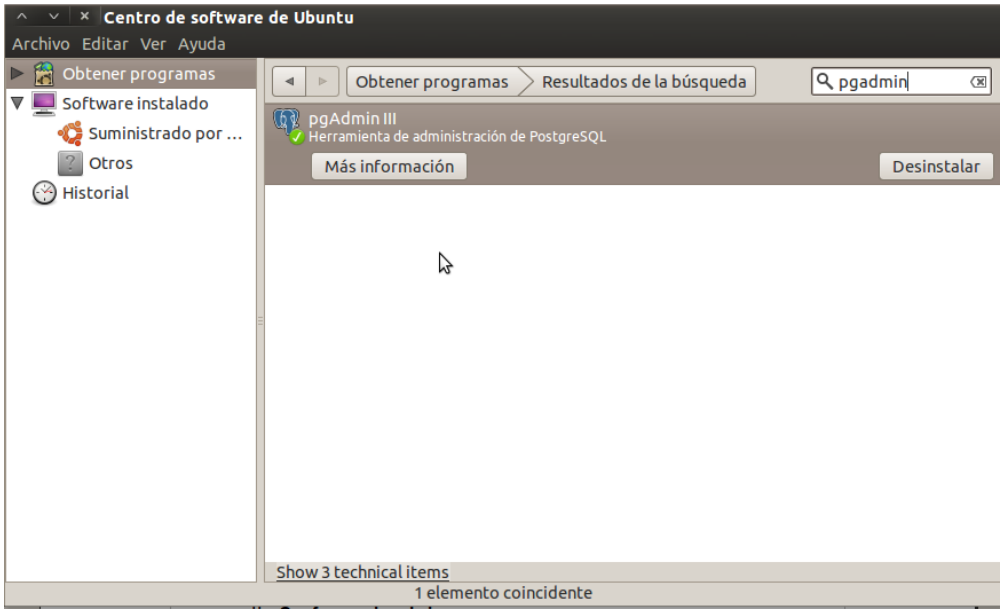


Fig. 11 Instalación de PgAdmin III

ii. Configuración de ingreso

Para ingresar al pgAdmin III simplemente vamos a aplicaciones/programas/pgAdmin III, y posteriormente hacemos clic en el botón de nueva conexión:

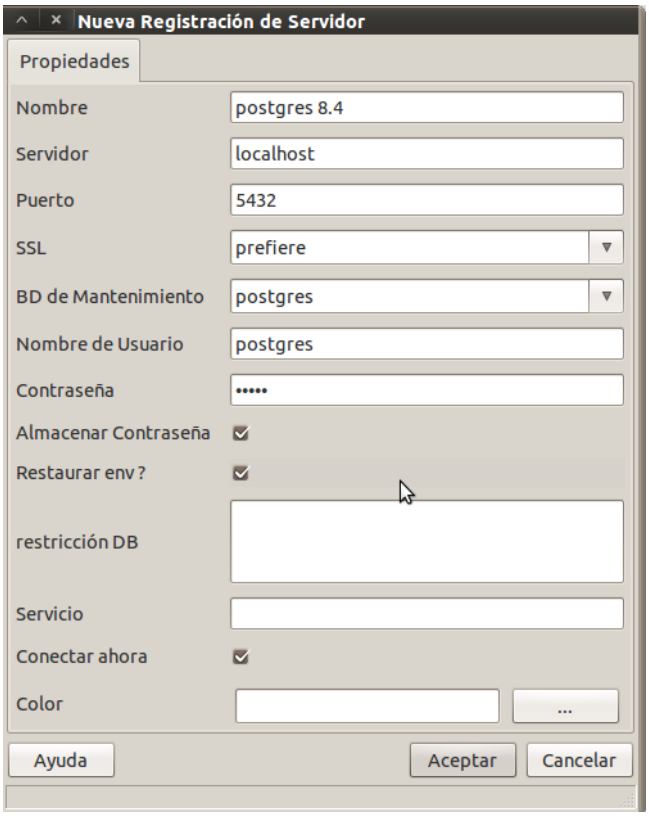


Fig. 12 Configuración de ingreso a PgAdmin III

Posteriormente nos saldrá la una ventana de conexión la cual la llenaremos de la siguiente forma:

Luego damos clic en aceptar y listo, ya estamos conectados a la base de datos.

iii. Creación de bases de datos

Si queremos crear una nueva base de datos desde pgAdmin III simplemente damos clic derecho sobre Bases de Datos y seleccionamos nueva base de datos, nos saldrá una ventana la cual debemos llenar de la siguiente forma:

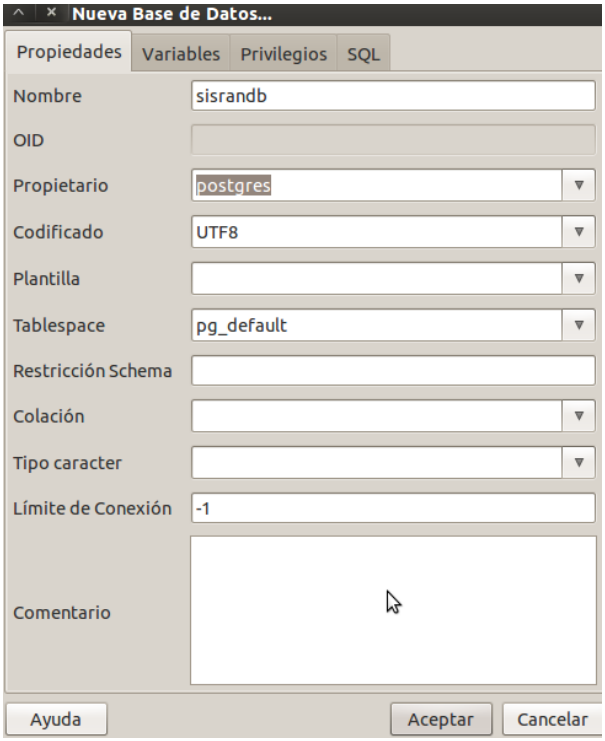


Fig. 13 Creación de Base de datos

Esto nos indica que todo lo que hacemos gráficamente se va creando un script SQL el cual al final será ejecutado al dar clic en aceptar.

iv. Creación de Tablas

- Para crear tablas usando el pgAdmin III debemos dar clic en el signo más ubicado junto a la base de datos en la que queremos crear la nueva tabla.

- Luego hacemos clic en esquemas y seleccionamos público.
- Finalmente hacemos clic derecho en público seleccionamos nuevo objeto y luego nueva tabla, saliéndonos una ventana la cual debemos llenarla de la siguiente forma:

Paso 1.

Fig. 14 Creación de tablas

Paso 2.

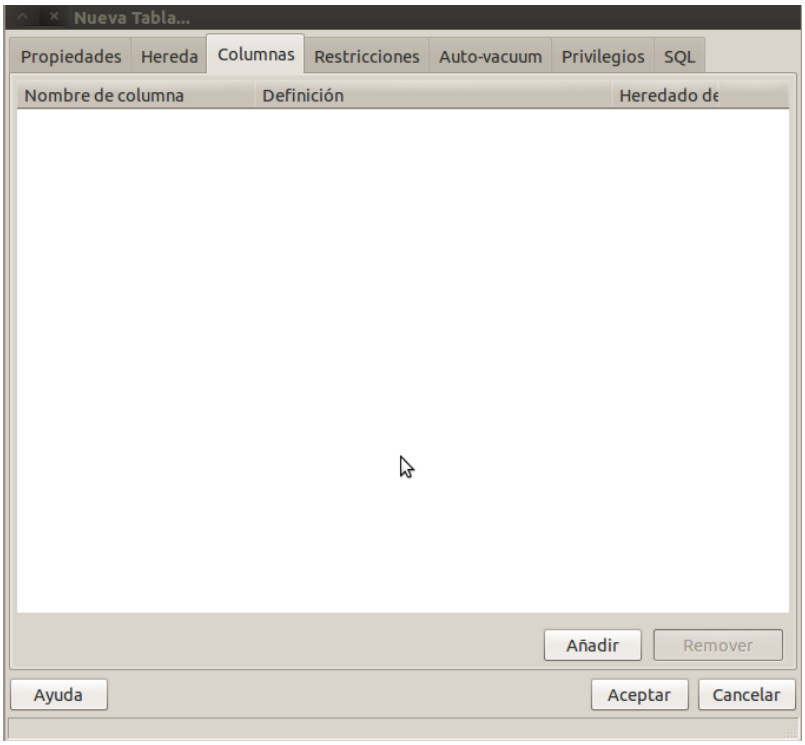


Fig. 15 Selección de columnas

Paso 3.

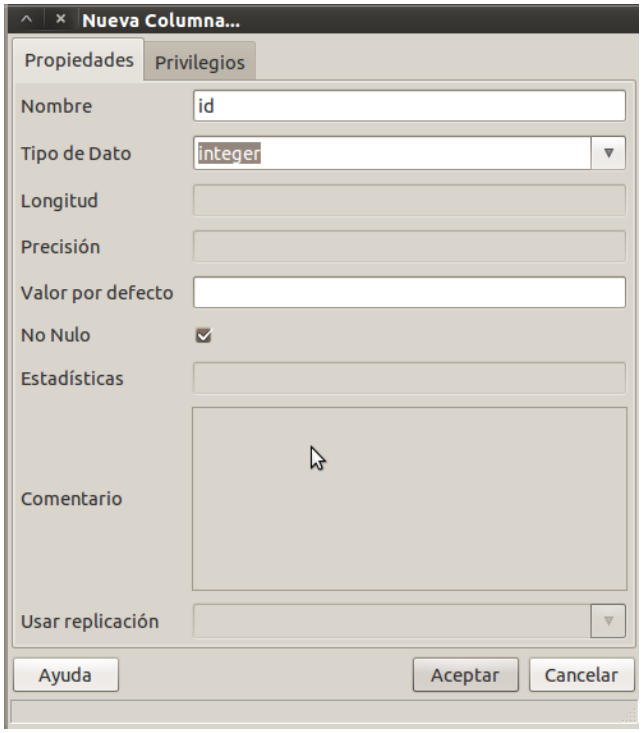


Fig. 16 Identificación de las Columnas

Paso 4.

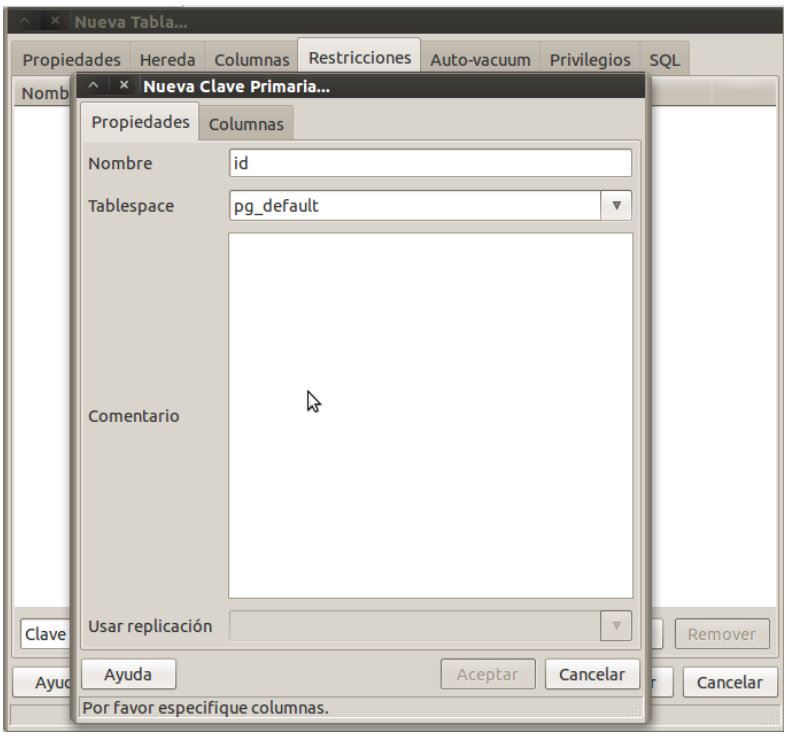


Fig. 17 Propiedades de las columnas

Paso 5.

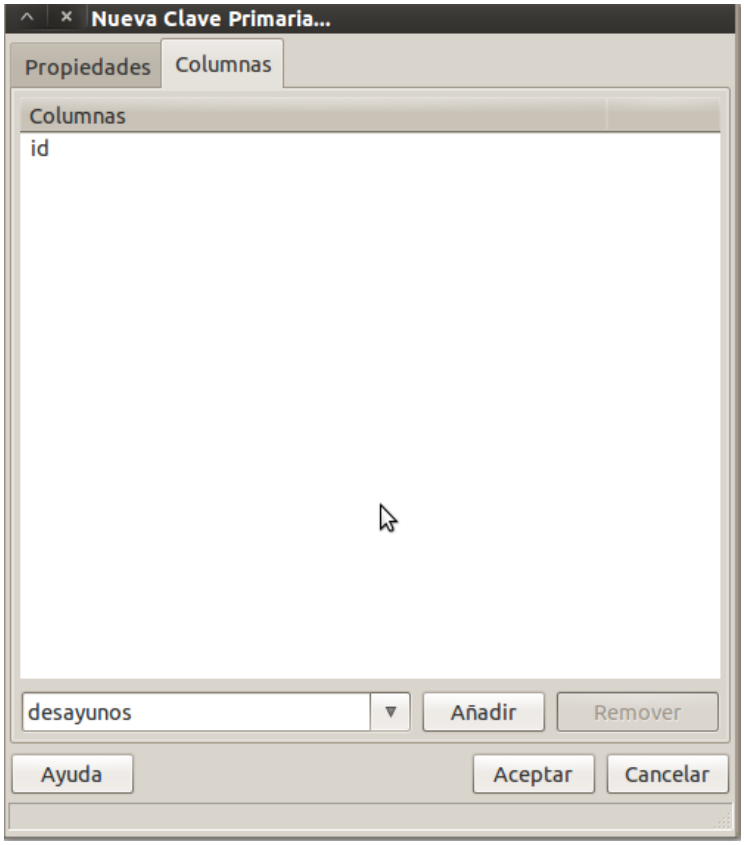


Fig. 18 Clave Primaria

Paso 6.

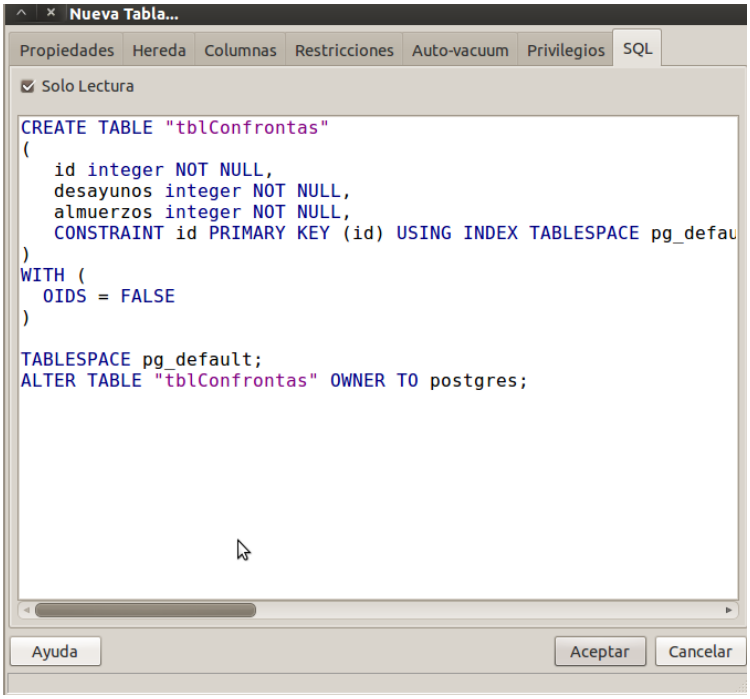


Fig. 19 Programación en SQL

Paso 7. Al finalizar los pasos anteriores explicados en forma grafica obtendremos la tabla Confronta (tblConfronta) creada.

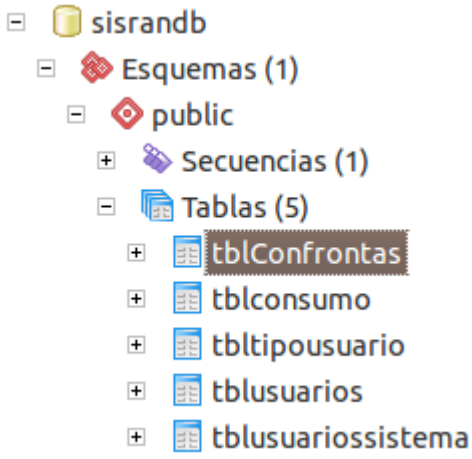


Fig. 20 Tabla Confronta creada

2.6 Crear Copias de Seguridad de la base de datos

En PostgreSQL la mejor forma de hacer copias de seguridad de los datos es usando pgAdmin III, la cual como ya explicamos anteriormente es una herramienta diseñada para administrar PostgreSQL gráficamente.

Para hacer una copia de seguridad de una base de datos utilizamos el siguiente comando:

$ pg_dump -h host -U usuario nombre_bd > nombre_bd.sql

Para hacer una copia de seguridad de todas las bases de datos PostgreSQL de un servidor, se usa este escript:

```
#!/bin/bash

 ## BEGIN CONFIG ##

HOST=localhost

BACKUP_DIR=tmp

## END CONFIG ##


if [ ! -d $BACKUP_DIR ]; then
```

```
  mkdir -p $BACKUP_DIR


POSTGRE_DBS=$(psql -h $HOST -U postgres -l | awk ' (NR > 2) && (/[a-zA-Z0-9]+[ ]+[\]/) && ( $0 !~ /template[0-9]/) { print $1 }');


 for DB in $POSTGRE_DBS ; do

  echo "* Backuping PostgreSQL data from $DB@$HOST..."

  pg_dump -h $HOST -U postgres $DB > $BACKUP_DIR/pg_$DB.sql

 done
```

Para restaurar una copia de seguridad:

```
sudo psql -d nombre_base_datos -f archivo.pgdump
```

CAPÍTULO 3: PRIMEROS PASOS CON NETBEANS IDE 6.9.1 y JASPERREPORTS

3.1 Introducción a NetBeans IDE 6.9.1

NetBeans IDE (Integrated Develoment Enviroment), es un proyecto de código abierto de gran éxito, fundado y patrocinado hasta la actualidad por la empresa Sun MicroSystems en Junio del 2000.

En principio NetBeans fue un proyecto estudiantil en República Checa. Y la meta era escribir un entorno de desarrollo integrado (IDE) para Java parecida a la de Delphi, por ello lo llamaron Xelfi.

Posteriormente el empresario Roman Stanek se intereso en el proyecto, teniendo como objetivo desarrollar unos componentes JavaBeans para redes. Jarda Tulach, quien diseñó la arquitectura básica de la IDE, propuso la idea de llamarlo NetBeans, a fin de describir este propósito.

NetBeans está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans, por ejemplo se puede trabajar con C, C++, Ruby, Phyton, PHP y diseño de aplicaciones web en general.

Sin lugar a duda, NetBeans se ha convertido en un IDE apto para la mayoría de los lenguajes de programación open so urce modernos llegando a ser, para la mayoría de programadores que trabajan con software libre, una gran herramienta y aliado a la hora de realizar proyectos de gran complejidad.

3.2 Creación de proyectos y diseño de Interfaces en NetBeans IDE 6.9.1

Como ya mencionamos anteriormente se puede crear una diversidad de tipos de proyectos, tal como proyectos web, de consola y de escritorio.

En esta parte vamos simplemente nos concentrarnos en la creación de ventanas y su respectivo diseño, realizando ejemplos simples y que el diseño de las interfaces del proyecto SISRAN 1.0 y sus respectivas funciones serán explicadas más adelante en el capítulo VI.

3.2.1 Nuevo Proyecto

Nos dijimos al menú Archivo/Nuevo Proyecto en donde nos saldrá la siguiente dialogo:

Fig. 21 Creación de Nuevo proyecto Java

Seleccionamos el tipo de proyecto que vamos a trabajas, en nuestro caso simplemente vamos seleccionar Java/ Aplicación Java y damos siguiente.

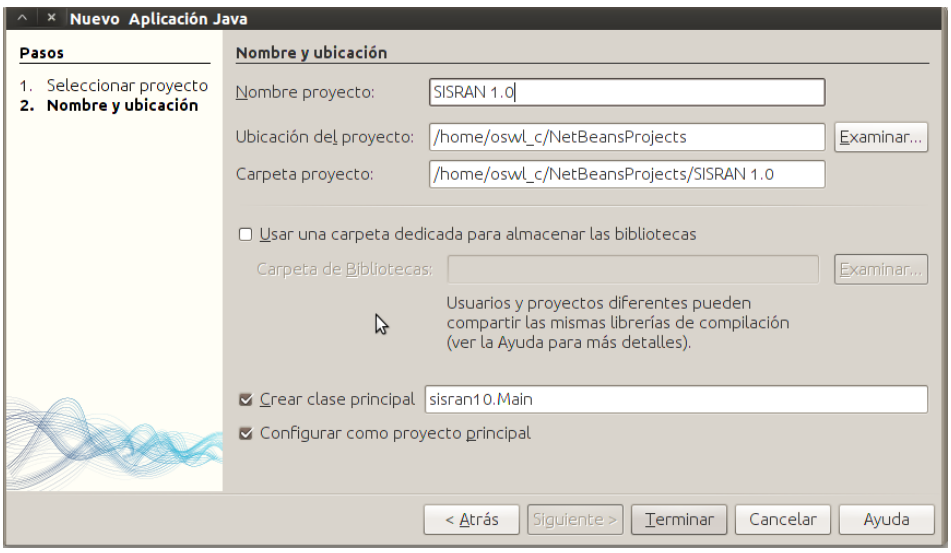


Fig. 22 Nombre y ubicación del proyecto

Escribimos en nombre del proyecto, hacemos clic en terminar.

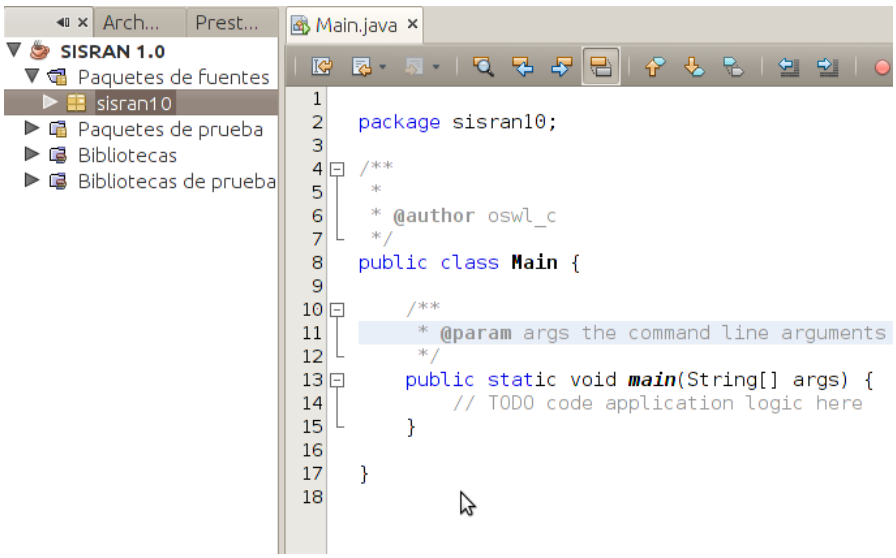


Fig. 23 Ventana principal de Java

Finalmente nos saldrá la siguiente ventana, Main.java, esta es la clase principal del proyecto, pero posteriormente vamos a reemplazarla con un formulario inicial, para lo cual vamos dar clic derecho sobre el paquete (sisran10) y seleccionamos Nuevo/ Otro donde nos aparecerá el siguiente cuadro de dialogo:

3.2.2 Formulario MDI

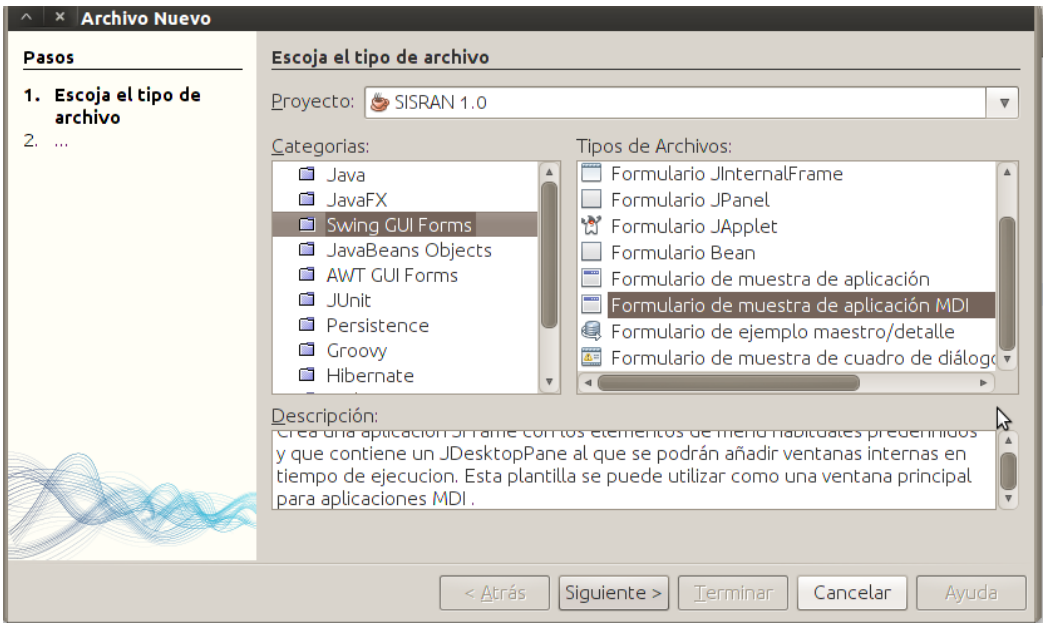


Fig. 24 Formulario MDI

Aquí vamos a seleccionar Swing GUI Forms/ Formulario de Muestra de aplicacion MDI

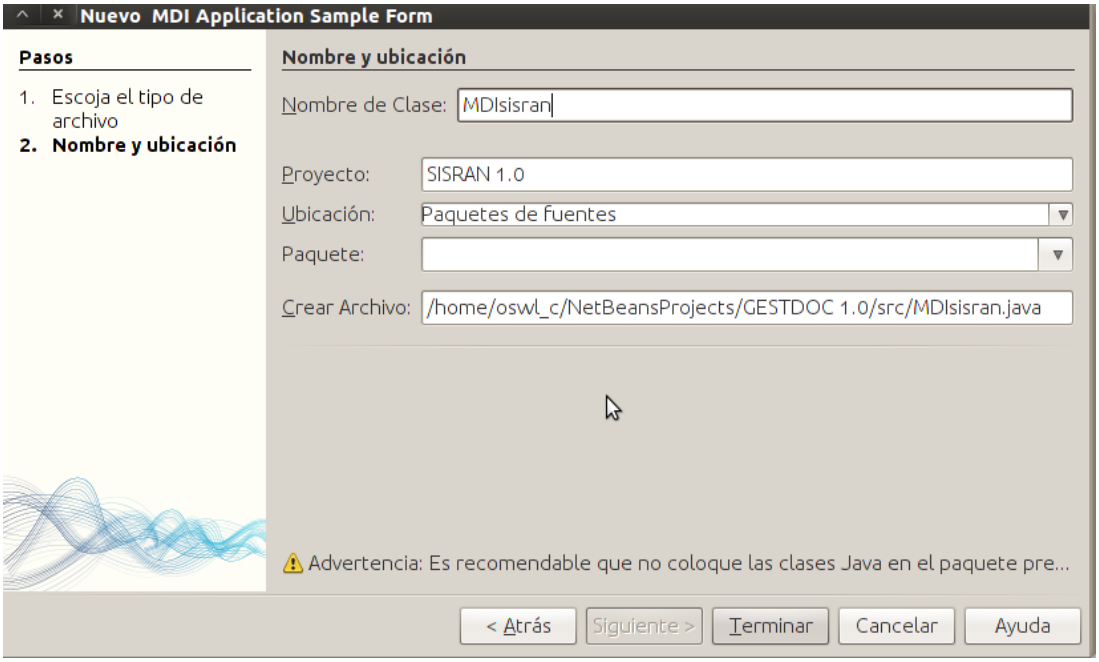


Fig. 25 Propiedades del formulario MDI

Colocamos en nombre y hacemos clic en Terminar, posteriormente se creará un nuevo formulario dentro del paquete sisran10

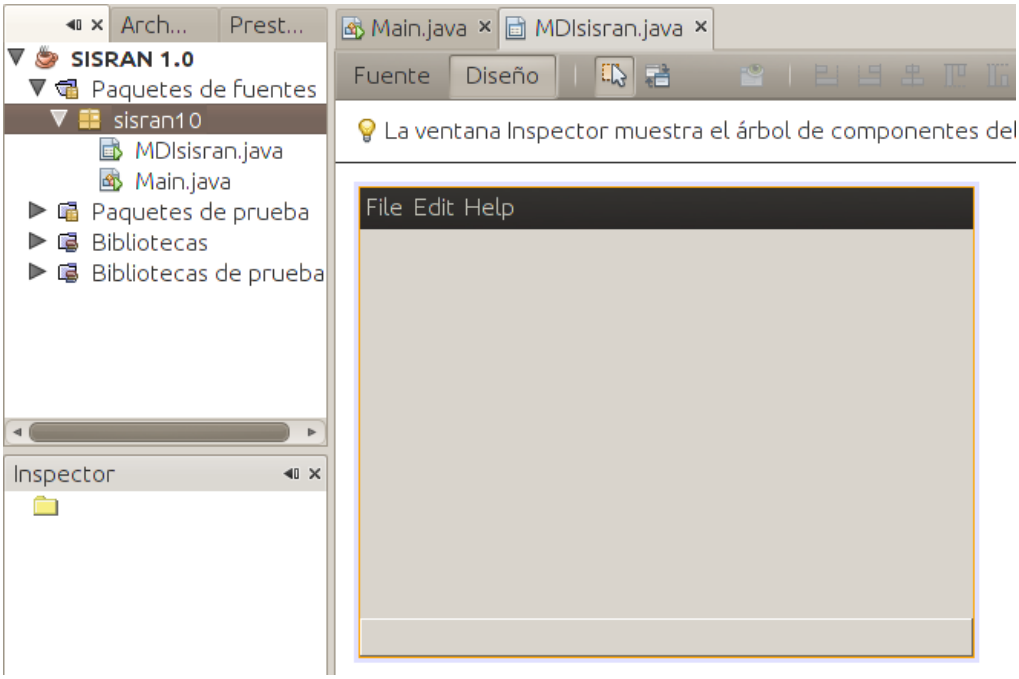


Fig. 26 Diseño del formulario MDI

Ahora vamos hacer que este nuevo formulario sea nuestro proyecto principal, para ello hacemos clic derecho sobre el proyecto SISRAN 1.0 y seleccionamos propiedades en donde nos saldrá el siguiente dialogo:

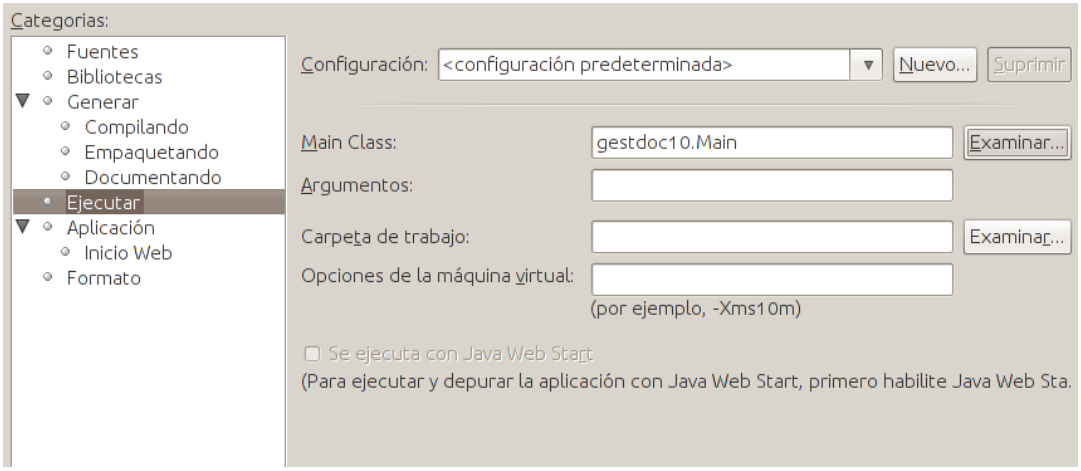


Fig. 27 Configuración Formulario MDI

Aquí seleccionamos Ejecutar y luego junto a Main Class hacemos clic en Examinar

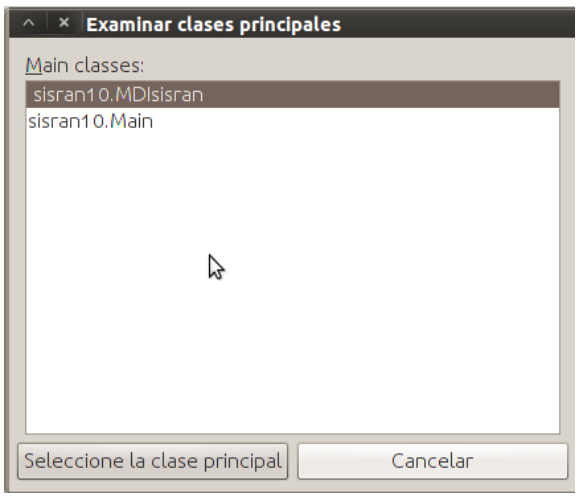


Fig. 28 Ventana de Clases

Aquí seleccionamos MDIsisran y hacemos clic en Seleccione la clase principal y hacemos clic en aceptar, ahora si es que se desea se puede eliminar la clase main, para ello hacemos clic derecho sobre dicha clase y en eliminar respondemos si en la pregunta que nos hará el sistema, y listo la clase esta borrada. Ahora solo nos resta comprobar si la clase MDIsisran es la principal, para ello hacemos clic en el botón ejecutar.

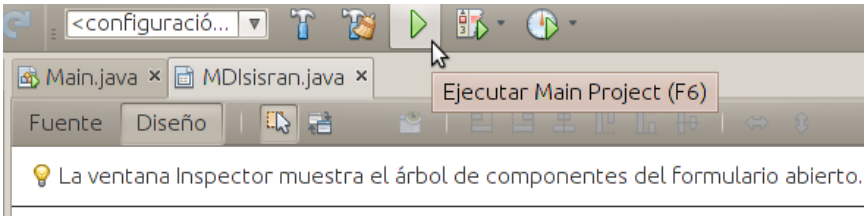


Fig. 29 Herramientas de ejecución

Y nos saldrá la siguiente ventana:

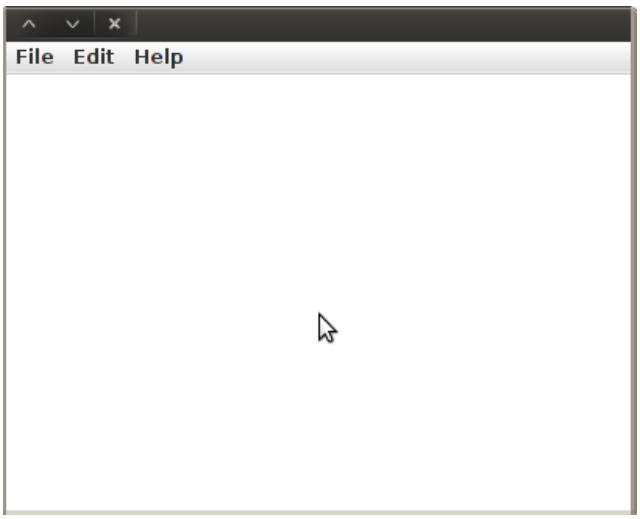


Fig. 30 Ventana de compilación

Esto significa que ya nos está corriendo el programa, ahora solo nos queda rediseñar nuestro formulario de acuerdo a nuestro programa. Quedándonos así:

Fig. 31 Programa Ejecutando

Ahora vamos a agregar los submenús, para ello vamos a la paleta y buscamos Menú Swing / Menú

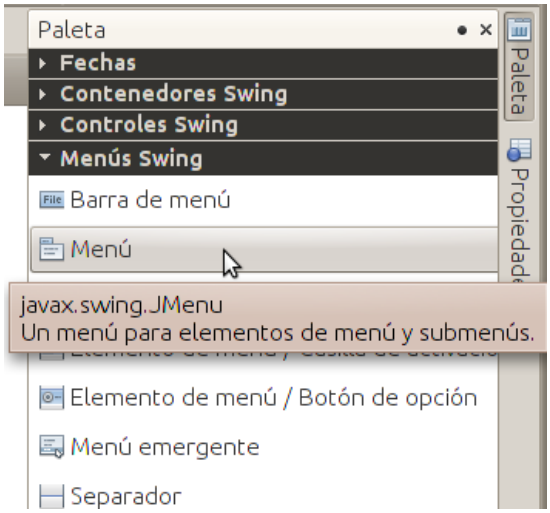


Fig. 32 Agregar Submenús

Lo arrastramos hacia El menú que deseemos y lo renombramos y al final nos va a quedar así:

Fig. 33 Selección de Submenús

Ahora para que nuestro formulario inicie de forma maximizada vamos a propiedades/ ExtendedState y le asignamos 6 ya que por defecto estará 0.

