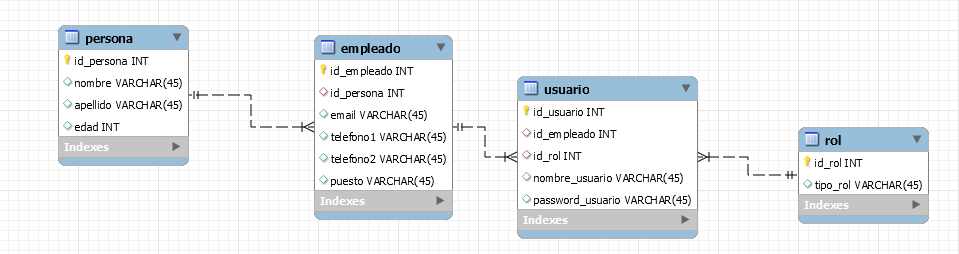
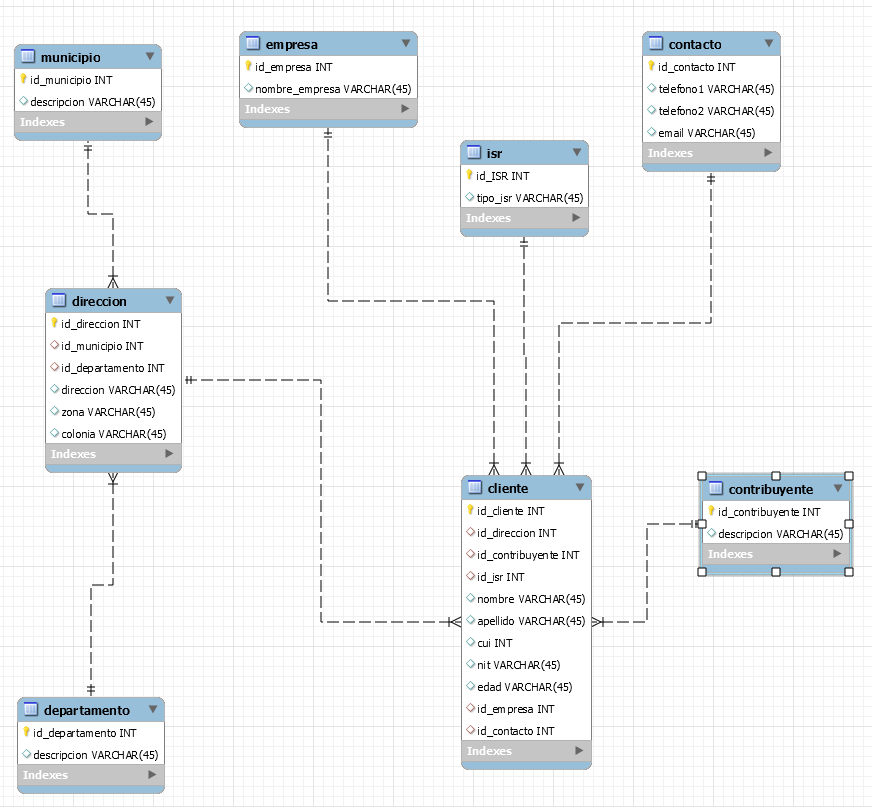
El proyecto esta creado en función de dos conjuntos de bases de datos que trabajan en paralelo uno del otro, así pues, se facilita el manejo de los usuarios permitidos en el sistema, como también permite un manejo organizado de los clientes presentes en el programa.

**Bases de datos relacionales**



Esta base de datos es la que se encarga de gestionar los usuarios permitidos en el sistema, cabe destacar que en este existen dos tipos de usuarios en el programa, uno con rol de administrador (ADMIN) y otro con rol de usuario normal (USER) en la cual el administrador tiene privilegios adicionales con respecto al usuario normal.

