**INGENIERÍA EN SISTEMAS COMPUTACIONALES**

**INTEGRANTES**

CHRISTIAN GUADALUPE CARRERA SALAS

GUSTAVO ÁNGEL CASTILLO ROJAS

LUIS DAVID HERNÁNDEZ PEREZ

JOSUÉ ISRAEL HUERTA CASTILLO

**MATERIA Y HORA**

INGENIERÍA DE SOFTWARE / 9:00-10:00 HR.

**PROFESORA**

EURI SALGADO ESCOBAR

**MANUAL TÉCNICO DEL SISTEMA DE CONTROL DE EQUIPO DE CÓMPUTO**

**Índice**

[**Introducción** 3](#_Toc43231062)

[**Base de datos** 3](#_Toc43231063)

[**Tablas** 3](#_Toc43231064)

[**Departamentos** 3](#_Toc43231065)

[**JefesDepto** 3](#_Toc43231066)

[**Equipos** 4](#_Toc43231067)

[**Procedimientos almacenados** 4](#_Toc43231068)

[**Sistema** 6](#_Toc43231069)

[**Librerías necesarias** 6](#_Toc43231070)

[**Clases del sistema** 6](#_Toc43231071)

[**ConexiónBD.java** 6](#_Toc43231072)

[**Login.java** 7](#_Toc43231073)

[**Equipos.java** 9](#_Toc43231074)

[**Deptos. java** 16](#_Toc43231075)

[**JefesDepto.java** 19](#_Toc43231076)

[**Reportes.java** 25](#_Toc43231077)

[**Elaboración de reportes** 29](#_Toc43231078)

# **Introducción**

El presente manual muestra el desarrollo del sistema para el control de equipo de cómputo. Se detallan los aspectos técnicos para un correcto entendimiento.

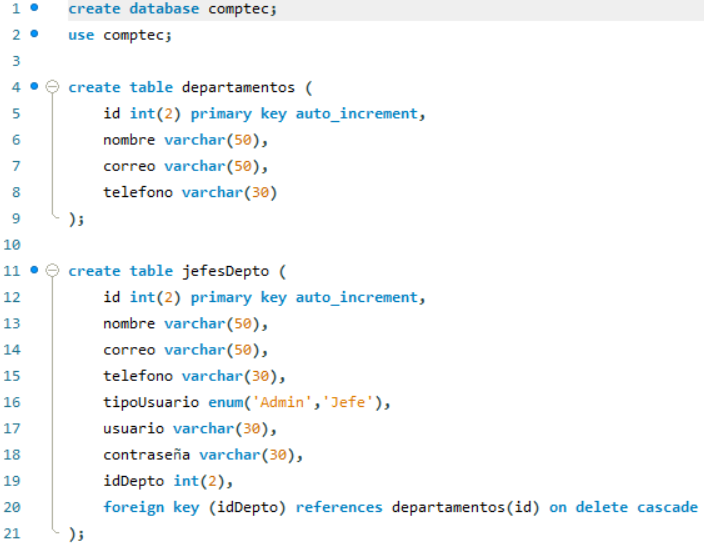
# **Base de datos**

Para poder almacenar toda la información es necesario que nuestro sistema tenga una base de datos. En este caso se trabajó con MySQL ya que es el con el que estamos más familiarizados y es muy sencillo utilizarlo. A continuación, se muestran las tablas y procedimientos almacenados que fueron necesarios para el desarrollo del sistema.

## **Tablas**

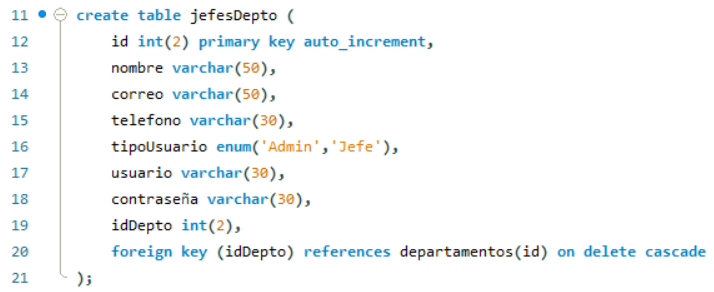
### **Departamentos**

Dentro de esta tabla como su nombre lo indica, se almacenará la información de los distintos departamentos existentes en el Tec. Madero. Podemos ver que se compone de varios atributos como el id, que, en este caso, será nuestra clave primaria y se ha definido como auto incrementable, con lo cual, al momento de realizar alguna inserción ya no será necesario definirlo porque se hará automáticamente. También tenemos otros atributos como el nombre, correo y teléfono.



### **JefesDepto**

Otra de las cosas que se deben almacenar son los jefes de departamento. En esta tabla observamos que se tiene un Id, el cual es nuestra llave primaria y es también auto incrementable, se tienen los datos del jefe como el nombre, correo, teléfono, un tipoUsuario que nos permitirá definir si es un administrador o un jefe, ya que cada uno tiene diferentes permisos en el sistema. De igual forma, se tiene un usuario y contraseña que servirán para poder iniciar sesión, y por último, un idDepto que utilizaremos para referirnos al departamento del que es jefe un determinado usuario. En el atributo idDepto se le añade "on delete cascade”, esto hace que si borramos un departamento también se eliminará la información de su jefe.



### **Equipos**

Si el sistema es para el control de equipo de cómputo, obviamente será necesario almacenar la información de los equipos. Se tienen como atributos, el id, número de inventario, número de serie, modelo, procesador, disco duro, estado, el cual puede ser bueno, regular o defectuoso. Por último, el id del departamento al que pertenece el equipo, también se le agregó “on delete cascade”, por lo cual, si eliminamos un departamento, de igual forma se eliminaran la información de los equipos que están asociados a él.



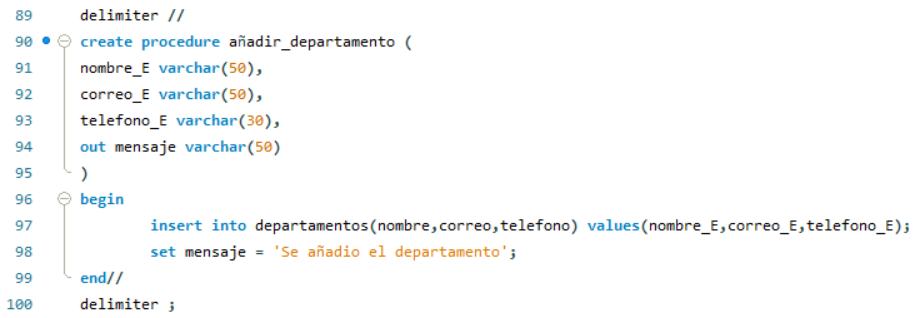
## **Procedimientos almacenados**

Los procedimientos almacenados son como los métodos que utilizamos en los lenguajes de programación. Reciben una serie de parámetros que utilizan para realizar un conjunto de acciones específicas.

Para poder realizar operaciones como mostrar los registros de alguna tabla, añadir, modificar o eliminar registros, desde una aplicación externa al SGBD, es necesario la creación de estos procedimientos almacenados.

En nuestro proyecto hemos definido diversos procedimientos que serán utilizados en las interfaces del sistema, para llevar a cabo las operaciones mencionadas anteriormente. A continuación, se muestran algunos de ellos.

En este procedimiento para añadir departamentos, se reciben los atributos que conforman la tabla de departamentos, a excepción del id, ya que se asigna automáticamente. También vemos que hay un parámetro de salida (out), el cual es un mensaje, el cual nos servirá para mostrar la notificación correspondiente al realizar la operación. Después tenemos el cuerpo del procedimiento, el cual especifica las operaciones que se realizarán.



Este otro procedimiento almacenado es para la modificación de los datos de un jefe de departamento. Se reciben los parámetros que ocuparemos y pasamos a la parte del cuerpo, en la cual podemos ver que se utilizan condicionales if, esto con la finalidad de verificar que el jefe este registrado en el sistema y que el id del departamento sea válido. Si se cumplen estas condiciones, se efectuarán las operaciones, de otra manera, se mostrará el mensaje correspondiente.



Se deben crear los procedimientos de mostrar registros, buscar, insertar, modificar y eliminar para las interfaces de departamentos, jefes de departamento y equipos. También existe otra interfaz que es la de reportes, para ello, ocuparemos procedimientos para generar un reporte de todos los equipos registrados o dependiendo el estado que seleccionemos.