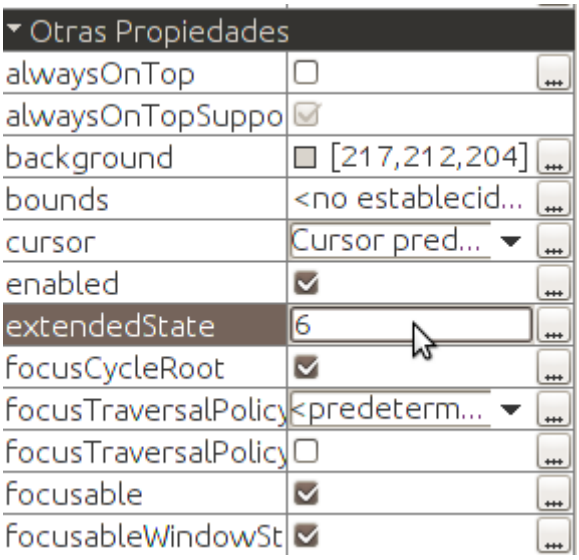


Fig. 34 Propiedades de los Menús

3.2.3 Formulario JInternalForm

Un JInternalForm es un formulario especialmente diseñado para que pueda ser agregado al interior de un formulario MDI, tal como en otros lenguajes se los conoce como formularios hijos.

Para ello hacemos clic derecho sobre el paquete sisran10 y seleccionamos nuevo/ otro y nos saldrá la siguiente ventana:

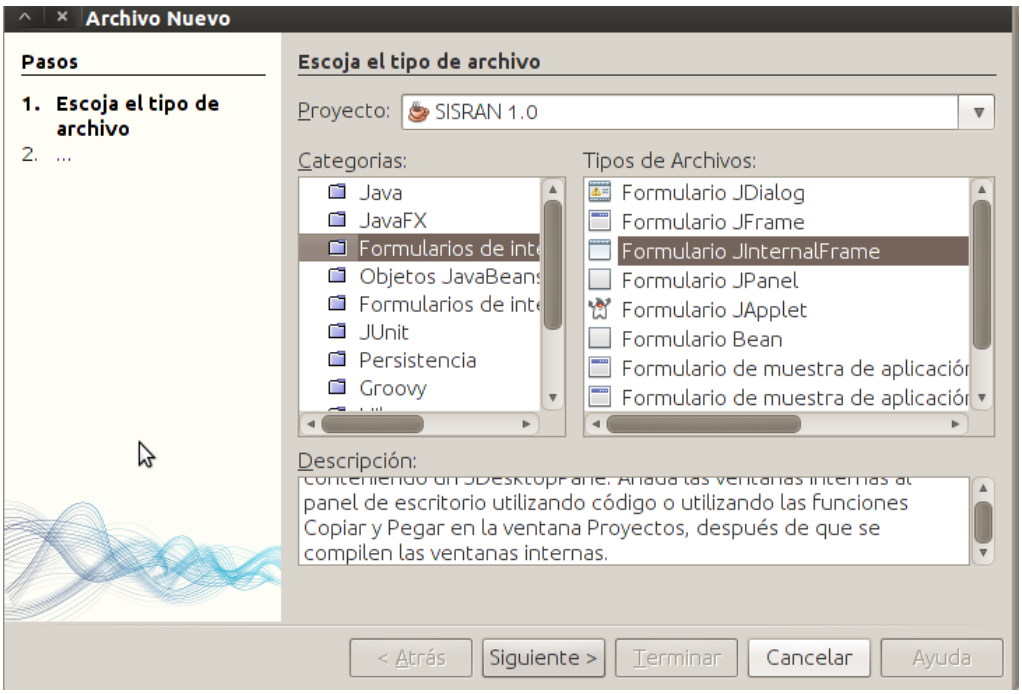


Fig. 35 Formulario JInternalForm

En este dialogo seleccionamos Formularios de interface grafica swing / Formulario JinternalFrame, y hacemos clic en siguiente.

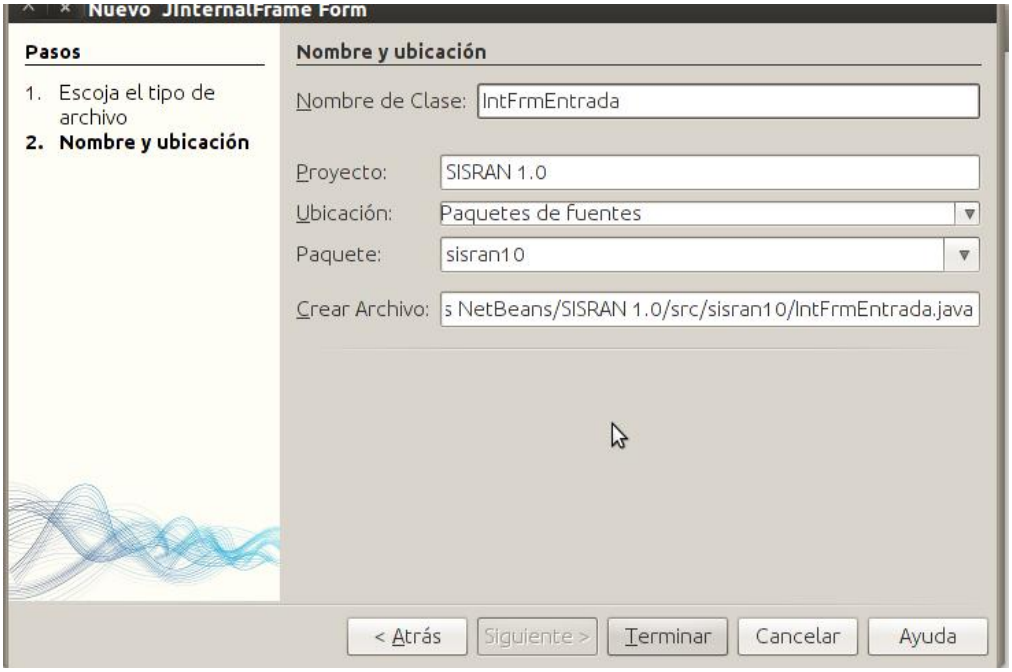


Fig. 36 Nombre y ubicación del formulario JinternalFrame

Y en esta ventana asignamos un nombre al Formulario y hacemos clic en Terminar.

Ahora simplemente nos queda realizar el diseño del formulario quedándonos de la siguiente forma:

En propiedades activamos la casilla de closable y en defaultCloseOperation le asignamos DISPOSE.

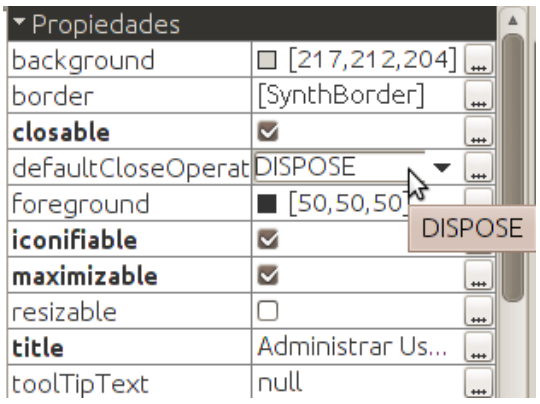


Fig. 37 Propiedades del Formulario JinternalFrame

Una vez hecho esto, solo nos falta hacer que el Formulario MDI pueda abrir el Formulario JinternalFrame, para ello abrimos el Formulario MDIsisran y seleccionamos Fuente

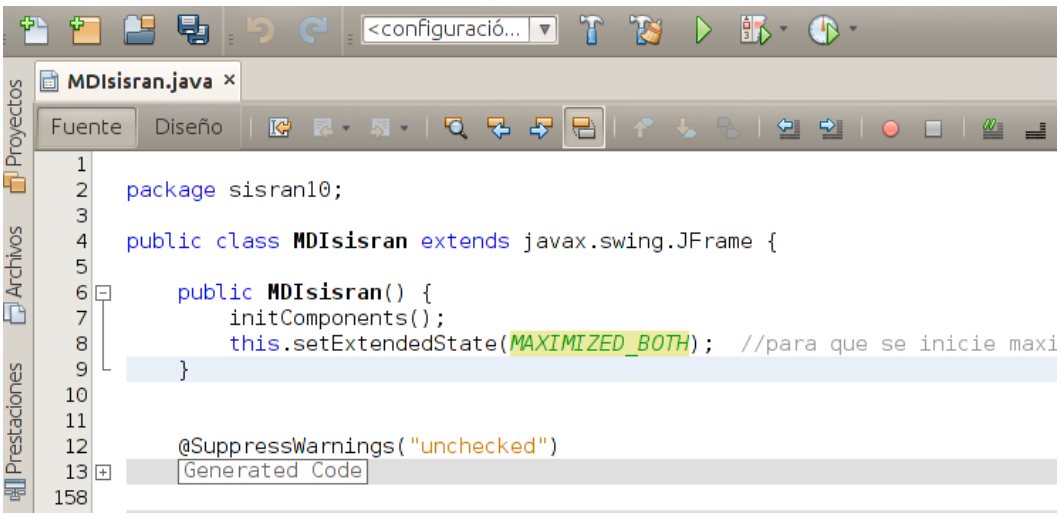


Fig. 38 Ubicación del Formulario

Aquí como podemos apreciar se encuentra el código fuente, por tanto aquí es en donde vamos a proceder a realizar la programación, ahora para que nuestra JinternalFrame pueda ser iniciado desde el Formulario MDI, procedemos de la siguiente manera:

nos vamos de nuevo al diseño del formulario y en el menú Administrar/ Usuario Rancho hacemos clic derecho y seleccionamos Eventos/ Action/ action permformed, una vez hecho clic en este botón nos enviara de nuevo a código fuente, justo en el nuevo método actionperformed del botón seleccionado, en donde vamos a colocar el siguiente código:

```
private void btnUsuarioRanActionPerformed(java.awt.event.ActionEvent evt) {
    IntFrmUsuarioRan usRan = new IntFrmUsuarioRan();
    desktopPane.add(usRan);
    usRan.setVisible(true);
}
```

Fig. 39 Programación del formulario

Explicando el código del grafico primero creamos una nueva instancia del Formulario JinternalFrame:

IntFrmUsuarioRan usRan = new IntFrmUsuarioRan();

Luego agregamos este objeto dentro del panel de escritorio del formulario MDIsisran

desktopPane.add(usRan);

Y finalmente establecemos la propiedad de visible a verdadero

usRan.setVisible(true);

Y con eso nuestro formulario JinternalFrame se lanzara dentro de nuestra aplicación.

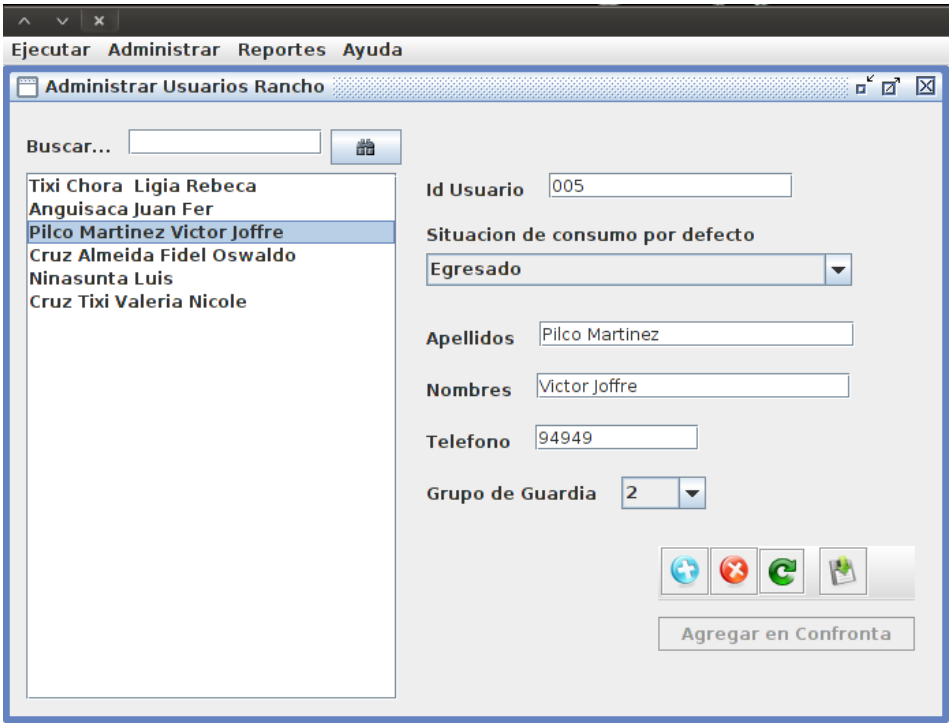


Fig. 40 Ejecución del formulario JinternalFrame

3.3 Programación en Java con NetBeans IDE 6.9.1

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90.

El nombre del lenguaje curiosamente fue acuñado en una cafetería frecuentada por algunos de los miembros del equipo de desarrollo. Pero no está claro si es un acrónimo o no, aunque algunas fuentes señalan que podría tratarse de las iníciales de sus creadores: **J**ames

Gosling, **A**rthur **V**an Hoff, y **A**ndy Bechtolsheim. Otros abogan por el siguiente acrónimo, **J**ust **A**nother **V**ague Acronym ("sólo otro acrónimo ambiguo más"). La hipótesis que más fuerza tiene es la que Java debe su nombre a un tipo de café disponible en la cafetería cercana, de ahí que el icono de java sea una taza de café caliente. Un pequeño signo que da fuerza a esta teoría es que los 4 primeros bytes (el número mágico) de los archivos .class que genera el compilador, son en hexadecimal, 0xCAFEBABE. A pesar de todas estas teorías, el nombre fue sacado al parecer de una lista aleatoria de palabras.

Java es un lenguaje de alto nivel y gran potencia que está orientado a objetos y justamente en este capítulo vamos a realizar un análisis de la programación orientada a objetos usando Java, por lo demás ya sean estructuras condicionales o de repetición son iguales al lenguaje C o C++, y para nuestro proyecto es la programación orientada a objetos será una herramienta fundamental para desarrollar nuestra aplicación.

3.3.1 ¿Qué es la Programación Orientada a Objetos?

La programación orientada a objetos es un paradigma de programación que utiliza objetos como elementos fundamentales en la construcción de la solución. Surge en los años 70 con Simula 67, un lenguaje diseñado para hacer simulaciones. Está basado en varias técnicas, incluyendo herencia, abstracción, polimorfismo y encapsulamiento y en la actualidad, existe gran variedad de lenguajes de programación que soportan la orientación a objetos.

3.3.2 ¿Qué es un objeto?

Un objeto es una abstracción de algún hecho o ente del mundo real que tiene atributos que representan sus características o propiedades y métodos que representan su comportamiento o acciones que realizan. Todas las propiedades y métodos comunes de los objetos se encapsulan o se agrupan en clases.

3.3.3 ¿Qué es una clase?

Una clase es una plantilla o un prototipo para crear objetos, por eso se dice que los objetos son instancias de clases. Por Ejemplo:

```
class Universidad{

String nombre;

String ubicación;

Universidad(String nombre, String ubicación){

this.nombre=nombre;

this.ubicacion=ubicacion

}

public void mostrarCaracteristicas(){

System.out.printf(nombre + " " + ubicacion);

}

}
```

La clase universidad es una plantilla sobre la cual se puede definir todas las características y comportamientos de los objetos de tipo universidad, el objeto de esta clase de definiría de la siguiente forma:

```
Universidad u = new  Universidad("ESPE-L", "Latacunga");

u.mostrarCaracteristicas();
```

3.3.4 Características de la Programación Orientada a Objetos

*** Abstracción:** denota las características esenciales de un objeto, donde se capturan sus comportamientos. El proceso de

abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real. La abstracción es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella podemos llegar a armar un conjunto de clases que permitan modelar la realidad o el problema que se quiere atacar.

*** Encapsulamiento:** Significa reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad. Esto permite aumentar la cohesión de los componentes del sistema.

*** Modularidad:** Se denomina Modularidad a la propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes. Estos módulos que se puedan compilar por separado, pero que tienen conexiones con otros módulos.

*** Principio de ocultación:** Cada objeto está aislado del exterior. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas, solamente los propios métodos internos del objeto pueden acceder a su estado. Esto asegura que otros objetos no pueden cambiar el estado interno de un objeto de maneras inesperadas, eliminando efectos secundarios e interacciones inesperadas.

*** Polimorfismo:** comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado. Cuando esto ocurre en "tiempo de ejecución", esta última característica se

llama asignación tardía o asignación dinámica.

*** Herencia:** las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo. Esto suele hacerse habitualmente agrupando los objetos en clases y estas en árboles o enrejados que reflejan un comportamiento común. Cuando un objeto hereda de más de una clase se dice que hay herencia múltiple.

*** Recolección de basura:** la recolección de basura o garbage collector es la técnica por la cual el entorno de objetos se encarga de destruir automáticamente, y por tanto desvincular la memoria asociada, los objetos que hayan quedado sin ninguna referencia a ellos. Esto significa que el programador no debe preocuparse por la asignación o liberación de memoria, ya que el entorno la asignará al crear un nuevo objeto y la liberará cuando nadie lo esté usando.

3.3.5 Ventajas de la Programación Orientada a Objetos

Como ya mencionamos anteriormente la programación orientada a objetos es un paradigma de programación que utiliza objetos o entidades como elementos fundamentales en la construcción de una solución, esto nos da una gran ventaja a la hora de plantear la solución al problema ya que este tipo de programación es muy coherente con la realidad en la que vivimos.

Por Ejemplo:

El objeto de tipo Usuario_Rancho debe estar relacionado con el objeto de tipo Confronta y cada uno de ellos tienen características que los diferencian de otros y que permiten a la vez interactuar entre si gráficamente mostraremos una relación simple entre estos dos objetos.

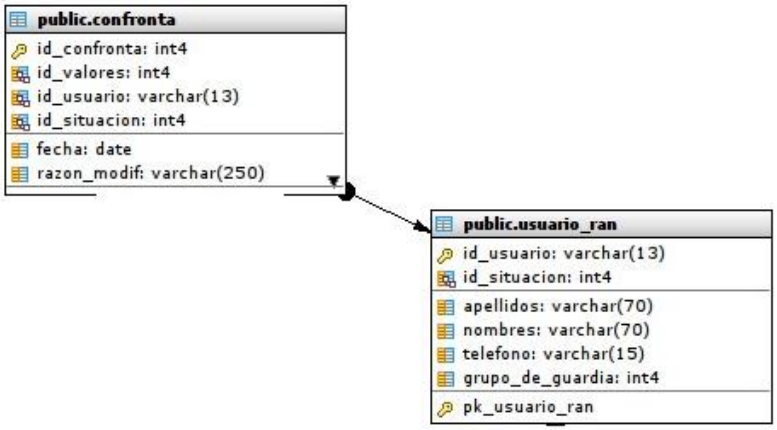


Fig. 41 Ejemplo Diagrama entidad relación

Es simple la Entidad confronta está relacionada con la Entidad Usuario_Ran mediante un campo clave llamado id_usuario.

Este concepto de programación difícilmente puede ser superado por algún otro paradigma de programación ya que al hablar de Entidades, sus características y comportamientos nos están trasladando de un ámbito tan complejo como es el de la programación, a un ámbito más comprensible como es el de la vida real.

3.3.6 Reportes con JasperReports

En NetBeans se puede realizar el diseño de Reportes muy fácilmente gracias al componente llamado JasperReports, el cual es un framework bastante completo para desarrollar reportes tanto web como desktop en Java. Este componente puede ser utilizado dentro de NetBeans o de forma externa como lo hemos utilizado en nuestro proyecto.

3.3.7 Instalación del JasperReports

Para instalar el plugin JasperReports dentro de NetBeans procedemos de la siguiente manera...

Primero debemos descargar el archivo .nbm para NetBeans que muestra el iReport dentro del mismo IDE, desde la siguiente dirección:

http://downloads.sourceforge.net/ireport/iReport-nb-0.9.2.nbm

Después de descargar el archivo .nbm, abrimos el IDE NetBeans. Entramos al menú Herramientas / Complementos, y en la ficha *Descargado*, hacemos clic en el botón Agregar plugins...

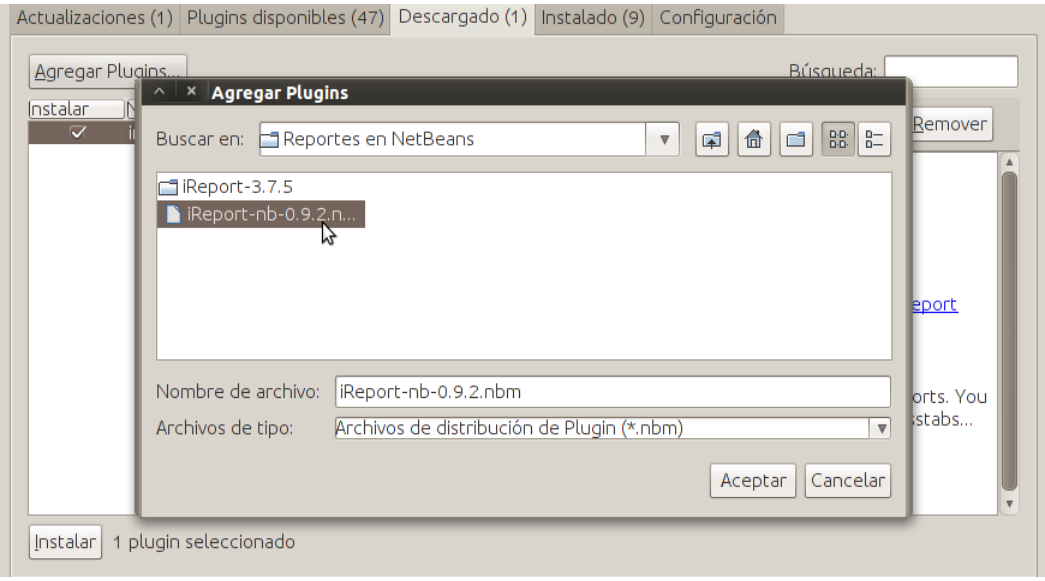


Fig. 42 Descarga de iReport

Seleccionamos el archivo .nbm que previamente hemos descargado. Después de esto se mostrará la ventana lista para instalarse el plugin:

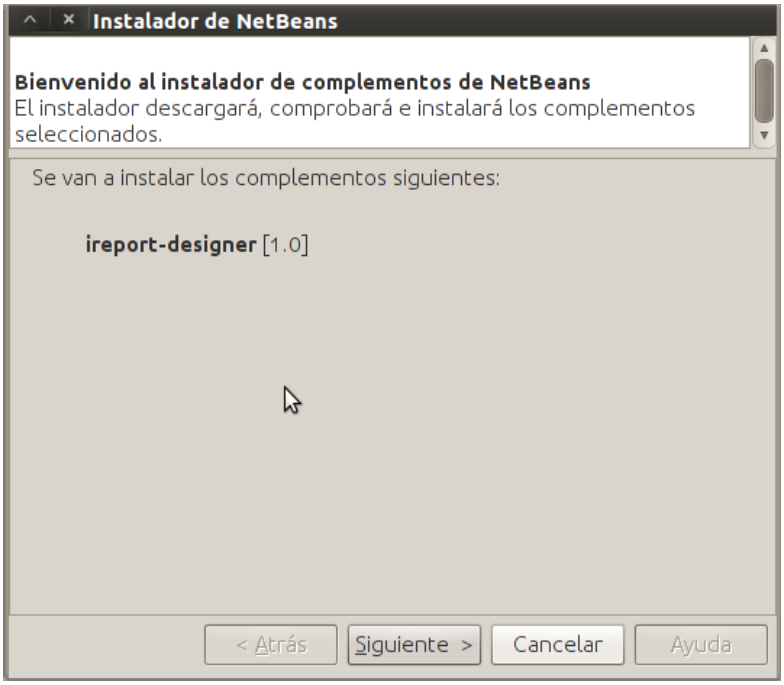


Fig. 43 Instalador de complementos de NetBeans

Hacemos clic en siguiente y aceptamos los términos de la licencia.

Fig. 44 Contrato de licencia

Nos saldrá una advertencia en la cual hacemos clic en Continuar

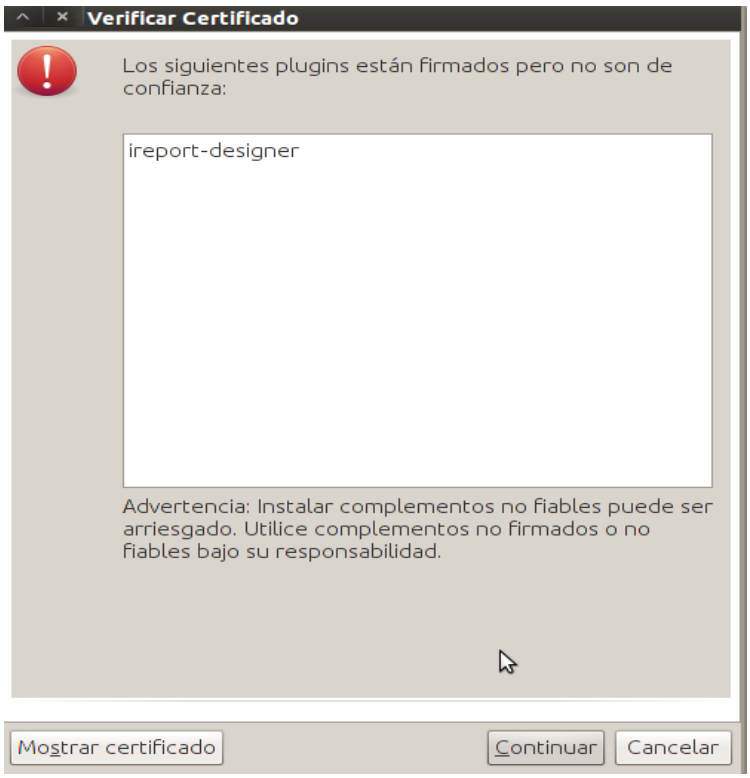


Fig. 45 Verificación de certificado

Listo con esto el complemento iReport ya está instalado dentro de NetBeans, ahora les mostraremos como trabajar con JasperIReports de forma externa al IDE, ya que de igual forma se lo puede hacer desde el interior de la aplicación como desde el exterior, pero la mejor forma es desde externamente ya que permite un mejor funcionamiento de nuestro programa a la hora de usar las bibliotecas del iReport, para ello realizamos los siguiente pasos:

Primero descargamos JasperIReports desde la siguiente página:
www.jarperreports.org

Ahora solo tenemos que descomprimir dicho archivo

Fig. 46 Instalador empaquetado de iReport

Damos doble clic y en la ventana gestor de archivadores hacemos clic en Extraer y le indicamos el lugar en donde queremos que se extraigan los archivos

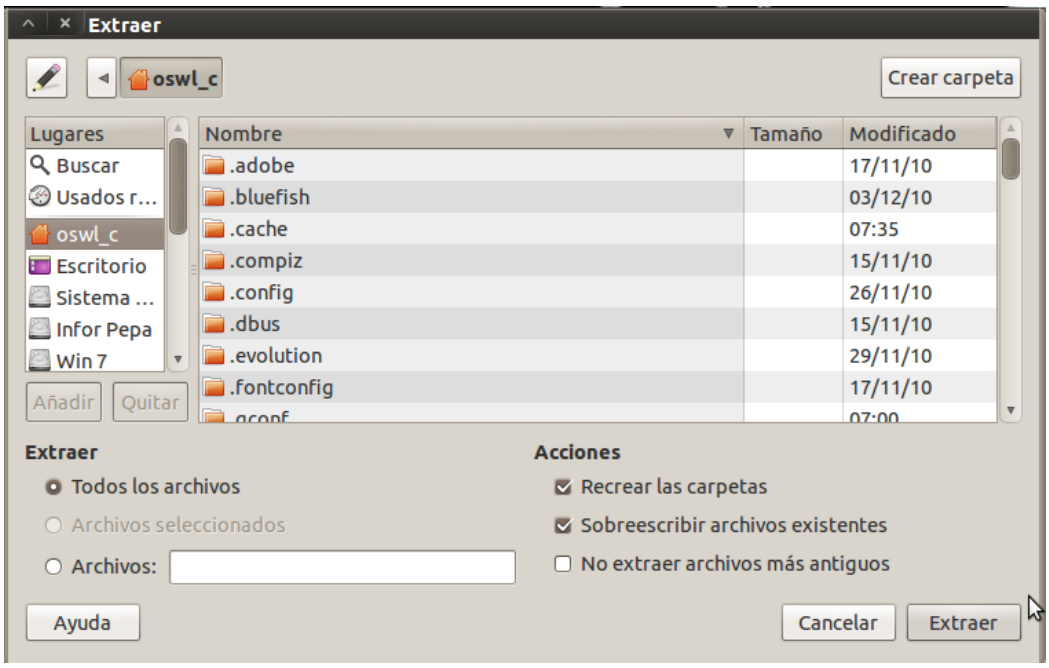


Fig. 47 Extraer archivos empaquetados

Hacemos clic en extraer. Ahora ingresamos en la carpeta extraída y abrimos la subcarpeta .bin

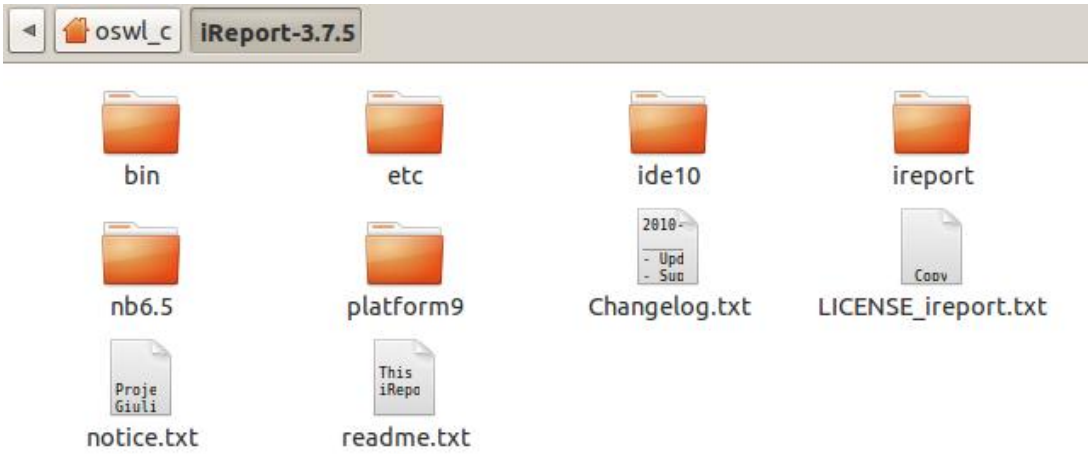


Fig. 48 Localización de carpetas y archivos

Dentro de esta carpeta vamos a encontrar los siguiente archivos

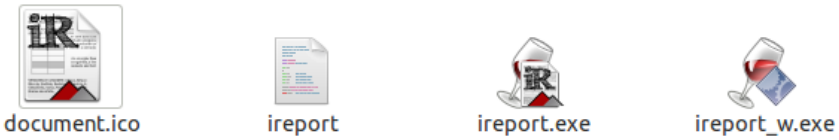


Fig. 49 Archivos de iReport

Ahora para Linux vamos a trabajar con el archivo ireport al cual tenemos que darle permisos de ejecución, para lo cual hacemos clic derecho sobre el archivo y seleccionamos propiedades y dentro de estas seleccionamos permisos.

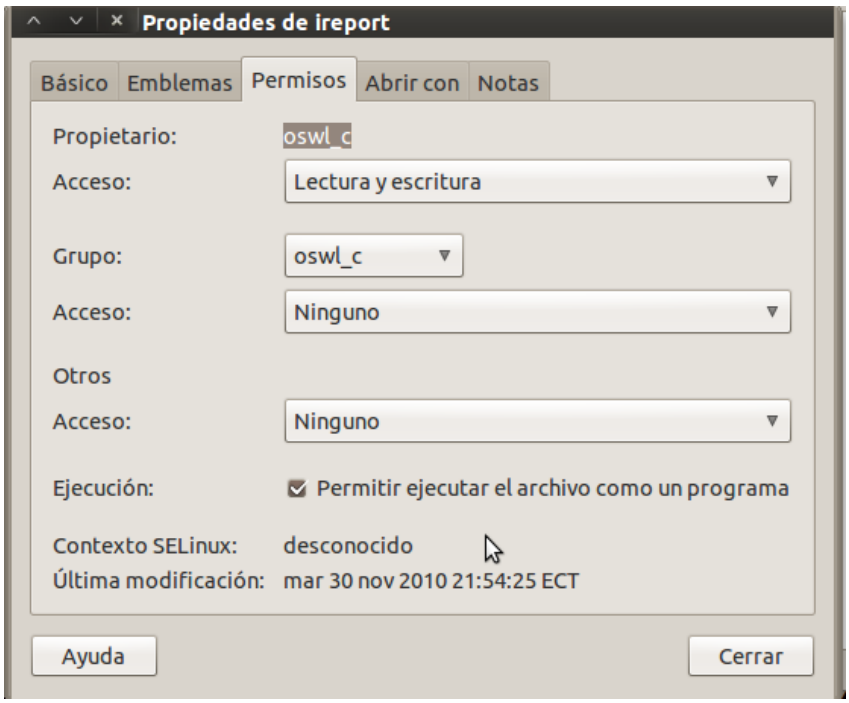


Fig. 50 Propiedades de iReport

Activamos la casilla Permitir ejecutar el archivo como un programa. Cerramos y hacemos doble clic en el archivo y luego hacemos clic en ejecutar

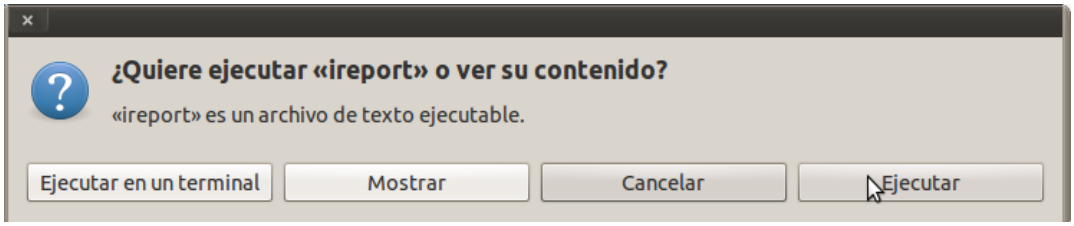


Fig. 51 Ventana para Ejecutar de iReport

Y se nos abrirá el Diseñador de JarperReports

Fig. 52 Portada de instalación iReport

3.3.8 Entorno de Diseño del iReport

Dentro del diseñador vamos a ver los siguientes elementos

Fig. 53 Entorno de diseño iReport

3.3.9 Conexión con la base de datos

Hacemos clic en The connection setup wisard, o asistente de conexión el cual abrirá el siguiente dialogo:

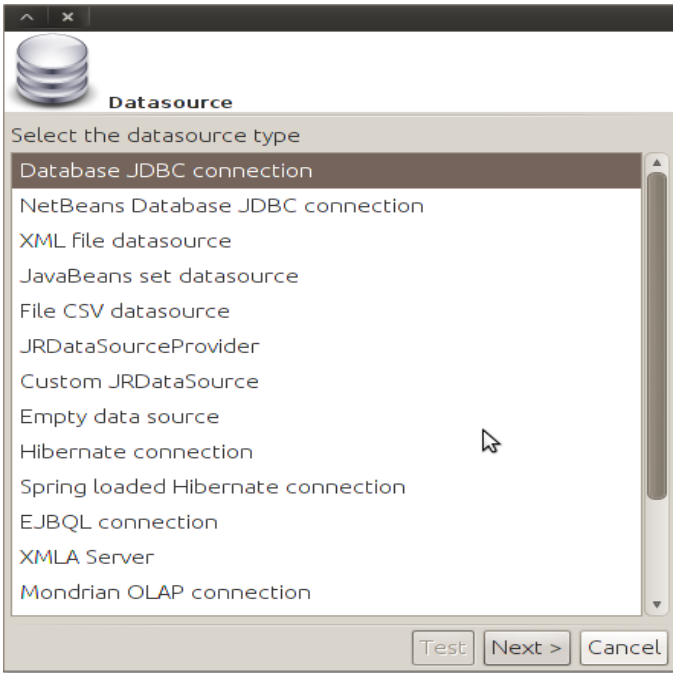


Fig. 54 Conexión de iReport con la Base de datos

En el cual seleccionamos como datasourse a DataBase JDBC Connection y hacemos clic en Next, y la ventana siguiente llenamos los datos de la siguiente forma:

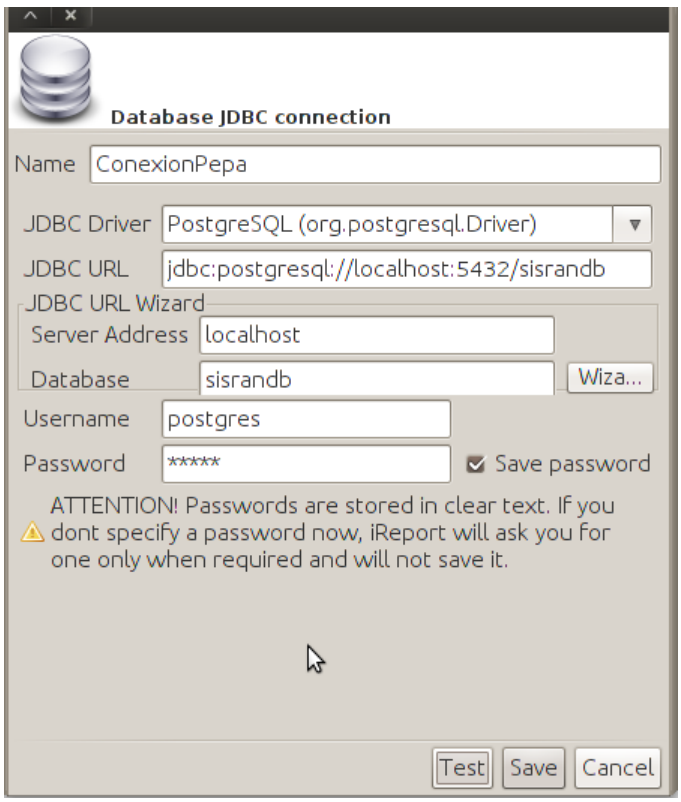


Fig. 55 Propiedades de la conexión

Hacemos clic en Test y nos debe salir lo siguiente:

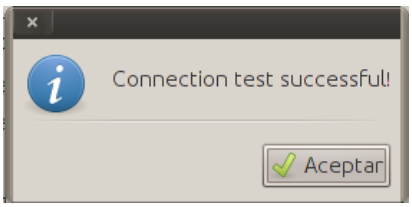


Fig. 56 Ventana de Test de conexión exitosa

Finalmente hacemos clic en Save.