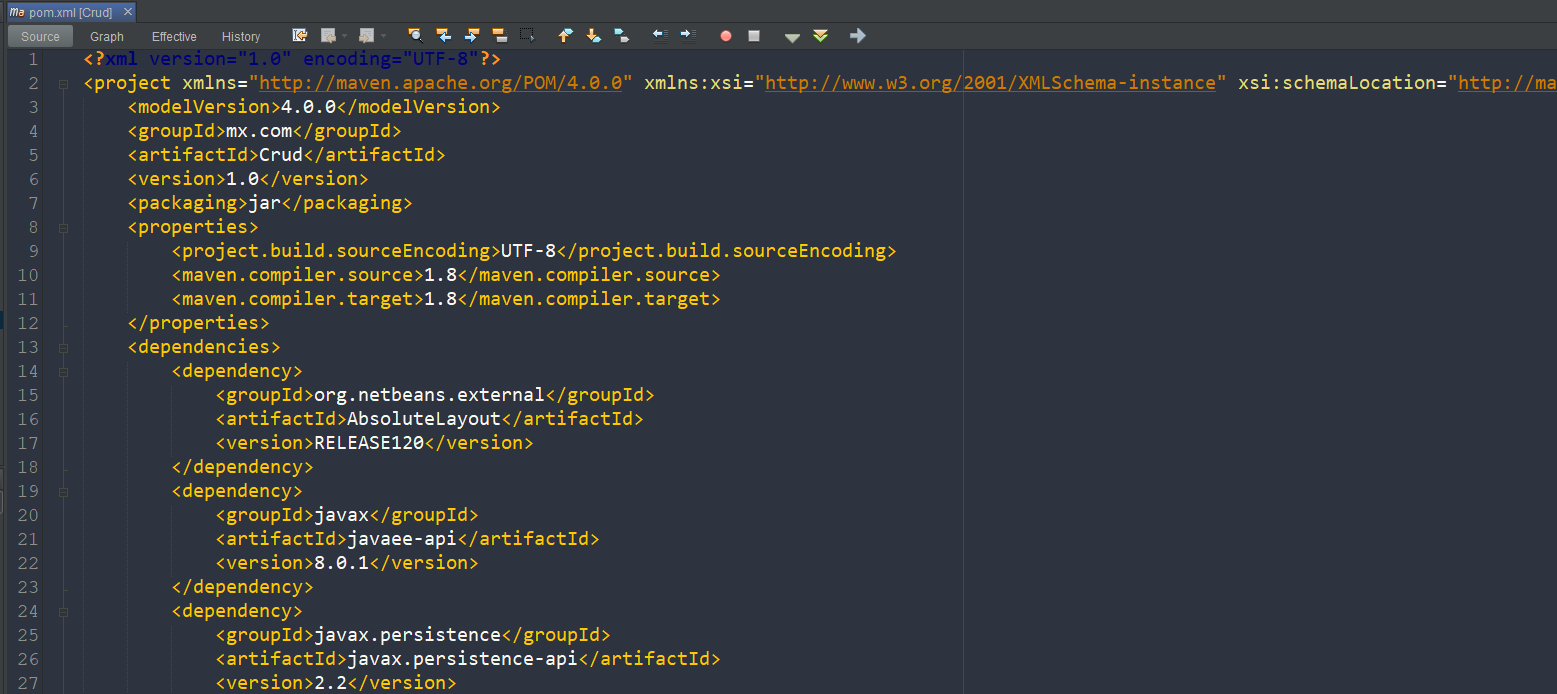


El conjunto de base de tabla de bases de datos anterior es el que ordena la información de los clientes registrados en el sistema, como podemos ver, esta dividida en varias tablas que son dependientes de la tabla cliente, así pues todos los datos agregados en cascada se trata de datos relacionados a un cliente en específico, a excepción de la clase dirección que posee dos llaves foráneas en las cuales se organiza la información de los objetos municipio y departamento para finalmente ser asociada a un objeto cliente.

Cabe destacar que todas las tablas de bases de datos poseen una llave primaria auto incrementable, con la cual se hace referencia a un objeto de tipo cliente.

**Estructura de la aplicación en NetBeans**

La aplicación esta creada gracias a la tecnología de Maven para gestionar las librerías, en las que fueron añadidas a el pom.xml:



Las cuales son:

**org.netbeans.external**: con esta librería se añaden elementos de diseño para la gestión de interfaces graficas.

**Javax**: paquete de java empresarial, el cual nos permite utilizar algunas velidaciones de entidad y también la inclusión del patron MVC.

**javax.persistence**: esta librería nos permite utilizar la persistencia no incluida en el paquete de java empresarial.

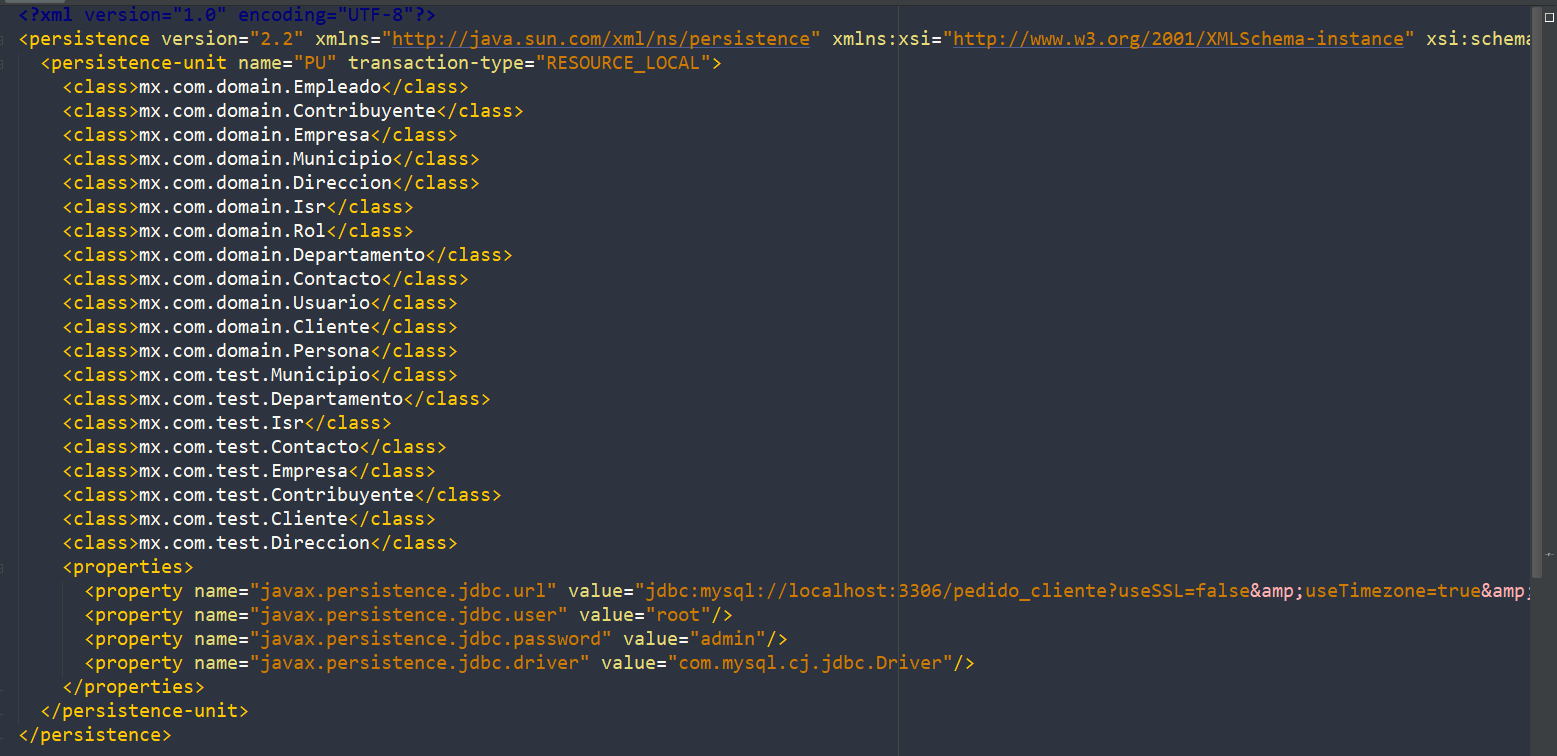
**org.eclipse.persistence**: agrega una implementación del API de JPA para realizar persistencia y lograr una base solidad para ejecutar nuestras operaciones.