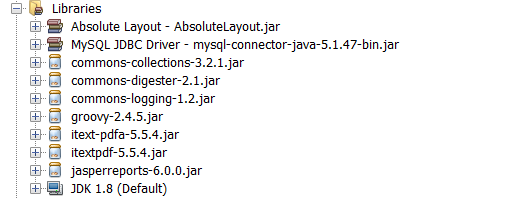
# **Sistema**

Una vez explicada la parte de la base de datos, ahora es el turno de hablar sobre cómo fue el proceso de creación del sistema. Antes que nada, el sistema está programado con Java y se utilizó el IDE NetBeans, ya que es lo que hemos venido manejando durante estos semestres de la carrera y es muy cómodo trabajar con él.

A continuación, se explican las diversas clases que componen el sistema, las librerías que fueron necesarias y la parte de la elaboración de reportes.

## **Librerías necesarias**

Al realizar una aplicación que implemente una base de datos, es necesario añadir el conector, en este caso es el mysql-connector-java-5.1.47-bin.jar, este nos permitirá utilizar instrucciones provenientes de la clase SQL para Java. Las otras librerías que se pueden observar, se ocuparán para la creación de los reportes utilizando el plugin JasperReports. Eso lo veremos más adelante.



## **Clases del sistema**

### **ConexiónBD.java**

Como su nombre lo indica, en esta clase se establece la conexión con la base de datos. Se crea un objeto connection que utilizaremos para asignarle la ruta de nuestra base de datos y que se pueda generar la conexión entre nuestro sistema y la base de datos. La parte de la ruta que dice “root”, “1234”, corresponde al usuario y contraseña de nuestra BD, cada quien podría tener unos diferentes, así que hay que asegurarse de que sea el correcto para evitar fallos.



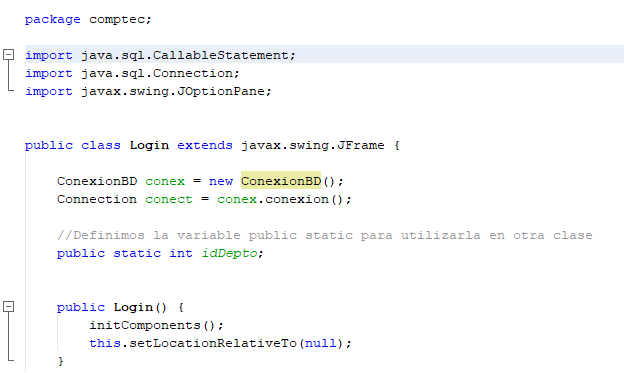
### **Login.java**

Las interfaces del sistema están construidas con JFrames, los cuales tienen una parte que corresponde al diseño de la ventana y otra al código o funcionalidad. Nuestra primera interfaz es la del login o inicio de sesión.

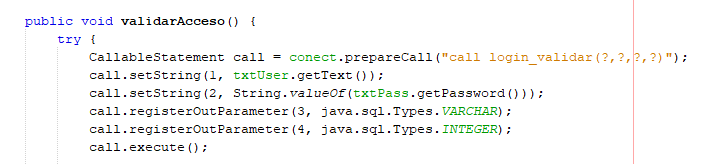
En la parte del diseño tenemos que la interfaz se compone de 2 labels que utilizamos para los textos de usuario y contraseña, 2 TextField para que podamos ingresar los datos y un botón, el cual, al presionarlo mandará llamar al método de validarAcceso() para permitirnos o no, el acceso al sistema.



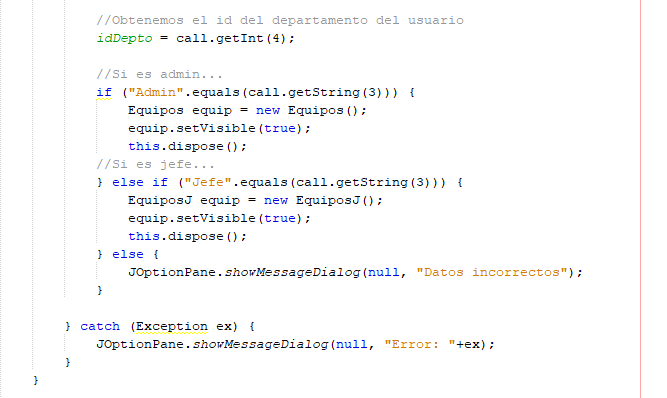
En la parte del código primeramente se declaran los objetos de la conexión que definimos en la clase anterior, además, creamos una variable llamada idDepto que nos servirá para capturar ese atributo del usuario con el que iniciemos sesión y de esta manera poder mostrarle solo lo que tiene permitido ver.



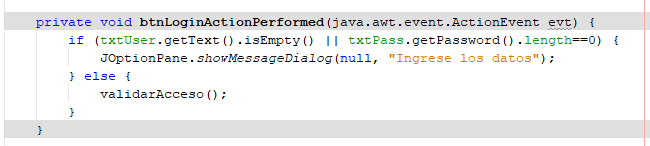
Se creó un método que nos permite validar el acceso al ingresar los datos de algún usuario. En la primera instrucción se crea el objeto call de la clase CallableStatement, con este objeto podremos llamar al procedimiento almacenado correspondiente. Entre comillas se escribe call nombreProcedimiento (?,?,?), esos signos de interrogación corresponden a la cantidad de parámetros tanto de entrada como de salida que ocupe el procedimiento. Después, procedemos a ir registrando los parámetros, en este caso los valores se obtendrán de las 2 cajas de texto de usuario y contraseña. Finalmente, ejecutamos el procedimiento.



Capturamos el idDepto, utilizando la información que nos otorga el 4to parámetro del procedimiento que ejecutamos. Dicho procedimiento al momento de proporcionarle los datos de algún usuario, nos devuelve el tipo de usuario que es y su idDepto. Con el tipo de usuario haremos la validación, si es “Admin” se mostrarán las interfaces donde podemos llevar a cabo operaciones de inserción, modificación o eliminación. Si es “Jefe” se mostraran las interfaces con el mismo diseño que las de administrador pero no será posible realizar las 3 operaciones ya mencionadas.



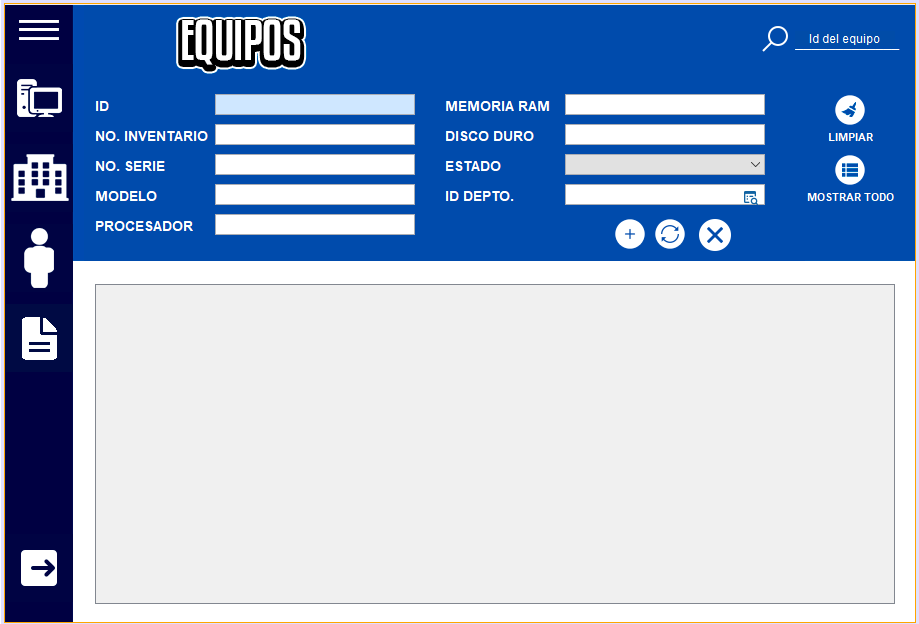
Finalmente, mediante un evento ActionPerformed que le hemos añadido al botón de iniciar sesión, se ejecutará el método de validarAcceso(), siempre y cuando no este vacío alguno de los 2 campos, usuario o contraseña. Si falta algún dato, se nos notificará.