3.3.10 Diseño del Informe

Hacemos clic en el asistente de diseño de formularios.

Fig. 57 Asistente para diseño de formularios

Luego nos aparecerá una ventana en la cual seleccionamos el diseño para nuestro reporte y hacemos clic en Launch Report Wizard.

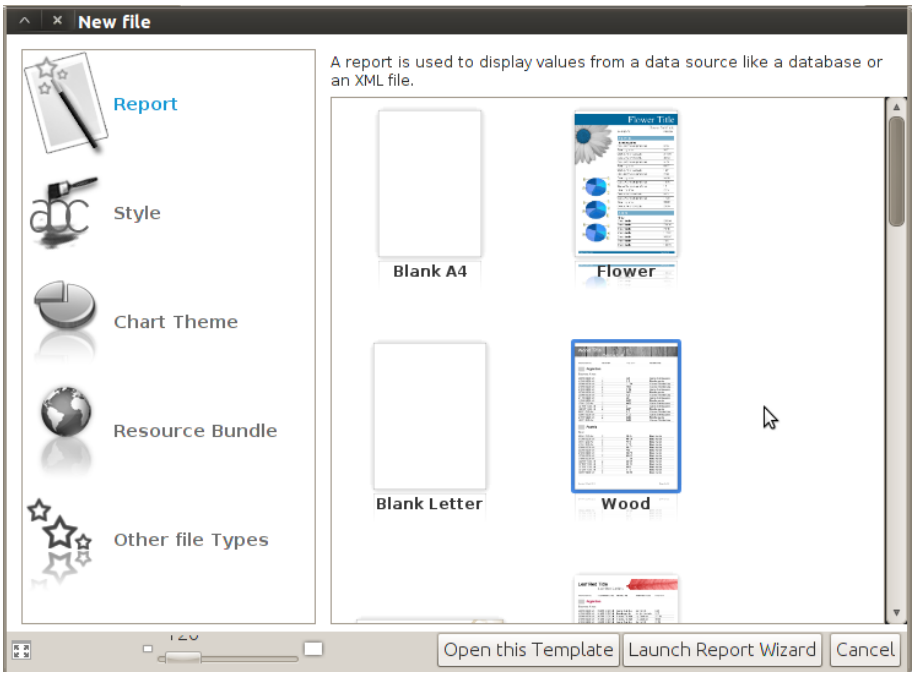


Fig. 58 Clases de formularios

Una vez abierto el asistente de creación de reportes le damos un nombre y hacemos clic en siguiente

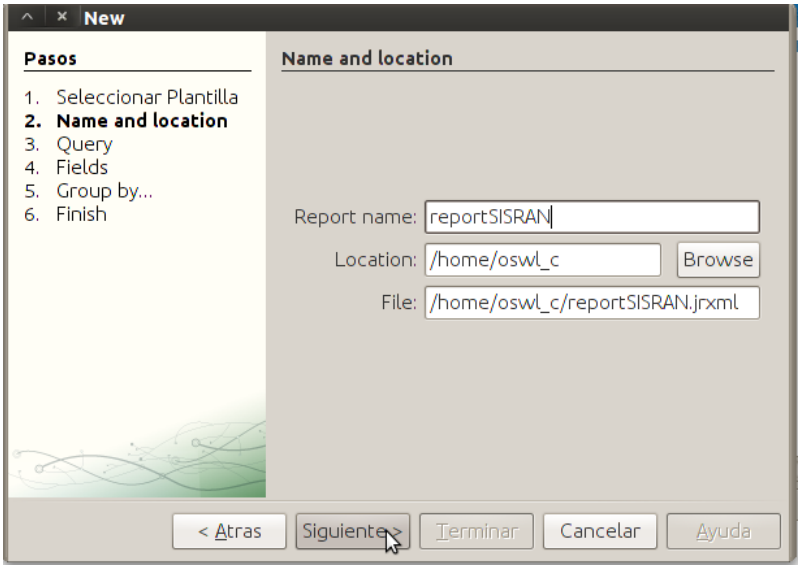


Fig. 59 Nombre y localización de Formularios

En la nueva ventana hacemos seleccionamos la conexión creada anteriormente y hacemos clic en Desing query, en donde seleccionamos la tabla o tablas que vamos a incluir en nuestro formulario

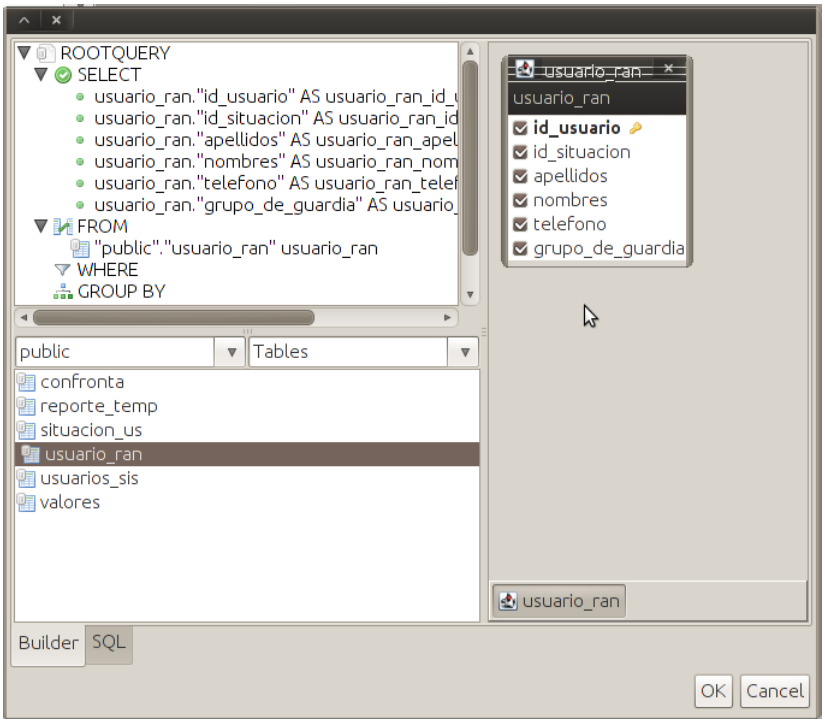


Fig. 60 Tablas para incluir en el formulario

Hacemos clic en OK y luego en Siguiente

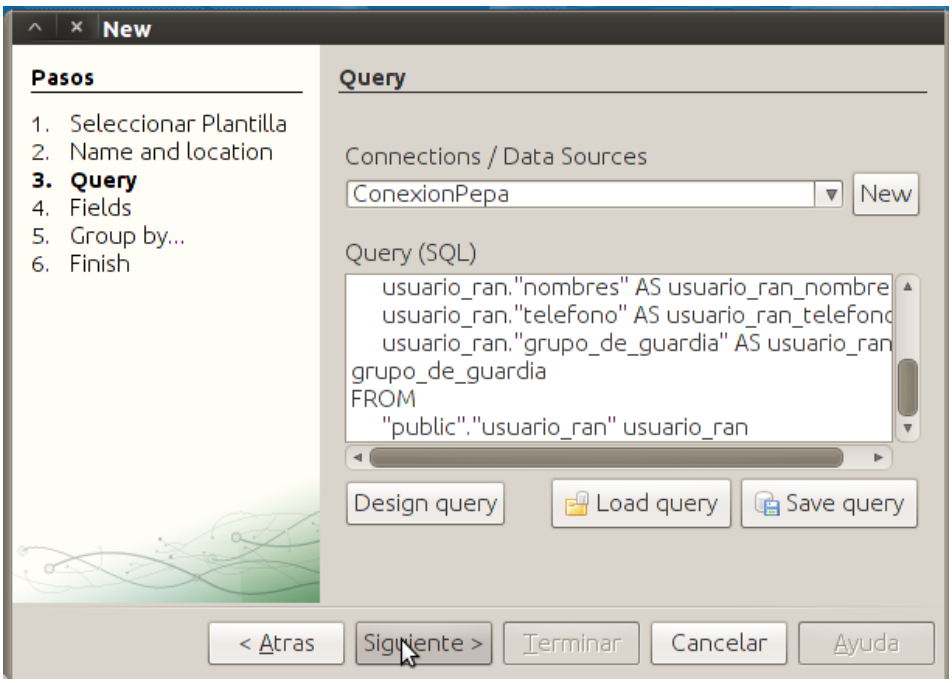


Fig. 61 Conexión del Query

Luego en la siguiente ventana seleccionamos todos los campos de la tabla que vayamos a utilizar en nuestro reporte

Fig. 62 Campos utilizados en el reporte

Y hacemos clic en siguiente y en la nueva ventana como no vamos a agrupar datos hacemos nuevamente clic en siguiente.

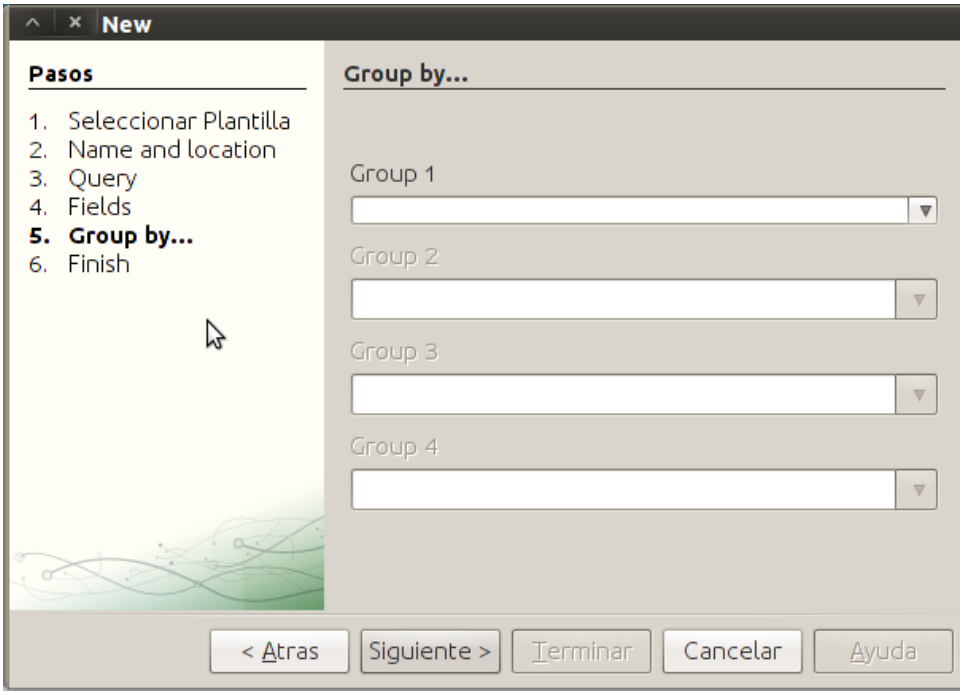


Fig. 63 Ventana para agrupar datos

Ya ahora hacemos clic en Terminar y listo nuestro reporte está listo para ser usado.

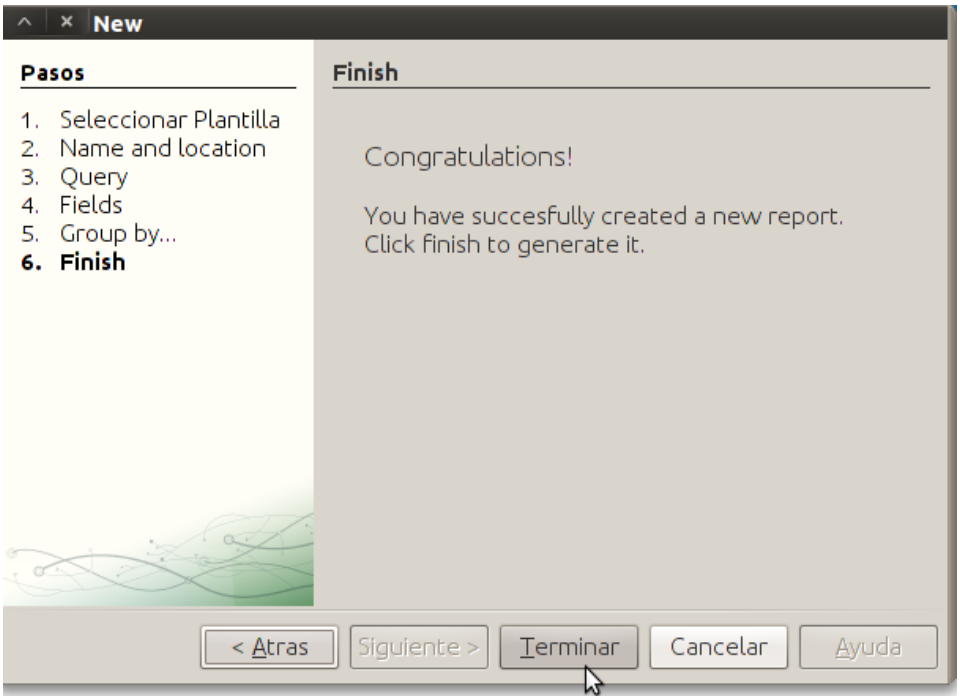


Fig. 64 Ventana de creación correcta

3.3.11 Manejo de campos dentro del informe

Ahora nuestro reporte luce más o menos así:

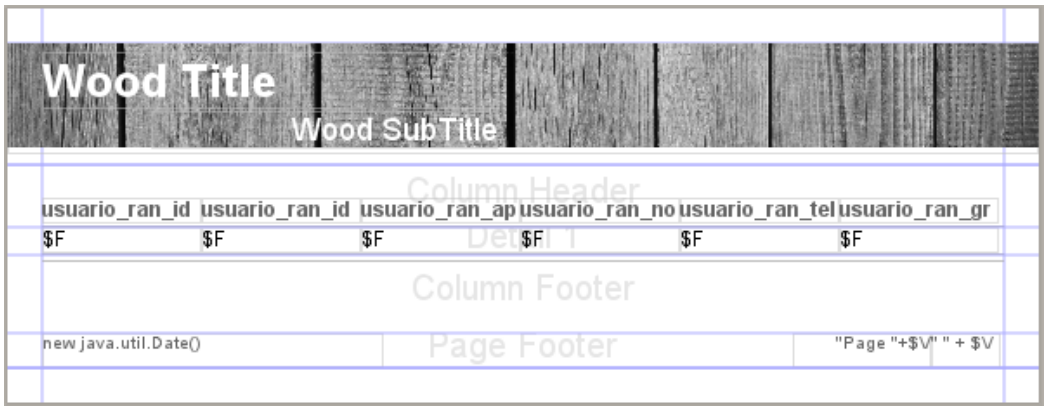


Fig. 65 Manejo de campos

Aquí simplemente debemos reacomodar los campos para visualizar un diseño más o menos así:

Fig. 66 Diseño de los campos para SISRAN 1.0

Ahora estamos listos para ver nuestros datos, para ello hacemos clic en Preview

Cedula	Apellidos	Nombres	Situacion	Total Consum.			Total Valores			
				Des.	Alm.	Mer.	Des.	Alm.	Mer.	Total
006	Anguisaca	Juan Fer	Egresado	5	5	5	4.25	7.5	4.75	16.5
001	Cruz Almeida	Fidel Oswaldo	Egresado	5	5	5	4.25	7.5	4.75	16.5
005	Pilco Martinez	Victor Joffre	Egresado	10	10	10	8.5	15.0	9.5	33.0
008	Ninasunta	Luis	Egresado	10	10	10	8.5	15.0	9.5	33.0
003	Tixi Chora	Ligia Rebeca	Alumno Militar No Residente	5	24	5	4.25	36.0	4.75	45.0
002	Cruz Tixi	Valeria Nicole	Alumno Militar Residente	25	25	25	21.25	37.5	23.75	82.5

Fig. 67 Vista de campos creados

3.3.12 Clase Lanzadora del Informe

Listo ya esta creado nuestro reporte ahora para visualizarlo desde NetBeans lo copiamos en la carpeta del proyecto y creamos la siguiente clase

```
package sisran10;

import java.sql.Connection;

import java.sql.DriverManager;

import net.sf.jasperreports.engine.*;

import net.sf.jasperreports.engine.util.JRLoader;

import net.sf.jasperreports.view.*;

public class IniciarReporte {

   Connection conn=null;

   IniciarReporte(String archivo) {

      try{

         conn=
DriverManager.getConnection("jdbc:postgresql://localhost:5432/
```

```
sisrandb","postgres","espel");
        }catch (Exception ex){msg.showMessageDialog(null,
ex.getMessage());}
        try{
            if(archivo == null){
                System.out.println("No se encuentra el archivo.");
                System.exit(2);
            }
            JasperReport masterReport= null;
            try{
                masterReport=
(JasperReport)JRLoader.loadObject(archivo);
            }catch (Exception ej) {
                System.out.println("Error cargando el reporte maestro:
" + ej.getMessage());
                System.exit(3);
            }
            JasperPrint
jasperPrint=JasperFillManager.fillReport(masterReport,null,conn
);
            JasperViewer jviewer= new
JasperViewer(jasperPrint,false);
            jviewer.setTitle("Reporte");
            jviewer.setVisible(true);
```

```
            conn.close();

        }catch(Exception j){System.out.println("Mensaje de
    Error:"+j.getMessage());}

    }

}
```

Mediante esta clase podremos abrir desde nuestro proyecto en NetBeans nuestro Reporte creado con el iReport de JasperReports y listo esto es todo en cuanto a reportes y ahora estamos listos para continuar con el siguiente capítulo.

CAPÍTULO 4: GESTIÓN DE BASES DE DATOS CON JAVA EN NETBEANS

4.1 Introducción a la gestión de base de datos.

En este capítulo nos centraremos en cómo conectar con una base de datos usando JDBC y también en el uso de JPA es decir Java Persistance API para la gestión de una base de datos.

Con estas herramientas el programador podrá fácilmente, realizar sus trabajos en cualquier plataforma, ya sea Linux o Windows. El entorno de desarrollo NetBeans es un gran apoyo a la hora de programar con bases de datos, ya que tiene asistentes para realizar las diferentes conexiones y que además crean automáticamente clases entidad, las mismas que serán usadas a los largo del desarrollo del nuevo sistema.

4.2 Pasos para agregar un nuevo Driver

Como ya mencionamos anteriormente, es necesario el uso de Drivers para realizar las conexiones entre una aplicación y el motor de base de datos, en nuestro caso estamos usando PostgreSQL 8.4 y en NetBeans IDE 9.6.1 el Driver por defecto para PostgreSQL es la versión 8.3, para agregar la nueva versión del driver simplemente procedemos de la siguiente forma:

Primero descargamos el driver de la siguiente página

www.postgresql.org/downloads

Una vez descargado nuestro driver abrimos el NetBeans y nos vamos al menú Herramientas / Bibliotecas en esta ventana buscamos y seleccionamos Driver JDBC PostgreSQL y hacemos clic en agregar archivo jar/carpeta

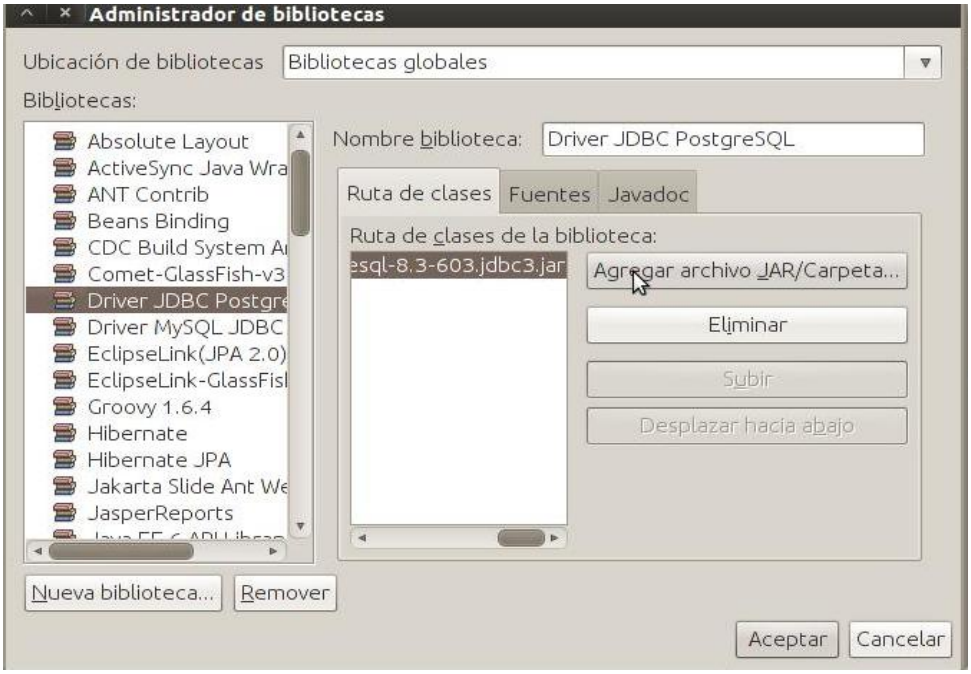


Fig. 68 Administrador de bibliotecas para agregar Driver

En la nueva ventana buscamos nuestro archivo descargado y hacemos clic en Agregar archivo Jar/Carpeta

Fig. 69 Examinar archivos JAR

Y hacemos clic en aceptar y listo nuestro nuevo driver está listo para ser usado.

4.3 Introducción a JDBC

Una vez agregado el nuevo Driver vamos a estudiar un poco acerca de las conexiones usando JDBC o Java Data Base Connectivity, la cual es una API que permite la ejecución de operaciones sobre bases de datos desde

el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

Para utilizar una base de datos particular, el usuario ejecuta su programa junto con la biblioteca de conexión apropiada al modelo de su base de datos, y accede a ella estableciendo una conexión, para ello provee el localizador a la base de datos y los parámetros de conexión específicos. A partir de allí puede realizar cualquier tipo de tareas con la base de datos a las que tenga permiso tal como es: consulta, actualización, creación, modificación y borrado de tablas, esto lo vamos a analizar más a fondo en el siguiente tema Pasos para la conexión entre NetBeans y PostgreSQL y más adelante en Gestión de bases de datos con JDBC.

4.4 Pasos para la conexión entre NetBeans y PostgreSQL mediante JDBC

Para realizar una conexión mediante JDBC, debemos usar una clase llamada Conexión en la cual vamos a insertar el siguiente código

```
package sisran10;

import java.sql.*;

public class Conexion {

    public Connection conn;

    public Statement stm;

    public Conexion() {

        try {

            conn = DriverManager.getConnection(

                    "jdbc:postgresql://localhost:5432/sisrandb/",

                    "postgres",
```

```
                "espel");

            stm                                                =
conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE);

        } catch (SQLException ex) {

            System.err.println(ex.getMessage());

        }

    }

}
```

Luego para que no nos salgan errores vamos a dar clic derecho en Bibliotecas y seleccionamos Agregar Archivo Jar/Carpeta, esto se lo hace para que junto a nuestro proyecto siempre vaya el archivo del Driver de PostgreSQL 8.4

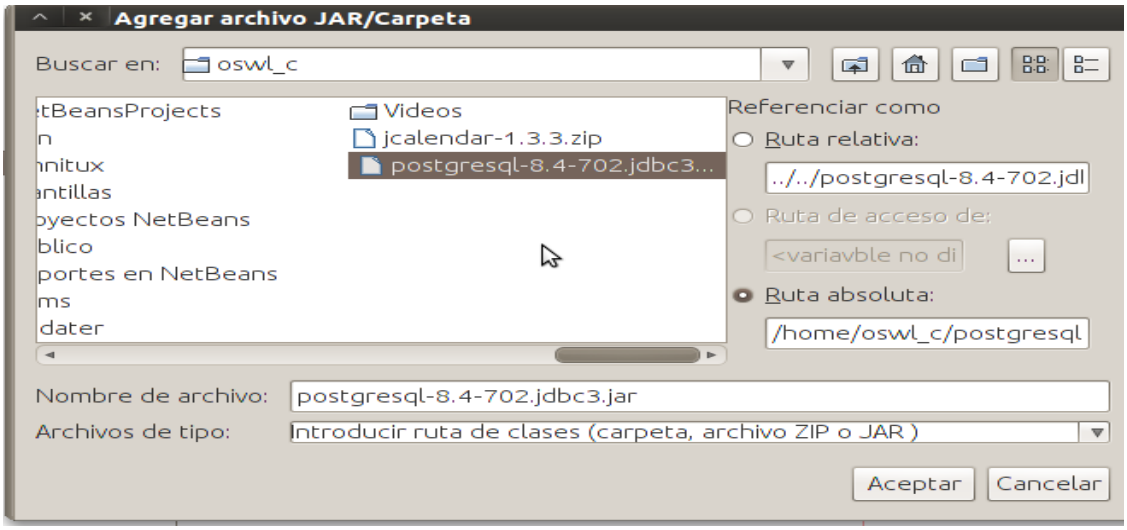


Fig. 70 Agregar archivo JAR

Al hacer clic en el botón aceptar nuestro proyecto ya puede establecer la conexión con PostgreSQL 8.4

4.5 Gestión de Bases de Datos Usando JDBC

Una vez creada la clase Conexión va a ser muy sencillo realizar tareas de gestión de bases de datos ya que simplemente vamos a usar Lenguaje SQL de la siguiente forma.

Primero creamos un objeto de tipo Conexión

```
Conexion c = new Conexion();
```

Con este objeto llamamos a la propiedad stm de tipo Statement, con la cual podremos ejecutar nuestras instrucciones SQL, de la siguiente forma:

```
ResultSet rs = c.stm.executeQuery("select IdUsuario from tblusuariossistema");
```

Y con este ResultSet ya podemos extraer los diferenctes registros, aquí hay un Ejemplo que nos muestra como cargar un ComboBox

```
 while(rs.next()){
        CBUsuario.addItem(rs.getString("IdUsuario")); }
```

Tan sencillo como eso es insertar nuevos registro tal como se lo hace en este ejemplo:

```
try{
        String us = this.txtUsuario.getText();
        String cl= this.txtClave.getText();
        c.stm.execute("inset into UsuarioSis (usuario,clave) values ('"+us+"','"+cl+"')");
 } catch (SQLException ex) {
        System.err.println(ex.getMessage());
 }
```

Como podemos ver solo hay que concatenar cadenas y listo es muy sencillo, al igual que realizar consultas, pero hay un método aun más potente para Gestionar Bases de datos el cual se lo denomina JPA.

4.6 Introducción a JPA

Ahora nos toca estudiar a la técnica de programación de bases de datos más potente y útil a la hora de realizar programas de gran complejidad, JPA significa Java Persistence API esto quiere decir que es la API de persistencia desarrollada para la plataforma Java EE e incluida en el estándar EJB3. Esta API busca unificar la manera en que funcionan las utilidades que proveen un mapeo objeto-relacional. El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos, como sí pasaba con EJB2, y permitir usar objetos regulares (conocidos como POJOs) y por cierto API significa Aplication Programing Interface, en español esto significa que es una Interfaz de Programación de Aplicaciones.

4.7 Pasos para la conexión entre NetBeans y PostgreSQL usando JPA

Para conectar NetBeans y PostgreSQL usando JPA simplemente realizamos los pasos que ya indicamos anteriormente para crear una nueva conexión, para ello nos vamos a Presentaciones / Bases de Datos

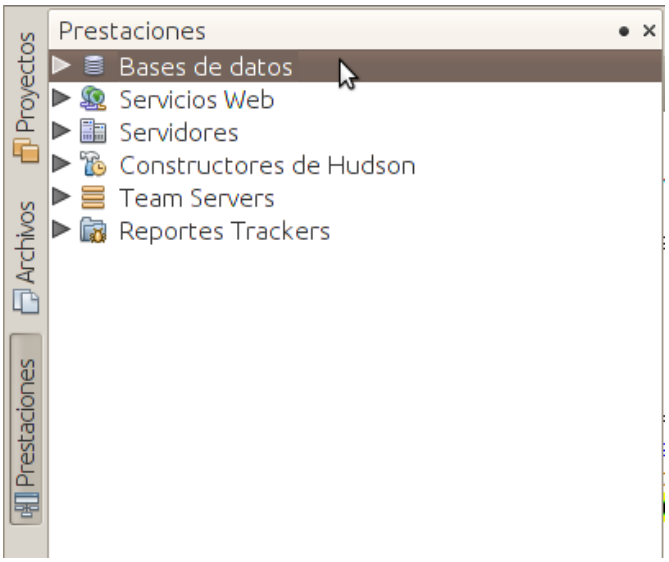


Fig. 71 Conexión entre NetBeans y PostgreSQL usando JPA

Seleccionamos Bases de datos, aquí hacemos clic derecho y seleccionamos nueva conexión de base de datos, en esta nueva ventana llenamos la información de la siguiente forma:

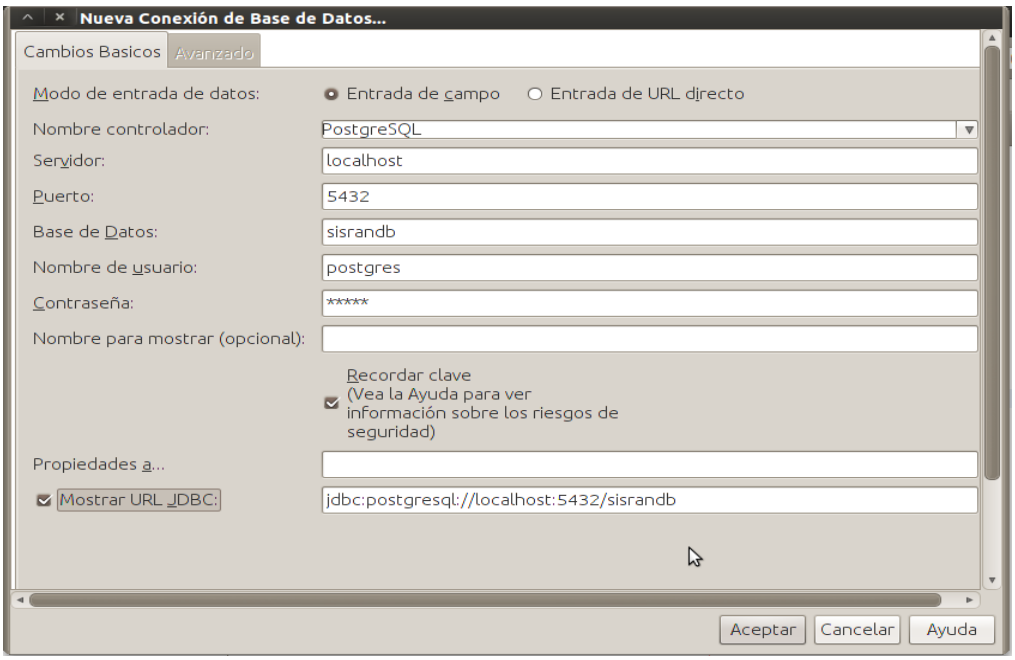


Fig. 72 Propiedades de la conexión

Hacemos clic en aceptar y en la nueva ventana que nos aparecerá seleccionamos el esquema public que es con el que vamos a trabajar y en donde se encuentran nuestras tablas creada en PostgreSQL

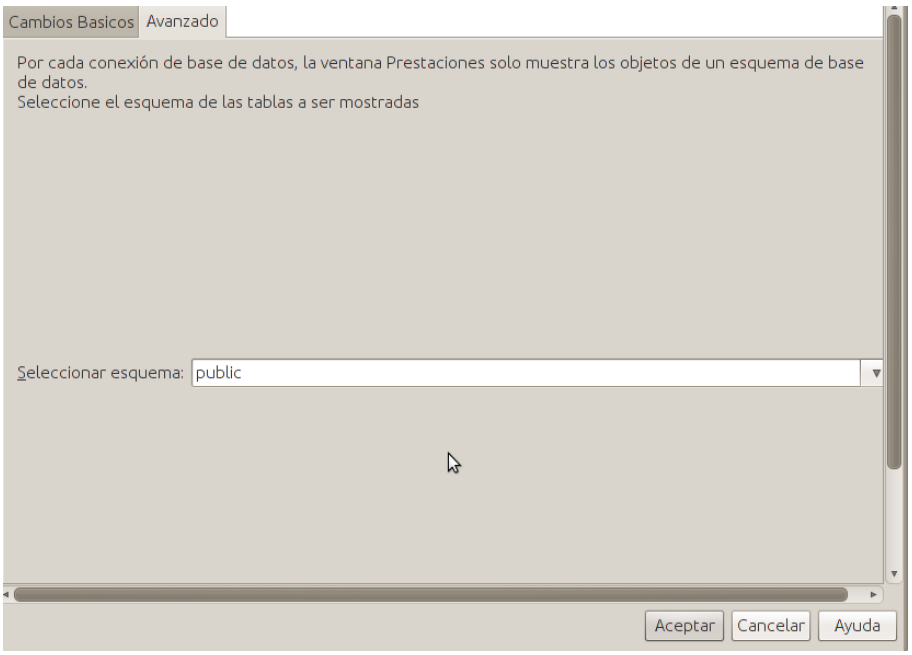


Fig. 73 Seleccionar esquema

Listo la conexión ha sido establecida y ahora podemos ver nuestras tablas creadas dentro de PostgreSQL

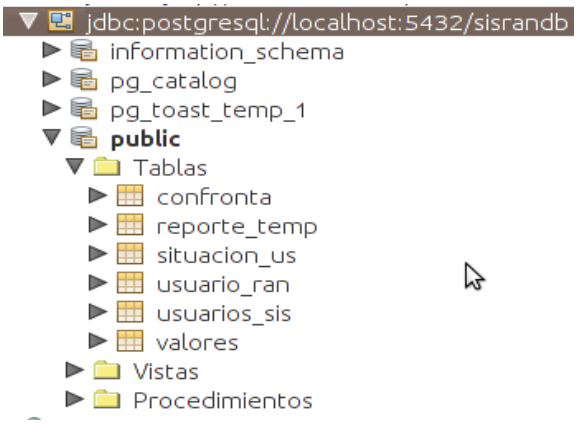


Fig. 74 Tablas creadas en PostgreSQL

Esta conexión nos servirá para crear una nueva unidad de Persistencia la cual nos conectara con la base de datos cada vez que lo requiramos, esta unidad de persistencia se crea automáticamente al crear Entidades a partir la bases de datos existentes, y para crear estas entidades procedemos de la siguiente forma:

Nos vamos a nuestro proyecto y creamos un nuevo paquete el cual lo llamaremos Entidades.

