

## Concesionario de Coches

Salvador Sánchez Rodríguez  
Silvia Kalbakdij Sánchez  
Dalila Berd Gómez  
Víctor Pinto San Macario  
David Gómez Cordero  
Nazareth Jiménez Vela  
Javier Rodríguez Vidal  
Rafael Fernández López

14 de mayo de 2008

# Índice general

<b>1. Especificación</b>	<b>5</b>
1.1. Coches . . . . .	5
1.2. Elementos de la Jerarquía . . . . .	6
<b>2. Diseño</b>	<b>13</b>
2.1. Diagrama de clases . . . . .	13
2.2. Métodos del diagrama de clases . . . . .	15
2.3. Diagrama de estados de Mensaje . . . . .	18
2.4. Diagrama de estados de Coche . . . . .	19
2.5. Diagrama de estados de Contable . . . . .	20
2.6. Diagrama de clases de la interfaz . . . . .	21
2.7. Diagramas de secuencias de sistema . . . . .	23
2.7.1. Diagrama de secuencias de visualizar coches vendidos por un comercial . . . . .	23
2.7.2. Diagrama de secuencias de vender un coche . . . . .	24
2.7.3. Diagrama de secuencias de eliminar un coche . . . . .	25
2.7.4. Diagrama de secuencias de eliminar un contable . . . . .	26
2.7.5. Diagrama de secuencias de eliminar un comercial . . . . .	28
2.7.6. Diagrama de secuencias de crear un contable . . . . .	29
2.7.7. Diagrama de secuencias de activar un contable cuando hay otro contable activo . . . . .	30
2.8. Diagramas de secuencias de interfaces . . . . .	31
2.8.1. Diagrama de secuencias de hacer logout . . . . .	31
2.8.2. Diagrama de secuencias de hacer login . . . . .	33
2.8.3. Diagrama de secuencias de activar un contable . . . . .	35
2.9. Diagrama de actividades visualizar coches vendidos por él mismo (comercial) . . . . .	36
2.10. Diagrama de actividades de visualizar coches en venta . . . . .	37
2.11. Diagrama de actividades de vender coche . . . . .	38
2.12. Diagrama de actividades de añadir coche en lista de ventas . . . . .	39
2.13. Diagrama de actividades de modificar datos de los coches vendidos por un comercial . . . . .	40
2.14. Diagrama de actividades de eliminar el contable activo . . . . .	41
2.15. Diagrama de componentes . . . . .	42
2.15.1. Bases de Datos . . . . .	42
2.15.2. Interfaces . . . . .	42
2.15.3. Gestores . . . . .	42
2.15.4. Expendedor . . . . .	42
2.15.5. Pago . . . . .	42
2.16. Pruebas . . . . .	46
2.16.1. Pruebas de Regresión y Sistema . . . . .	46
2.16.2. Pruebas de la interfaz gráfica . . . . .	49
2.16.3. Pruebas específicas . . . . .	51
2.17. Apariencia de la interfaz . . . . .	52
2.17.1. Interfaz Comercial . . . . .	52
2.17.2. Interfaz Base de Datos de Coche . . . . .	52
2.17.3. Interfaz Contable . . . . .	52
2.17.4. Interfaz de Gestión de Coches . . . . .	53
2.17.5. Interfaz Gestión de Contables . . . . .	53
2.17.6. Interfaz Jefe . . . . .	54

2.17.7. Interfaz Lista Contables . . . . .	54
2.17.8. Interfaz Login . . . . .	55
2.17.9. Interfaz Modificar Cuenta . . . . .	55
2.17.10 Interfaz principal mensajes . . . . .	56
2.17.11 Interfaz nuevo mensaje . . . . .	56
2.18. Invariantes de la Interfaz . . . . .	57
2.18.1. Interfaz principal mensajes . . . . .	57
2.18.2. Interfaz login . . . . .	57
2.18.3. Interfaz ventana gestión contable . . . . .	58
2.18.4. Interfaz mandar mensajes . . . . .	58
2.18.5. Interfaz modificar cuenta . . . . .	58
2.19. Investigación de los componentes utilizados en el sistema . . . . .	58
2.19.1. Propuestas para el componente de pago . . . . .	58
2.19.2. Propuestas para el componente de bases de datos . . . . .	58
2.20. Implementación . . . . .	60
2.20.1. Información sobre las tablas . . . . .	60
2.20.2. Lenguaje utilizado . . . . .	60