**Mysql**: nos permite utilizar el driver para la gestión de base de datos me mySQL.

**com.itextpdf**: nos permite utilizar las herramientas necesarias para la gestión de archivos de tipo pdf.

**org.apache.poi**: el conjunto de estas librerías nos permite utilizar las herramientas necesarias para la gestión de archivos de tipo Excel desde java.

Archivo Persistence.xml:

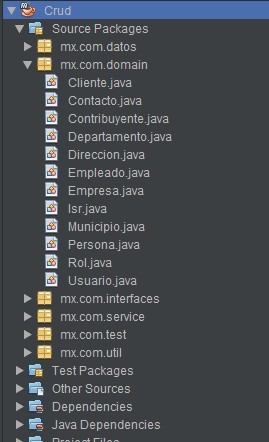


En este archivo se agregan las clases de entidad generadas por ingeniería inversa con herramientas del IDE, además se agregan las propiedades pertinentes para la aplicación del API de JPA, como vemos es una aplicación de recursos locales, además lleva la ruta, contraseña y usuario necesarios para la gestión permitida de esta base de datos.

La aplicación sigue un estándar en su modelo aplicado, siguiendo el patrón MVC (Modelo-Vista-Controlador) se crea una sólida base estructural para soportar y ejecutar cualquier proceso requerido, así pues, se basa en la creación de objetos entidad de las bases de datos para lograr un CRUD completamente aplicado a esta. Los paquetes presentes en la aplicación son:

* **mx.com.domain**

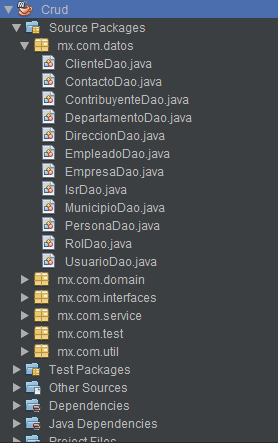
Se trata de un paquete contenedor de las clases entidad, creadas a partir de ingeniería inversa asistido por herramientas del IDE y configuraciones previamente definidas en la base de datos.

Como vemos en la imagen existe una clase entidad para cada tabla de base de datos presentes en la configuración de bases de datos, en las cuales están configuradas para su persistencia en cascada, de modo que facilite su manejo y mantenimiento.

Este conjunto de clases cumple la función de modelo en el patrón de diseño MVC aplicado.

* **mx.com.datos**

Se trata de un conjunto de clases que corresponde a la capa de gestión de datos, cumplen la función de crear las conexiones pertinentes a la base de datos para ejecutar las acciones de CRUD solicitadas por el usuario.



Este conjunto de clases contiene una estructura similar en cada una de las clases presentes en el paquete, conteniendo todo un objeto instancia de la clase ‘EntityManager’ para manejar las operaciones sobre la base de datos.

Dichas operaciones son:

**listarObjetos**: regresa la lista de objetos presentes en la base de datos correspondiente.

**insertarObjeto**: persiste un objeto creado por el usuario en la base de datos.

**modificarObjeto**: modifica un objeto ya creado en la base de datos.

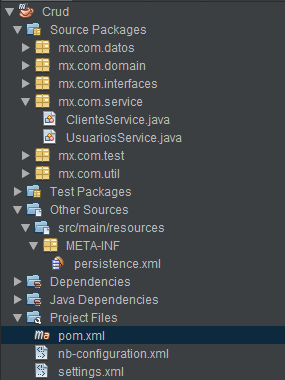
Solo las clases ClienteDao y UsuarioDao poseen métodos diferentes a las demás clases, dada la naturaleza de la relación de las bases de datos. Estas clases contienen métodos mas aplicados a las necesidades del usuario.

En la clase clienteDao se agregan los métodos para búsquedas personalizadas, para eliminar clientes y además para reiniciar la lista de clientes utilizando un Query personalizado en referencia a la base de datos de Cliente.

La clase de usuarioDao existen los métodos para gestionar los usuarios permitidos para hacer login al sistema, no difiere mucho del común de las clases anteriores, salvo los métodos eliminarUsuario que elimina un objeto de tipo usuario y el método encontrarPorId, que regresa un objeto según el id de este.

* **mx.com.service**

se trata de un paquete que reúne las clases de servicio, en el que existen solo dos clases, ClienteService y UsuarioService respectivamente, este par de clases son las que se encargan de la conexión con la capa de datos a su vez de proporcionar al cliente una conexión solida con los métodos empleados en la capa de datos.