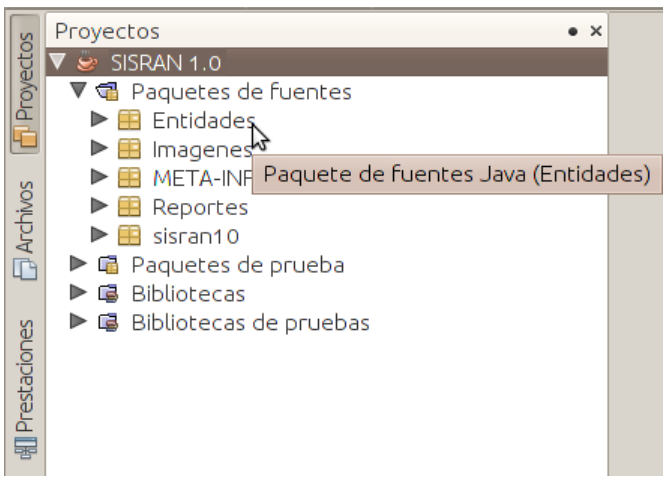


Fig. 75 Creación de paquetes

Ahora hacemos clic derecho sobre el paquete Entidades y seleccionamos Nuevo/ Otro, en donde seleccionamos Persistencia / Clase entidad a partir

de base de datos y hacemos clic en siguiente.

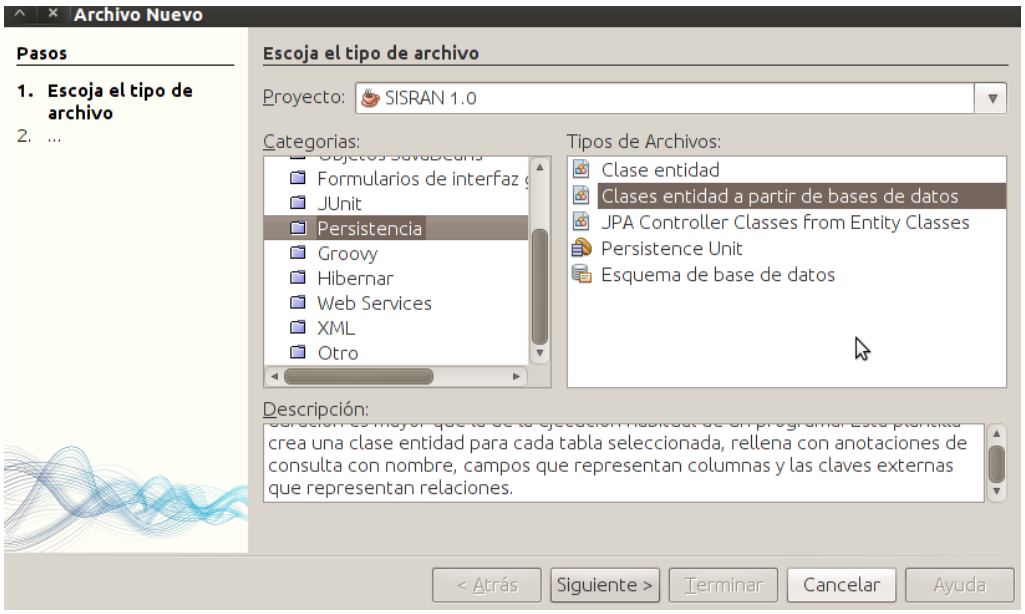


Fig. 76 Escoger tipo de archivo

En esta venta seleccionamos la conexión anteriormente creada y las tablas que vamos a utilizar, estas tablas serán las nuevas Clases Entidades.

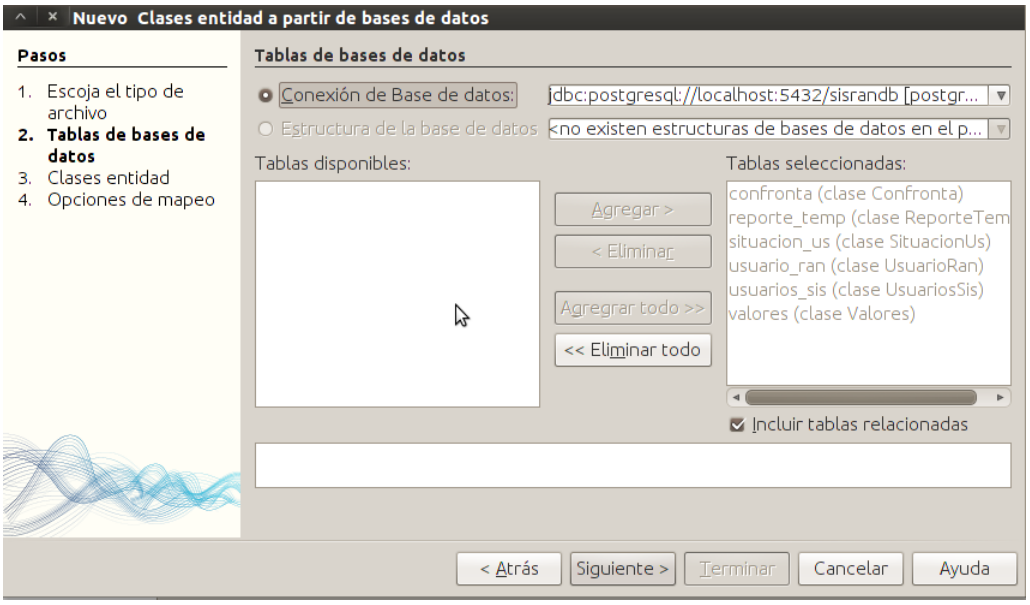


Fig. 77 Clase entidades

Hacemos clic en siguiente y nos aparecerá una ventana en donde nos da la opción de modificar los nombres de nuestras entidades, en esta ventana simplemente hacemos clic en siguiente.

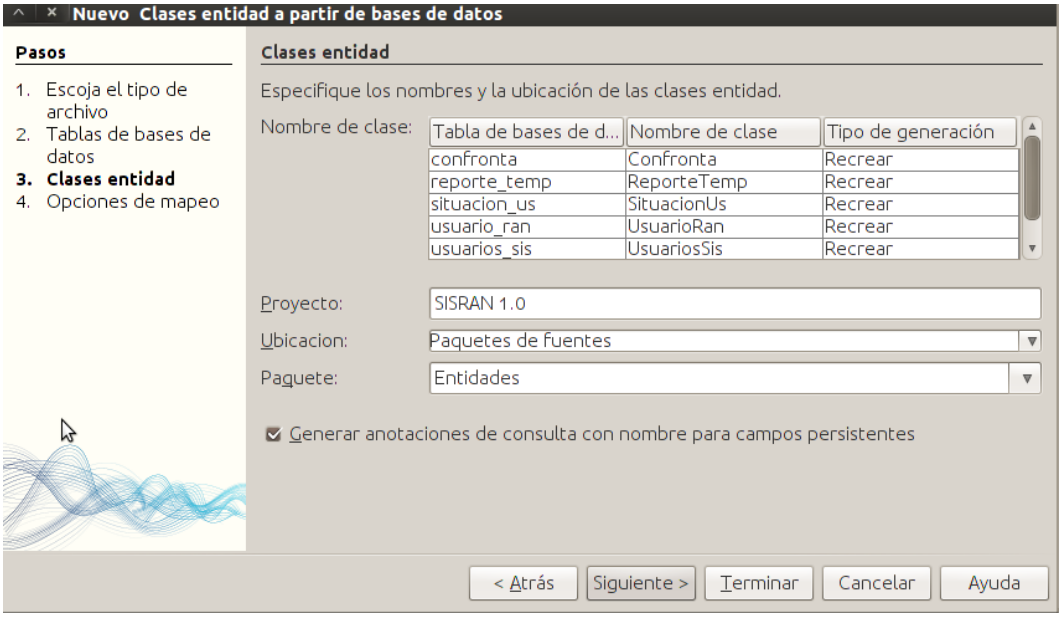


Fig. 78 Propiedades de clase entidades

Ahora nos aparecerá una nueva venta en la cual nos da algunas opciones de mapeo pero dejamos por defecto todos los campos y hacemos clic en Terminar

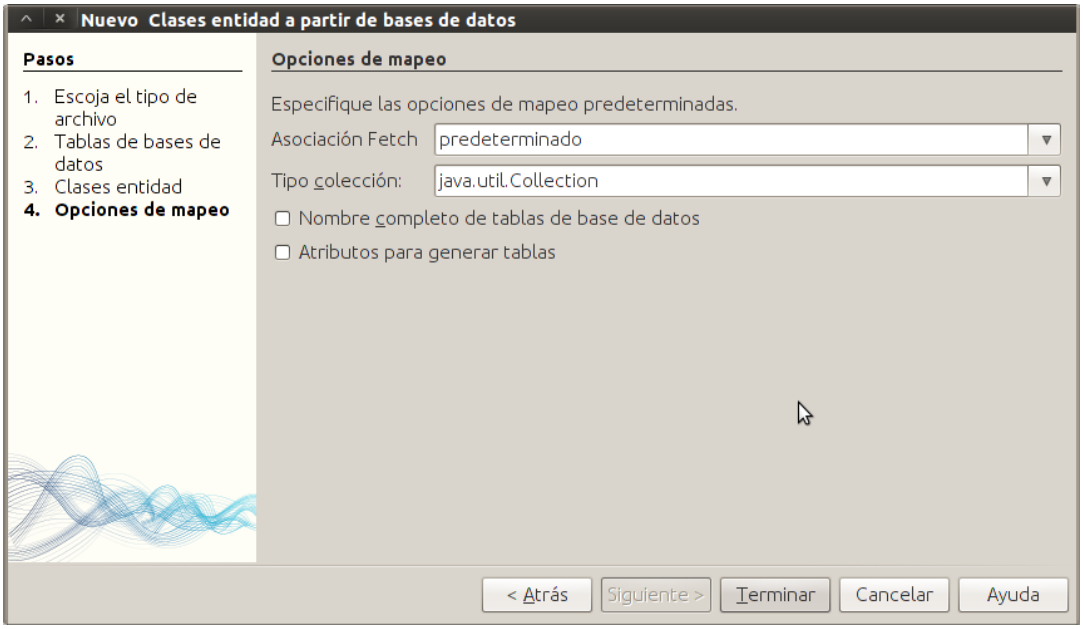


Fig. 79 Opciones de mapeo

Listo ya esta creada la Unidad de Persistencia dentro del paquete META-INF, y nuestras entidades dentro del paquete Entidades.

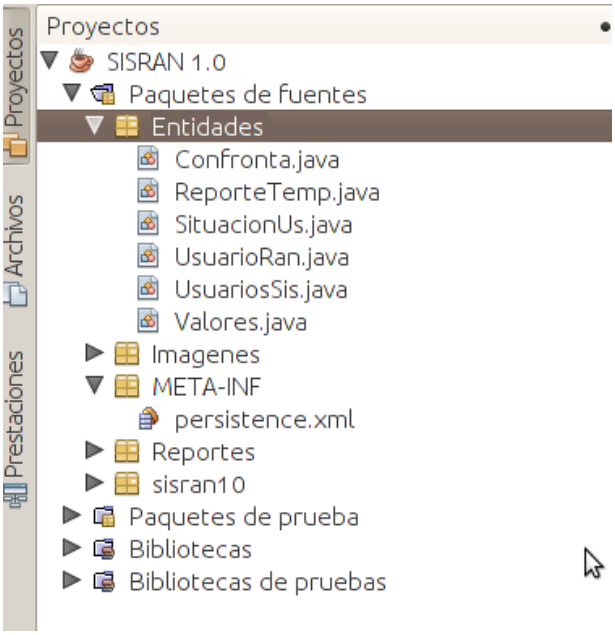


Fig. 80 Unidad de persistencias

4.8 Gestión de Bases de Datos Usando JPA

La gestión de bases de datos se basan principalmente en cuatro cosas: altas, bajas, cambios y consultas, este último es un tema aun mucho más extenso ya que existen consultas anidadas y algunas funciones como por ejemplo SUM() que permite sumar el contenido de algunos registros, pero esto se facilita con JPA ya que nos permite manejar las Tablas que se encuentran en la base de datos como Objetos. Para ello debemos tomar en cuenta la declaración de dichos objetos y las posibles operaciones que no permite realizar una librería llamada EclipseLink (JPA 2.0) que es la versión libre de TopLink Essentials de Oracle, aunque no es tan buena como Hibernate que es la mejor biblioteca de persistencia que existe en el mercado, EclipseLink es de código abierto, y esperamos que en futuras versiones ya pueda igualar y tal vez superar a Hibernate.

Ahora vamos a explicar las operaciones antes mencionadas de forma detallada para un mejor entendimiento.

4.8.1 Crear un Administrador de Entidades o entityManager

Para ello en la paleta seleccionamos Persistencia/ Administrador de Entidades y lo arrastramos a nuestro formulario

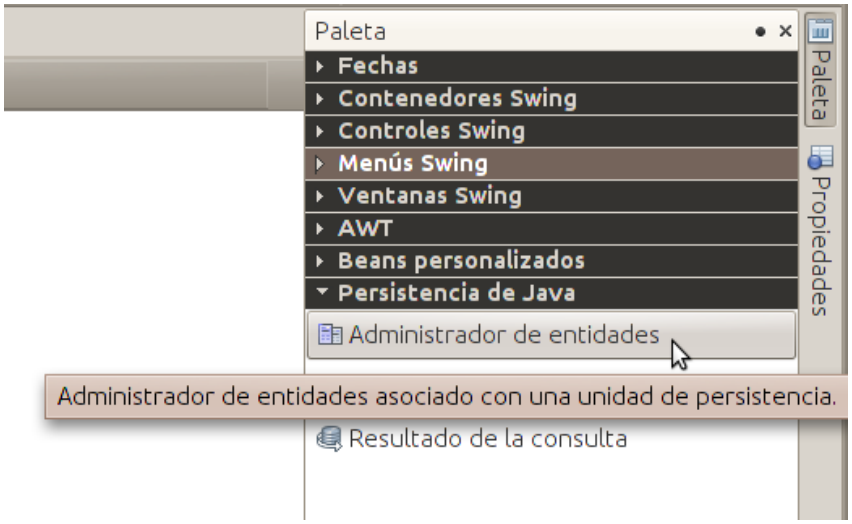


Fig. 81 Administrador de entidades

Ahora lo renombramos como em, para ello vamos al Inspector y seleccionamos Otros Componentes

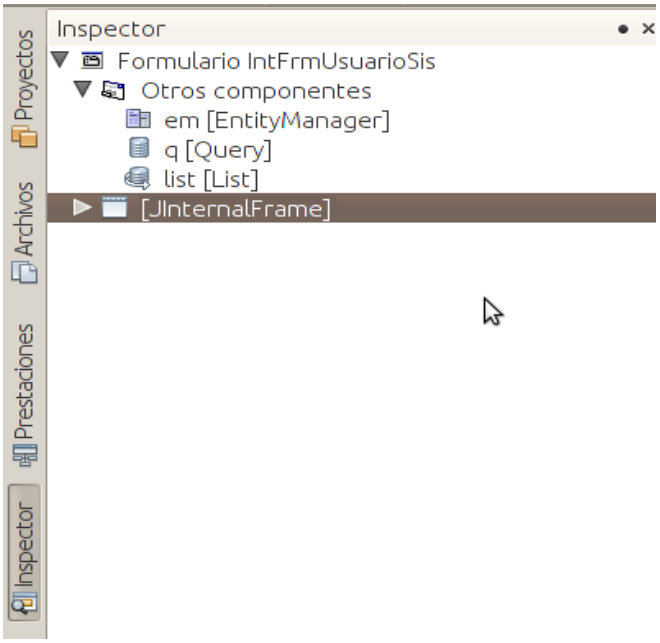


Fig. 82 Inspector para renombrar

Luego seleccionamos entityManager y en propiedades vamos a código y cambiamos el nombre.

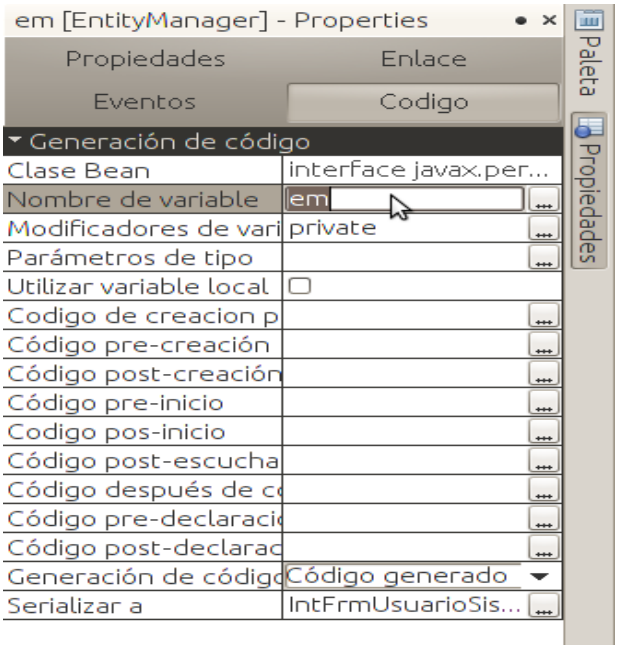


Fig. 83 Código de EntyManager

Ahora con este Administrador de Entidades ya podemos manejar las tablas y sus datos sin ningún problema.

4.8.2 Crear un nuevo registro

Para crear un nuevo registro procedemos de la siguiente forma: Primero tenernos que declarar un objeto del tipo de la Entidad que queramos agregar un nuevo registro.

```
em.getTransaction().begin();

UsuarioRan u = new UsuarioRan(“123”,”Pilco Joffre”);

em.persistence(u);

em.getTransaction().commit();
```

4.8.3 Modificar un registro

```
em.getTransaction().begin();

UsuarioRan u = em.find(UsuarioRan.class, “123”);

u.setNombre(“Pilco Victor”);

em.getTransaction().commit();
```

4.8.4 Eliminar un registro

```
em.getTransaction().begin();

UsuarioRan u = em.find(UsuarioRan.class, “123”);

em.remove(u);

em.getTransaction().commit();
```

Como podemos ver en toda transacción que realicemos siempre va entre em.getTransaction().begin(); y em.getTransaction().commit(); esto es muy importante ya que si no colocamos alguna de estas dos instrucciones los cambios no van a surtir efecto y nos va a dar algún error en tiempo de ejecución.

4.8.5 Realizar consultas

Realizar consulta con JPA es muy sencillo y el único requisito es tener un conocimiento básico de instrucciones SQL

Para ello debemos crear un objeto de tipo Query y asignarle un Query creado mediante el Administrador de Entidades Ej.:

```
Query q = em.createQuery("SELECT c FROM Confronta c
WHERE c.fecha = :fecha AND c.situacionUs = :situacionUs");
```

En esta instrucción SQL nos fijamos que tiene dos parámetro fecha y situacionUs, y estos parámetros normalmente usando otra técnica de programación se los debería reemplazar con variables las cuales deben ser concatenadas pero antes de ello fecha por ejemplo debe ser transformada a un formato acorde con el que tiene PostgreSQL para fechas y luego transformarlas a tipo String para poder concatenarlas dentro de la instrucción SQL,

Pero gracias a la magia del JPA y EclipseLink (JPA 2.0) esto te resume en dos simples instrucciones adicionales Ej.:

```
Query q = em.createQuery("SELECT c FROM Confronta c
WHERE c.fecha = :fecha AND c.situacionUs = :situacionUs");

q.setParameter("fecha", fecha);

q.setParameter("situacionUs", sitUs);
```

El método setParameter nos permite establecer los parámetros necesarios para realizar nuestras consulta por ejemplo fecha puede estar en el formato de java.util.Date o en el formato java.sql.Date, que igual reconoce estos tipos de formatos, o simplemente para establecer la fecha actual seria

```
q.setParameter("fecha", new Date());
```

Y así nos evitamos realizar tantas instrucciones y en especial para el manejo de objetos de tipo fecha son muy complicados.

Ahora vamos a mostrar como el resultado de estas consultas pueden ser usados por otros objetos o también como arrays de objetos para facilitar su navegación y optimizar recursos del sistema.

Ya en este punto tenemos un conocimiento básico de manejo de datos con JPA y con librerías EclipseLink (JPA 2.0), ahora vamos a relacionar estos objetos con otros que nos van a permitir un mejor manejo de nuestros datos.

Por ejemplo este resultado lo podemos asignar como parte de una Lista Ej.:

```
List<Confronta> cList = q.getResultList();
```

Ahora dentro de list esta la colección de Objetos de tipo Confronta que es un entidad previamente creada.

Con el objeto list podremos recorrer todos los elementos que pertenecen a la tabla confronta de nuestra base de datos por ejemplo si queremos mostrar el ID de cada objeto basta con realizar el siguiente proceso:

```
for(int i=0; i<cList .size(); i++)

    System.out.println(cList .get(i).getIdConfronta());
```

Tan sencillo como eso de igual forma podríamos usar esta técnica para asignar nuevos valores a esta clase por ejemplo supongamos que la clase confronta tiene un campo llamado almuerzos y quiero que a este campo se le asigne su contenido multiplicado por 2 y sumado 3 seria sencillamente el siguiente proceso:

```
for(int i=0; i<cList .size(); i++)

    cList.setAlmuerzos( cList.getAlmuerzos *2 +3);
```

Tan sencillo como eso y eso que para el planteamiento del problema tenemos más líneas que la misma solución.

4.9 Consultas usando un jList

Ahora vamos a explicar cómo se puede usar esta poderosa técnica de programación junto a componentes Swing gráficos, como por ejemplo un JTable o un Jlist, estos dos componentes son similares en propiedades pero el Jlist tiene la ventaja de ser mas estético a la hora de trabajar con bases de datos, así que nos vamos a concentrar en un Jlist.

Para cargar datos en un Jlist vamos a utilizar la conexión creada anteriormente, ya no será necesario crear los objetos de tipo List y de tipo confronta por ejemplo, ya que al realizar la conexión también serán creados estos objetos automáticamente.

Para ello nos vamos a la Paleta/ Controles Swing/ Lista

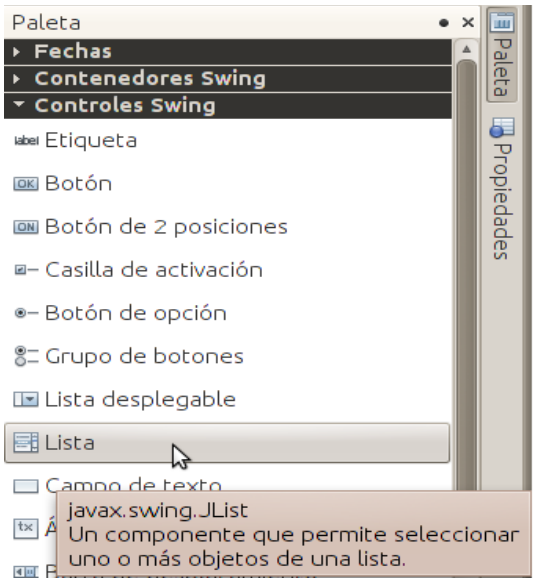


Fig. 84 Paleta para realizar consultas

Arrastramos este componente a nuestro formulario y nos va a quedar más o menos así

Fig. 85 Formularios de listas

4.9.1 Enlace de datos

Ahora sobre la lista hacemos clic derecho y seleccionamos Enlazar/ Elements y nos aparecerá el siguiente dialogo

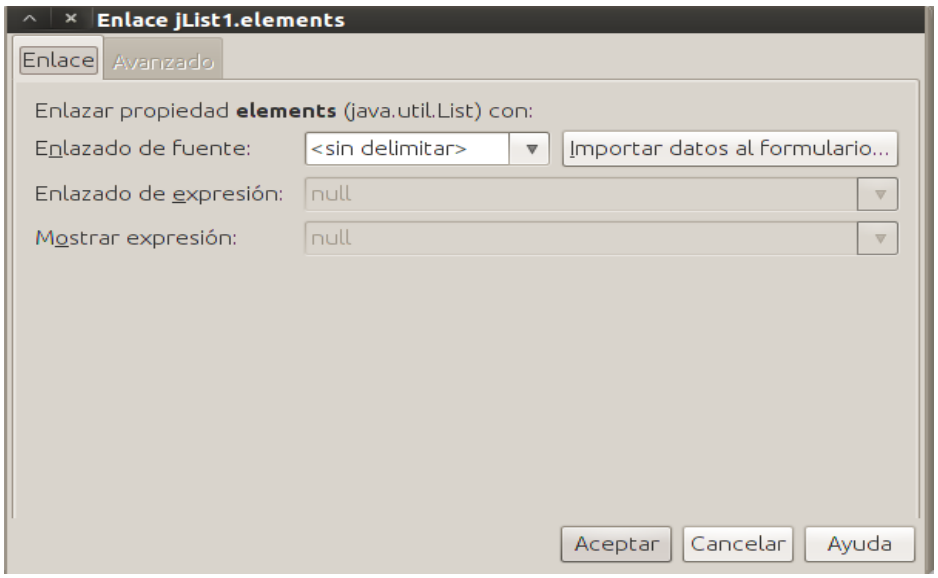


Fig. 86 Enlace de Jlist

Hacemos clic en importar datos al formulario y nos aparecerá un dialogo en el que podemos seleccionar la conexión que vamos a utilizar

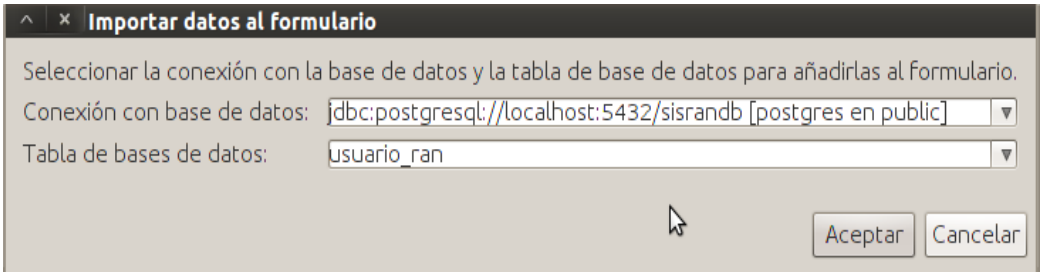


Fig. 87 Importar datos al formulario

Seleccionamos la conexión jdbc:postgresql://localhost:5432/sisrandb [postgresql en public] y de esta conexión seleccionamos la Tabla usuario_ran y hacemos clic en aceptar y nos aparecerá la siguiente ventana.

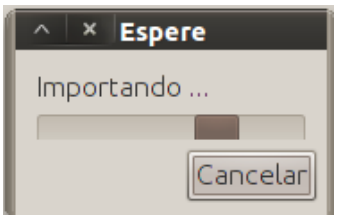


Fig. 88 Procesando importación

4.9.2 Especificación de campos a mostrar

Ahora vamos a especificar los campos que se mostraran en el Jlist para ello colocamos en Mostrar Expresión los siguiente: ${apellidos} ${nombres}, para especificar que en el Jlist mostraremos los campos apellidos y nombres de los usuarios de rancho y hacemos clic en guardar

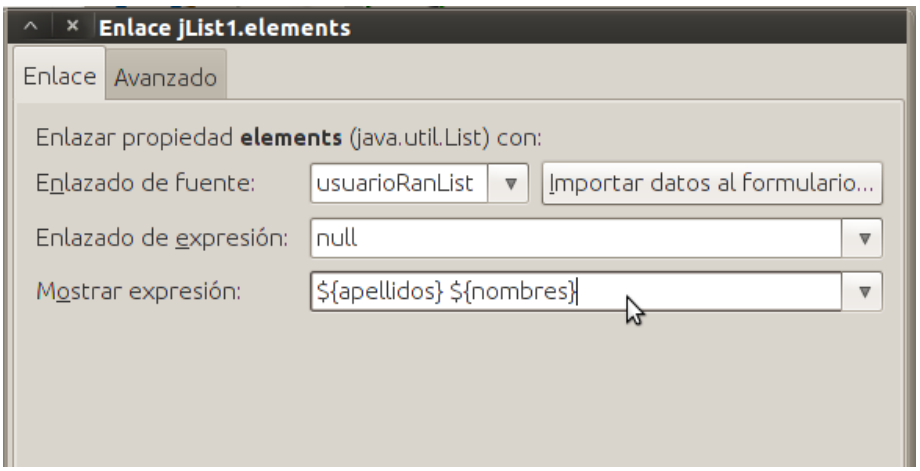


Fig. 89 Especificación de campos a mostrar

Los datos están listos para ser mostrados y también para ser manipulados fácilmente con nuestro programa. Si ejecutamos nuestro archivo se mostraran los apellidos y los nombres de los usuarios de rancho.

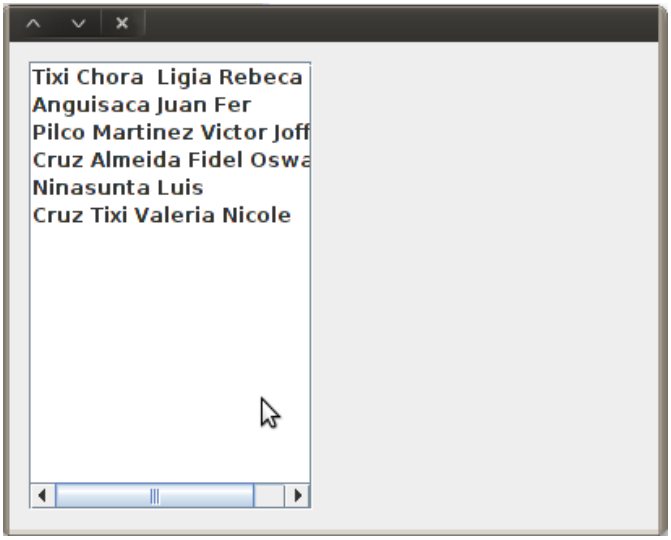


Fig. 90 Datos listos para manipular

4.9.3 Personalización de variables tipo campo

Al hacer clic derecho sobre nuestro Jlist, seleccionando la opción personalizar código podremos manipular las variables que se han generado automáticamente al importar la tabla ususarios_ran a nuestro formulario. Los siguientes campos son los más importantes para la manipulación de datos y a estos los vamos a renombrar para facilitar su manejo.

Fig. 91 Personalizar variables

A las siguientes variables las vamos a renombrar y explicar su función:

EntityManager (em).- está encargado de administrar las clases entidad.

UsuarioRanQuery (q).- en esta variable se asigna las sentencias SQL de consulta.

usuarioRanList(usRL).- en esta variable de tipo lista se asignan todos los elementos de la tabla y también se pueden modificar mediante esta lista.

4.9.4 Lista observable

En este punto es necesario realizar un pequeña modificación a nuestra lista usRL, haciéndola observable, es decir que cada cambio que realicemos se lo vera inmediatamente en nuestro Jlist sin necesidad de realizar algún tipo de recarga de datos. Para ello realizamos los dos pasos siguientes.

- nos dirigimos hacia la ventana Inspector y hacemos clic sobre el campo usRL

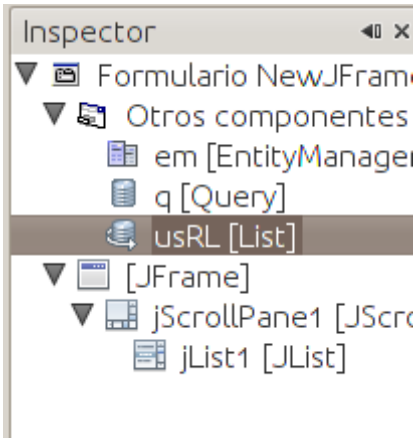


Fig. 92 Lista observable

- en la ventana propiedades hacemos activamos la casilla observable

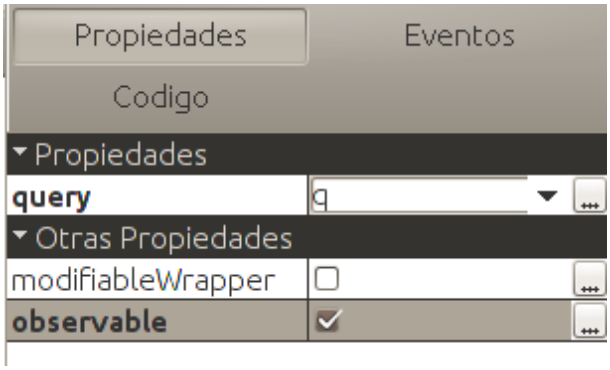


Fig. 93 Propiedades de lista

Ahora que nuestra variable usRL y nuestra lista están enlazadas y son observables todos sus cambios es mucho mas fácil el manejo de nuestros datos, esta administración de datos (nuevo, editar, eliminar, guardar y consultas), son de la misma forma que indicamos en nuestro tema anterior manejo de datos con JPA.

CAPÍTULO 5: CREACIÓN DE CÓDIGOS DE BARRAS Y USO DEL LECTOR

5.1 Introducción a Código de Barras

El **código de barras** es un código basado en la representación mediante un conjunto de líneas paralelas verticales de distinto grosor y espaciado que en su conjunto contienen una determinada información. De este modo, el código de barras permite reconocer rápidamente un artículo en un punto de la cadena logística y así poder realizar inventario o consultar sus características asociadas. Actualmente, el código de barras está implantado masivamente de forma global.

Es un sistema que permite la identificación de las unidades comerciales y logísticas de forma única, global y no ambigua. Este conjunto de barras y espacios codifican pequeñas cadenas de caracteres en los símbolos impresos.

La correspondencia o mapeo entre la información y el código que la representa se denomina *simbología*. Estas simbologías pueden ser clasificadas en dos grupos atendiendo a dos criterios diferentes:

- *Continua o discreta*: los caracteres en las simbologías continuas comienzan con un espacio y en el siguiente comienzan con una barra (o viceversa). Sin embargo, en los caracteres en las simbologías discretas, éstos comienzan y terminan con barras y el espacio entre caracteres es ignorado, ya que no es lo suficientemente ancho.

- *Bidimensional o multidimensional*: las barras en las simbologías bidimensionales pueden ser anchas o estrechas. Sin embargo, las barras en las simbologías multidimensionales son múltiplos de una anchura determinada (X). De esta forma, se emplean barras con anchura X, 2X, 3X, y 4X.

5.1.1 Nomenclatura básica

- **Módulo:** Es la unidad mínima o básica de un código. Las barras y espacios están formados por un conjunto de módulos.
- **Barra:** El elemento oscuro dentro del código. Se hace corresponder con el valor binario 1.
- **Espacio:** El elemento claro dentro del código. Se hace corresponder con el valor binario 0.
- **Carácter:** Formado por barras y espacios. Normalmente se corresponde con un carácter alfanumérico.

Funciones técnicas de los carácteres contenidos en un código de barras:
1: Quiet Zone.
2: Carácter inicio (derecha), Carácter terminación (izquierda).
3: Carácter de datos.
4: Checksum.

Ejemplo de datos contenidos en un código de barras GTIN 13:

- P: prefijo GS1 (por ejemplo, el número 84 correspondería a España)
- Código de empresa: código asignado a las empresas registradas (5-8 dígitos)
- Código de producto: dígitos en blanco para el propietario de la marca
- C: dígito de control.