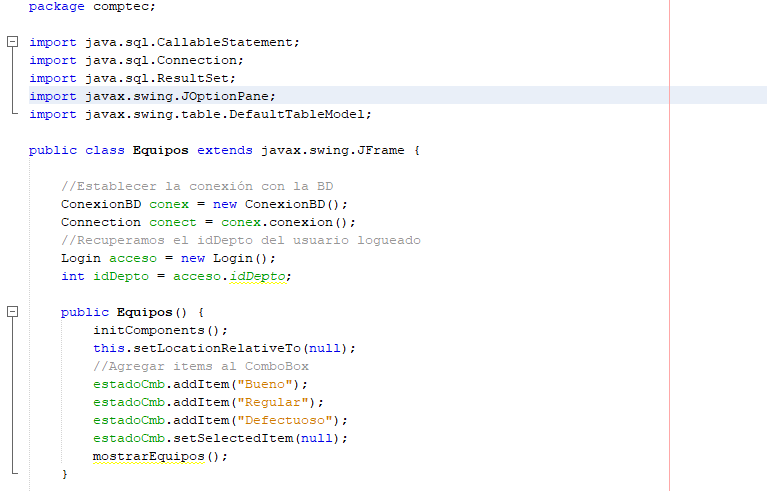
### **Equipos.java**

En la parte del diseño, nuestra interfaz está constituida por los 5 botones en el lado izquierdo, los primero 4 nos permiten el acceso a las diferentes interfaces del sistema, y el 5to nos permite cerrar sesión. En el medio tenemos los diferentes TextField para poder capturar los datos ingrese el usuario, así como 3 botones para las operaciones de inserción, modificación y borrado. Una barra de búsqueda, un botón para limpiar los campos y uno para mostrar todos los registros.

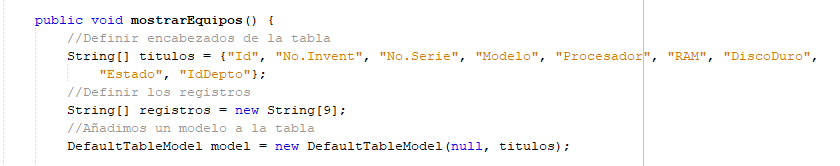
En la parte inferior tenemos una tabla que se llenará con los equipos que estén registrados en el sistema.



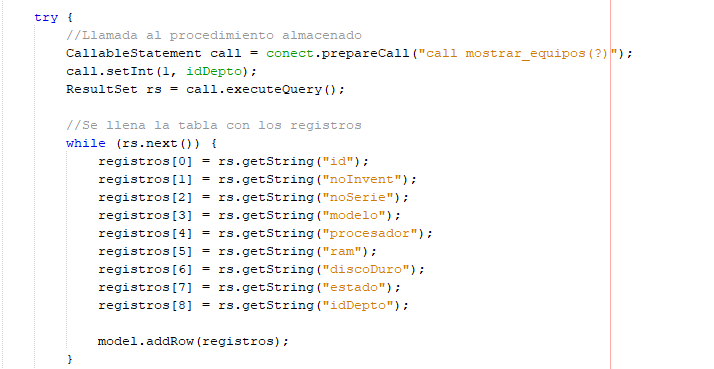
En la parte del código tenemos la creación de la conexión y tenemos una variable idDepto con la cual recuperamos el valor que se le asigno en la clase login al momento de iniciar sesión con un usuario. En el método constructor añadimos las diferentes opciones que contiene el ComboBox que utilizamos para los estados en los que se encuentran los equipos, los cuales como ya se dijo, pueden ser bueno, regular o defectuoso.



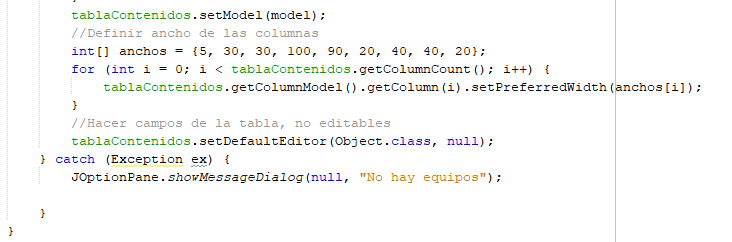
Tenemos un método llamado mostrarEquipos() el cual permite cargar la información de los equipos registrados cada vez que ingresamos a la interfaz. Como mencionamos, los registros se cargan en una tabla, por lo que es necesario prepararla añadiendo los títulos de las columnas.



Mas adelante tenemos el llamado al procedimiento almacenado, al cual le enviaremos como parámetro el idDepto para que se muestre la información que corresponda al usuario que ha iniciado sesión. Se crea un ciclo while con los diferentes atributos de los equipos, para ir llenando nuestra tabla de resultados.



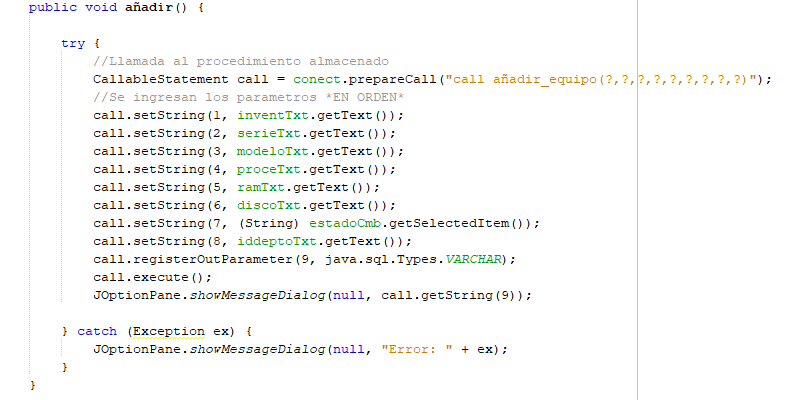
Por último, se ajustan los anchos de las columnas para que se muestren bien los datos. En caso de excepciones, se capturan y se muestra un mensaje.

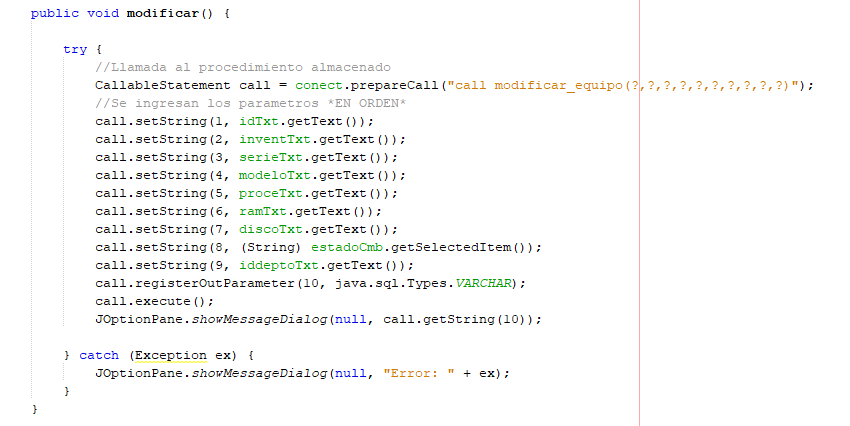


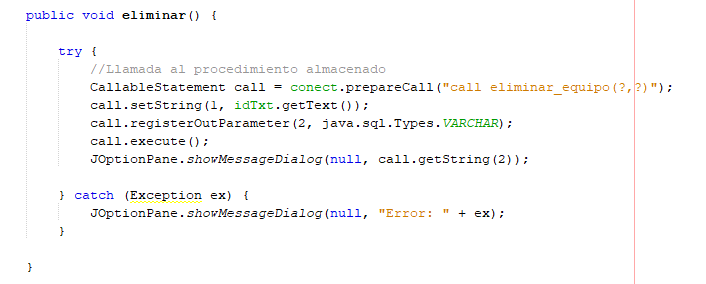
Se definió también un método para realizar la búsqueda de los equipos. Al buscar un equipo se actualizará la tabla mostrando únicamente el equipo en cuestión, por lo que, la funcionalidad de este método es similar al anterior de mostrar equipos, lo único que cambia es el procedimiento al que llamamos.



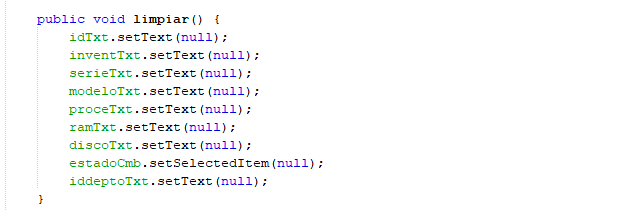
Los siguientes métodos son los de añadir, modificar y eliminar. En cada uno de ellos mandamos llamar al procedimiento necesarios, le pasamos los parámetros que requiera y ejecutamos la llamada. Estos procedimientos almacenados tienen un parámetro de salida, el cual es un mensaje que nos notificará el resultado de la operación, si se realizó con éxito o si surgió algún problema, dicho mensaje aparecerá en una ventana de dialogo.