La clase cliente tiene los métodos básicos de gestión de base de datos asistido por la capa de datos, en la cual existen las funciones de:

**listarCliente**s: regresa una lista de los objetos clientes presentes en la base de datos

**agregarCliente**: agrega un objeto cliente a la base de datos, cabe destacar la persistencia en cascada, una vez creado el objeto con sus respectivos elementos de llaves foráneas estas se persisten todas al mismo tiempo

**modificarCliente**: modifica un objeto cliente ya creado en la base de datos

**eliminarCliente**: elimina un cliente presente en la base de datos

**eliminarTodosLosClientes**: elimina todos los clientes, pero sin reiniciar la base de datos

**elementos de búsqueda**: se trata de métodos utilizados para encontrar objetos según valores que estos presentan.

**imprimirClientePdf**: esta llama a la clase del paquete mx.com.util encargada de la creación de hojas pdf con los datos presentes en la tabla de clientes.

**imprimirClienteExcel**: este método llama a la clase del paquete mx.com.util encargado de la creación de tablas Excel con los datos presentes en la tabla de clientes.

**reiniciarListaDeClientes**: este método elimina todos los datos de la tabla de clientes y reinicia el contador de llave primaria.

La clase UsuarioService posee los métodos necesarios para la gestión de usuarios permitidos en el sistema, dicha gestión esta reservado solo para usuarios de tipo administrador (ADMIN), dichos métodos son:

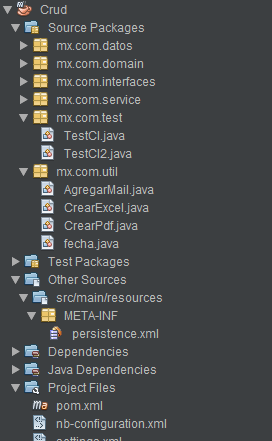
**listarUsuarios**: regresa una lista de los usuarios registrados en el sistema.

**encontrarPorId**: regresa un usuario que coincide con una llave primaria que recibe el método.

**insertarUsuario**: elimina un usuario según el id usuario recibido por el método.

**modificarUsuario**: modifica un usuario ya existente en el sistema

**logIn**: este recibe los parámetros de usuario y contraseña y los compara con la base de datos de usuarios permitidos, de ser coincidentes con valores existentes este permite el acceso al sistema.

* **mx.com.test**

se trata de un paquete de prueba, en el que se realizaron las pruebas pertinentes a la conexión de base de datos, dichas clases están inactivas en su totalidad, por ende, no tienen repercusión en el performance de la aplicación final.

* **mx.com.util**

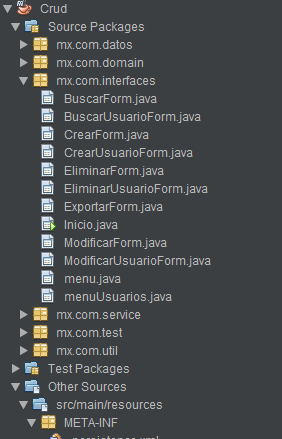
en este paquete se guardan todas las clases utilizadas para aplicaciones extra del sistema, que no tienen que ver con manejo de bases de datos.

**Clase AgregarMail:** en esta clase se agrega una validación de formato tipo mail, al entrar un valor se comprueba si es un correo o no.

**Clase CrearExcel**: en esta clase contiene los métodos de la dependencia de ApachePoi utilizados para la gestión y creación de libros en Excel, al ser llamado el método de esta clase directamente toma los valores presentes en la base de datos de clientes y los transforma en una tabla de Excel organizada.

**Clase CrearPdf**: esta clase contiene los métodos de la dependencia itextpdf utilizados para la gestión y creación de archivos en PDF, al ser llamado este método de la clase directamente toma los valores presentes en la base de datos de clientes y los transforma en una tabla organizada creada en pdf.

**Clase fecha**: contiene un único método con una simple función, obtener los valores de la fecha actual y los retorna como una cadena de caracteres, dicha clase es utilizada para darle nombre a los archivos creados por la clase CrearExcel y CrearPdf.

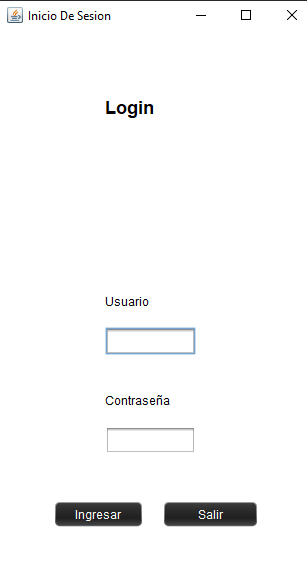
* **mx.com.domain**

En este paquete se alojan todas las clases que crean las interfaces graficas presentes en el sistema, es el paquete que directamente hace uso de la capa de servicio, por ende, tiene acceso a todos los metodos de gestion de clientes.

Dichas clases se subdividen en dos categorias, una a la que pueden acceder los usuarios comunes y un menu extra a la que solo pueden acceder usuarios de tipo administrador.