5.1.2 Principales técnicas de creación de códigos de barras

La impresión de códigos de barras en la tienda web de VWR es realmente sencilla. Ponemos a su disposición diversas opciones que le

permitirán crear los códigos de barras que necesita y comenzar a disfrutar de las ventajas de esta solución. Para realizar todas las opciones que se describen a continuación deberá iniciar sesión. Si aún no dispone de un perfil, haga clic aquí para obtener uno.

Es posible imprimir códigos de barras desde prácticamente cualquier ubicación del centro en la que exista una lista de números de artículos. El lugar más cómodo es la zona de la lista de compra. Puede elegir:

5.1.3 Imprimir en formato de papel de etiqueta avery

Puede especificar un formato e imprimir los códigos de barras para pegar las etiquetas en las estanterías en las que están colocados sus productos.

Para alternar entre los formatos que necesite, utilice el cuadro desplegable de la lista de compra. Al hacer clic en el cuadro de creación de códigos de barras, se creará un código de barras que podrá imprimir si lo desea. Encontrará este papel en el almacén de suministros de su oficina local.

5.1.4 Imprimir un mini catálogo personalizado

Haga clic en la opción de creación de páginas del catálogo en PDF, en la página de la lista de compra. Así, generará un catálogo completo con imágenes y mucho más, incluidos códigos de barras. De esta manera podrá crear el catálogo que desee y solicitar artículos con esta fantástica nueva función.

Aquí encontrará algunos ejemplos que ilustran cómo quedan los códigos de barras:

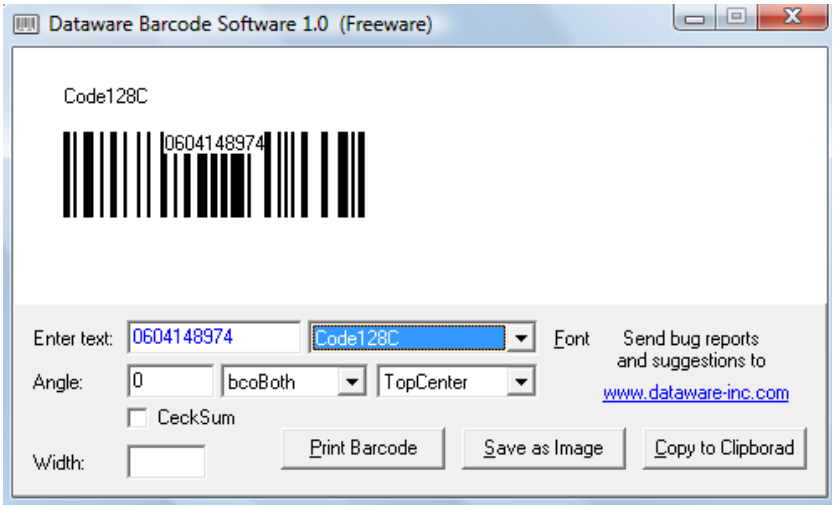


Fig. 94 Software para crear códigos de barra

Fig. 95 Ejemplo de código 128C C.I. Chimborazo

Fig. 96 Ejemplo de código 128C con C.I. Pichincha

5.2 Función del lector de códigos de barras

Un escáner para lectura de códigos de barras básicas consiste en el escáner propiamente dicho, un decodificador y un cable que actúa como interfaz entre el decodificador y el terminal o la computadora.

La función del escáner es leer el símbolo del código de barras y proporcionar una salida eléctrica a la computadora, correspondiente a las barras y espacios del código de barras. Sin embargo, es el decodificador el que reconoce la simbología del código de barras,

analiza el contenido del código de barras leído y transmite dichos datos a la computadora en un formato de datos tradicional.

Un escáner puede tener el decodificador incorporado en el mango o puede tratarse de un escáner sin decodificador que requiere una caja separada, llamada interfaz o emulador. Los escáneres sin decodificador también se utilizan cuando se establecen conexiones con escáneres portátiles tipo “batch” (por lotes) y el proceso de decodificación se realiza mediante el Terminal propiamente dicho.

5.3.1 Ventajas del uso de Códigos de Barras

- Rapidez en la captura y lectura de datos
- Mayor precisión en la información.
- Rastreo preciso en actividades
- Mejor control de entradas y salidas
- Reducción de errores
- Los equipos de lectura e impresión de códigos de barras son fáciles de conectar e instalar.

5.3.2 Descripción del Lector MS9520 Voyager®

Con un diseño y un sistema avanzado de exploración, el Voyager es, hoy en día, el lector de códigos de barras de una línea más avanzado del mercado.

El Voyager 9520 se ha concebido como un lector muy agresivo con una gran profundidad de campo y una velocidad de lectura que es casi el doble que el siempre recordado MS 951. Por supuesto este producto mantiene el exclusivo sensor infrarrojo patentado por Metrologic y un sistema de control que permite una activación totalmente automática y su uso como lector de "manos libres".

El Voyager puede operar en el modo "manos libres" cuando se sitúa sobre su soporte. Basta con la presentación del código para que el lector realice automáticamente la lectura. Además es programable para lecturas a corto o largo alcance tanto en el modo automático como manual, con lo que incrementa su eficiencia y productividad al mismo tiempo que su diseño ergonómico lo hace muy cómodo para su manejo.

Fig. 97 Lector Láser de Código de Barras Metrologic MS9520 Voyager

5.2.1 Características Técnicas

- Con conector para teclado PS/2, permite modo de operación manual y manos libres, incluye base.
- Distancia máxima de lectura 8"
- Características: Activación en corto y largo alcance, Fácil programación por menú de códigos de barras Metroselect ó software en Windows Metroset2
- Fuente luminosa: Diodo láser visible de 650 nm 10 nm
- Velocidad de lectura: 72.2 líneas por segundo
- Modo de exploración: Una línea
- Interfaz: Emulación de teclado
- Indicadores led: Rojo = lectura correcta, Verde = láser activo y listo para leer, Amarillo = modo de disparo automático sobre la base
- Cable: Estándar 2.7 m en espiral.

CAPÍTULO 6: DESARROLLO DEL SISTEMA DE CONTROL DE RANCHO SISRAN 1.0

6.1 Análisis y Especificación de requisitos de software

6.1.1 Introducción

La Especificación de requerimientos de Software (ERS), es un documento base para el desarrollo del Sistema SISRAN (Sistema de Control de Rancho mediante lector de código de barras para la ESPE Extensión Latacunga, utilizando Software Libre). Este documento se desarrolla siguiendo el estándar IEEE 830-1998 “Estándar IEEE Práctica Recomendada para la Especificación de Requisito Software”.

6.1.2 Análisis del problema de cobro de rancho y recolección de requisitos

En la ESPE extensión Latacunga y a nivel Ejército Ecuatoriano se tiene el servicio de rancho, el problema principal radica en la gran cantidad de personal que consume el mismo sin un control eficiente, ocasionando inconvenientes a la hora de realizar los reportes de consumo individual, ya que es indispensable cuadrar el valor total de descuentos con la suma de todos los confrontas entregadas al encargado del rancho durante el transcurso del mes.

En la realización de estos reportes de descuento es muy importante tomar en cuenta que existe personas que no han consumido el rancho todos los días, y que en algunos casos han estado en una situación especial como son: Licencias Anuales Planificadas (L.A.P), y a estas personas no se les debería cobrar en absoluto el mes de consumo de rancho, también puede darse otra situación que es muy común como es el permiso de 15 días que tienen derecho por alumbramiento, o a su vez por comisiones, cursos y pases especiales que deben ser registrados oportunamente para evitar cobros erróneos y valores excesivos en los descuentos individuales.

Por otra parte en la directiva de cobro de rancho vigente esta la disposición que se tiene que cobrar todos los almuerzos del mes, y desayunos de martes y jueves además de las guardias que se cobra día completo. Esto está justificado ya que existen personas que por alguna razón no prevista no consumen el rancho del día y la confronta una vez pasada al encargado del rancho esa comida simplemente se pierde, por ese motivo es indispensable cobrar esos días a fin de evitar pérdidas para los encargados del rancho.

Todo debe estar estrictamente planificado y en concordancia con las confrontas que se emiten a diario, y todo esto está estipulado detalladamente en la directiva de rancho de la unidad en vigencia.

Por todo esto es necesario crear un sistema de control de rancho que sea eficiente y flexible para realizar cambios importantes pero justificados para evitar problemas al momento de realizar los reportes de descuentos de rancho y confrontas diarias para el encargado del rancho.

Al final del mes el sistema deberá emitir reportes de descuento de rancho que estén en concordancia con las confrontas de rancho emitidas a diario.

Si por alguno motivo en especial se necesitan realizar cambios en el sistema deberá pedir una justificación que al final se incluirá en el reporte, además del registro de los usuarios que realizan dichos reajustes, como por ejemplo en el caso de los ranchos especiales que por lo general son más costosos que los ranchos comunes.

6.1.3 Ámbito del Sistema

El sistema ha realizar se denominará SISRAN 1.0 el cual deberá solventar a cabalidad todos los puntos expuestos anteriormente, para lo cual será necesario crear políticas de cobro que permitan al sistema automatizar eficientemente todos los procesos necesarios para realizar los reportes mensuales de cobro de rancho y emisión diario de confrontas.

6.2 Definiciones, acrónimos y abreviaturas

6.2.1 Definiciones

Administrador	Usuario con privilegios para establecer altas, bajas y cambios o configuraciones necesarias en la gestión del Sistema SISRAN 1.0
Responsable del control	Persona que inicializa el sistema y realiza el control diario de las diferentes comidas: desayuno, almuerzo y merienda.
Usuario consumidor	Persona que posee una tarjeta para el servicio de Rancho.
Rancho	Definición que se le da a la comida que se consume un las unidades militares.
Confronta	Cálculo del personal que va a consumir el rancho diariamente.

6.2.2 Acrónimos

ERS	Especificación de Requisitos de Software Estándar IEEE 830-1998

6.2.3 Abreviaturas

SISRAN	Sistema de Control de Rancho mediante lector de código de barras para la ESPE Extensión Latacunga, utilizando Software Libre.

6.3 Referencias

Estándar IEEE 830-1998 (IEEE Recomended Practice for Software Requirements Specification).

6.4 Visión general del documento

El documento consta de tres secciones; La primera contiene una visión general del sistema a desarrollar. La segunda sección describe el funcionamiento del sistema, gestión de datos asociados y factores que inciden el sistema. Y en la última sección se definen los requisitos que debe satisfacer el sistema.

6.5 Descripción general

En esta sección se detalla de forma general el sistema, con el objetivo de conocer las principales funciones que realizara, tanto con los datos, restricciones, y cualquier factor que afecte al desarrollo del mismo.

6.5.1 Perspectiva del producto

El sistema debe tener un enlace directo con la base de datos del personal de Oficiales Voluntarios, Alumnos militares de planta y personal administrativo que se encuentra con el pase en la ESPE extensión Latacunga.

El funcionamiento esencial del sistema consistirá en que todos los valores y las confrontas deberán ser pre-llenadas al iniciar el mes para posteriormente realizar las modificaciones de acuerdo a los casos que se vayan suscitando en el transcurso de ese mes.

Por ejemplo:

El CBOP. PILCO usuario de rancho es un alumno egresado y como tal el sistema comprende que consume solo días de guardia sin embargo podría ser castigado y estos días de consumo obligatorio deberán ser registrados en el sistema, evitando de ese modo errores de cobro de rancho

6.5.2 Funciones del Sistema

En forma general el sistema deberá dar y soportar las siguientes gestiones:

- Gestión de usuarios de sistema
- Gestión de usuarios consumidores de rancho
- Gestión de situaciones de consumo de rancho
- Gestión de Valores
- Gestión de Confrontas
- Elaboración de reportes
- Verificación de consumo

- **Gestión de usuarios del sistema**

Estos usuarios del sistema se los puede clasificar en Administrador, y Responsable de Control de consumo.

- El Administrador es aquel que tiene acceso a todas las funciones del sistema, altas, bajas y cambios.
- El Responsable del control de consumo de rancho, en este caso el usuario puede ser el encargado de la preparación del rancho o en su reemplazo el cocinero de turno.

- **Gestión de usuarios consumidores de rancho**

Usuarios consumidores son todos los que hacen uso del servicio de rancho en la ESPE extensión Latacunga que tienen asignado una tarjeta.

- **Gestión de Situaciones de consumo de rancho**

Permitirá realizar las funciones de Cambio o Registro de Situación de uno o varios usuarios.

Definiendo como situaciones de consumo a los siguientes casos:

1. Clase de Semana
2. Guardia
3. Destacamento

4. Permiso
5. Alumno militar egresado
6. Alumno militar residente
7. Alumno militar no residente
8. Fines de semana
9. Feriados
10. Castigados
11. Alumnos civiles
12. Docentes

Nota: Se pueden añadir más situaciones de consumo especificando en detalle el consumo tanto en desayuno, almuerzo o merienda

Para realizar un cambio, deberá ingresar la cédula, los apellidos y nombres del usuario.

- **Gestión de Valores**

Permitirá realizar las funciones de ingreso, modificación y eliminación de valores de acuerdo a las directivas de rancho vigentes y en otros casos como: ranchos especiales (navideños, cumpleaños, etc.), estos valores tienen que ser detallados para desayunos, almuerzos y meriendas, además que deben tener un registro de la razón por la que se ha creado ese nuevo valor de cobro.

- **Gestión de Confrontas**

Permitirá realizar las funciones de pre llenado y modificación de las confrontas diarias, controlando guardias y fines de semana.

En las guardias se considerara el número de grupos de guardia y el grupo con el que inicia el mes.

Los cambios en las situaciones de los usuarios deben ser realizados por el administrador del sistema, en este caso la interfaz del sistema deberá ser amigable teniendo en cuenta los siguientes campos:

1. Lista de usuarios, de donde vamos a seleccionar el usuario cuya situación va a ser modificada
2. Lista de situaciones, en la cual vamos a seleccionar la nueva situación que vamos a asignarle al usuario.
3. Lista de valores, en la cual se selecciona los valores con los que vamos a realizar los cálculos.
4. Un selector de fecha inicial y uno de fecha final, esto nos permitirá realizar dichas modificaciones dentro de un rango específico de tiempo.
5. Búsqueda, permite filtrar usuarios, situaciones y valores.

- **Elaboración de Reportes**

Producirá los respectivos Reportes de consumo de rancho en forma mensual o diaria. Los reportes en forma mensual servirán para realizar los cobros individuales de rancho al personal de consumidores del mismo, los reportes diarios de rancho serán usados como confronta diaria del rancho.

- **Verificación de consumo**

El procedimiento de verificación de consumo es la comprobación de la tarjeta a través del código de barras y a su vez la emisión del respectivo ticket de consumo, y si un usuario que no está considerado en confronta no recibirá el ticket impreso que le permitirá consumir su rancho.

6.6 Restricciones

Las restricciones que se deberán tener en cuenta al momento de desarrollar el sistema, tanto en hardware como en software son los siguientes:

6.6.1 Software

- No se puede instalar cualquier otro paquete informático en el computador asignado al uso exclusivo para el Sistema de Control de Rancho SISRAN Ver 1.0.

- No se puede eliminar el registro anterior de un usuario, es decir el historial.

Metodología de desarrollo

- Orientada a objetos

Sistema Base

- Sistema Operativo Ubuntu 10.10

Base de datos

- PostgreSQL 8.4

Entorno de desarrollo

- Netbeans IDE 6.9.1

Lenguaje de desarrollo

- Java

6.7 Requisitos Específicos

Esta sección contiene los requisitos a un nivel de detalle suficiente como para permitir a los diseñadores elaborar un sistema que satisfaga estos requisitos, y que permita al equipo de pruebas planificar y realizar las pruebas que demuestren si el sistema satisface los requisitos.

6.7.1 Requisitos Funcionales

i. Gestión de Usuarios del Sistema

El sistema permitirá:

- Req(01) Ingresar un nuevo Usuario.
- Req(02) Se debe considerar dentro de los datos de los usuarios los siguientes campos: ID de usuario, contraseña y tipo de usuario (Administrador o de Control).
- Req(03) Eliminar usuario seleccionado.
- Req(04) Modificar los Datos de un Usuario.

- Req(05) Consulta de usuarios en forma general.

ii. Gestión de Usuarios de Rancho

El sistema permitirá:

- Req(06) Ingresar un nuevo Usuario y considerarlo en confronta a partir de la fecha de ingreso al sistema.
- Req(07) Se debe considerar dentro de los datos de los usuarios los siguientes campos: ID de usuario(cedula), situación de consumo, Apellidos, Nombres, teléfono y grupo de guardia.
- Req(08) Eliminar usuario seleccionado.
- Req(09) Modificar los Datos de un Usuario.
- Req(10) Consulta de usuarios en forma general.
- Req(11) Consulta de usuarios en forma individual o filtro por apellidos.
- Req(12) Modificación de grupos de guardia

iii. Gestión de Situaciones de consumo

El sistema permitirá:

- Req(13) Ingresar un nuevo situación de consumo.
- Req(14) Se debe considerar dentro de los datos de las situaciones los siguientes campos: ID de situación (automático), Nombre de la situación, detalle de desayuno, almuerzo, merienda. Este detalle se deberá considerar en que 0 si no consume o 1 si consume. Ejemplo: Alumno Militar no residente (desayuno=0, almuerzo=1, merienda =0), seleccionados con un ckeckbox.
- Req(15) Eliminar situación seleccionada.
- Req(16) Modificar los Datos de una situación.
- Req(18) Consulta de situaciones en forma general.

iv. Gestión de valores

El sistema permitirá:

- Req(19) Ingresar un nuevo valor de consumo.
- Req(20) Se debe considerar dentro de los valores los siguientes campos: ID de valores (automático), detalle del valor del desayuno, detalle del valor del almuerzo, detalle del valor de la merienda, y detalle o concepto del nuevo valor establecido. Ejemplo: Rancho especial (desayuno=0.80, almuerzo=3.20, merienda =0.95).
- Req(21) Eliminar valores seleccionados.
- Req(22) Modificar valores seleccionados.
- Req(23) Consulta de valores en forma general.

.

v. Gestión de confrontas

El sistema permitirá:

- Req(24) El usuario administrador debe tener la posibilidad de realizar un pre llenado de las confrontas de todo el mes, considerando las diferentes situaciones en cada usuario consumidor del rancho.
- Req(25) Se debe considerar los siguientes campos: ID de confronta (automático), ID de valores, ID de usuario, ID. De situación, fecha y razón de la modificación si es que la hay.
- Req(26) Realizar un pre llenado de los grupos de guardia, considerando el número de grupos y en qué grupo de guardia inicia el mes
- Req(27) Modificar las confrontas de forma individual y colectiva de acuerdo a la situación que se presente en el transcurso del mes.
- Req(28) Realizar un pre-llenado automático de los fines de semana para evitar cobros exagerados del rancho
- Req(29) El pre-llenado se debe realizar de acuerdo al siguiente orden:
    1. se debe llenar la confronta, mensual considerando la situación individual de cada usuario por defecto.

2. Se debe pre-llenar los fines de semana
3. Se debe pre-llenar los grupos de guardia.
4. Cualquier modificación posterior, por ejemplo: semana, cuartelería, etc.

- Req(30) El sistema debe incorporar un método de búsqueda fácil.
- Req(31) El sistema debe permitir la selección de los valores de cobro de rancho.
- Req(32) El sistema debe tener interfaces amigables para el usuario.

vi. Elaboración de Reportes

El sistema permitirá:

- Req(33) El sistema deberá producir informes diarios, los mismos que servirán como confrontas de rancho entregadas al encargado de la realización de los descuentos del rancho.
- Req(34) El sistema debe producir informes mensuales las cuales servirán como reporte para los descuentos mensuales del rancho, los mismos que serán entregados en pagaduría.
- Req(35) Los reportes deberán ser fáciles de entender para todos los usuarios.

vii. Verificación de Consumo

- Req(37) El sistema deberá realizar la verificación del código de barras del ID impreso en la tarjeta de los usuarios que van a consumir el rancho.
- Req(38) Si el sistema verifica que el usuario está considerado en la confronta debe emitir un mensaje que puede consumir el rancho, caso contrario emitirá un mensaje que no puede hacer uso del servicio.

6.7.2 Requisitos De Interfaces Externas

a. Interfaces de Usuario

Las interfaces de usuario se desarrollan en un ambiente de ventanas y el trabajo se lo realizara con el teclado, el mouse y Lector Láser de Código de Barras Metrologic MS9520 Voyager.

b. Interfaces de Hardware

Se trabajara en plataforma cliente/servidor.

c. Interfaces de Software

Se conecta con la Base de datos del personal que hace uso del rancho en la ESPE-L.

6.7.3 Requisitos De Rendimiento

No se ha definido.

6.7.4 Requisitos De Desarrollo

El ciclo de vida para el desarrollo del producto será el secuencial básico.

6.7.5 Requisitos Tecnológicos

a. Hardware: Pc. Pentium IV o superior, Impresora, Lector Láser de Código de Barras Metrologic MS9520 Voyager

b. Sistema Base: Sistema Operativo Ubuntu 10.10

c. Base de datos: PostgreSQL 8.4.

d. Lenguaje de programación: NetBeans IDE 6.9.1 y Java

6.7.6 Atributos del sistema

a. Seguridad

Cuando el administrador del sistema desee abrirlo, debe ingresar su ID o Nombre de usuario y la clave o password, si los datos ingresados no son los correctos se le indicara un mensaje de error.

Los tipos de usuarios que se van a contemplar, y las labores que corresponden a cada uno de ellos son:

- **Administrador del Sistema**

  Es el usuario que tiene acceso a todas las funciones del sistema, por lo tanto tiene permisos para gestionar: usuarios, valores, situaciones y reportes, además de realizar el mantenimiento y actualización continua de los privilegios asignados.

- **Responsable del control de Rancho**

  Es el usuario que puede ser el encargado de la preparación del rancho o en su reemplazo el cocinero de turno.

  Cabe mencionar que el solo es responsable de inicializar el sistema y reportar inmediatamente cualquier error en el mismo, a su vez si un usuario no está registrado en el sistema y desea consumir el rancho, este simplemente lo registrara para posteriormente informar al administrador para ingresarlo al sistema.

- **Usuarios Consumidores del rancho**

  Son todos los Usuarios que hacen uso del servicio de rancho en la ESPE extensión Latacunga únicamente se registrara a través de una tarjeta.

6.7.7 Diagrama de Casos de Uso

Para el sistema de control de rancho SISRAN 1.0 se ha diseñado el siguiente diagrama de casos de uso, en base a la información obtenida en el Documento de especificación de requisitos (ANEXO A).

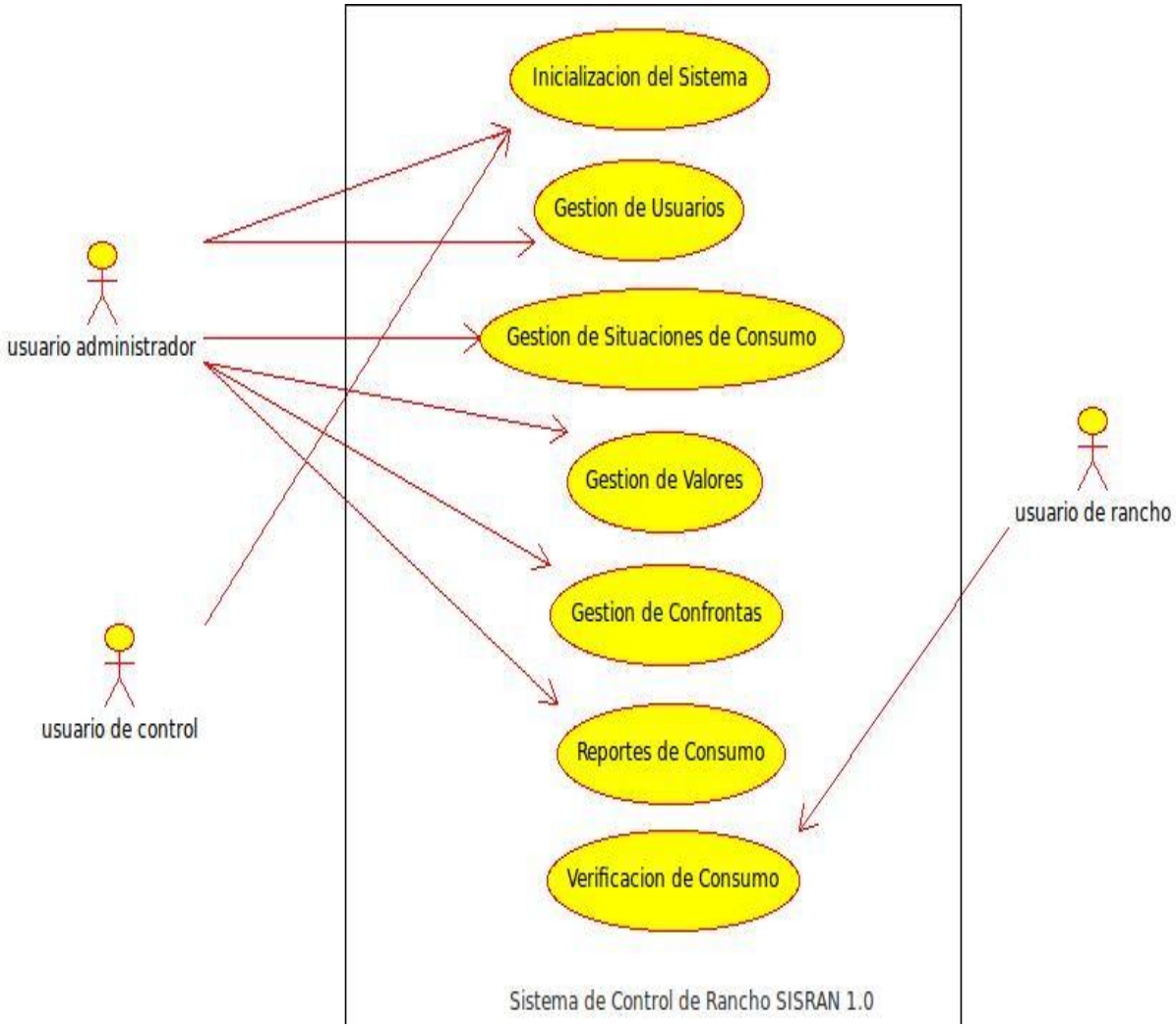


Diagrama 1. Diagrama de Caso de uso para Usuarios del SISRAN 1.0

6.7.8 Casos de Uso Expandidos

i. Inicialización de Sistema

Nombre:	Inicialización del Sistema
Autor:	Oswaldo Cruz
Fecha:	01-FEB-2011
Descripción: Permite que los usuarios administrador y de control puedan inicializar el sistema.	
Actores: • Usuario Administrador • Usuario de Control	
Precondiciones: El usuario debe haberse loggeado en el sistema.	
Flujo Normal: 1. El actor selecciona el usuario de una lista desplegable 2. El usuario ingresa la contraseña 3. El sistema comprueba la contraseña 4. Si la contraseña es correcta ingresa al sistema	
Flujo Alternativo: 4. Si la contraseña es incorrecta el sale un mensaje de error y pide nuevamente la contraseña.	
Pos condiciones: El sistema debe identificar si se trata de usuario administrador o de control.	

Tabla 1. Caso de uso expandido Inicialización del sistema

ii. Gestión de Usuarios

Nombre:	Gestión de Usuarios
Autor:	Oswaldo Cruz
Fecha:	01-FEB-2011
Descripción: Permite gestionar usuarios tanto del sistema como usuarios del servicio de rancho.	
Actores: Usuario Administrador.	
Precondiciones: El usuario debe haberse loggeado en el sistema y ser identificado como Administrador.	
Flujo Normal: 1. El actor decide entre crear un nuevo usuario, editar o eliminar un usuario existente. 2. El sistema recuerda cual fue la acción que decidió tomar el actor. 3. El momento que el usuario administrador hace clic en el botón guardar el sistema se encarga de realizar todas las transacciones requeridas por el usuario administrador. 4. Si los cambios se realizaron correctamente, el sistema enviara el mensaje “Las operaciones solicitadas se realizaron con éxito”	
Flujo Alternativo: 4. Si los cambios no se realizaron correctamente, el sistema enviara el mensaje “Error, datos no guardados”	
Pos condiciones: El sistema deberá actualizar la lista de usuarios luego de guardar los cambios.	

Tabla 2. Caso de uso expandido Gestión de usuarios

iii. Gestión de Situaciones de Consumo

Nombre:	Gestión de Situaciones de Consumo
Autor:	Oswaldo Cruz
Fecha:	01-FEB-2011
Descripción: Permite gestionar las situaciones de consumo de rancho.	
Actores: Usuario Administrador.	
Precondiciones: El usuario debe haberse loggeado en el sistema y ser identificado como Administrador.	
Flujo Normal: 1. El administrador decide entre crear una nueva situación de consumo, editar o eliminar una situación existente. 2. El sistema recuerda cual fue la acción que decidió tomar el actor. 3. El momento que el usuario administrador hace clic en el botón guardar el sistema se encarga de realizar todas las transacciones requeridas por el usuario administrador. 4. Si los cambios se realizaron correctamente, el sistema enviara el mensaje “Las operaciones solicitadas se realizaron con éxito”.	
Flujo Alternativo: 4. Si los cambios no se realizaron correctamente, el sistema enviara el mensaje “Error, datos no guardados”	
Pos condiciones: El sistema deberá actualizar la lista de situaciones luego de guardar los cambios.	

Tabla 3. Caso de uso expandido Gestión de situaciones de consumo

iv. Gestión de Valores

Nombre:	Gestión de Valores
Autor:	Oswaldo Cruz
Fecha:	01-FEB-2011
Descripción: Permite gestionar los valores de cobro de rancho.	
Actores: Usuario Administrador.	
Precondiciones: El usuario debe haberse loggeado en el sistema y ser identificado como Administrador.	
Flujo Normal: 1. El administrador decide entre crear nuevos valores de cobro, editar o eliminar valores existentes. 2. El sistema recuerda cual fue la acción que decidió tomar el actor. 3. El momento que el usuario administrador hace clic en el botón guardar el sistema se encarga de realizar todas las transacciones requeridas por el usuario administrador. 4. Si los cambios se realizaron correctamente, el sistema enviara el mensaje “Las operaciones solicitadas se realizaron con éxito”.	
Flujo Alternativo: 4. Si los cambios no se realizaron correctamente, el sistema enviara el mensaje “Error, datos no guardados”	
Pos condiciones: El sistema deberá actualizar la lista de valores de cobro luego de guardar los cambios.	

Tabla 4. Caso de uso expandido Gestión de valores

v. Gestión de Confrontas

Nombre:	Gestión de Confrontas
Autor:	Oswaldo Cruz
Fecha:	01-FEB-2011
Descripción: Permite gestionar las confrontas de rancho.	
Actores: Usuario Administrador.	
Precondiciones: 1. El usuario debe haberse loggeado en el sistema y ser identificado como Administrador. 2. Las confrontas podrán ser modificadas según sea la situación especial de cada usuario de rancho.	
Flujo Normal: 1. El administrador da clic en el botón pre-llenar confronta. 2. El sistema realiza el cálculo de las confrontas de todos los usuarios, a partir del primero hasta el último día del mes. 3. En el transcurso del mes el administrador registra todos los cambios de acuerdo a las situaciones que se vayan presentando. 4. Si los cambios se realizaron correctamente, el sistema enviara el mensaje “Las operaciones solicitadas se realizaron con éxito”.	
Flujo Alternativo: 4. Si los cambios no se realizaron correctamente, el sistema enviara el mensaje “Error, datos no guardados”	
Pos condiciones: El sistema deberá permitir los cambios en las guardias y otras situaciones.	

Tabla 5. Caso de uso expandido Gestión de confrontas

vi. Elaboración de Reportes de Consumo

Nombre:	Reportes de Consumo
Autor:	Oswaldo Cruz
Fecha:	01-FEB-2011
Descripción: Permite imprimir los reportes de consumo diarios y mensuales.	
Actores: Usuario Administrador.	
Precondiciones: El usuario debe haberse loggeado en el sistema y ser identificado como Administrador.	
Flujo Normal: 1. El administrador selecciona entre confronta mensual o diaria y hace clic en el botón generar informe. 2. El sistema llena una tabla temporal dentro de la base de datos, esta tabla contiene los siguientes campos: cedula, apellidos, nombres, desayunos, almuerzos, meriendas y total. 3. El sistema muestra al usuario administrador el reporte listo para imprimir.	
Flujo Alternativo: 3. Si no existen los datos el sistema emitirá el mensaje, “el reporte esta vacio”.	
Pos condiciones: Los reportes diarios y el reporte mensual deben concordar, es decir que la suma que se cobra a los usuarios debe ser igual a la que se ha pasado al encargado de preparar el rancho.	

Tabla 6. Caso de uso expandido Reportes de consumo

vii. Verificación de Consumo

Nombre:	Verificación de Consumo
Autor:	Oswaldo Cruz
Fecha:	01-FEB-2011
Descripción: El sistema verifica si el usuario está considerando en la confronta de rancho.	
Actores: Usuario Consumidor de Rancho.	
Precondiciones: El usuario debe identificarse mediante una tarjeta con código de barras, la cual será leída por el lector de código de barras.	
Flujo Normal: 1. El usuario consumidor pasa la tarjeta por el lector, permitiendo que el sistema lea el código personal del usuario, es decir su número de cedula. 2. El sistema busca al usuario y verifica si está considerado en la confronta, para ello también debe considerar la hora actual para saber si es desayuno, almuerzo o merienda. 3. Si el usuario está considerado, el sistema muestra en pantalla el nombre del usuario e imprime un ticket para que acceda al servicio.	
Flujo Alternativo: 3. Si el usuario no está considerado, el sistema emitirá el mensaje “usuario no considerado en confronta”.	
Pos condiciones: Ninguna.	