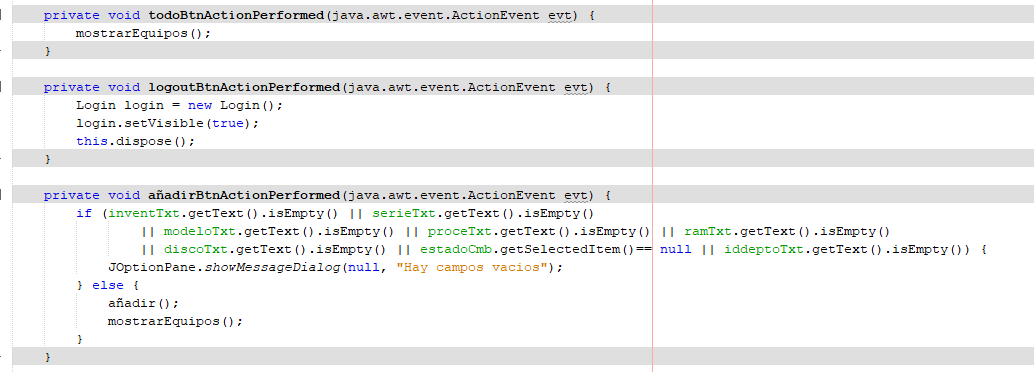
Como se comentó, se tiene una botón para limpiar los campos el cual al presionarlo activa este método llamado limpiar(), lo único que hace es poner como nulo o vacío el contenido de los diferentes TextFields.



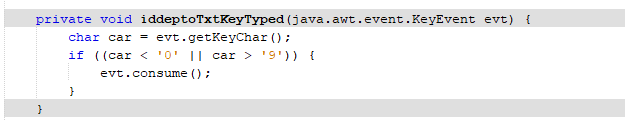
Mas adelante tenemos los eventos actionPerformed de cada botón de la interfaz. Los botones de la izquierda dijimos que nos sirven para entrar a las diferentes ventanas del sistema, por lo cual, debemos crear un objeto de la clase correspondiente para que al presionar dicho botón nos mande a esa interfaz y en la que estamos actualmente, se oculte.



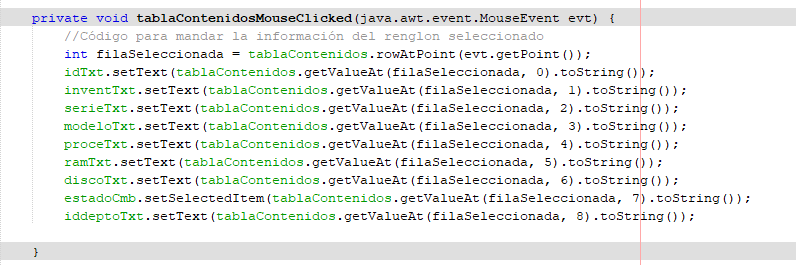
Algunos botones tienen condiciones que se deben cumplir para poder realizar la función que tienen asignadas, de lo contrario nos aparecerá un mensaje de que existe algún problema. Por ejemplo, en el botón para realizar la operación de añadir, necesitamos ninguno de los TextField este vacío, ya que todos son necesarios.



Otras restricciones, como el que solo podamos introducir ciertos caracteres en algún campo, en este en el idDepto, ya que, en la base de datos está definido como entero y si llegamos a introducir una letra, nos saldría un error. Por eso mejor establecemos que solo se puedan introducir los dígitos del 0-9.

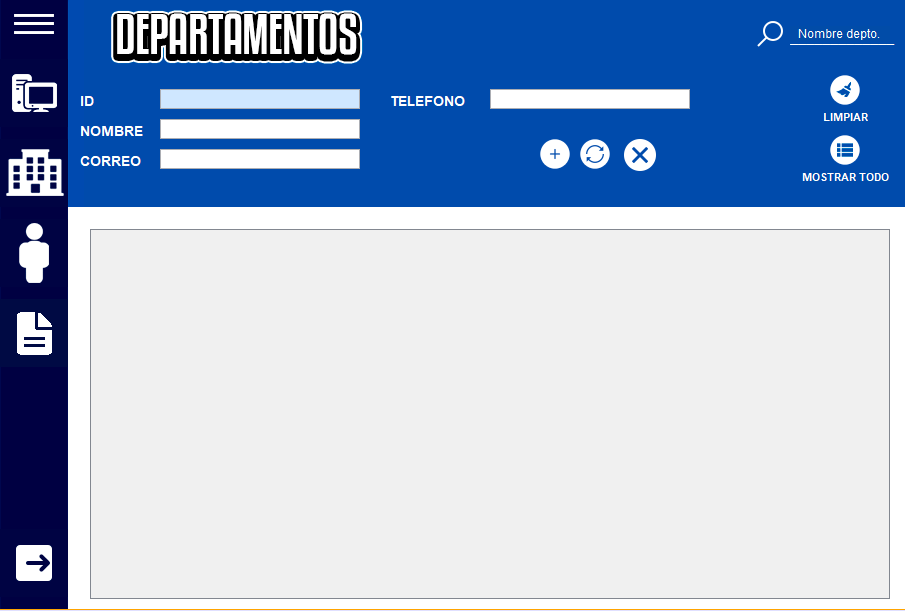


Nuestro sistema tiene la funcionalidad de que, al seleccionar un registro, se carguen sus datos en los TextField de arriba, esto nos facilita las operaciones de modificación y eliminación, puesto que ya no tenemos que estar escribiendo los datos de algún equipo. Esto se logra con el evento de MouseClicked, el cual obtiene el contenido de toda la fila seleccionada y lo envía a los campos de texto.

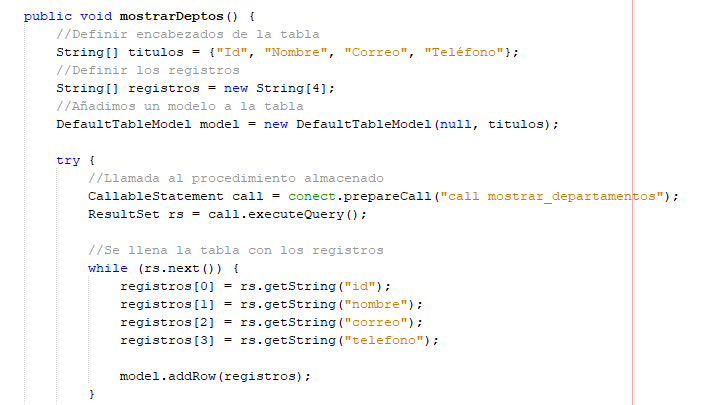


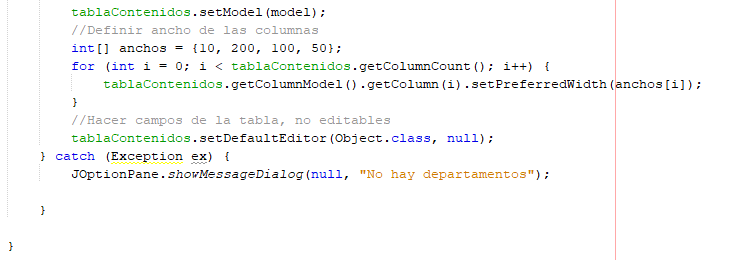
### **Deptos. java**

Establecimos un diseño similar para todas las interfaces, con la finalidad de que los usuarios pueden familiarizarse rápidamente con el sistema y se sientan más cómodos. Esta interfaz de departamentos también cuenta con los accesos a las otras interfaces, tiene campos de texto, botones para las operaciones de inserción, modificación y eliminación, botones para limpiar y mostrar todo, barra de búsqueda y una tabla en donde se muestran todos los departamentos existentes.



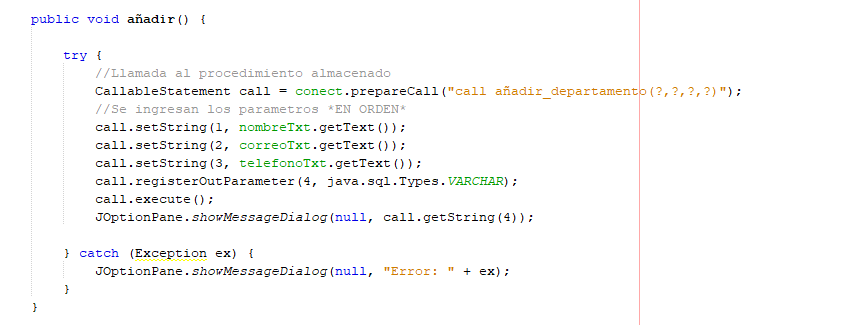
El código para este tipo de aplicaciones puede ser reutilizado casi en su totalidad, por ejemplo, en esta interfaz de departamentos también tenemos un método para mostrar los registros, solo que ahora utilizaremos el procedimiento almacenado de mostrar departamentos, todo lo demás es similar al método que utilizamos en la clase Equipos.java.