# Índice de figuras

1.1. Jerarquía del sistema . . . . .	5
1.2. Diagrama de casos de uso para el comercial . . . . .	10
1.3. Diagrama de casos de uso para el contable . . . . .	11
1.4. Diagrama de casos de uso para el jefe . . . . .	12
1.5. Diagrama de casos de uso para un usuario genérico (no identificado todavía) . . . . .	12
2.1. Diagrama de clases . . . . .	14
2.2. Diagrama de estados de Mensaje . . . . .	18
2.3. Diagrama de estados de Coche . . . . .	19
2.4. Diagrama de estados de Contable . . . . .	20
2.5. Diagrama de componentes . . . . .	43

# Índice de cuadros

1.1. Tabla de Requisitos . . . . .	8
1.2. Tabla de Requisitos y Localización . . . . .	9
2.1. Métodos de Coche . . . . .	15
2.2. Métodos Cuenta Comercial . . . . .	16
2.3. Métodos Cuenta Contable . . . . .	16
2.4. Métodos Cuenta Jefe . . . . .	17
2.5. Base de Datos de los Contables . . . . .	44
2.6. Base de Datos de los Comerciales . . . . .	44
2.7. Base de Datos de los Coches . . . . .	44
2.8. Base de Datos de los Usuarios . . . . .	45

# Capítulo 1

## Especificación

El Proyecto a desarrollar es una aplicación informática que gestiona una pequeña empresa dedicada a la compra y venta de coches de segunda mano y kilómetro cero.

### 1.1. Coches

Nuestros principales elementos en la empresa serán los coches que tendrán la siguiente información que podrá ser consultada por nuestros empleados en los casos especificados posteriormente. Estos datos son:

- Fabricante del coche
- Modelo
- Kilometraje
- Año de Fabricación
- Matrícula del coche
- Color
- Número de Bastidor
- Precio con el que fue comprado el coche
- Precio de venta, que será fijado por el jefe
- Fecha de venta
- DNI del comprador

La empresa sigue una estructura jerárquica. Los elementos de esta jerarquía son: jefe, contable y comercial. La estructura de ésta viene determinada por la siguiente figura:

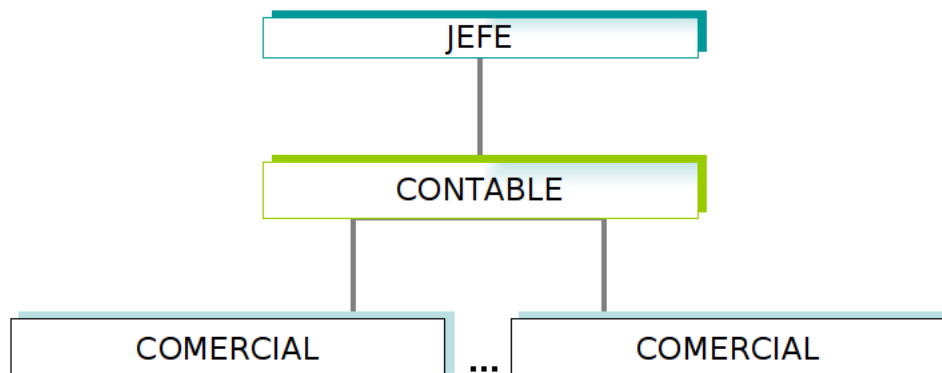


Figura 1.1: Jerarquía del sistema

La figura anterior representa un posible estado del sistema en un cierto instante. Porque aunque ésta no lo muestra puede haber más de un contable aunque uno solo activo (más tarde se explicará el significado de este término) y más comerciales. Todos los comerciales están asociados a todos los contables.

## 1.2. Elementos de la Jerarquía

### ■ Jefe

El jefe de la empresa será imprescindible en el sistema, es decir, siempre debe existir uno y sólo uno, y tendrá la potestad de realizar las siguientes acciones.

- Será el único que pueda crear, modificar y eliminar una cuenta de contable.
- El jefe tendrá derecho a modificar el capital de la empresa, bien sea ingresando o extrayendo dinero.
- Al ser el supervisor de las finanzas, se le notificará todos los movimientos que se hagan por medio de mensajes informativos y tendrá que confirmar todas las salidas de dinero (compra de coches, modificación de sueldos, etc).

### ■ Contable

El siguiente en la jerarquía de la empresa será el contable. En nuestro sistema sólo existirá un único contable activo aunque podrán existir otros contables que no podrán realizar acción alguna. El encargado de activar/desactivar contables será el jefe.