Tabla 7. Caso de uso expandido Verificación de consumo

6.7.9 Diagramas de secuencia

Los siguientes diagramas de secuencia fueron diseñados de acuerdo con el diagrama de casos de uso expuesto en el tema 6.1.2.

i. Inicialización de Sistema

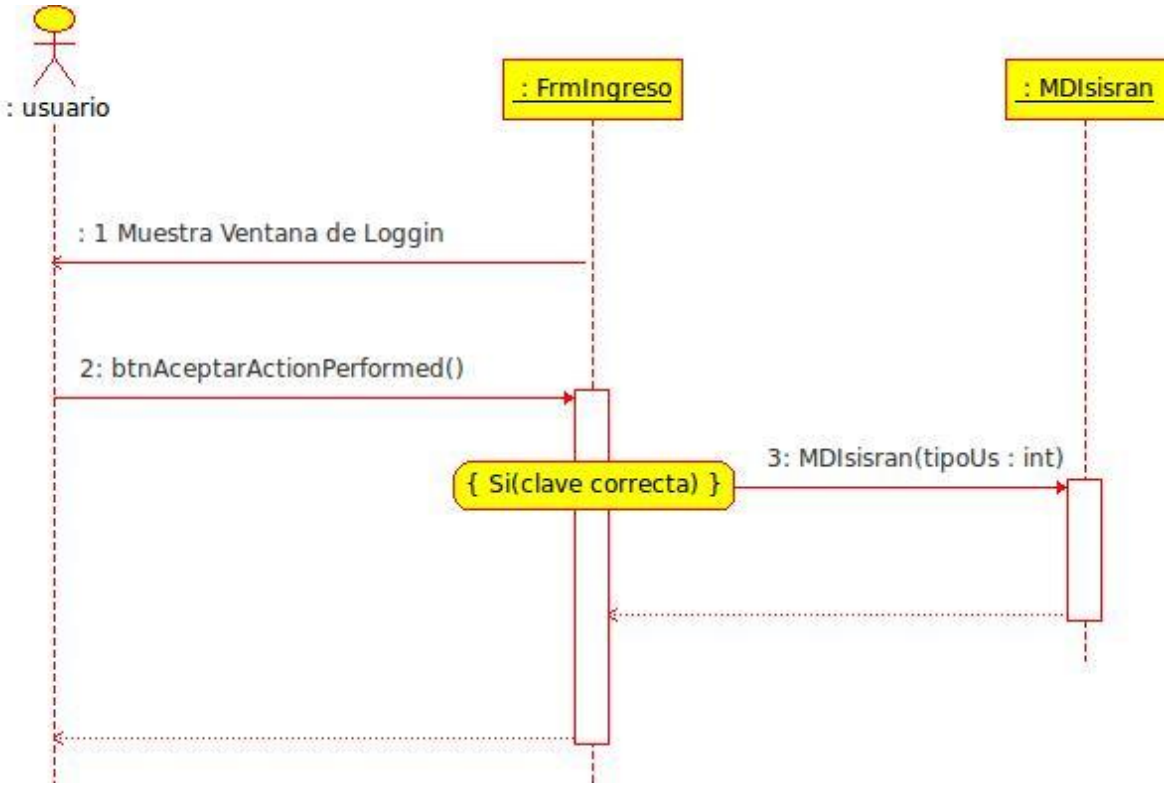


Diagrama 2. Diagrama de secuencia Inicialización de Sistema

ii. Gestión de Usuarios de Sistema

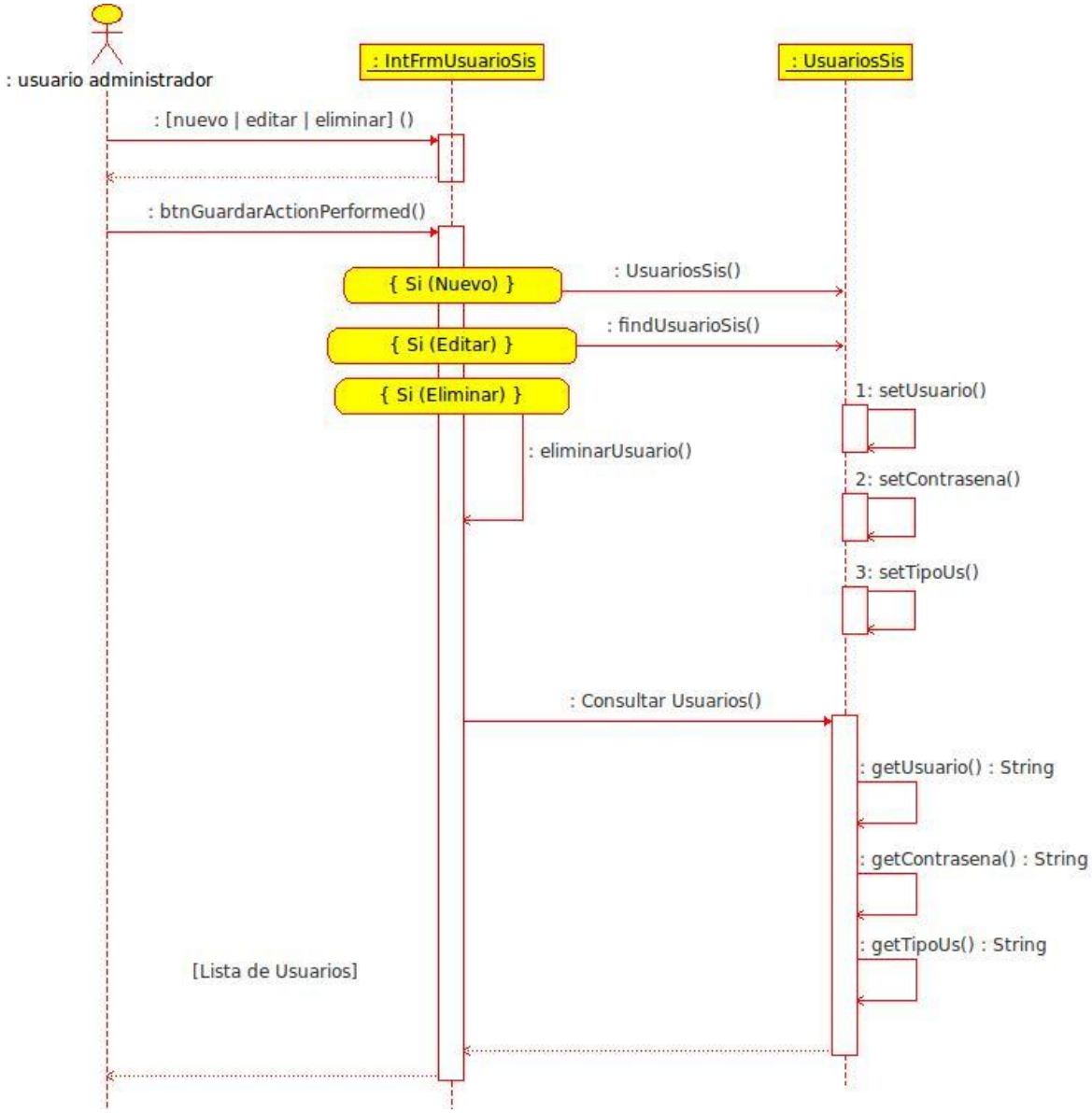


Diagrama 3. Diagrama de secuencia Gestión de usuarios del Sistema

iii. Gestión de Usuarios de Rancho

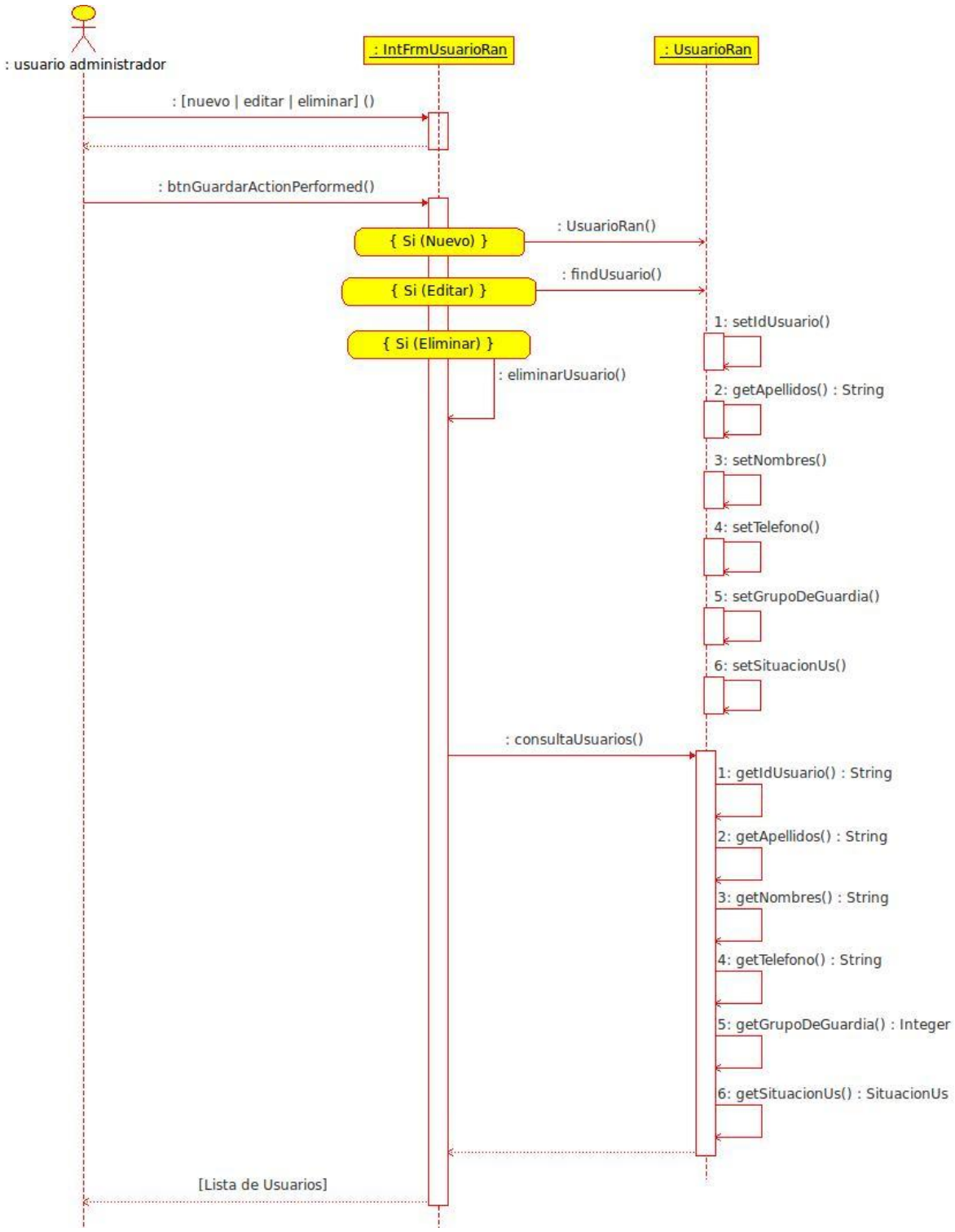


Diagrama 4. Diagrama de secuencia Gestión de usuarios de Rancho

iv. Gestión de Situaciones de Consumo

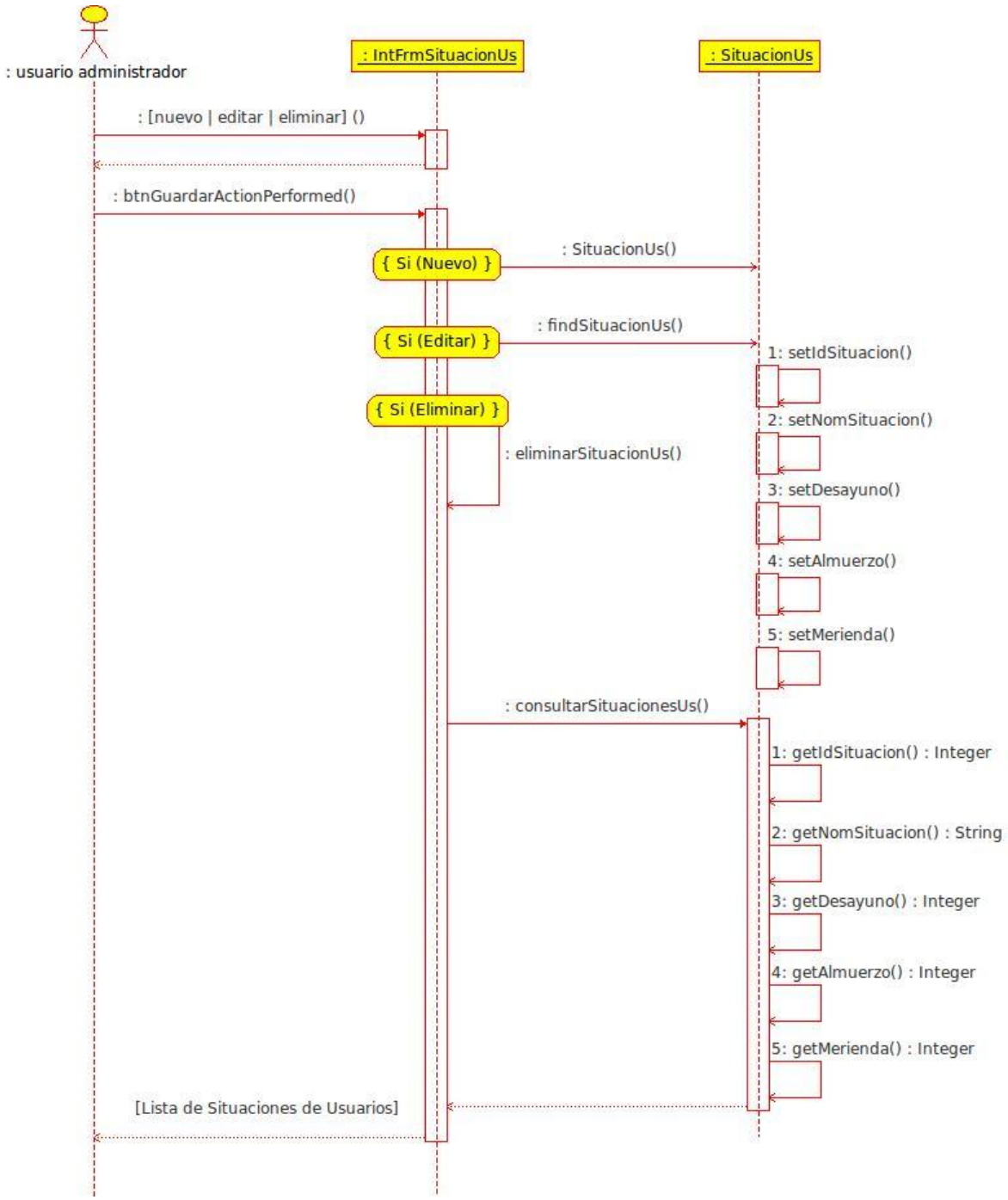


Diagrama 5. Diagrama de secuencia Gestión de situaciones de consumo

v. Gestión de Valores

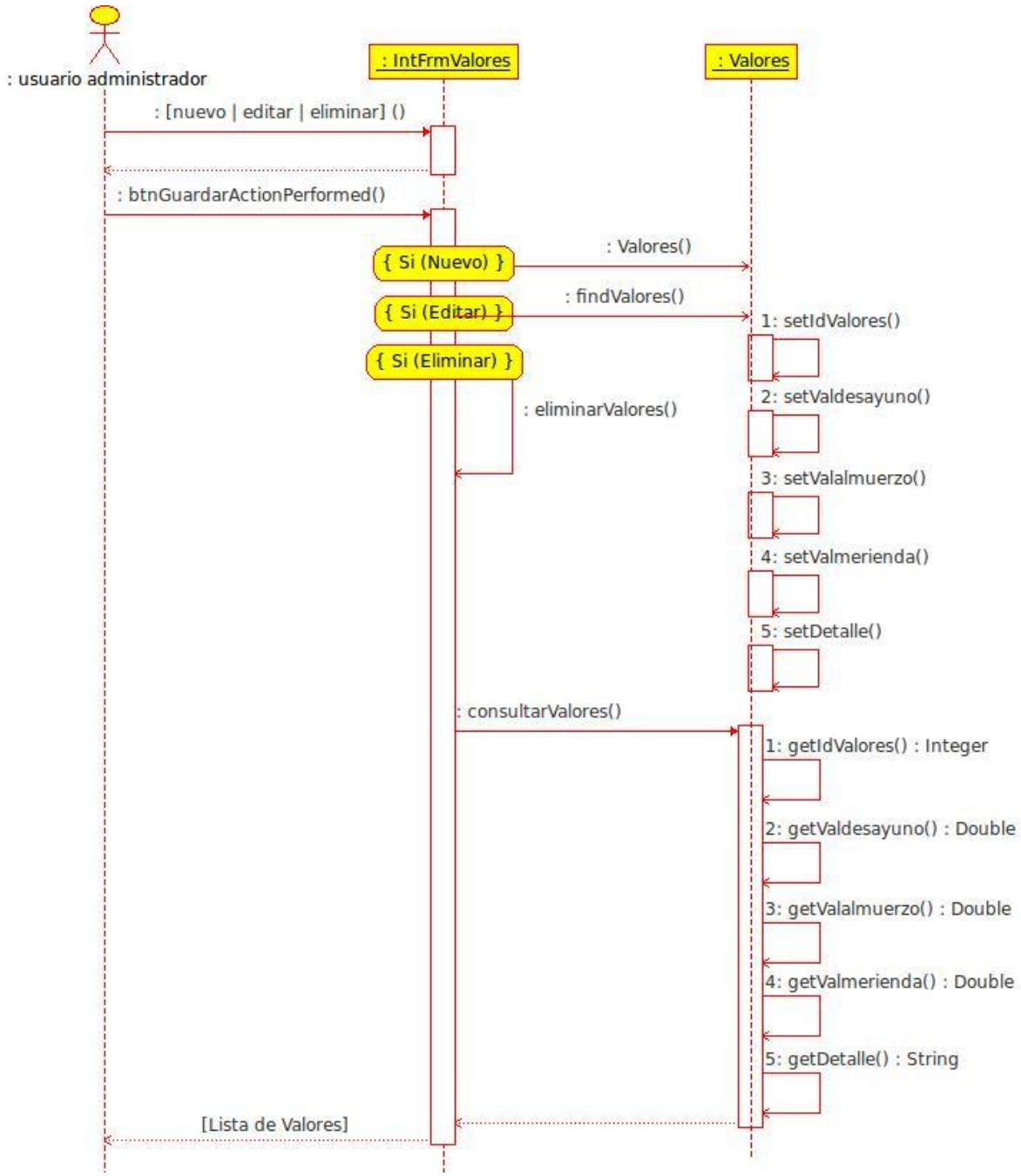


Diagrama 6. Diagrama de secuencia Gestión de valores

vi. Gestión de Confrontas

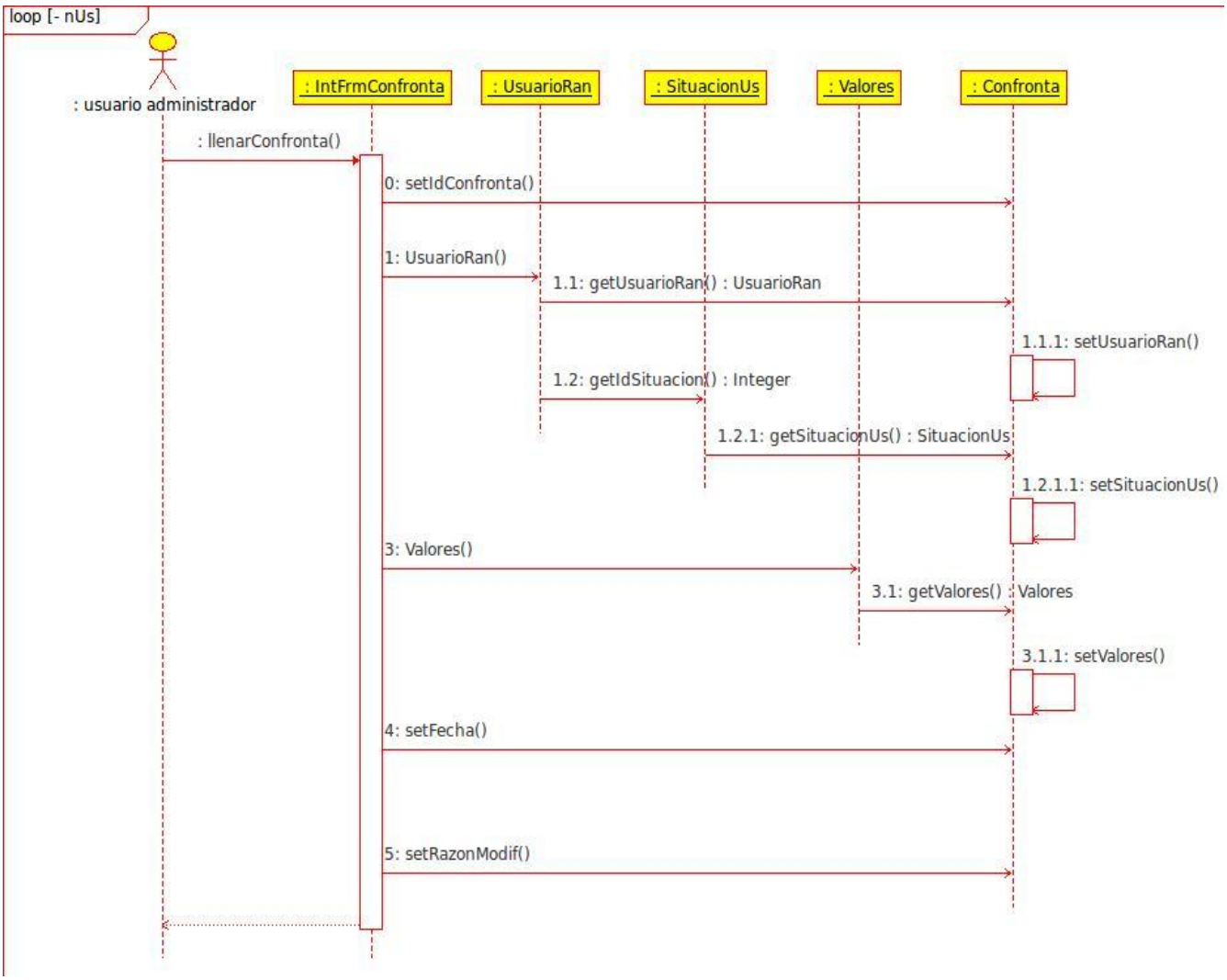


Diagrama 7. Diagrama de secuencia Gestión de confrontas

vii. Elaboración de Reportes de Consumo

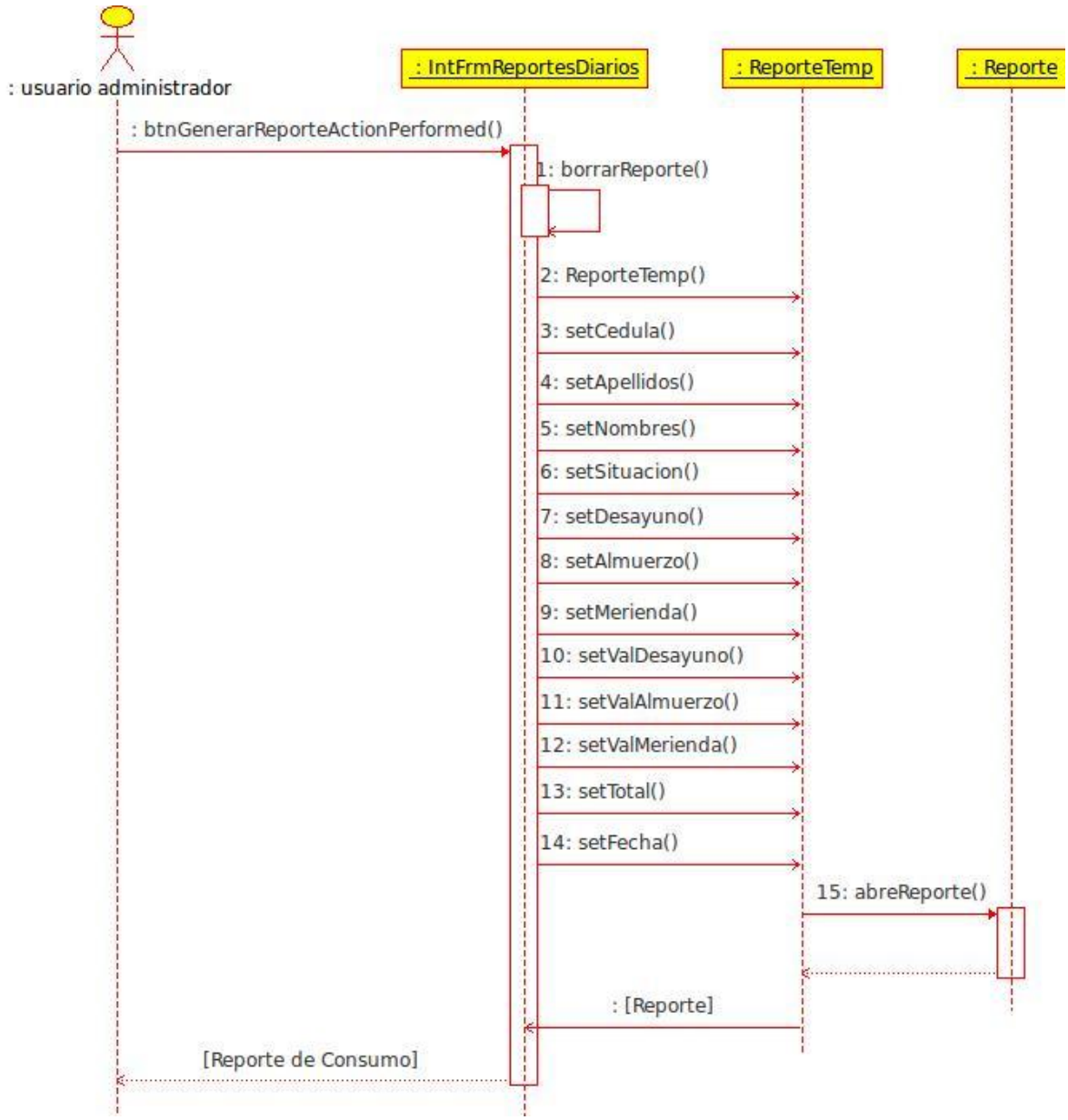


Diagrama 8. Diagrama de secuencia Elaboración de reportes de consumo

viii. Verificación de Consumo

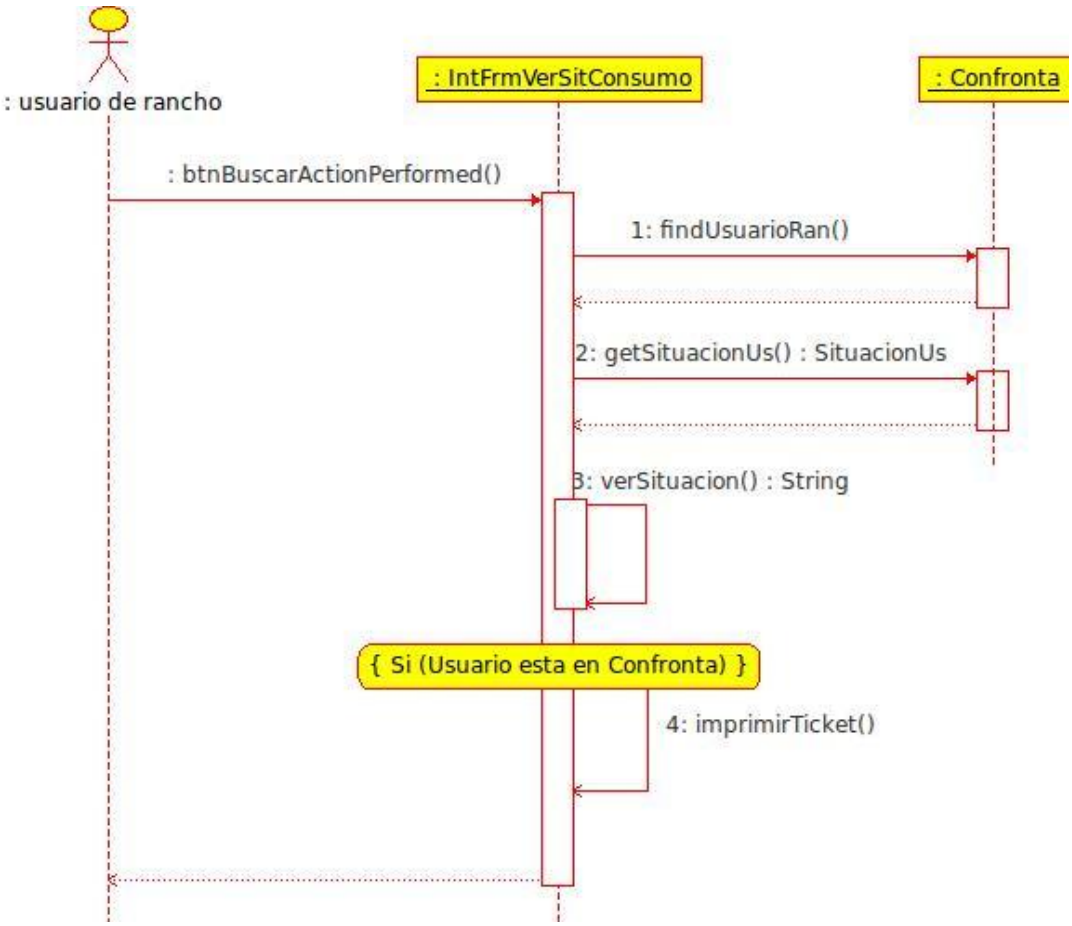
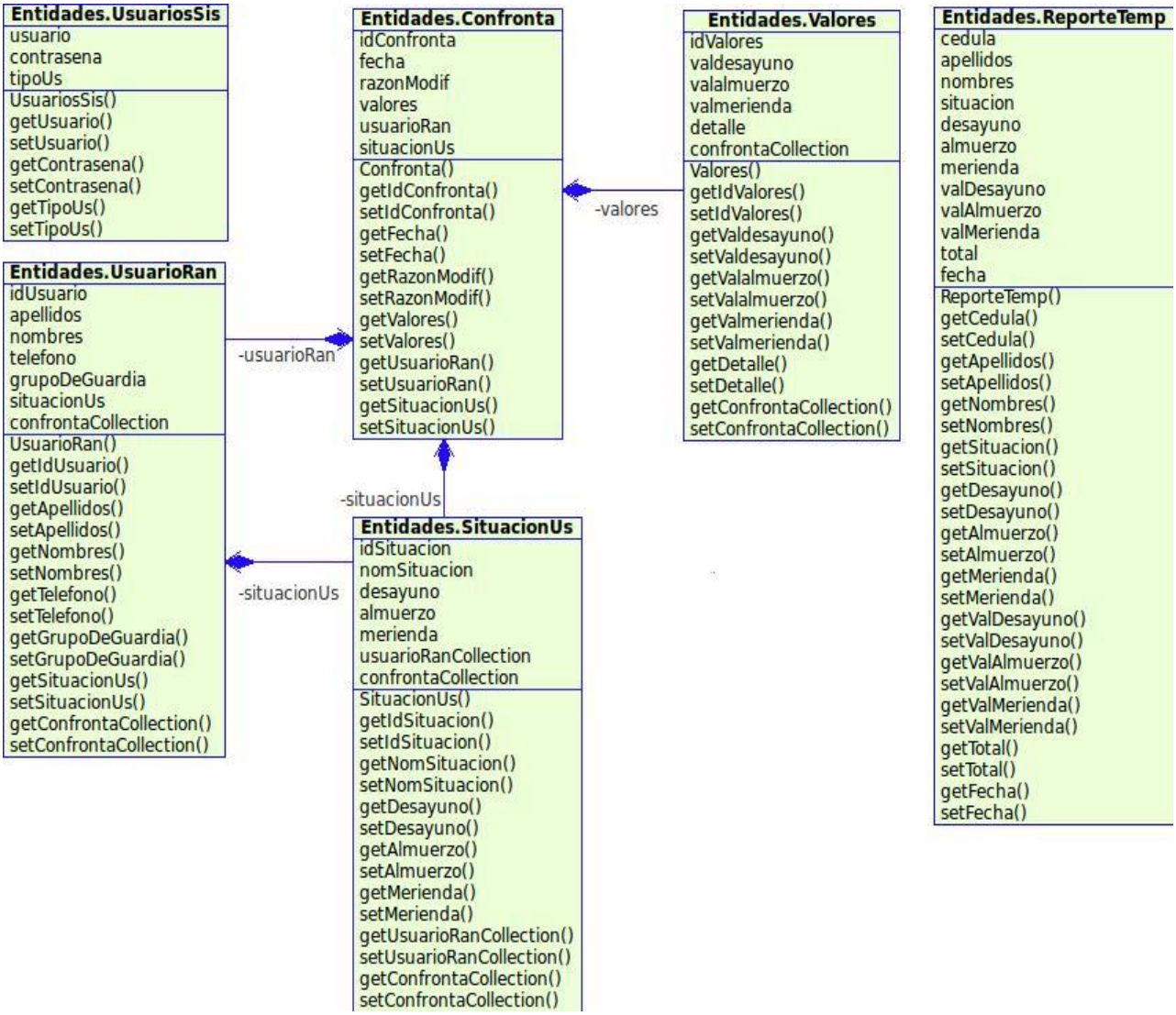
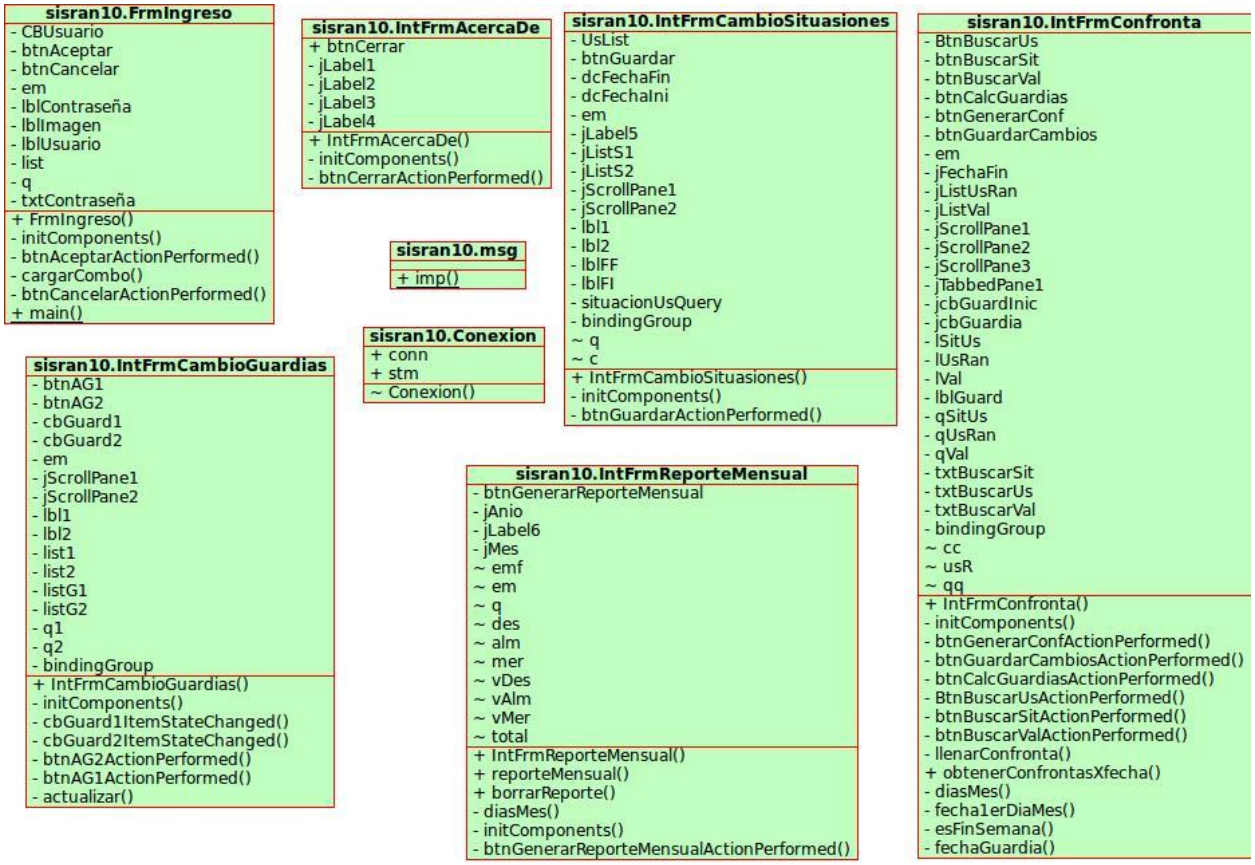


Diagrama 9. Diagrama de secuencia Verificación de consumo

6.7.10 Diagramas de clases

Una vez realizado el análisis correspondiente del diagrama de casos de uso, procedemos a realizar un diseño de las clases más importantes que se utilizaran en el sistema de control de rancho SISRAN 1.0.




sisran10.FrmIngreso
- CBUsuario
- btnAceptar
- btnCancelar
- em
- lblContraseña
- lblImagen
- lblUsuario
- list
- q
- txtContraseña
+ FrmIngreso()
- initComponents()
- btnAceptarActionPerformed()
- cargarCombo()
- btnCancelarActionPerformed()
+ main()
sisran10.IntFrmAcercaDe
+ btnCerrar
- jLabel1
- jLabel2
- jLabel3
- jLabel4
+ IntFrmAcercaDe()
- initComponents()
- btnCerrarActionPerformed()
sisran10.msg
+ imp()
sisran10.Conexion
+ conn
+ stm
~ Conexion()
sisran10.IntFrmCambioGuardias
- btnAG1
- btnAG2
- cbGuard1
- cbGuard2
- em
- jScrollPane1
- jScrollPane2
- lbl1
- lbl2
- list1
- list2
- listG1
- listG2
- q1
- q2
- bindingGroup
+ IntFrmCambioGuardias()
- initComponents()
- cbGuard1ItemStateChanged()
- cbGuard2ItemStateChanged()
- btnAG2ActionPerformed()
- btnAG1ActionPerformed()
- actualizar()
sisran10.IntFrmCambioSituasiones
- UsList
- btnGuardar
- dcFechaFin
- dcFechaIni
- em
- jLabel5
- jListS1
- jListS2
- jScrollPane1
- jScrollPane2
- lbl1
- lbl2
- lblFF
- lblFI
- situacionUsQuery
- bindingGroup
~ q
~ c
+ IntFrmCambioSituasiones()
- initComponents()
- btnGuardarActionPerformed()
sisran10.IntFrmReporteMensual
- btnGenerarReporteMensual
- jAnio
- jLabel6
- jMes
~ emf
~ em
~ q
~ des
~ alm
~ mer
~ vDes
~ vAlm
~ vMer
~ total
+ IntFrmReporteMensual()
+ reporteMensual()
+ borrarReporte()
- diasMes()
- initComponents()
- btnGenerarReporteMensualActionPerformed()
sisran10.IntFrmConfronta
- BtnBuscarUs
- btnBuscarSit
- btnBuscarVal
- btnCalcGuardias
- btnGenerarConf
- btnGuardarCambios
- em
- jFechaFin
- jListUsRan
- jListVal
- jScrollPane1
- jScrollPane2
- jScrollPane3
- jTabbedPane1
- jcbGuardInic
- jcbGuardia
- lSitUs
- lUsRan
- lVal
- lblGuard
- qSitUs
- qUsRan
- qVal
- txtBuscarSit
- txtBuscarUs
- txtBuscarVal
- bindingGroup
~ cc
~ usR
~ qq
+ IntFrmConfronta()
- initComponents()
- btnGenerarConfActionPerformed()
- btnGuardarCambiosActionPerformed()
- btnCalcGuardiasActionPerformed()
- BtnBuscarUsActionPerformed()
- btnBuscarSitActionPerformed()
- btnBuscarValActionPerformed()
- llenarConfronta()
+ obtenerConfrontasXfecha()
- diasMes()
- fecha1erDiaMes()
- esFinSemana()
- fechaGuardia()

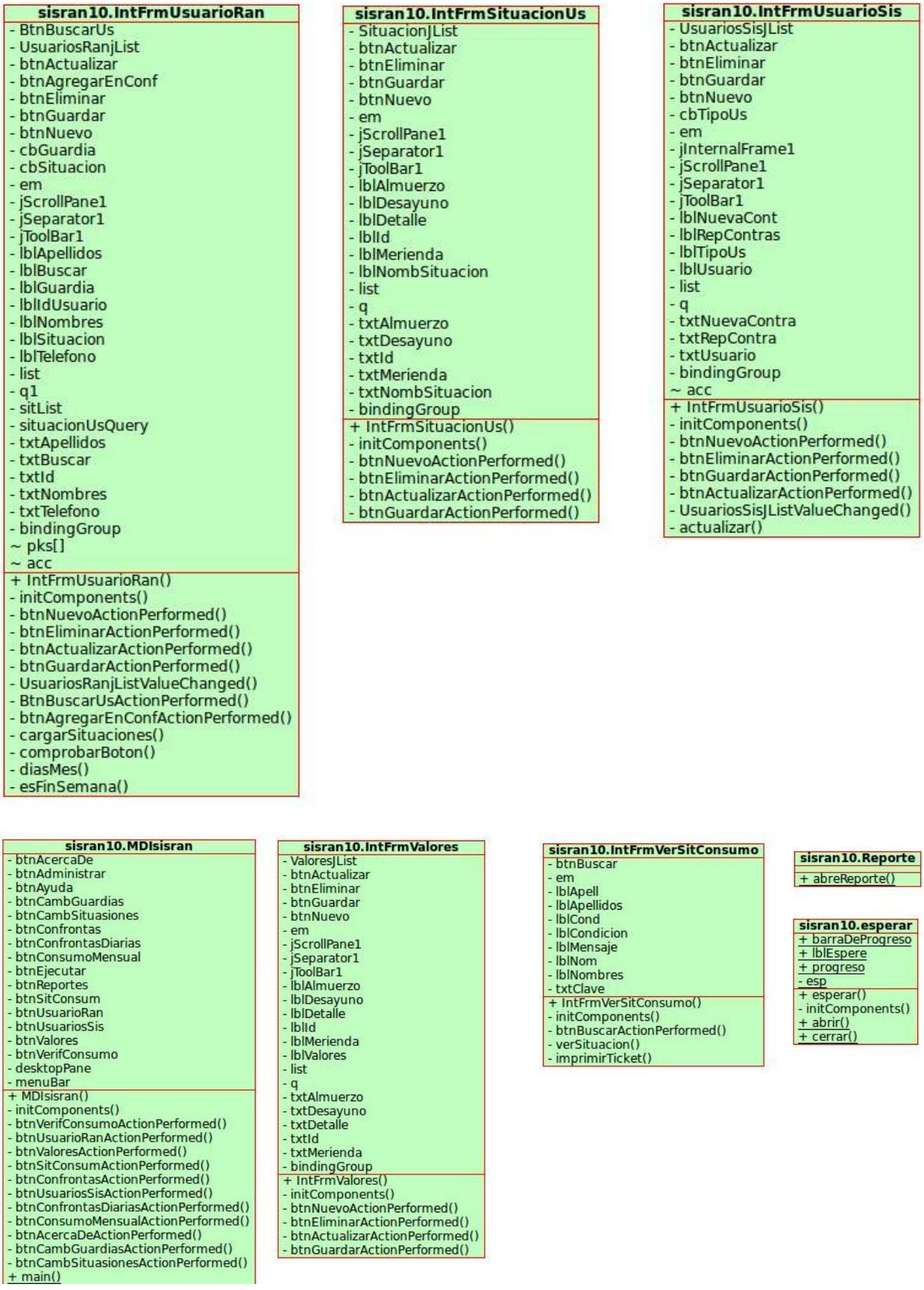


Diagrama 10. Diagramas de Clases

6.7.11 Diagramas de base de datos

i. Diagrama Entidad-Relación

Para nuestro sistema hemos diseñado el diagrama de clases con anterioridad, y este diagrama contiene esencialmente a las clases Entidad, encargadas de conectarse y administrar los datos que se encuentran en las tablas de nuestra base de datos SISRANDB en PostgreSQL. Por lo tanto el diagrama de clases es muy similar al diagrama entidad relación que se implementara en nuestra base de datos.