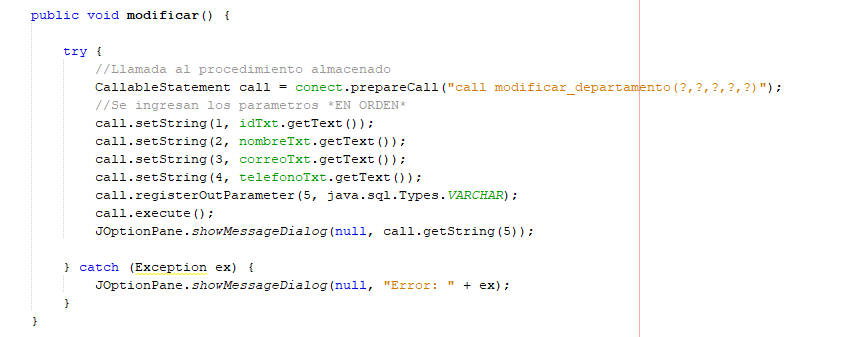


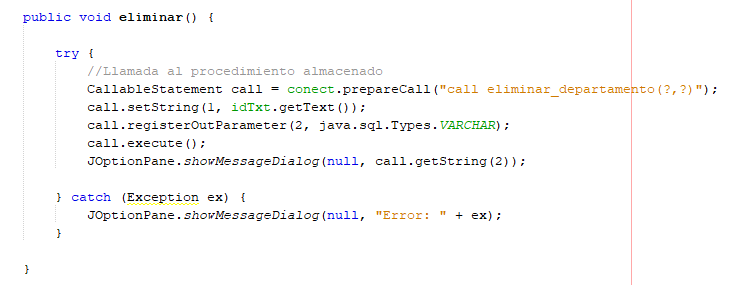


Desde luego también se tienen las funciones para la búsqueda, inserción, modificación y borrado de departamentos. Como podemos observar, solo vamos cambiando el procedimiento al que llamamos y ajustando el tamaño de las columnas de la tabla.

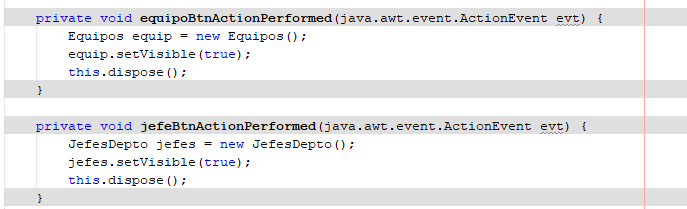




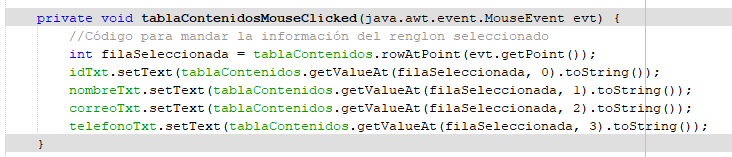




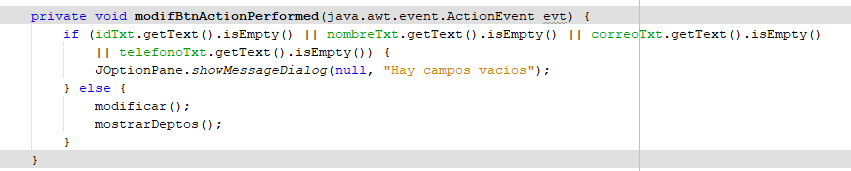
Los accesos a las otras ventanas.



La función para cargar los datos del departamento seleccionado en los campos de texto.



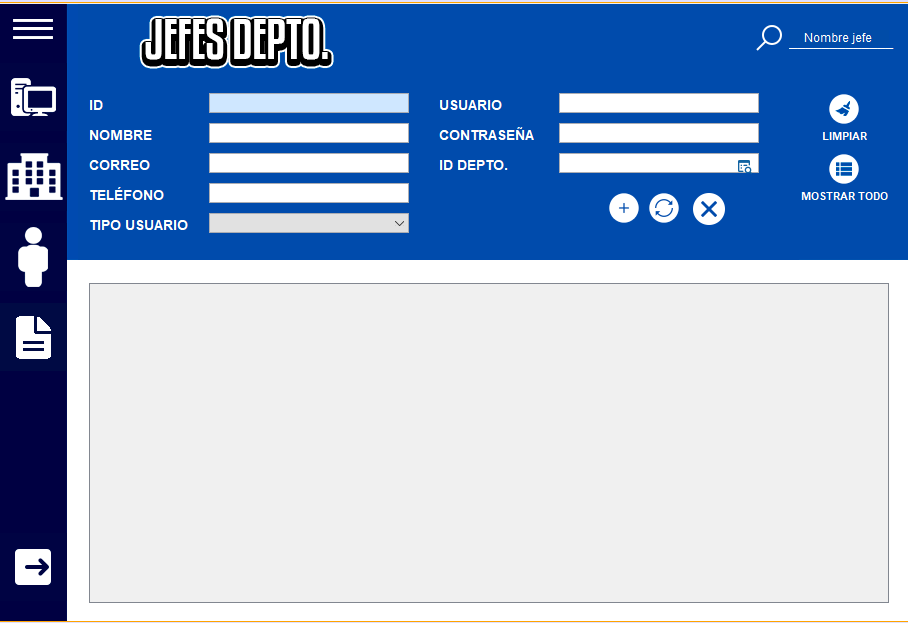
Restricciones en los eventos de botón que se deben cumplir para que pueda llevarse a cabo la funcionalidad que poseen.



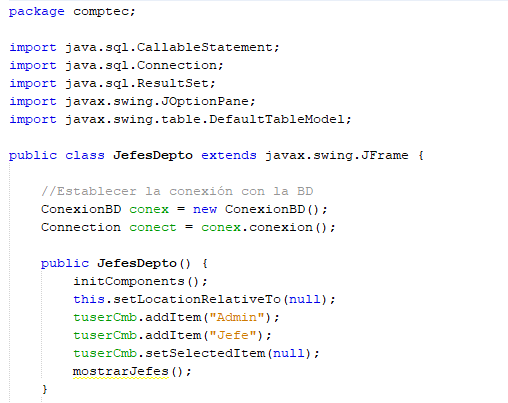
En general, las funciones que contienen las diversas interfaces son muy similares y como se mencionó, eso nos permitió reutilizar código y agilizar el desarrollo del sistema.

### **JefesDepto.java**

La 3ra interfaz del sistema es la de jefes de departamento. El diseño es el siguiente:



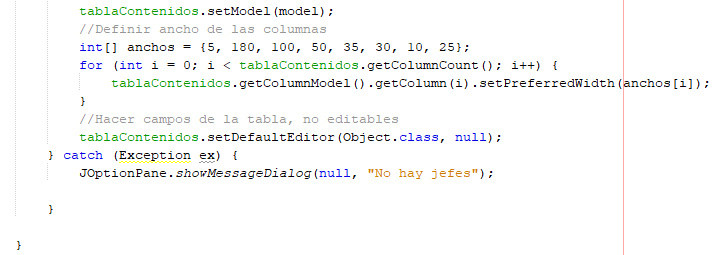
En la parte del código tenemos que primero y como en las demás interfaces, se debe crear el objeto conexión que nos permite realizar las operaciones con la base de datos. También se añaden las opciones al ComboBox de tipoUsuario, las cuales son Admin y Jefe.



Se tiene el primer método el cual es el de mostrarJefes(), en el cual primero definimos los encabezados y después mandamos llamar al procedimiento para que nos entregue los resultado y así poder llenar nuestra tabla de jefes.

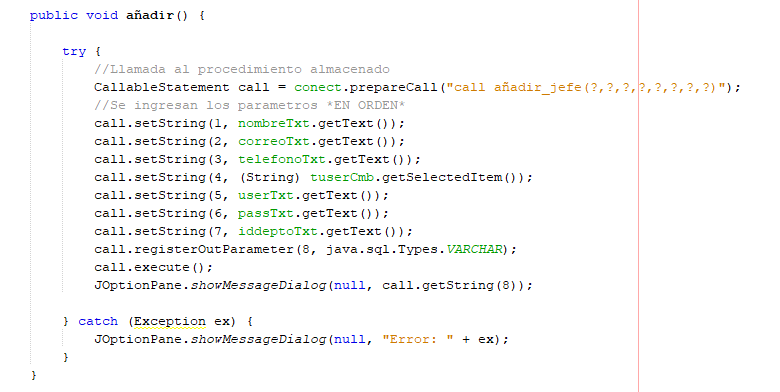


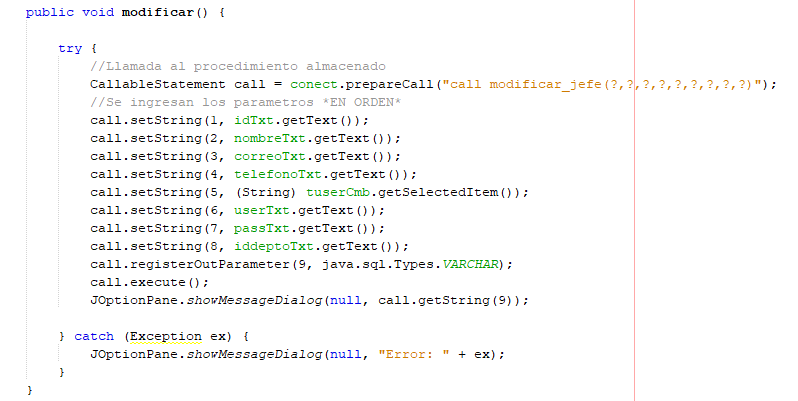
Se ajustan las columnas, ya que cada tabla tiene un número diferente de atributos y algunos son más extensos que otros, por lo cual, hay que elegir el ancho correcto para que se puedan visualizar por completo.