El contable será el encargado de:

- Realizar los pedidos de coches y añadirlos al sistema si son aceptados por el jefe.
- Podrá modificar las características de los coches del sistema notificando esta acción al jefe.
- También podrá modificar los datos de una venta en caso de que se haya cometido algún error (nombre erróneo, etc).
- Visualizará los datos referentes a las ventas de cualquier comercial.
- Podrá añadir, modificar y eliminar comerciales.
- A la hora de modificar un comercial sólo podrá variar su sueldo notificando este hecho al jefe.
- Podrá ver todos los coches alojados en el sistema, tanto vendidos como no vendidos.
- Eliminar los coches que han pasado el periodo obligatorio de permanencia en la aplicación. Este tiempo será de 3 años.

El contable exclusivamente se dedica a tareas administrativas no pudiendo vender coches.

En caso de no disponer de ningún contable (porque se haya eliminado o desactivado el último contable) el jefe ocupará este rol.

### ■ Comercial

El último de la jerarquía será el/los comerciales. Cuyas funciones en la empresa serán:

- Vender coches.
- Visualizar todos los coches disponibles para vender.
- Cambiar los datos referentes a una venta que haya realizado (erratas en el DNI del comprador, etc). En caso de cambio se notificará al jefe.
- Visualizar todas las ventas que haya realizado pero no la del resto de comerciales.

Podrá haber un número “ilimitado” de comerciales teniendo que existir al menos uno. En caso de no disponer de ningún comercial, el contable ocupará este rol.

La aplicación dispondrá de un sistema de mensajería que podrá ser utilizado por los diferentes

usuarios del sistema, siempre y cuando hayan iniciado su sesión. Los mensajes disponibles de este servicio dispondrán de un campo de texto, un campo de asunto, y las diferentes direcciones del remitente y destinatario. A la vez, los mensajes serán usados por el sistema para comunicar a los usuarios de la aplicación de las diversas modificaciones relevantes de su cuenta.



Los requisitos de la aplicación son los siguientes:

Requisitos	Descripción
LOG1	Antes de poder realizar cualquier operación es necesario que el usuario esté logueado.
LOG2	Para poder realizar el login es necesario introducir el identificador de usuario y la contraseña correcta.
LOG3	Al hacer login, si la cuenta corresponde con una cuenta contable no sólo es necesario que el identificador y la contraseña de usuario sean correctas, sino también que dicho contable esté activo.
LOG4	Dependiendo del tipo de cuenta que corresponda una vez que el usuario está logueado el sistema ofrecerá un conjunto de operaciones distintas de acuerdo a el caso.
LOG5	Los usuarios podrán modificar su contraseña pero no su identificador.
SISTEMA1	Tiene que haber al menos un comercial en el sistema.
SISTEMA2	Tiene que haber al menos un contable en el sistema.
SISTEMA3	Sólo hay un jefe en el sistema (y sólo uno).
SISTEMA4	Una misma persona puede tener como máximo una cuenta por cada cargo (puesto que ID == DNI).
SISTEMA5	Cualquier cambio en el sistema es notificado al jefe vía mensaje generado por el sistema excepto, por cambios resultantes de sus propias acciones.
SISTEMA6	Cualquier salida o entrada de dinero provoca que automáticamente se actualice el capital.
SISTEMA7	Si un coche pendiente de entrar en la lista de coches en venta es rechazado por el jefe, éste se elimina del sistema.
SISTEMA8	El contable sólo puede eliminar un coche de los coches vendidos si éste se vendió al menos tres años atrás.
SISTEMA9	Todos los usuarios pueden mandarse mensajes entre ellos.
SISTEMA10	Un mensaje puede tener varios destinatarios pero sólo un remitente.
SISTEMA11	Existen dos tipos de mensajes: los generados por el sistema y los creados por los usuarios.
SISTEMA13	La aplicación dispone de un capital inicial. Dicho capital, sólo sera gestionado por el jefe, no teniendo acceso a el ningún otro usuario.
JEF1	Si no hay contables en el sistema el jefe debe asumir ese puesto.
JEF2	El jefe puede añadir y eliminar contables así como modificar su sueldo.
JEF3	El jefe puede activar y desactivar contables.
JEF4	El jefe puede modificar el capital de la empresa.
JEF5	El jefe tiene que confirmar mediante mensajes la compra de coches solicitada por un contable.
CONT1	Siempre hay un contable activo en el sistema y sólo uno.
CONT2	Si no hay comerciales en el sistema el contable debe asumir ese puesto.
CONT3	Sólo el contable puede añadir coches a la lista de coches en venta.
CONT4	Sólo se añaden coches por el contable a la lista de coches en venta si son confirmados por el jefe.
CONT5	Sólo el contable puede eliminar coches de la lista de los coches vendidos.
CONT6	El contable puede modificar y visualizar los datos de cualquier coche, tanto vendidos como en venta.
CONT7	El contable tiene acceso a la lista de comerciales.
CONT8	El contable puede añadir y eliminar comerciales así como modificar su sueldo.
CONT9	Sólo puede realizar acciones de contable el contable activo.
COM1	Únicamente los comerciales pueden vender coches.
COM2	El comercial tiene acceso a los coches en venta.
COM3	El comercial puede visualizar únicamente los coches vendidos por el mismo.
COM4	Cuando el comercial vende un coche, éste es eliminado de la lista de coches en venta, y es añadido a la lista de coches vendidos.
COM5	Un comercial puede modificar únicamente el DNI de un comprador relativo a una venta realizada por él mismo.
SISTEMA12	Cualquier otra funcionalidad que no quede expresada implícita o explícitamente por los requisitos anteriores no deberá implementarse en el sistema .