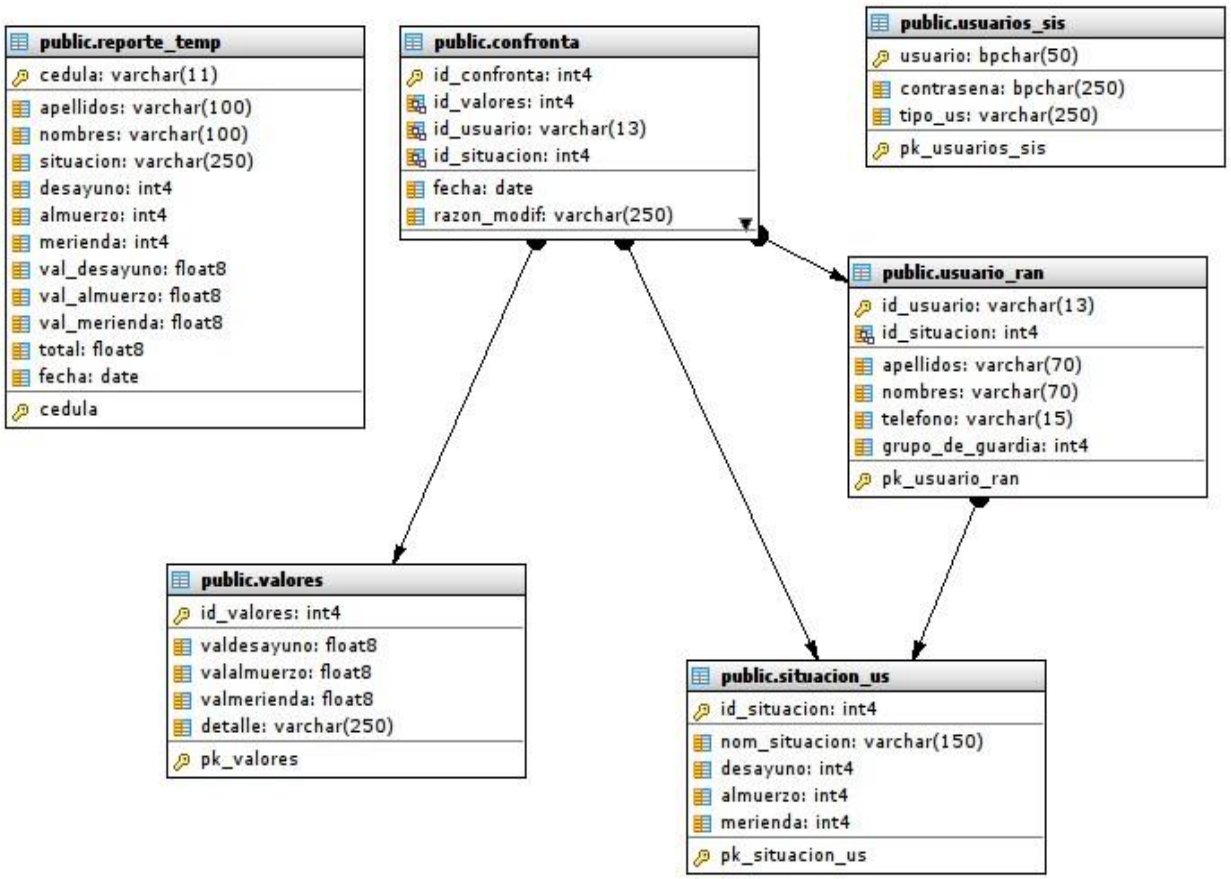


Diagrama 11. Diagrama Entidad – Relación

La implementación de las tablas se lo puede realizar mediante el programa PostgreSQL Maestro o también mediante el pgDesigner.

Adicionalmente el archivo sisrandb.sql, el cual puede importarse mediante el programa pgAdmin III creando automáticamente las tablas, se encuentra el Cd del sistema de control de rancho SISRAN 1.0 (ANEXO C).

6.8 Programación

6.8.1 FrmIngreso.java

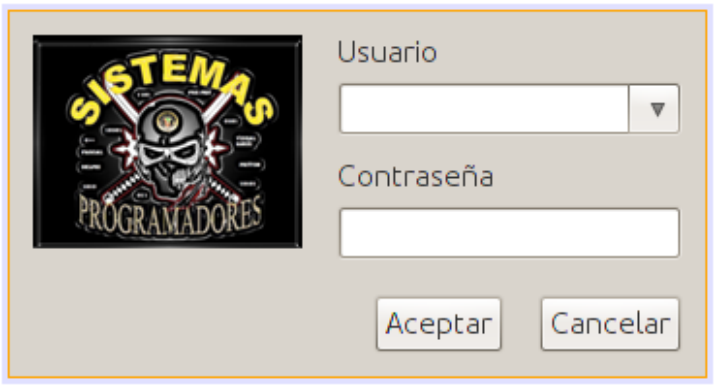


Fig. 98 Ventana de Ingreso al sistema

En esta clase verificamos el ingreso de nuestros usuarios y también el tipo de usuario que es (Usuario Administrador o Usuario de Control), en el constructor especificamos cual va a ser el botón predeterminado es decir si el usuario hace enter se ejecutara el botón btnAceptar, además de esto se llama a la función cargarCombo.

```
public class FrmIngreso extends javax.swing.JDialog {

    public FrmIngreso(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
        this.setLocationRelativeTo(null); //posicionar ventana en el centro de la pantalla
        this.getRootPane().setDefaultButton(btnAceptar); //establecer aceptar como por defecto
        cargarCombo();
    }
```

En la función cargar combo vamos a leer el nombre de los usuario y vamos

a añadirlo a nuestro jComboBox

```
private void cargarCombo(){
    for(int i=0; i<=list.size()-1; i++)
        this.CBUsuario.addItem(list.get(i).getUsuario());
}
```

Dentro del botón aceptar colocamos el siguiente código que buscara a nuestro usuario con la función em.find(), para posteriormente obtener la contraseña y compararla con la que ingreso el usuario, además de esto controlamos también si es usuario administrador (1) o de control (0) y enviamos el resultado dentro del constructor del formulario principal.

```
private void btnAceptarActionPerformed(java.awt.event.ActionEvent evt) {
    UsuariosSis u = em.find(UsuariosSis.class, this.CBUsuario.getSelectedItem().toString());

    if(u.getContrasena().equals(this.txtContraseña.getText())){
        int tipoUs = 0;
        if(u.getTipoUs().equals("Administrador SISRAN 1.0")) tipoUs=1;
        MDIsisran ms = new MDIsisran(tipoUs);
        ms.setVisible(true);
        this.setVisible(false);
    }else{
        msg.showMessageDialog(null, "CLAVE INCORRECTA...");
        this.txtContraseña.setText("");
    }
}
```

6.8.2 MDIsisran.java

Fig. 99 Formulario MDI para SISRAN

Este es el formulario principal, aquí nuestro usuario administrador podrá realizar la gestión de toda la información que concierne al sistema de control de rancho y el usuario de control podrá iniciar el subprograma de verificación de confronta es decir para que usuario.

Además en este formulario se deshabilitara los botones de administración para los usuarios control, esto se lo realizara en el constructor de la clase.

```
public MDIsisran(int tipoUs) {
    initComponents();
    this.setExtendedState(MAXIMIZED_BOTH);  //para que se inicie maximizada la ventana
    if(tipoUs==0)
        this.btnAdministrar.setEnabled(false);
        this.btnReportes.setEnabled(false);
}
```

Cada botón abre un jInternalFrame, es decir al resto de ventanas que este caso son internas a este formulario para ello creamos el siguiente código en cada botón, aquí tres ejemplos de lo que se hace para cada ventana interna.

```
private void btnVerifConsumoActionPerformed(java.awt.event.ActionEvent evt) {
    IntFrmVerSitConsumo verCons = new IntFrmVerSitConsumo();
    desktopPane.add(verCons);
    verCons.setVisible(true);
}

private void btnUsuarioRanActionPerformed(java.awt.event.ActionEvent evt) {
    IntFrmUsuarioRan usRan = new IntFrmUsuarioRan();
    desktopPane.add(usRan);
    usRan.setVisible(true);
}

private void btnValoresActionPerformed(java.awt.event.ActionEvent evt) {
    IntFrmValores val = new IntFrmValores();
    desktopPane.add(val);
    val.setVisible(true);
}
```

6.8.3 IntFrmUsuarioSis.java

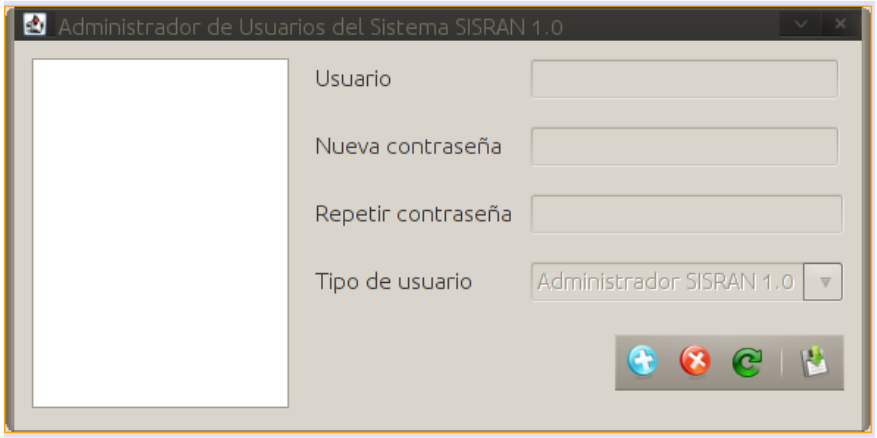


Fig. 100 Formulario Tipos de usuario

Dentro de esta clase el usuario administrador podrá crear nuevos usuario de tipo administrador y también de control, para ello usamos las técnicas de programación Java Percistence Api, que aprendimos en los capítulos anteriores.

```
private void btnNuevoActionPerformed(java.awt.event.ActionEvent evt) {
    UsuariosSis u = new UsuariosSis();
    em.persist(u);
    list.add(u);
    this.UsuariosSisJList.setSelectedIndex(list.size()-1);
    this.txtUsuario.requestFocus();
}

private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
    int r = msg.showConfirmDialog(null, "Seguro que desea eliminar este registro");
    int i = this.UsuariosSisJList.getSelectedIndex();
    if(i>=0 && r==0){
        UsuariosSis v = list.get(i);
        list.remove(v);
        em.remove(v);
        acc='x';
    }
}
```

```
private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {
    String pw = this.txtNuevaContra.getText();
    String tp = this.cbTipoUs.getSelectedItem().toString();
    int i = this.UsuariosSisJList.getSelectedIndex();

    if(pw.equals(this.txtRepContra.getText()) || acc=='x'){
        try{
            if(acc!='x'){
                list.get(i).setContrasena(pw);
                list.get(i).setTipoUs(tp);
            }
            em.getTransaction().commit();
            em.getTransaction().begin();
            msg.showMessageDialog(null, "Los cambios se han realizado con exito.");
        } finally{
              this.txtRepContra.setText("");
              acc='e';
        }
    } else msg.showMessageDialog(null, "Error, las contraseñas no coinciden...");
}
```

Además de crear un nuevo, eliminar y guardar también está la función actualizar y si es seleccionado un usuario en el jList, se mostrara también que tipo de usuario es para ser posteriormente modificado.

```
private void btnActualizarActionPerformed(java.awt.event.ActionEvent evt) {
    em.getTransaction().rollback();
    em.getTransaction().begin();
    list.clear();
    list.addAll(q.getResultList());
}

private void UsuariosSisJListValueChanged(javax.swing.event.ListSelectionEvent evt) {
    int i = this.UsuariosSisJList.getSelectedIndex();
    if(i>=0){
        UsuariosSis u = list.get(i);
        if(u.getTipoUs()!=null)
            this.cbTipoUs.setSelectedItem(u.getTipoUs().toString());
    }
}
```

6.8.4 IntFrmUsuarioRan.java

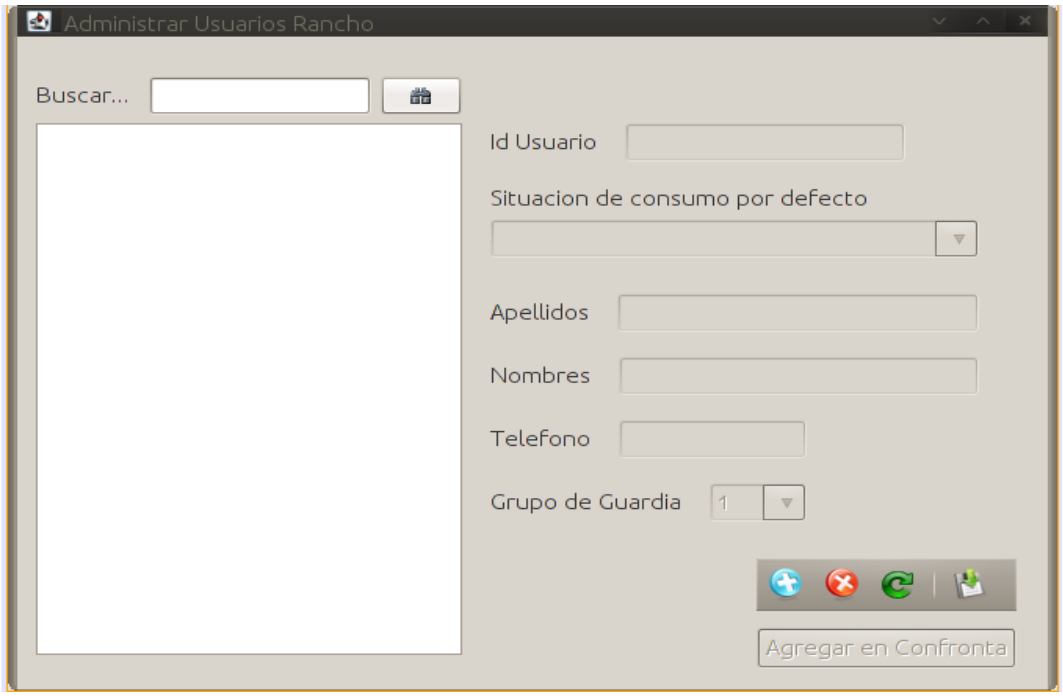


Fig. 101 Formulario Administrar Usuarios de Rancho

En esta clase el usuario podrá realizar altas, bajas y cambios en la base de datos del usuario de rancho, esto se lo realiza de similar forma que con la clase IntFrmUsuarioSis.

```
    private void btnNuevoActionPerformed(java.awt.event.ActionEvent evt) {
        UsuarioRan v = new UsuarioRan();
        em.persist(v);
        list.add(v);
        this.UsuariosRanjList.setSelectedIndex(list.size()-1);
        this.txtId.requestFocus();
}

    private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
        int r = msg.showConfirmDialog(null, "Seguro que desea eliminar este registro");
        int i = UsuariosRanjList.getSelectedIndex();
        if(i>=0 && r==0){
            UsuarioRan s = list.get(i);
            list.remove(s);
            em.remove(s);
            acc='x';
        }
}

    private void btnActualizarActionPerformed(java.awt.event.ActionEvent evt) {
        em.getTransaction().rollback();
        em.getTransaction().begin();
        list.clear();
        list.addAll(q1.getResultList());
    }
```

```
private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        int i = this.cbSituacion.getSelectedIndex();
        int j = this.UsuariosRanjList.getSelectedIndex();
        int g = Integer.parseInt(cbGuardia.getSelectedItem().toString());

        if(acc=='e'){
           SituacionUs us = em.find(SituacionUs.class, pks[i]);
           list.get(j).setSituacionUs(us);
           list.get(j).setGrupoDeGuardia(g);
        }
        em.getTransaction().commit();
        em.getTransaction().begin();
        msg.showMessageDialog(null, "Los cambios se han realizado con exito.");
    }finally{acc='e';}
```

```
private void UsuariosRanjListValueChanged(javax.swing.event.ListSelectionEvent evt) {
   int i = this.UsuariosRanjList.getSelectedIndex();
   if(i>-1){
       UsuarioRan u = list.get(i);
       if(u.getSituacionUs()!=null && u.getGrupoDeGuardia()!=null){
           this.cbSituacion.setSelectedItem(u.getSituacionUs().getNomSituacion().toString()
           this.cbGuardia.setSelectedItem(u.getGrupoDeGuardia().toString());
           comprobarBoton(u);
       }
    }
```

También tenemos el siguiente método de búsqueda de usuarios, este método seleccionara al usuario más próximo que tenga el apellido especificado en la caja de texto donde se ingresa el apellido a buscar.

```
private void BtnBuscarUsActionPerformed(java.awt.event.ActionEvent evt) {
    String ap = this.txtBuscar.getText();
    Query query = em.createQuery("SELECT u FROM UsuarioRan u WHERE u.apellidos like :apellidos");
    query.setParameter("apellidos", ap + "%");
    List<UsuarioRan> listU = q1.getResultList();
    for(int i=0; i<listU.size(); i++){
        for(int j=0; j<list.size(); j++){
            if(listU.get(i).getIdUsuario().equals(list.get(j).getIdUsuario())){
                this.UsuariosRanjList.setSelectedIndex(j);
                int r = msg.showConfirmDialog(null, "Buscar Siguiente ...");
                if(r!=0){
                    j=list.size();
                    i=listU.size();
                }
            }
        }
    }
```

Además en el siguiente método todos los nuevos usuarios serán ingresados directamente en confronta simplemente haciendo clic en el botón “agregar en confronta”.

```
private void btnAgregarEnConfActionPerformed(java.awt.event.ActionEvent evt) {
    int r = msg.showConfirmDialog(null, "<html>Va a agregar un nuevo usuario a la confronta mensual,"
                                    + "<br>haga esto solo si ya ha pre-llenado la confronta mensual"
                                    + "<br>caso contrario cancele la a operacion.");
    if(r==0){
        int guardia = Integer.parseInt(msg.showInputDialog("Cual es el grupo de guardia de hoy..."));
        int nGrupos = Integer.parseInt(msg.showInputDialog("Cuantos grupos de guardia hay..."));

        Date fecha = new Date();
        UsuarioRan u = list.get(this.UsuariosRanjList.getSelectedIndex());
        SituacionUs finSem = em.find(SituacionUs.class, 9);
        SituacionUs guard = em.find(SituacionUs.class, 4);
        Query q = em.createQuery("SELECT c FROM Confronta c WHERE c.fecha = :fecha");
        q.setParameter("fecha", fecha);
        List<Confronta> conf = q.getResultList();
        if(!conf.isEmpty()){
```

```
        for(int i=fecha.getDay(); i<=diasMes(); i++){
            Confronta c = new Confronta(fecha, null, conf.get(0).getValores(), u, u.getSituacionUs());
            if(esFinSemana(fecha))
                    c.setSituacionUs(finSem);
            if(guardia==u.getGrupoDeGuardia())
                    c.setSituacionUs(guard);

            em.persist(c);
            em.getTransaction().commit();
            em.getTransaction().begin();


            if(guardia<nGrupos)guardia++;
            else guardia=1;
            fecha.setDate(fecha.getDate()+1);
        }
        msg.showMessageDialog(null, "Los cambios se han realizado correctamente.");
        comprobarBoton(u);
    }
}
```

También tenemos otros métodos que nos facilitan la programación y también hacer reusable el código para otras funciones, como por ejemplo el método cargarSituaciones (), el cual tiene el siguiente código:

```
private void cargarSituaciones(){
    for(int i=0; i<=sitList.size()-1; i++){
        pks[i] = sitList.get(i).getIdSituacion();
        this.cbSituacion.addItem(sitList.get(i).getNomSituacion());
    }
}
```

En necesario también implementar el método comprobarBoton (), el cual es usado dentro del método UsuariosRanjListValueChanged (), el cual permite la selección de un usuario, para de esta forma saber si es usuario nuevo o antiguo en el sistema.

```
private void comprobarBoton(UsuarioRan usRan){
    Date fecha = new Date();
    Query query = em.createQuery("SELECT c FROM Confronta c WHERE c.fecha = :fecha AND c.usuarioRan = :usRan");
    query.setParameter("fecha", fecha);
    query.setParameter("usRan", usRan);
    List<Confronta> c = query.getResultList();
    if(c.size()>0)
        this.btnAgregarEnConf.setEnabled(false);
    else
        this.btnAgregarEnConf.setEnabled(true);
}
```

También tenemos un método para calcular el número de días del mes y otro para conocer si es fin de semana, para posteriormente realizar el pre llenado de la confronta.

```
private int diasMes(){
    Calendar cal = Calendar.getInstance();
    int dias = cal.getActualMaximum(Calendar.DAY_OF_MONTH);
    return dias;
}

private Boolean esFinSemana(Date fecha){
   Boolean fs = false;
   String dia = fecha.toString();
   String d = dia.substring(0, 3);
   if(d.equals("Sat") || d.equals("Sun"))
       fs=true;
   return fs;
}
```

6.8.5 IntFrmSituacionUs.java

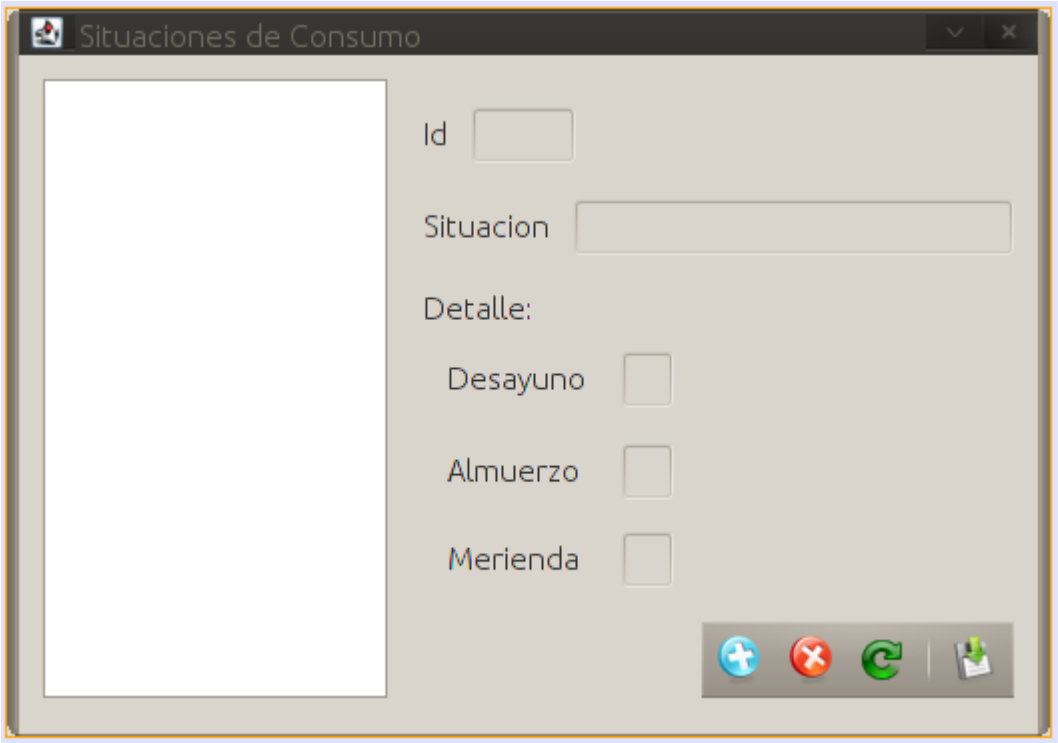


Fig. 102 Formulario Situaciones de consumo

En esta clase se realizan altas, bajas y cambios sin ningún método adicional.

```
private void btnNuevoActionPerformed(java.awt.event.ActionEvent evt) {
    SituacionUs v = new SituacionUs();
    em.persist(v);
    list.add(v);
    this.SituacionJList.setSelectedIndex(list.size()-1);
    this.txtNombSituacion.requestFocus();


private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
    int r = msg.showConfirmDialog(null, "Seguro que desea eliminar este registro");
    int i = SituacionJList.getSelectedIndex();
    if(i>=0 && r==0){
        SituacionUs s = list.get(i);
        if(s.getIdSituacion()!=9 && s.getIdSituacion()!=4){
            list.remove(s);
            em.remove(s);
        } else
            msg.showMessageDialog(null, "<html>No puede eliminar este elemento, ya que es<br>"
                                        + "indispensable para otros procesos...");
    }
```

```
private void btnActualizarActionPerformed(java.awt.event.ActionEvent evt) {
    em.getTransaction().rollback();
    em.getTransaction().begin();
    list.clear();
    list.addAll(q.getResultList());


private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        em.getTransaction().commit();
        em.getTransaction().begin();
        msg.showMessageDialog(null, "Los cambios se han realizado con exito.");
    }finally{}
```

6.8.6 IntFrmValores.java

En esta clase se realizan altas, bajas y cambios de los valores con los que se calcula la confronta, de igual forma se utiliza Java Persistence Apis o JPA para realizar la programación de los métodos.

```
private void btnNuevoActionPerformed(java.awt.event.ActionEvent evt) {
    Valores v = new Valores();
    em.persist(v);
    list.add(v);
    this.ValoresJList.setSelectedIndex(list.size()-1);
    this.txtDesayuno.requestFocus();
}

private void btnEliminarActionPerformed(java.awt.event.ActionEvent evt) {
    int r = msg.showConfirmDialog(null, "Seguro que desea eliminar este registro");
    int i = ValoresJList.getSelectedIndex();
    if(i>=0 && r==0){
        Valores v = list.get(i);
        list.remove(v);
        em.remove(v);
    }
}
```

```
private void btnActualizarActionPerformed(java.awt.event.ActionEvent evt) {
    em.getTransaction().rollback();
    em.getTransaction().begin();
    list.clear();
    list.addAll(q.getResultList());
}

private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        em.getTransaction().commit();
        em.getTransaction().begin();
        msg.showMessageDialog(null, "Los cambios se han realizado con exito.");
    }finally{}
}
```

6.8.7 IntFrmConfronta.java

En esta clase el usuario administrador realizara todas las gestiones correspondientes a confrontas, incluyendo las modificaciones de forma individual y colectiva, para ello es necesario mostrar un listado de usuarios, situaciones y valores, de donde el usuario administrador seleccionara los datos que desee modificar, el uso de esta ventana se lo explicará con más detalle en el manual de usuario.

Fig. 103 Registro de confrontas

Tenemos la pestaña en donde el usuario administrador podrá pre llenar las confrontas de todo el mes.

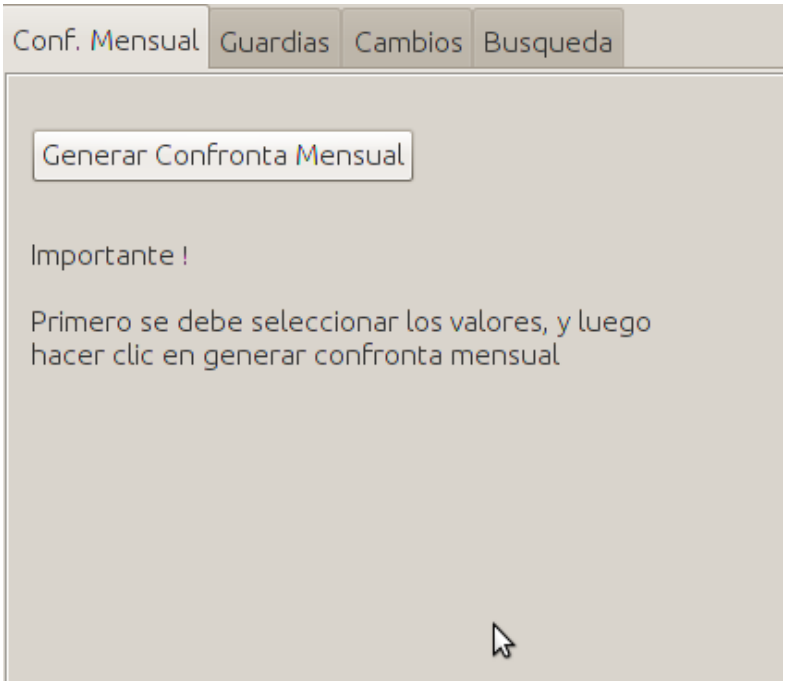


Fig. 104 Ventana para generar confrontas

En la pestaña guardias el usuario administrador podrá realizar los cálculos de las guardias de todos los usuarios de rancho.

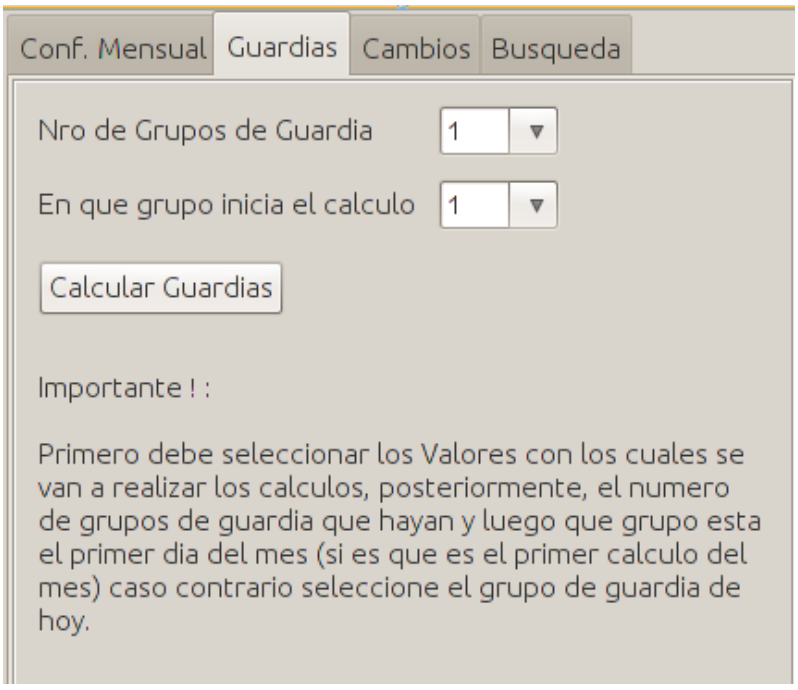


Fig. 105 Ventana para cálculo de guardias

En la pestaña cambios el usuario administrador podrá realizar los cambios de forma individual y por periodos de tiempo.

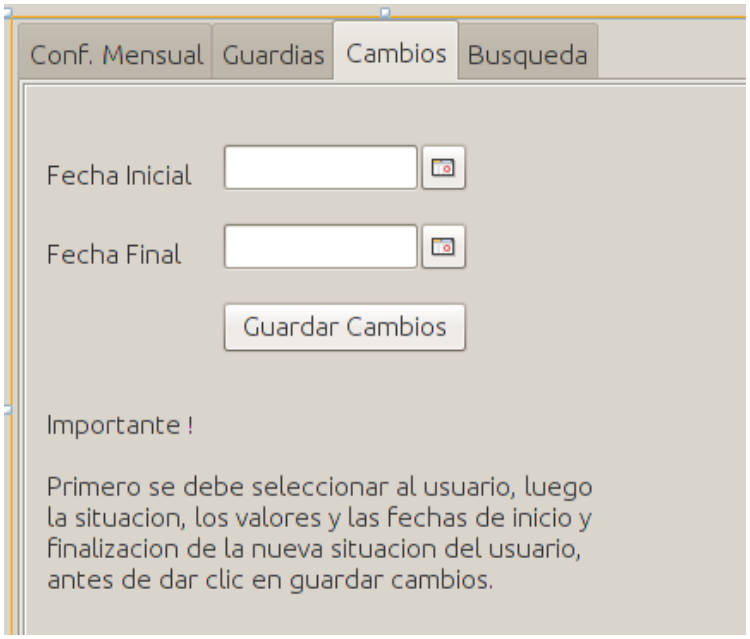


Fig. 106 Ventana para cambio de fechas

La pestaña búsqueda facilitara al usuario administrador encontrar los datos necesarios ya sean usuario, situaciones o valores para realizar los cambios correspondientes.