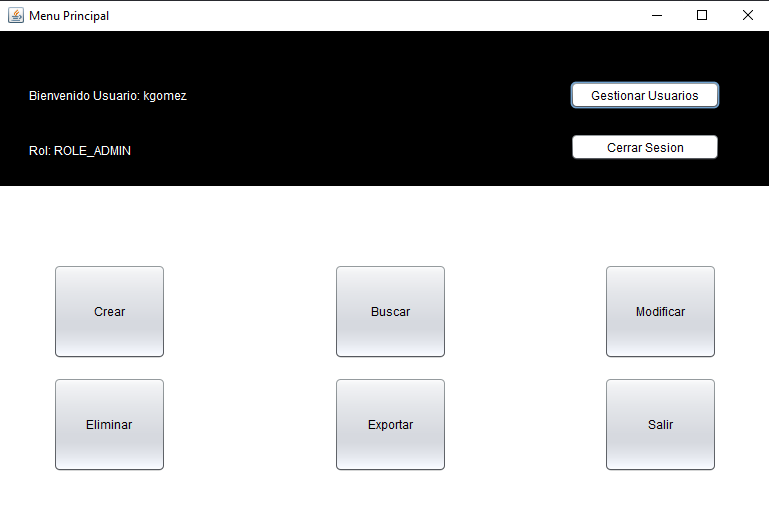
Siendo estos los que tienen mayor privilegios, ya que pueden agregar clientes como crear nuevos usuarios.

**Interfaz: Inicio.**



En esta interfaz se permite la entrada al sistema, dicho sistema obtiene los datos introducidos a los cuadros de texto de usuario y contraseña, para después ejecutar el método logIn en la clase UsuariosService, al encontrar y validar estas credenciales se concede el acceso al sistema.

**Interfaz: menú.**

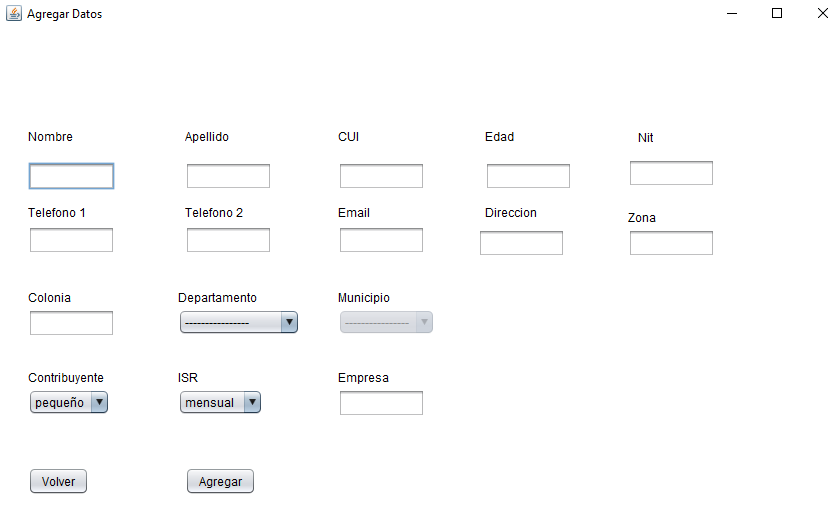
****

En esta interfaz de menú principal se puede acceder a las interfaces básicas que permiten el control de la base de datos de cliente, además de esto en la parte superior izquierda se muestra el usuario que hizo Login al sistema, además de su tipo de ususario.

Ya que el usuario en sesión tiene privilegios de administrador tiene habilitado el botón de gestión de usuarios, el cual conduce a un submenú de gestión de usuarios, e inferior a este se encuentra un botón para cerrar sesión sin salir del sistema.

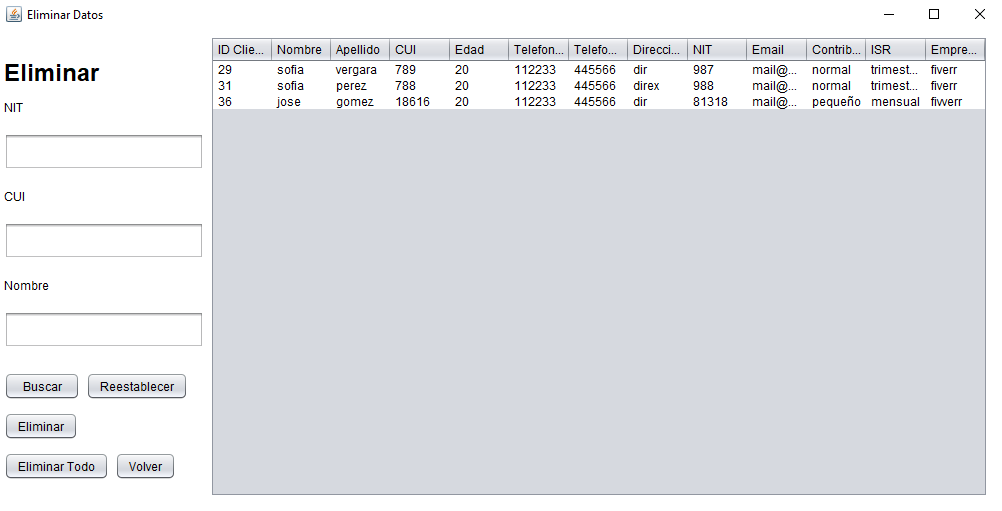
Además, cuenta con un botón de Salir, para directamente cerrar el sistema.

**Interfaz: crearForm.**

****

Se accede a esta interfaz desde el botón crear en el menú principal, esta se encarga de construir el objeto cliente con los datos agregados en los campos de texto, cabe destacar que los campos de texto están sujetos a varias validaciones, por ejemplo, emails válidos, campos de números de teléfonos, o sin tolerancia a campos vacíos de los campos para asegurar que el objeto persistido sea correcto.

**Interfaz: eliminarForm.**



Se accede a esta interface desde el botón de Eliminar en el menú principal, permite acceder a una búsqueda de los elementos presentes en la base de datos para posteriormente poder ser seleccionado y eliminado.

El botón buscar toma los elementos introducidos en los campos de texto y posteriormente realiza una búsqueda tomando alguno que no este vacío, este arroja los elementos que coinciden en la tabla, los cuales pueden ser seleccionados y eliminados.