Cuadro 1.1: Tabla de Requisitos

Para poder implementar los anteriores requisitos, hemos realizado las siguientes implementaciones:

Requisitos	Localización
LOG1	Diagrama de secuencias “Login”.
LOG2	Diagrama de secuencias “Login”.
LOG3	Diagrama de secuencias “Login”.
LOG4	Diagrama de secuencias “Login”.
LOG5	En cualquiera de las bases de datos tanto usuarios como de contables y comerciales no se ofrece ningún metodo que permita modificar el identificador.
SISTEMA1	CuentaComercial.allInstances() → size() >= 1
SISTEMA2	CuentaContable.allInstances() → size() >= 1
SISTEMA3	CuentaJefe.allInstances() → size() == 1
SISTEMA4	a) CuentaContable.allInstances() → forAll(c1   CuentaContable.allInstances() → forAll(c2   c1 <> c2 implies (c1.id <> c2.id))) b) CuentaContable.allInstances() → forAll(c1   CuentaContable.allInstances() → forAll(c2   c1 <> c2 implies (c1.id <> c2.id)))
SISTEMA5	En cualquier Diagrama de secuencia donde se haga alguna modificación del sistema. Ejemplo: Diagrama de secuencias de vender coche.
SISTEMA6	En cualquier Diagrama de actividades donde se haga alguna modificación del capital.
SISTEMA7	Diagrama de actividades “Añadir Coche a la Lista de Ventas”.
SISTEMA8	Diagrama de secuencias “Eliminar Coche Vendido”.
SISTEMA9	Diagrama de clases (relaciones de multiplicidad de los usuarios con los mensajes)
SISTEMA10	Diagrama de clases (relaciones de multiplicidad de los usuarios con los mensajes).
SISTEMA11	Diagrama de secuencias, donde en cada una de las acciones podemos observar la creación de los mensajes.
SISTEMA13	Diagrama de clases (atributo Capital y método modificarCapital).
JEF1	Diagrama de secuencias “Eliminar Contable”.
JEF2	Diagrama de clases (métodos crearContable y eliminarContable de la clase CuentaJefa). Diagrama de secuencias “Eliminar Contable”.
JEF3	Diagrama de clases (métodos activarContable y desactivarContable). Diagrama de secuencias “Activar Contable”. El diagrama de secuencias de “Desactivar Contable” no lo adjuntamos porque es similar al de “Eliminar Contable”.
JEF4	Diagrama de clases (método modificarCapital).
JEF5	Diagrama de actividades “Visualizar coches en venta”.
CONT1	context: CuentaContable: self.allInstances() → select(c   c.activo) → size() = 1.
CONT2	Diagrama de secuencias “Eliminar Comercial”.
CONT3	Diagrama de clases (método añadirCoche).
CONT4	Diagrama de actividades “Añadir Coche a la Lista de Venta”.
CONT6	Diagrama de clases (métodos modificarCoche, visualizarListaVendidos, visualizarListEnVenta).
CONT7	Diagrama de clases (método consultaComerciales) y diagrama de componentes.
CONT8	Diagrama de clases (métodos modificarComercial, eliminarComercial)
COM1	Diagrama de clases (método venderCoche)
COM2	Diagrama de clases (método visualizarCochesVenta).
COM3	Diagrama de secuencias “Visualizar Coches Vendidos”.
COM4	Diagrama de secuencias “Vender Coche”.
COM5	Consecuencia de COM3, ya que antes de modificar tiene que visualizar los coches vendidos, los cuales han tenido que ser vendidos por él mismo. Lo vemos también en el diagrama de actividades “Modificar Datos de Coches Vendidos por el Comercial”.