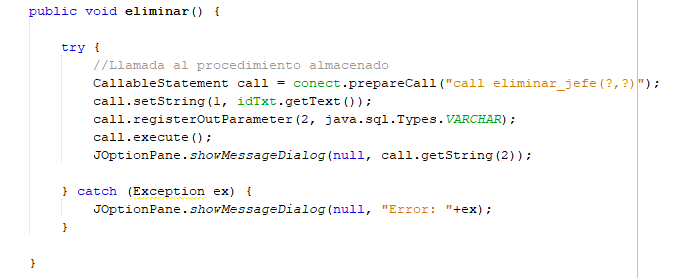


En esta interfaz como vimos en el diseño, también contamos con los botones de buscar, insertar, modificar y eliminar jefes, por lo que será necesario definir las funcionalidades para estos.

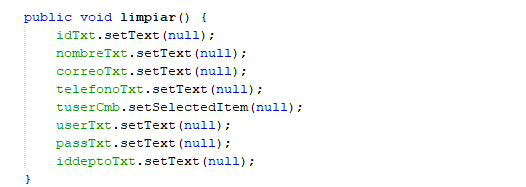




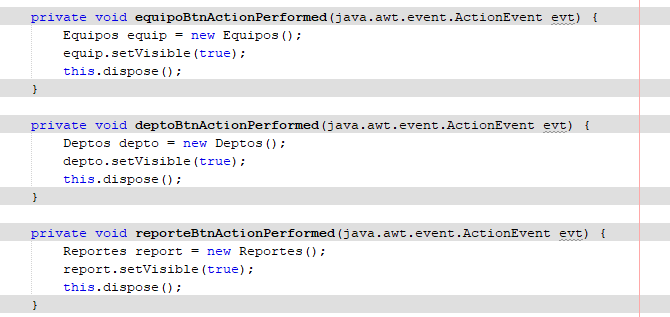




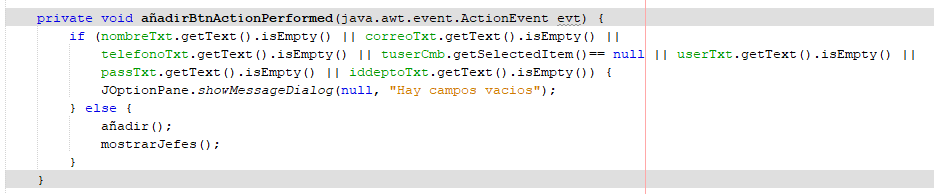
Desde luego, también se cuenta con la función de limpiar campos, de esta manera no tendremos que estarlos borrando uno por uno. Se establecerá su contenido como nulo o vacío.

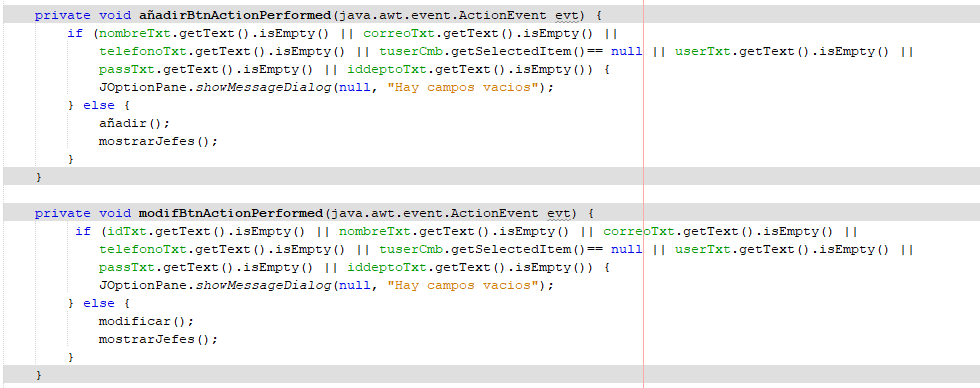


A los botones de acceso a las interfaces, se les asignó un evento actionPerformed que se activa al presionar el botón. Para poder cambiarnos a otra ventana fue necesario crear un objeto de dicha interfaz, hacerlo visible y ocultar en el que nos encontramos al momento de presionar el botón.

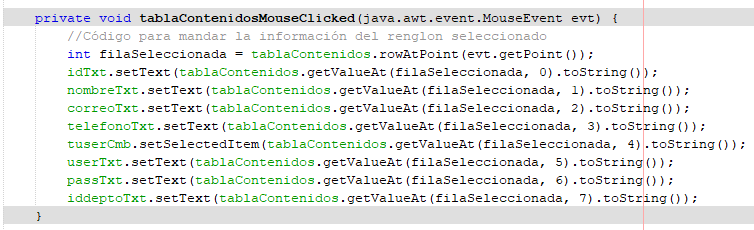


Se añaden condiciones a algunos botones para asegurarnos de que contemos con lo necesario para realizar la operación. De otra manera, no podremos proceder.

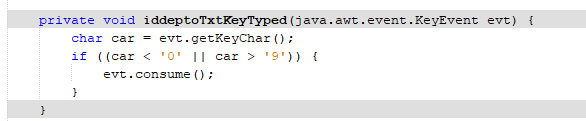




Se le añadió un evento del tipo MouseClicked a la tabla de resultados que nos permitirá que, al seleccionar algún registro, se carguen sus datos en los campos de texto que tenemos en la parte superior de la interfaz, esto nos ayudará a agilizar operaciones como las modificaciones de los datos de un jefe o su eliminación.

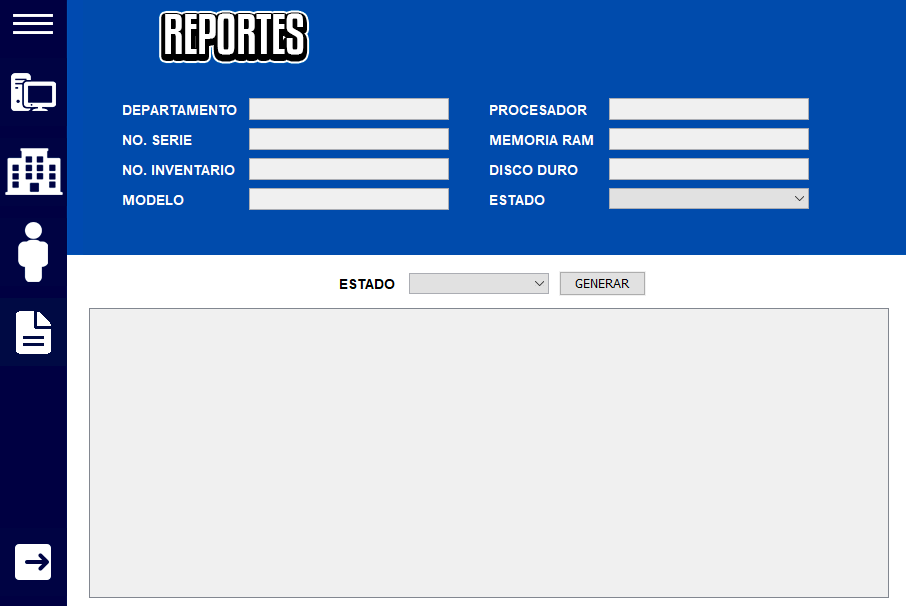


En la base de datos se declararon algunos atributos como enteros, por eso se decidió establecer que solo se puedan introducir números del 0 al 9 para que no puedan surgir errores y el usuario se dé cuenta de que solo se aceptan números.