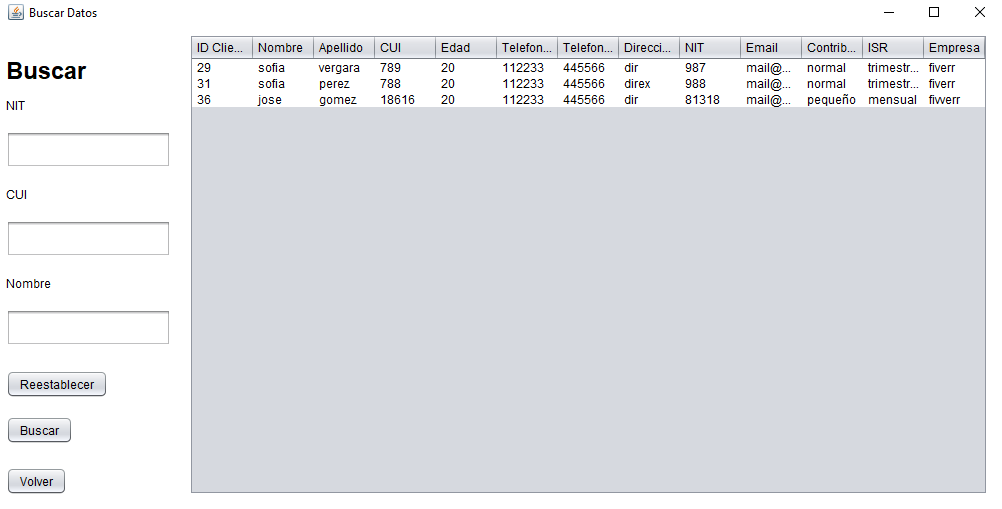
El botón reestablecer devuelve la tabla al estado inicial, cabe destacar que si se elimino un elemento este ya no aparecerá.

El botón eliminar toma el id del cliente seleccionado y ejecuta el método de eliminarCliente presente en la capa de servicio, en la clase ClienteService.

El botón eliminar todo borra todos los datos presentes en al base de datos utilizando el método eliminarTodosLosClietes en la capa de servicio, además usa el método reiniciarListaDeClientes para reestablecer los el id y comenzar de nuevo desde 1.

El botón de volver permite volver al menú principal.

**Interfaz: buscarForm.**



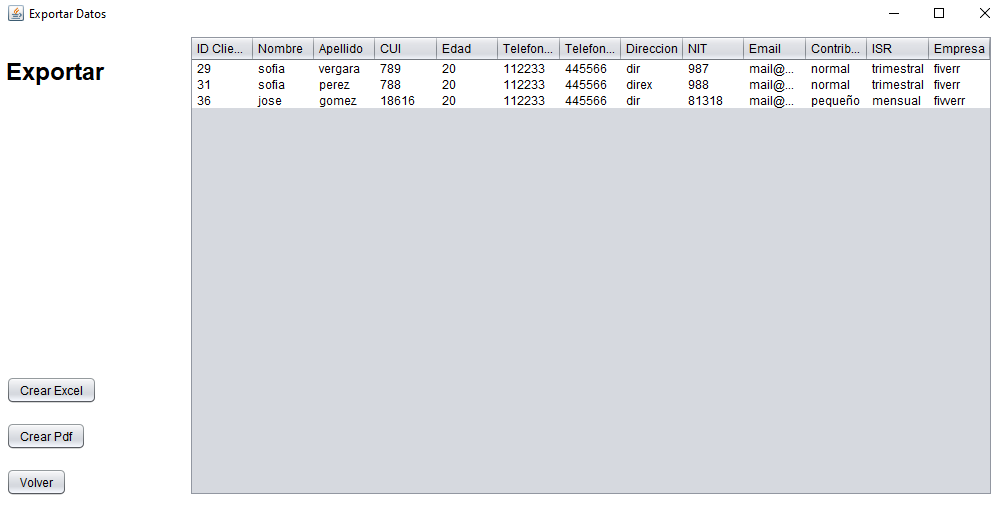
Se accede a esta interface desde el botón de buscar en el menú principal, permite realizar una búsqueda según valores introducidos en los campos de texto.

El botón buscar toma los valores introducidos en el campo de texto y los compara con valores presentes en la tabla, este reescribe la tabla y solo muestra los valores con los que coinciden los datos introducidos.

El botón reestablecer toma los valores iniciales de la tabla y los devuelve a su posición original.

El botón de volver permite regresar al menú principal.

**Interfaz: exportarForm**

**.**

Se accede a estar interfaz desde el botón exportar en el menú principal, permite visualizar los datos presentes en la base de datos de clientes además de crear los archivos pdf y Excel correspondientes.