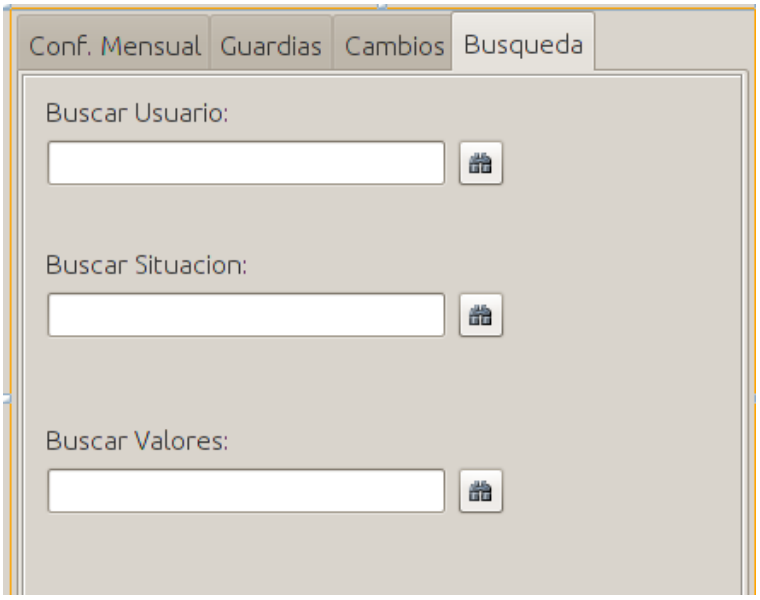


Fig. 107 Ventana para búsqueda

Como se ha venido mencionando a lo largo de los anteriores capítulos, el sistema de control de rancho SISRAN 1.0. Realiza un pre llenado de las confrontas diarias durante todo el mes, esto se lo realiza con el siguiente método:

```
private void llenarConfronta(){
    Date fecha= fechalerDiaMes();
    java.sql.Date d = new java.sql.Date(fecha.getTime());
    Confronta[] con = obtenerConfrontasXfecha(d.toString());
    SituacionUs finSem = em.find(SituacionUs.class, 9);
    int k = this.jListVal.getSelectedIndex();
    if(con.length == 0){
        if(k>=0){
           for(int j = 0; j<diasMes(); j++){
                for(int i=0; i<=lUsRan.size()-1; i++){
                    Confronta c = new Confronta(
                            fecha, null, lVal.get(k), lUsRan.get(i),
                            lUsRan.get(i).getSituacionUs());
                    if(esFinSemana(fecha))
                        c.setSituacionUs(finSem);
                    em.getTransaction().begin();
                    em.persist(c);
                    em.getTransaction().commit();
                }
                fecha.setDate(fecha.getDate()+1);
            }
           msg.showMessageDialog(null, "La confronta mensual "
                   + "fue pre-llenada correctamente.");
        } else
            msg.showMessageDialog(null, "Por favor seleccione "
                    + "un elemento de la columna valores que se "
                    + "usara para llenar la confronta.");
    } else
        msg.showMessageDialog(null, "El mes seleccionado ya fue pre-llenado.");
}
```

También tenemos un método para consultar las confrontas ya guardadas.

```
public Confronta[] obtenerConfrontasXfecha(String Xfecha) {
    Confronta confrontas[] = new Confronta[0];
    try {
        Query q = em.createQuery("select C from Confronta as C where C.fecha = '" + Xfecha +"'");
        confrontas = (Confronta[]) q.getResultList().toArray(confrontas);
    } catch (Exception e) {
        System.err.print(e.getMessage());
        em.getTransaction().rollback();
    } finally {
        return confrontas;
    }
}
```

Además de esto existen métodos para conocer el número de días del mes, y cuál es el primer día del mes.

```
private int diasMes(){
    Calendar cal = Calendar.getInstance();
    int dias = cal.getActualMaximum(Calendar.DAY_OF_MONTH);
    return dias;
}

private Date fechalerDiaMes(){
   Date f = new Date();
   Date fecha=new Date(f.getYear(), f.getMonth(), 1);
   return fecha;
}
```

El siguiente método nos permite conocer si una fecha es fin de semana.

```
private Boolean esFinSemana(Date fecha){
   Boolean fs = false;
   String dia = fecha.toString();
   String d = dia.substring(0, 3);
   if(d.equals("Sat") || d.equals("Sun"))
       fs=true;
   return fs;
}
```

El siguiente método tiene un propósito muy importante, ya que nos permite conocer el grupo de guardia actual, si no está registrado ningún grupo de guardia el cálculo se lo realiza a partir del primer grupo de guardia, caso contrario, la fecha actual y eliminara las guardias registradas a partir de la fecha actual, esto nos servirá en el caso de que hayan cambios en los grupos de guardia, es decir si a partir del 20 de enero hay nuevos grupos de guardia, el sistema deberá calcular los nuevos grupos de guardia a partir de esta fecha.

```
private Date fechaGuardia(){
    SituacionUs g = em.find(SituacionUs.class, 4);
    Date d = fechalerDiaMes();
    qq = em.createQuery("SELECT c FROM Confronta c WHERE c.fecha = :fecha "
            + "AND c.situacionUs = :situacionUs");
    qq.setParameter("fecha", d);
    qq.setParameter("situacionUs", g);
    cc = qq.getResultList();
    if(cc.size()>0){
        d = new Date();
        for(int i=d.getDay(); i<=diasMes(); i++){
            qq.setParameter("fecha", d);
            cc = qq.getResultList();
            for(int j=0; j<cc.size(); j++){
                usR = em.find(UsuarioRan.class, cc.get(j).getUsuarioRan().getIdUsuario());
                cc.get(j).setSituacionUs(usR.getSituacionUs());
            }
            d.setDate(d.getDate()+1);
        }
        d = new Date();
    }
    return d;
}
```

En el siguiente método tenemos calculamos las guardias de cada usuario de rancho, en todo el mes.

```
private void btnCalcGuardiasActionPerformed(java.awt.event.ActionEvent evt) {
    Date fecha = fechaGuardia();
    int guardia = Integer.parseInt(jcbGuardInic.getSelectedItem().toString());
    int nGrupos = Integer.parseInt(jcbGuardia.getSelectedItem().toString());
    SituacionUs su = em.find(SituacionUs.class, 4);
    int k = this.jListVal.getSelectedIndex();
    if(k>-1){
        Valores val = lVal.get(k);

        for(int dm = 1; dm<=diasMes(); dm++){
            String d = new java.sql.Date(fecha.getTime()).toString();
            String g = String.valueOf(guardia);
             Query q = em.createQuery("SELECT c FROM Confronta as c WHERE"
                     + " c.usuarioRan.grupoDeGuardia = "+ g +" "
                     + "and c.fecha = '" + d +"'");
```

```
            List<Confronta> conf = q.getResultList();
            for(int i=0; i<conf.size(); i++){
                em.getTransaction().begin();
                conf.get(i).setSituacionUs(su);
                conf.get(i).setValores(val);
                em.getTransaction().commit();
            }
            if(guardia<nGrupos)
                guardia++;
            else
                guardia=1;
            fecha.setDate(fecha.getDate()+1);
        }
        msg.showMessageDialog(null, "Los cambios se han realizado con exito.");
    } else
        msg.showMessageDialog(null, "Por favor seleccione los valores "
                + "para realizar los calculos.");
}
```

El siguiente método nos permite guardar los cambios todos los cambios suscitados en todo mes, registrados por el administrador del sistema de control de rancho.

```
    private void btnGuardarCambiosActionPerformed(java.awt.event.ActionEvent evt) {
        try{
            int i = this.jListUsRan.getSelectedIndex();
            int j = this.jListSitUs.getSelectedIndex();
            int k = this.jListVal.getSelectedIndex();

            Date d1 = this.jFechaIn.getDate();
            Date d2 = this.jFechaFin.getDate();

            UsuarioRan us = lUsRan.get(i);
            SituacionUs sit = lSitUs.get(j);
            Valores val = lVal.get(k);

            String razon = msg.showInputDialog("Por favor indique la razon "
                    + "por la cual realiza el presente cambio");

            for(Date d = d1; d.getTime()<=d2.getTime(); d.setDate(d.getDate()+1)){
                java.sql.Date f = new java.sql.Date(d.getTime());
                Query q = em.createQuery("select C from Confronta as C where "
                        + "C.fecha = '" + f.toString()
                        +"' and C.usuarioRan.idUsuario = '"+ us.getIdUsuario() +"'");
```

```
                em.getTransaction().begin();
                Confronta c = (Confronta) q.getSingleResult();
                c.setSituacionUs(sit);
                c.setValores(val);
                c.setRazonModif(razon);
                em.getTransaction().commit();
            }
                msg.showMessageDialog(null, "Cambio realizado correctamente");

        } catch (Exception ex) {
            System.err.print(ex);
            msg.showMessageDialog(null, "Por favor verifique que los campos "
                    + "esten seleccionado correctamente.");
        }

    }
```

Finalmente para la clase confronta tenemos los métodos de búsqueda de usuarios, situaciones y valores. Como estos métodos son similares, solamente vamos a mostrar el método buscar usuarios.

```
private void BtnBuscarUsActionPerformed(java.awt.event.ActionEvent evt) {
    String ap = this.txtBuscarUs.getText();
    Query q = em.createQuery("SELECT u FROM UsuarioRan u WHERE u.apellidos like :apellidos");
    q.setParameter("apellidos", ap + "%");
    List<UsuarioRan> list = q.getResultList();
    for(int i=0; i<list.size(); i++){
        for(int j=0; j<lUsRan.size(); j++){
            if(list.get(i).getIdUsuario().equals(lUsRan.get(j).getIdUsuario())){
                this.jListUsRan.setSelectedIndex(j);
                int r = msg.showConfirmDialog(null, "Buscar Siguiente ...");
                if(r!=0){
                    j=lUsRan.size();
                    i=list.size();
                }
            }
        }
    }
}
```

6.8.8 IntFrmCambioGuardias.java

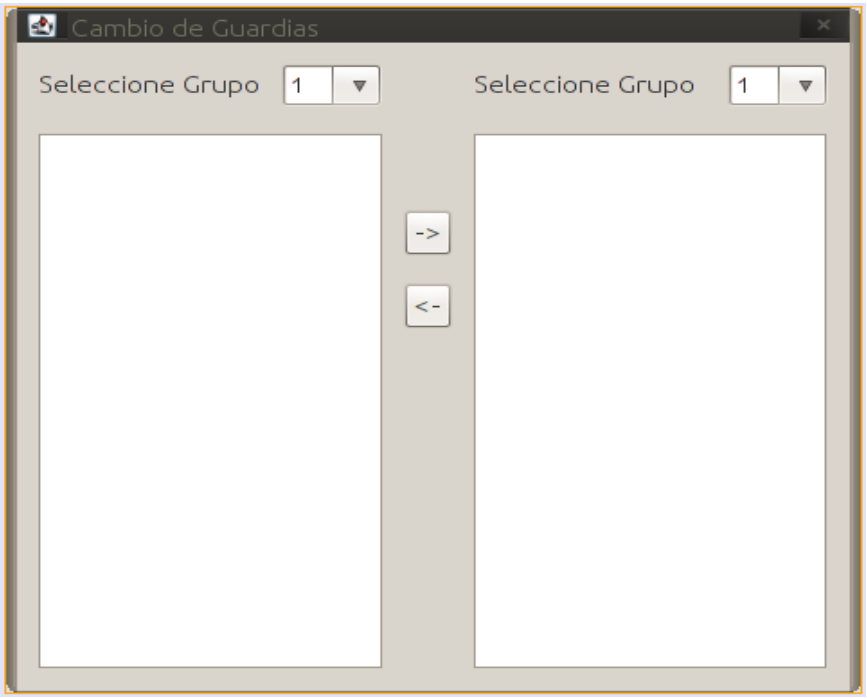


Fig. 108 Formulario Cambio de guardias

En esta clase el usuario administrador podrá realizar la modificación de los grupos de guardia de una forma fácil y rápida.

En los siguientes métodos se cargaran automáticamente al seleccionar el grupo de guardia, el listado de personal que le corresponde a ese grupo.

```
private void cbGuard1ItemStateChanged(java.awt.event.ItemEvent evt) {
    q1.setParameter("grupoDeGuardia", Integer.parseInt(
            this.cbGuard1.getSelectedItem().toString()));
    actualizar();
}

private void cbGuard2ItemStateChanged(java.awt.event.ItemEvent evt) {
    q2.setParameter("grupoDeGuardia", Integer.parseInt(
            this.cbGuard2.getSelectedItem().toString()));
    actualizar();
}
```

El siguiente método permite actualizar los listados de acuerdo al parámetro dado, es decir de acuerdo al grupo de guardia seleccionado por el usuario administrador.

```
private void actualizar(){
   list1 = q1.getResultList();
   list2 = q2.getResultList();
   JListBinding j1 = SwingBindings.createJListBinding(
           UpdateStrategy.READ, list1, listG1);
   JListBinding j2 = SwingBindings.createJListBinding(
           UpdateStrategy.READ, list2, listG2);
   j1.setDetailBinding(ELProperty.create("${apellidos} ${nombres}"));
   j2.setDetailBinding(ELProperty.create("${apellidos} ${nombres}"));
   bindingGroup.addBinding(j1);
   bindingGroup.addBinding(j2);
   bindingGroup.bind();
}
```

Estos dos métodos se encargan de cambiar de grupo de guardia al usuario seleccionado.

```
private void btnAG2ActionPerformed(java.awt.event.ActionEvent evt) {
    int g = Integer.parseInt(this.cbGuard2.getSelectedItem().toString());
    int i = listG1.getSelectedIndex();
    if(i>=0){
        em.getTransaction().begin();
        list1.get(i).setGrupoDeGuardia(g);
        em.getTransaction().commit();
        actualizar();
    }
}

private void btnAG1ActionPerformed(java.awt.event.ActionEvent evt) {
    int g = Integer.parseInt(this.cbGuard1.getSelectedItem().toString());
    msg.showMessageDialog(null, g);
    int i = listG2.getSelectedIndex();
    if(i>=0){
        em.getTransaction().begin();
        list2.get(i).setGrupoDeGuardia(g);
        em.getTransaction().commit();
        actualizar();
    }
}
```

6.8.9 IntFrmCambioSituaciones.java

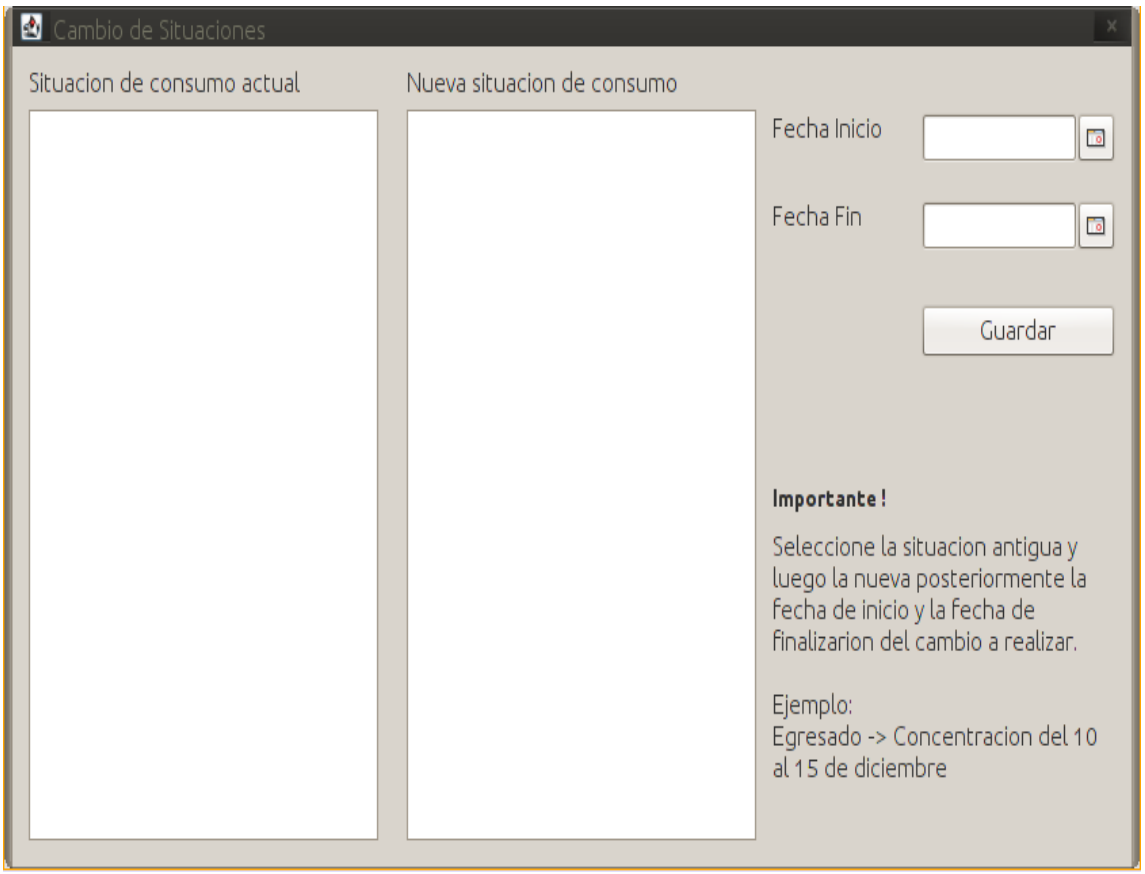


Fig. 109 Formulario Cambio de situaciones

En esta clase el usuario administrador podrá realizar cambios, agrupando al personal por situaciones, como por ejemplo el grupo de Egresados o el grupo que se encuentra de semana.

El siguiente método es el encargado de realizar los cambios, cabe resaltar que estos cambios se los realiza por periodos de tiempo, para que luego de este tiempo el usuario vuelva a tener su situación por defecto.

Esto se lo puede utilizar en casos emergentes como por ejemplo que los egresados sean concentradores un fin de semana.

```
private void btnGuardarActionPerformed(java.awt.event.ActionEvent evt) {
    int s1 = this.jListS1.getSelectedIndex();
    int s2 = this.jListS2.getSelectedIndex();
    Date fIni = this.dcFechaIni.getDate();
    Date fFin = this.dcFechaFin.getDate();
    if(s1>-1 && s2>-1 && fIni!=null && fFin!=null){
        SituacionUs sitUs1 = UsList.get(s1);
        SituacionUs sitUs2 = UsList.get(s2);
        for(Date f=fIni; f.getTime()<=fFin.getTime(); f.setDate(f.getDate()+1)){
            q = em.createQuery("SELECT c FROM Confronta c WHERE c.fecha = :fecha "
                    + "AND c.situacionUs = :situacionUs");
            q.setParameter("fecha", f);
            q.setParameter("situacionUs", sitUs1);
            c = q.getResultList();
            for(int i=0; i<c.size(); i++){
                em.getTransaction().begin();
                c.get(i).setSituacionUs(sitUs2);
                em.getTransaction().commit();
            }
        }
        msg.showMessageDialog(null, "Los cambios se realizaron con exito");
    }
}
```

6.8.10 IntFrmReporteMensual.java

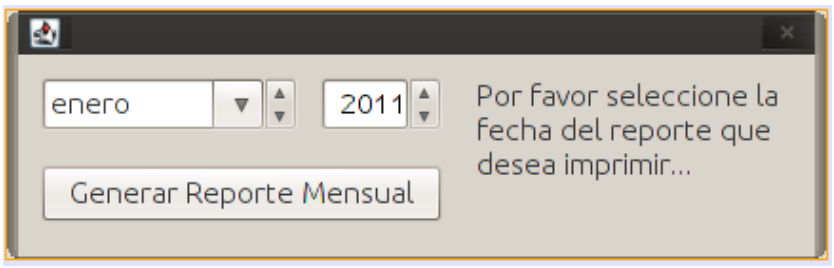


Fig. 110 Formulario reporte mensual

Con esta clase el usuario administrador podrá imprimir la tabla de descuentos de rancho de todo el mes y de cualquier mes que desee, es decir como la información se encuentra en la base de datos, simplemente tendrá que seleccionar el mes y el año que desea imprimir.

Para ello el siguiente método realiza un llenado de toda la información del mes en una tabla temporal para que posteriormente esta información sea tomada por el iReport y se pueda imprimir dicha información.

```
public void reporteMensual(){
    borrarReporte();
    int aa=this.jAnio.getYear() - 1900;
    int mm=this.jMes.getMonth();
    int nDias = diasMes(aa,mm,1);
    Date fecha = new Date(aa, mm, nDias);

    for(int d=nDias; d>=1; d--){

        q = em.createQuery("SELECT c FROM Confronta c WHERE c.fecha = :fecha");
        q.setParameter("fecha", fecha);

        Confronta[] c = new Confronta[0];
        c = (Confronta[]) q.getResultList().toArray(c);

        for(int i=0; i<c.length; i++){
            em.getTransaction().begin();
            ReporteTemp r = null;
            if(d==nDias){

            des   = c[i].getSituacionUs().getDesayuno();
            alm   = c[i].getSituacionUs().getAlmuerzo();
            mer   = c[i].getSituacionUs().getMerienda();
            vDes  = c[i].getValores().getValdesayuno() * des;
            vAlm  = c[i].getValores().getValalmuerzo() * alm;
            vMer  = c[i].getValores().getValmerienda() * mer;
            total = vDes + vAlm + vMer;
            r = new ReporteTemp();
            em.persist(r);
            r.setCedula(c[i].getUsuarioRan().getIdUsuario());
            r.setApellidos(c[i].getUsuarioRan().getApellidos());
            r.setNombres(c[i].getUsuarioRan().getNombres());
            r.setSituacion(c[i].getSituacionUs().getNomSituacion());
            r.setDesayuno(des);
            r.setAlmuerzo(alm);
            r.setMerienda(mer);
            r.setValDesayuno(vDes);
            r.setValAlmuerzo(vAlm);
            r.setValMerienda(vMer);
            r.setTotal(total);
            r.setFecha(fecha);
        } else {
```

```
                r = em.find(ReporteTemp.class, c[i].getUsuarioRan().getIdUsuario());

                des   = c[i].getSituacionUs().getDesayuno() + r.getDesayuno();
                alm   = c[i].getSituacionUs().getAlmuerzo() + r.getAlmuerzo();
                mer   = c[i].getSituacionUs().getMerienda() + r.getMerienda();
                vDes  = c[i].getValores().getValdesayuno() * des;
                vAlm  = c[i].getValores().getValalmuerzo() * alm;
                vMer  = c[i].getValores().getValmerienda() * mer;
                total = vDes + vAlm + vMer;

                r.setDesayuno(des);
                r.setAlmuerzo(alm);
                r.setMerienda(mer);
                r.setValDesayuno(vDes);
                r.setValAlmuerzo(vAlm);
                r.setValMerienda(vMer);
                r.setTotal(total);
            }

            em.getTransaction().commit();
        }
        fecha.setDate(fecha.getDate()-1);
    }
    Reporte.abreReporte();
}
```

Los siguientes métodos se utilizan para borrar la tabla creada anteriormente, permitiendo un nuevo llenado de la misma con datos actualizados. También tenemos el método que permite obtener el número de días del mes, esto es muy importante ya que permite delimitar el informe desde la fecha del primer día de mes hasta la del último día.

```
public void borrarReporte(){
     try{
        Connection conn= DriverManager.getConnection(
                "jdbc:postgresql://localhost:5432/sisrandb","postgres","espel");
        Statement stm = conn.createStatement();
        stm.execute("delete from reporte_temp");
     }catch (Exception ex){msg.showMessageDialog(null, ex.getMessage());}
}

private int diasMes(int aa, int mm, int dd){
     Calendar cal = Calendar.getInstance();
     cal.set(aa, mm, dd);
     int dias = cal.getActualMaximum(cal.DAY_OF_MONTH);
     return dias;
 }
```

6.8.11 IntFrmReportesDiarios.java

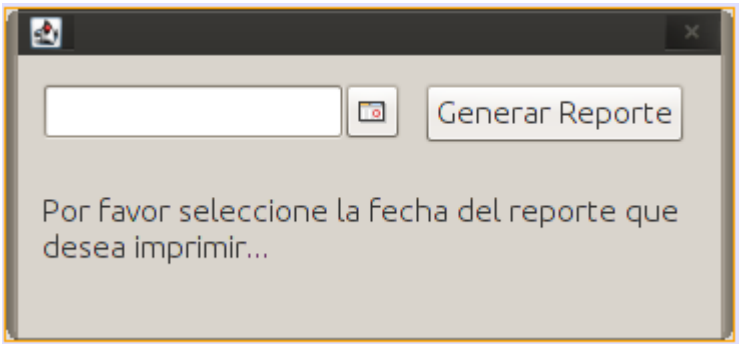


Fig. 111 Formulario Reportes diarios

Esta clase es muy similar a la anterior, con la diferencia que el usuario administrado imprime los consumos diariamente, es decir son las confrontas que se emiten desde la oficina de personal hasta el encargado del rancho, con la finalidad de preparar el número requerido de alimentos, evitando los excesos y la falta de los mismos.

```
public void confrontaDiaria(){
    borrarReporte();

    Date fecha = this.jFecha.getDate();
    q = em.createQuery("SELECT c FROM Confronta c WHERE c.fecha = :fecha");
    q.setParameter("fecha", fecha);

    Confronta[] c = new Confronta[0];
    c = (Confronta[]) q.getResultList().toArray(c);

    for(int i=0; i<c.length; i++){
        des   = c[i].getSituacionUs().getDesayuno();
        alm   = c[i].getSituacionUs().getAlmuerzo();
        mer   = c[i].getSituacionUs().getMerienda();
        vDes  = c[i].getValores().getValdesayuno() * des;
        vAlm  = c[i].getValores().getValalmuerzo() * alm;
        vMer  = c[i].getValores().getValmerienda() * mer;
        total = vDes + vAlm + vMer;
```

```
            em.getTransaction().begin();
            ReporteTemp r = new ReporteTemp();
            em.persist(r);
            r.setCedula(c[i].getUsuarioRan().getIdUsuario());
            r.setApellidos(c[i].getUsuarioRan().getApellidos());
            r.setNombres(c[i].getUsuarioRan().getNombres());
            r.setSituacion(c[i].getSituacionUs().getNomSituacion());
            r.setDesayuno(des);
            r.setAlmuerzo(alm);
            r.setMerienda(mer);
            r.setValDesayuno(vDes);
            r.setValAlmuerzo(vAlm);
            r.setValMerienda(vMer);
            r.setTotal(total);
            r.setFecha(fecha);
            em.getTransaction().commit();
        }
        Reporte.abreReporte();
    }
```

De igual forma que la anterior clase también tiene un método de borrado de la tabla temporal, con la finalidad de eliminar datos antiguos.

```
public void borrarReporte(){
     try{
        Connection conn= DriverManager.getConnection(
                "jdbc:postgresql://localhost:5432/sisrandb","postgres","espel");
        Statement stm = conn.createStatement();
        stm.execute("delete from reporte_temp");
     }catch (Exception ex){msg.showMessageDialog(null, ex.getMessage());}
}
```

6.8.12 IntFrmVerSitConsumo.java

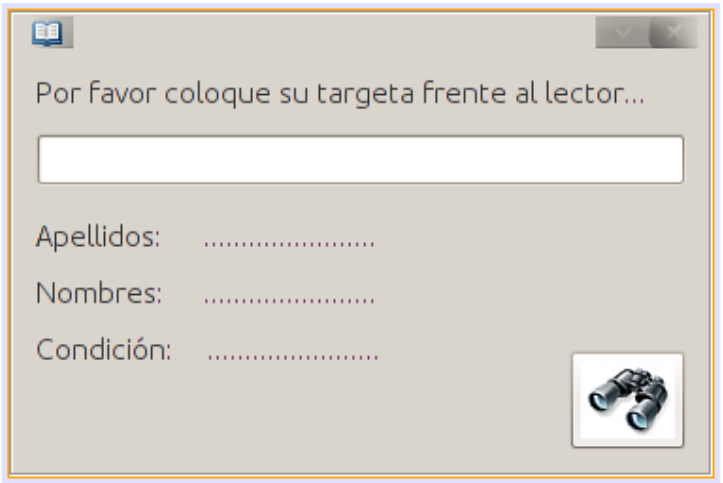


Fig. 112 Formulario Ver situación de consumo

En esta clase es en donde mediante el lector de código de barras, el sistema verifica si el usuario consumidor del rancho está considerando en confronta.

Para ello el siguiente método primero verifica si el usuario esta en confronta y luego verifica el horario.

```
private void btnBuscarActionPerformed(java.awt.event.ActionEvent evt) {
    Date fecha = new Date();
    String idUsuario = this.txtClave.getText();
    Query q = em.createQuery("select c from Confronta c where "
            + "c.fecha = :fecha and c.usuarioRan.idUsuario = :idUsuario");
    q.setParameter("fecha", fecha);
    q.setParameter("idUsuario", idUsuario);
    List<Confronta> con = q.getResultList();
    if(con.size()>0){
        Confronta c = con.get(0);
        this.lblApellidos.setText(c.getUsuarioRan().getApellidos());
        this.lblNombres.setText(c.getUsuarioRan().getNombres());
        this.lblCondicion.setText(verSituacion(c));
    } else {
        this.lblApellidos.setText(".....................");
        this.lblNombres.setText(".....................");
        this.lblCondicion.setText(".....................");
    }
}

private String verSituacion(Confronta c){
    int hora = new Date().getHours();
    int des = c.getSituacionUs().getDesayuno();
    int alm = c.getSituacionUs().getAlmuerzo();
    int mer = c.getSituacionUs().getMerienda();
    String situacion = "El usuario no esta considerado en confronta";

    if( (hora>=6  && hora<=9 &&  des==1) ||
        (hora>=12 && hora<=15 && alm==1) ||
        (hora>=17 && hora<=20 && mer==1)  )
    {
        situacion = "Confronta normal";
        imprimirTicket();
    }

    return situacion;
}

private void imprimirTicket(){
    msg.showMessageDialog(null, "Ticket impreso...");
}
```

6.8.13 Reporte.java

Esta clase es la encargada de realizar el llenado del reporte y de abrir el reportViewer, es decir la vista previa de la impresión del reporte.

```
public class Reporte {

    public static void abreReporte() {
      try{
         JasperReport jr = (JasperReport)JRLoader.
                 loadObject("reporteConfrontas.jasper");
         Connection con = DriverManager.getConnection(
                 "jdbc:postgresql://localhost:5432/sisrandb","postgres","espel");
         JasperPrint jp = JasperFillManager.fillReport(jr, null, con);
         JasperViewer.viewReport(jp, false);
      }catch(Exception ex){ex.printStackTrace();}
   }
}
```

6.8.14 msg.java

Esta clase es en realidad derivada de otra llamada JoptionPane, esto se lo hizo con la finalidad de evitar escribir...

javax.swing.JOptionPane.showMessageDialog(null, "mensaje");

... resumiéndose en:

msg.showMessageDialog(null, "mensaje");

```
package sisran10;

import javax.swing.JOptionPane;

public class msg extends JOptionPane{

}
```

6.9 Ejecución y Pruebas

Durante la ejecución de prueba del programa no se encontraron anomalías especiales, sin embargo será necesario que el usuario de control tenga una capacitación adecuada a fin de evitar errores a la hora de ingresar los datos de los usuarios y de realizar las modificaciones y cálculos de consumo.

CAPÍTULO 7: CONCLUSIONES Y RECOMENDACIONES

7.1 Conclusiones

Al finalizar el presente trabajo de investigación podemos concluir que:

- El uso de software libre utilizado en la elaboración de este proyecto de tesis como: Ubuntu, NetBeans, PostqgeSQL, fue determinante tanto a nivel legal por poseer una GLP, Distribución Open Source, y económico por ser de código abierto, por lo que los costos fueros muy bajos comparados con el software propietario de Microsoft.

- Java es un lenguaje muy potente y robusto, posee una gran cantidad de librerías FREE que facilitan en gran parte el desarrollo de tareas, aunque es un poco más complicado de usar porque es orientado a objetos.

- El Modelamiento de datos mediante el empleo de diagramas UML ayudan a entender de una mejor manera el sistema y posteriormente sirve para codificar más rápido el proyecto

- El Lector Láser de Código de Barras Metrologic MS9520 Voyager es el lector más utilizado en el mercado ya que cuenta con un sistema de lectura avanzado y un moderno diseño.

7.2 Recomendaciones

Al finalizar el presente trabajo de investigación recomendamos que:

- Utilizar software libre para realizar las diferentes aplicaciones ya que a mas de no necesitar licencia, son estables y de fácil manejo.
- Elaborar una lista de todos los requisitos sin omitir detalles, lo que ayuda a la implementación de los mismos y cumplir son los objetivos propuestos.
- Realizar pruebas en cada una de las fases de Desarrollo del proyecto con el fin de comprobar el buen funcionamiento de la aplicación.
- Se use el código de barras dentro de la cadena logística, ya que permite reconocer rápidamente un artículo, información personal o características especiales y así se puede realizar inventarios o listas asociadas de acuerdo a su aplicación.
- Los código de barras para Usuarios consumidores sean anexados al reverso del carnet estudiantil de la ESPE-L para un mejor control.

GLOSARIO

- **GLP.-** Licencia General Pública.
- **Open Sourse.**- Libertad a los usuarios para ejecutar, distribuir, estudiar, cambiar y mejorar el software.
- **UML.**- Lenguague Modeling Unified, Lenguaje de Modelamiento Unificado.
- **JDBC**.- Java Data Base Connectivity, Es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java.
- **JPA**.- Java Persistence API, Persistencia Java.
- **API**.- Aplication Programing Interface, Interfaz de Programación de Aplicaciones.
- **JAVA.-** Lenguaje de programación orientado a objetos. En la actualidad es lenguaje muy extendido y permite realizar diferentes aplicaciones especialmente en el internet y la programación de aplicaciones.
- **FREE SOFTWARE.-** Software Libre, No posee licencia de propietario.

Direcciones Web

Netbeans y Java

- http://es.wikipedia.org/wiki/NetBeans
- www.lawebdelprogramador.com
- http://elvex.ugr.es/decsai/java/pdf/2B-Java.pdf
- http://www.scribd.com/doc/16600726/Guia-Como-Programar-Java-Con-Netbeans-PDF
- http://downloads.sourceforge.net/ireport/iReport-nb-0.9.2.nbm
- www.jarperreports.org
- www.netbeans.org

PostgresQL

- http://laboratorio.is.escuelaing.edu.co/labinfo/doc/Manual_Basico_de_PostgreSQL.pdf
- http://postgresql.uci.cu/system/files/Manual%20del%20usuario%20de%20PostgreSQL.pdf
- www.postgresql.org/downloads

Ubuntu

- www.ubuntu.ec
- www.wikipedia.com

Lector de código de barras

- http://www.monografias.com/trabajos11/yantucod/yantucod.shtml
- http://www.idconsultants.us/

UML

- http://www.dcc.uchile.cl/~psalinas/uml/modelo.html
- http://sel.unsl.edu.ar/licenciatura/ingsoft2/UML-DiagramaClaseObjeto.pdf
- http://es.wikipedia.org/w/index.php?title=caso_de_uso

Referencias Bibliográficas

- Linux Guía de Instalación y administración (Vicente López Camacho)
- Introducción a la programación JAVA (John S. Deam)
- Desarrollo de Base de datos en JAVA (Martin Rinehart)
- Manual de Java (Patrick Naughton)

ESCUELA POLITÉCNICA DEL EJÉRCITO

EXTENSIÓN LATACUNGA

Víctor Joffre Pilco Martínez

C.I. 1716376015

AUTOR

Fidel Oswaldo cruz Almeida

C.I. 0604148965

AUTOR

Ing. José Luis Carrillo

DIRECTOR DE LA CARRERA DE SISTEMAS E INFORMÁTICA

Dr. Rodrigo Vaca

SECRETARIO ACADÉMICO ESPE-L

Latacunga, Marzo del 2011