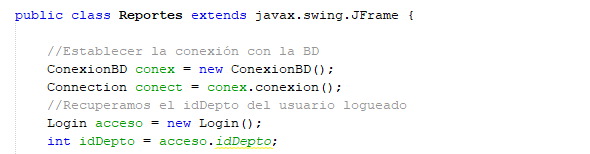


### **Reportes.java**

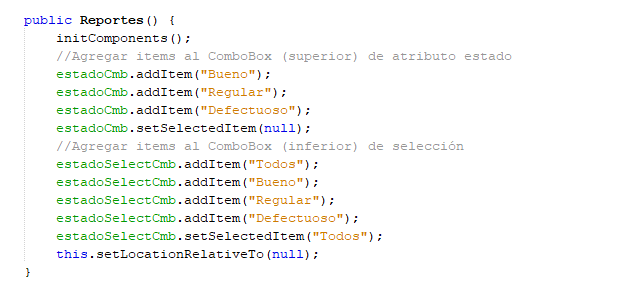
La última interfaz del sistema es la de reportes. Como podemos ver en el diseño, ya no tenemos los botones para insertar, modificar, eliminar y algunos otros más, puesto que, en esta interfaz lo que se hace es recopilar los equipos que están registrados y mediante un comboBox podemos seleccionar un estado para que se nos muestren los equipos correspondientes a esa selección y así poder generar el reporte deseado.



En la parte del código, de igual forma como en todas las demás interfaces, es necesario declarar la conexión además de obtener el idDepto del usuario que ha iniciado sesión para poder mostrar los resultados que correspondan.



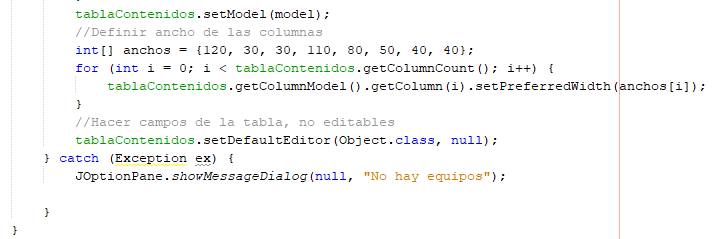
En el método constructor se van agregando las opciones de los ComboBox que utilizamos además de hacer que se centre la ventana con la instrucción setLocationRelativeTo.



El siguiente método corresponde al reporte general. El atributo estado solo puede tener 3 valores: bueno, regular o malo. Nosotros agregamos una opción en el ComboBox para mostrar todos los equipos registrados sin importar su estado, por lo cual, fue necesario crear un procedimiento almacenado para un reporte general y otro para un reporte dado un estado.

Se declaran los encabezados de las columnas y se hace una llamada al procedimiento, enviándole los atributos que pudiera necesitas. Posteriormente, se llena la tabla con los resultados.

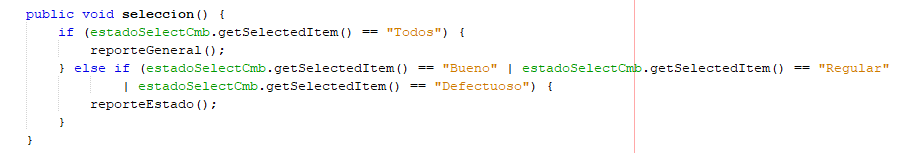




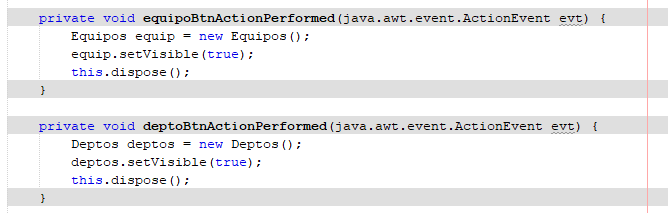
El reporte por estado, en el cual solo se ocupará cambiar el procedimiento que teníamos antes por el de reporte\_estado(?,?). Mediante el ComboBox que tenemos arriba de la tabla le enviaremos el valor del estado que seleccionamos para que nos muestre los resultados. Si no hubiera equipos en ese estado, se nos notificará.



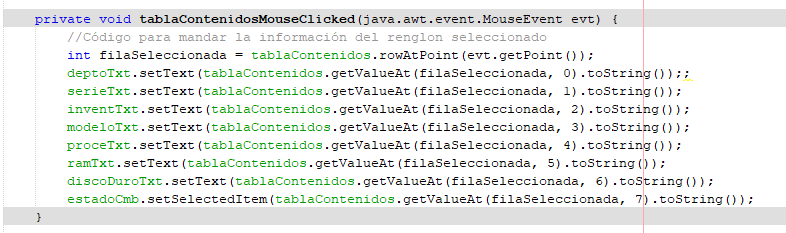
Esta es la funcionalidad que se le asigna al ComboBox. Si se tiene seleccionada la opción de Todos, se mandará llamar al método reporteGeneral(), de otra manera, se aplicará reporteEstado().



Se programan los botones de acceso a las distintas interfaces.

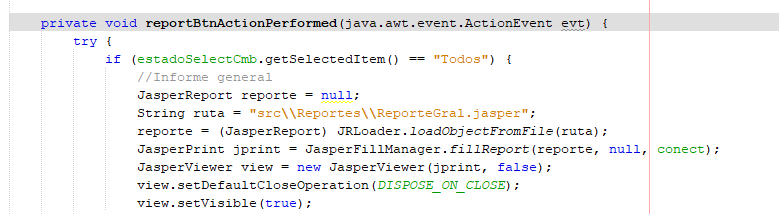


También se añade la funcionalidad de mandar los valores del registro seleccionado a los campos de texto.



A continuación, tenemos la funcionalidad del botón para generar el reporte. Se tiene una condición de que, si el ComboBox tiene seleccionada la opción “Todos”, se abrirá el reporte general. Para los reportes se utilizó el plugin JasperReports, eso se explicará en la siguiente sección.

Se especifica la ruta del reporte y las siguientes instrucciones serán para poder desplegar la ventana que mostrará el reporte generado.



Si el ComboBox tiene alguna de las otras opciones (bueno, regular, defectuoso), se mandará llamar al reporte por estado, dicho reporte se le ha añadido que pida el estado para poder realizar la búsqueda de los equipos que correspondan. En instrucción que dice **parametro.put("status", "'" + estadoSelectCmb.getSelectedItem().toString() + "'");**

Status es el nombre del parámetro que tenemos que enviarle, se ha programado para que lo tome del ComboBox de estado. Una vez que lo haya obtenido, se procede a generar el reporte.



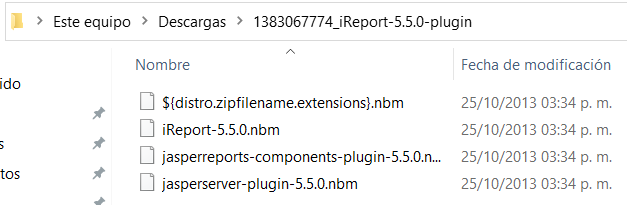
## **Elaboración de reportes**

Para poder crear los reportes hicimos uso de un plugin de NetBeans llamado JasperReports, el cual tiene la habilidad de entregar contenido enriquecido al monitor, a la impresora o a ficheros PDF, HTML, XLS, CSV y XML. En este caso la usaremos para crear nuestra plantilla del reporte.

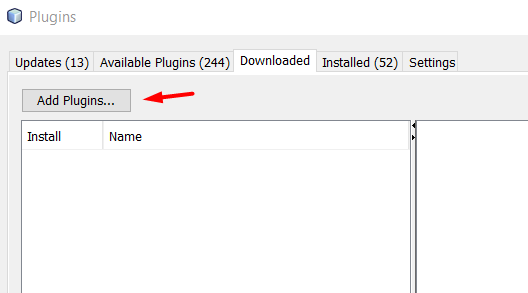
Lo primero que se debe hacer para utilizar esta herramienta, es descargar el plugin, el cual esta disponible en la siguiente dirección:

**http://plugins.netbeans.org/plugin/4425/ireport**

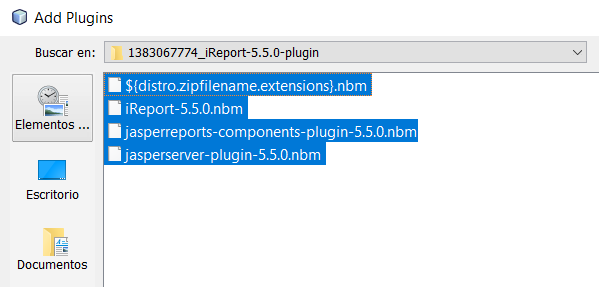
Una vez que lo hayamos descargado y extraído, nos quedara una carpeta con los siguientes archivos:



Los cuales debemos de añadir a nuestro NetBeans. Para eso abrimos el IDE y nos iremos a la pestaña de Herramientas – Plugings – Instalados y le daremos añadir plugins.

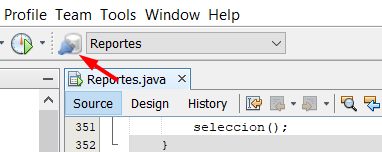
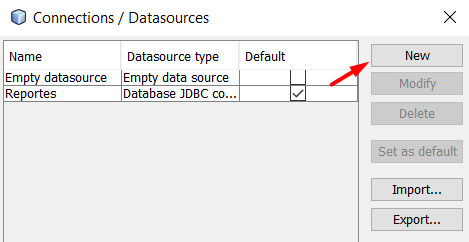


Deberemos ir a la carpeta descargada y seleccionar todos para añadir a nuestro IDE.