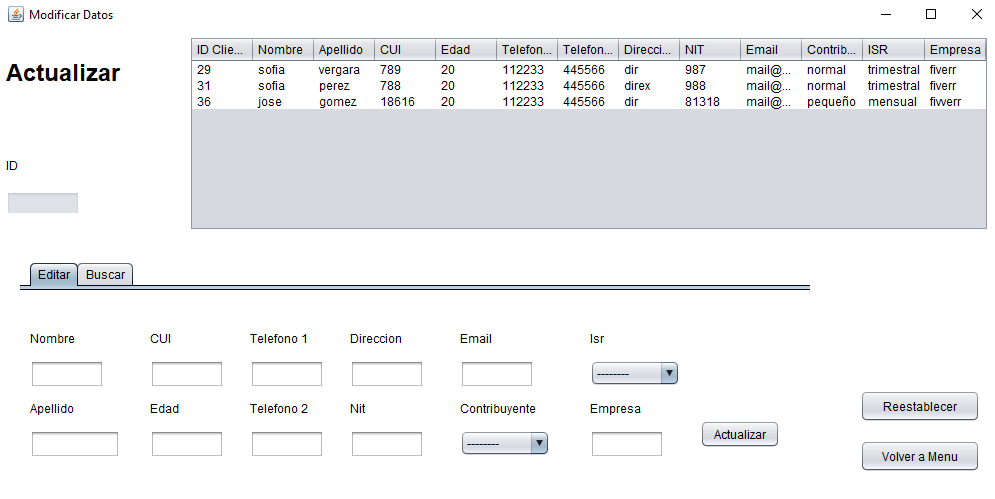
El botón crear Excel hace referencia al método imprimirClienteExcel presente en la capa de servicio específicamente en la clase de ClienteService.

El botón Crear Pdf hace referencia al método imprimirClientePdf, presente también en la capa de servicio, en la misma clase del anterior.

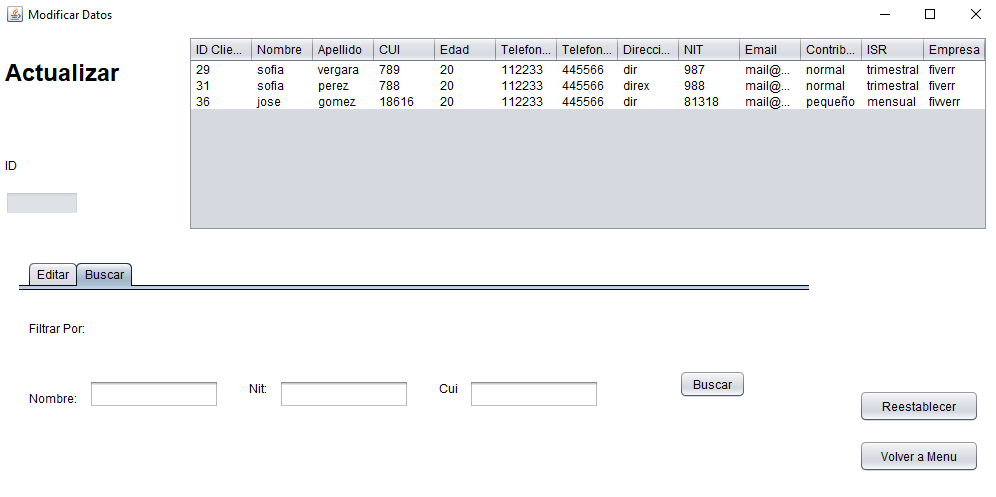
El botón Volver permite volver al menú principal.

**Interfaz: modificarForm.**

se accede a esta interface desde el botón de modificar en el menú principal, cumple con la función de permitir modificar un objeto ya existente en la base de datos de clientes, se compone a su vez de dos submenús uno que permite editar un registro seleccionado en la tabla superior:



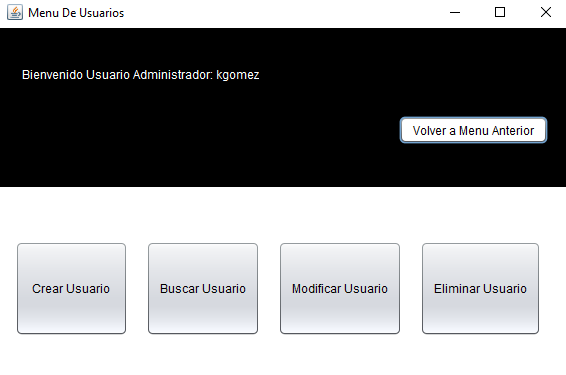
El botón actualizar toma los elementos del objeto seleccionado y los actualiza por los valores modificados por el usuario en los campos de texto.

Y la pestaña Buscar que permite filtrar los elementos de la tabla para facilitar su obtención:

Donde el botón buscar filtra los elementos ingresados en los cuadros de texto para facilitar el trabajo de modificación.

Dos botones que son locales de la interfaz son los botones de Reestablecer y Volver a Menú, que permiten reestablecer los valores de la lista a los valores iniciales y regresar al menú principal respectivamente.

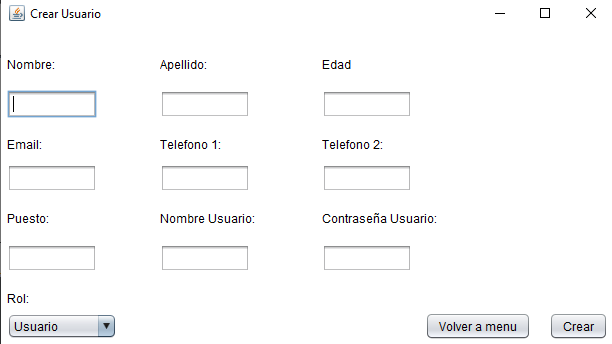
**Interfaz: menuUsuarios.**

****

A este menú solo se puede acceder con privilegios de administrador, esto desde el botón Gestión de usuarios en el menú principal, como tal es una interface parecida al menú principal de la aplicación, solo que con este se controlan los usuarios permitidos para hacer login.

Comparte la característica de mostrar el usuario con el que se hizo login, y añade un botón para volver al menú principal de la aplicación.