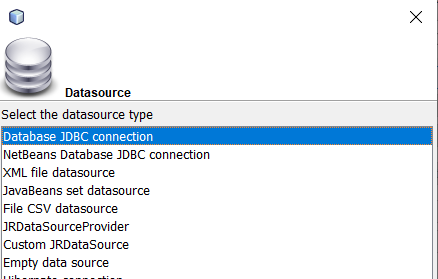


En este caso ya lo tenemos instalado, pero en caso de que no, después de seleccionar le daremos instalar y se abrirán una serie de ventanas como ocurre normalmente al instalar un programa. Finalmente se reiniciará nuestro NetBeans y habrá quedado correctamente instalado.

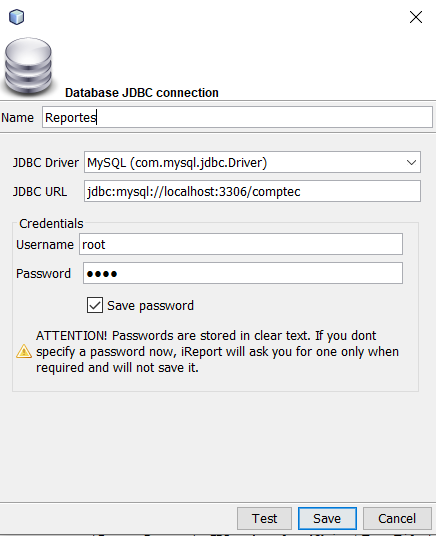
También es necesario crear una conexión con la base de datos (aparte de la establecida en la clase conectarBD.java) para que el plugin de JasperReports tenga acceso a toda la información de nuestra base de datos. Nos vamos al icono de conexión y daremos en New.

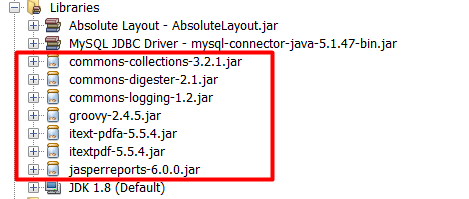
Ahora escogemos la opción de Database JDBC connection.



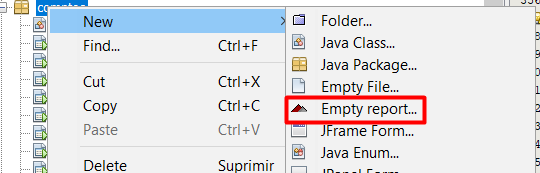
En la siguiente ventana debemos darle un nombre a la conexión, establecer el JDBC Driver al que dice MySQL, definir la dirección de nuestra base de datos, así como el usuario y contraseña que nos permiten el acceso. Le damos test y save.



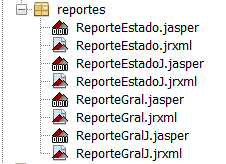
También se añaden las librerías necesarias de JasperReports. Estas vienen incluidas en el repositorio del proyecto.



Ya con estos pasos podremos empezar con la creación de los reportes. Si le damos crear un nuevo archivo en el proyecto, podemos ver que ahora nos sale la opción de crear un reporte vacío de JasperReports.

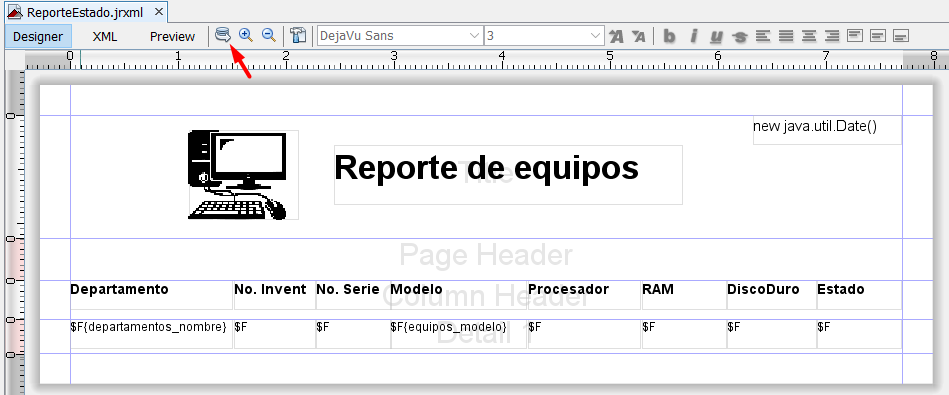


En este caso nosotros hemos creado los siguientes:

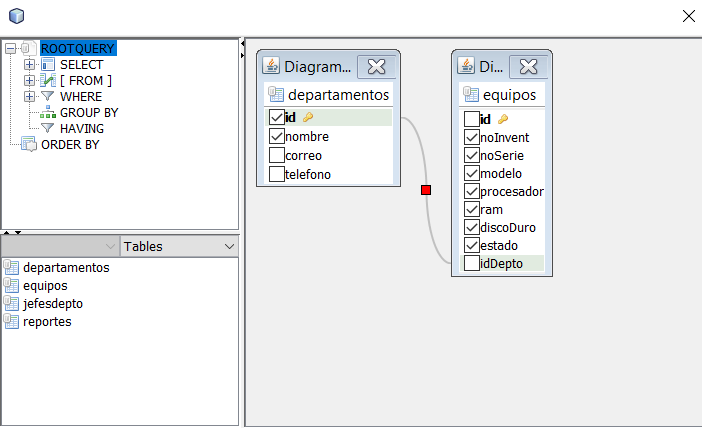


Los que terminan en ‘J’ son los que se generan al entrar como un usuario jefe de departamento, fue necesario crear 2 de cada uno, ya que, en nuestro sistema existe la restricción de que un administrador puede ver todos los equipos existentes en todos los departamentos, pero un jefe solo verá los del suyo.

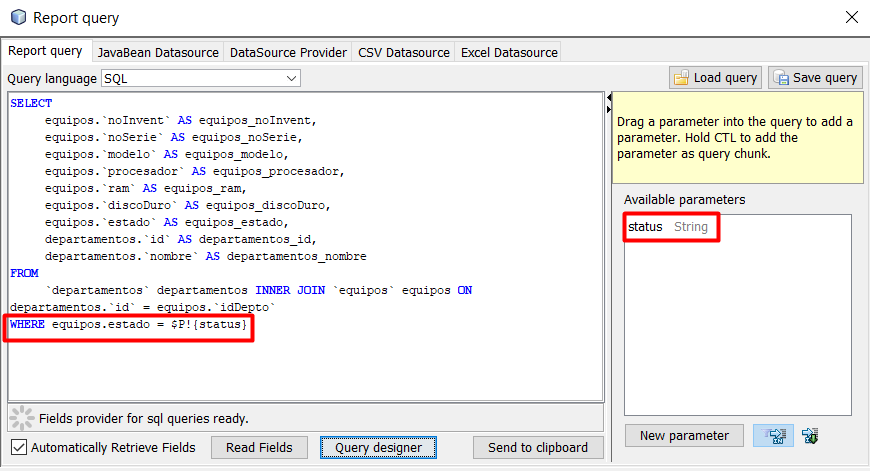
Nos aparece la plantilla del reporte, en este caso ya la tenemos completa pero originalmente sale vacía. Para seleccionar la información que mostraremos en el reporte, daremos click en el icono de la base de datos de arriba.



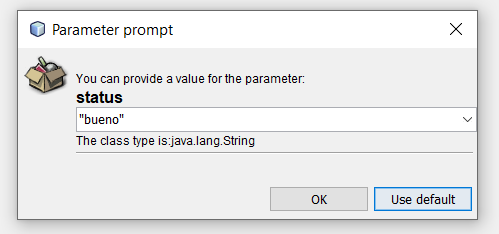
Nos aparece una ventana donde podremos seleccionar las tablas y los atributos que necesitaremos para nuestro reporte.



Una vez seleccionadas las tabla y atributos, lo siguiente será diseñar la consulta, igual como si estuviéramos en MySQL. JasperReports también nos permite añadir parámetros cuyo valor será asignado desde la interfaz del sistema. En este caso nosotros definimos un parámetro llamado status que corresponde al estado en el que se encuentren los equipos de cómputo.



El plugin también nos permite probar el reporte antes de generarlo desde la aplicación. Por ejemplo, le ponemos que nos muestre los equipos en el estado “bueno”.





Y esto es en todo lo correspondiente al desarrollo de nuestro sistema de control de equipo de cómputo.