**Interfaz: crearUsuarioForm.**

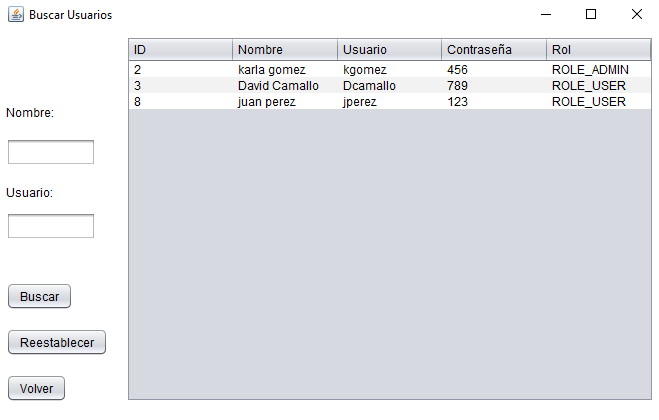
****

En esta interface esta habilitada la función de crear un usuario, en la que permite seleccionar el rol que cumplirá este usuario, administrador o usuario común.

El botón crear cumple la función de crear el objeto usuario y persistirlo a la base de datos, utilizando la clase modelo usuario y aplicando el método de insertarUsuario en la clase UsuariosService.

El botón Volver a menú permite volver al menú de usuarios.

**Interfaz: BuscarUsuarioForm.**

****

esta interface cumple con la función de organizar y mostrar los datos de la tabla de usuarios permitidos en el sistema, y además su contraseña y rol asignado.

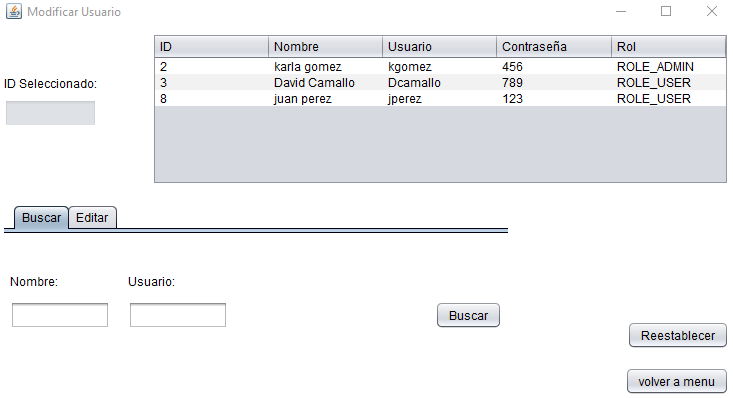
El botón de buscar filtra los datos de la tabla según los datos proporcionados en los cuadros de texto.

El botón de Reestablecer regresa la tabla al estado inicial.

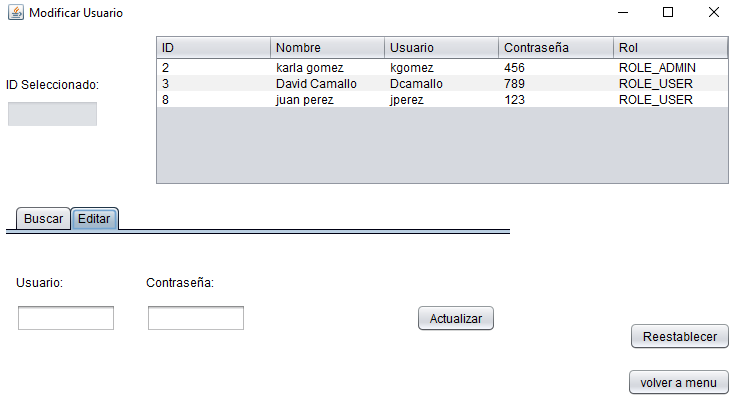
El botón Volver permite regresar al menú de usuarios.

**Interfaz: ModificarUsuarioForm.**

Esta interface, cumple con la función de mostrar los datos de los usuarios presentes en el sistema, y permite realizar una búsqueda según los valores agregados en los campos de texto.

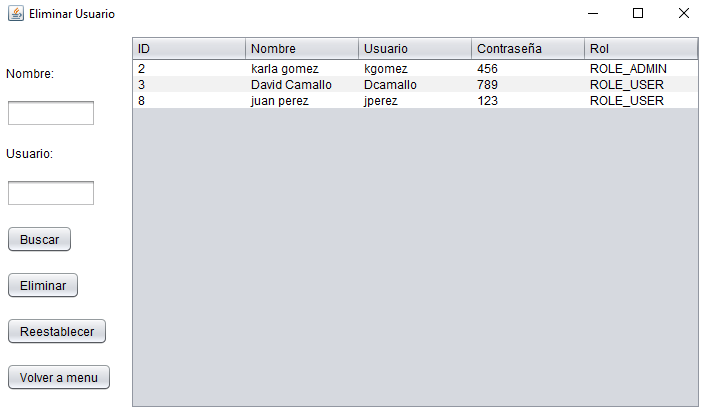


Dicha pestaña tiene un botón que realiza la búsqueda y filtra los elementos de la tabla utilizando métodos nativos de componentes jtable.



Esta pestaña permite editar los elementos del usuario seleccionado, pero no permite modificar su rol asignado, esta es estática para todos los usuarios.

**Interfaz: EliminarUsuarioForm.**

****

Esta interface cumple con la función de eliminar un usuario seleccionado, además de añadir una herramienta de filtrado para facilitar el proceso de eliminación.

El botón de buscar filtra los elementos de la tabla según los valores agregados en los campos de nombre y usuario.

El botón eliminar toma el valor del id seleccionado y llama el método eliminarUsuario presente en la clase usuarioServicer, para lograr eliminar el objeto, sin embargo, no se permite eliminar el usuario con el que se hizo login.

El botón de Reestablecer permite reiniciar los valores de la tabla a su valor original.

El botón Volver a menú permite volver al menú de usuarios.