Cuadro 1.2: Tabla de Requisitos y Localización

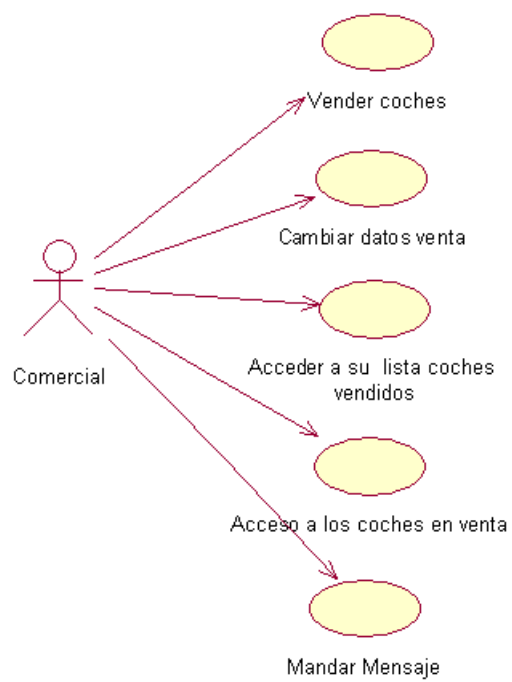


Figura 1.2: Diagrama de casos de uso para el comercial

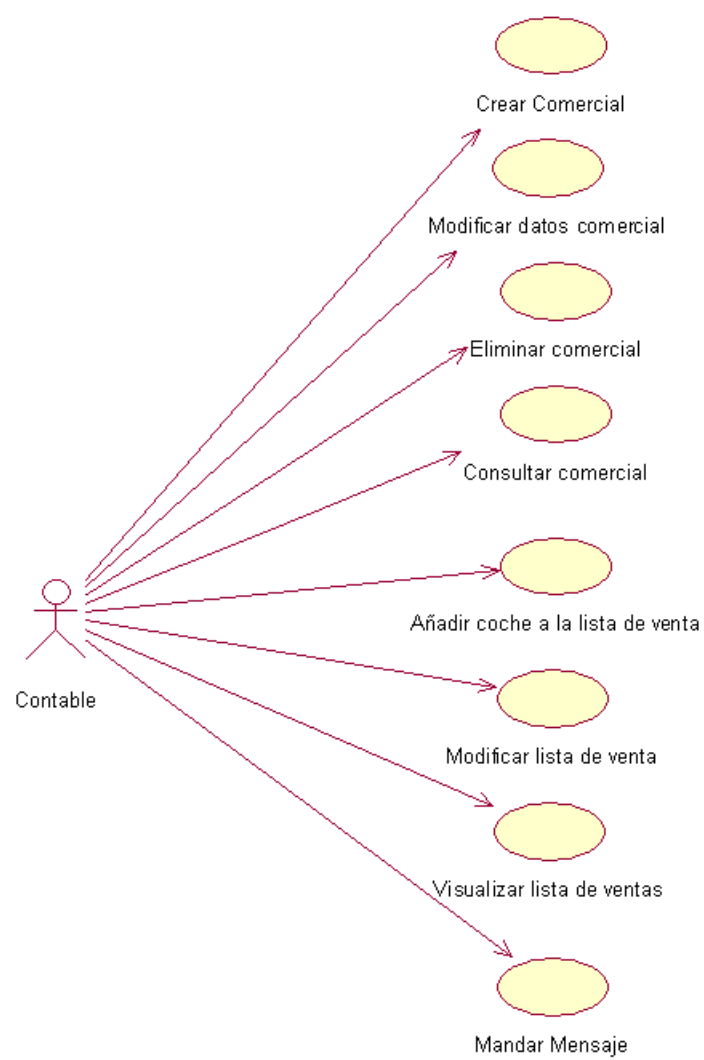


Figura 1.3: Diagrama de casos de uso para el contable

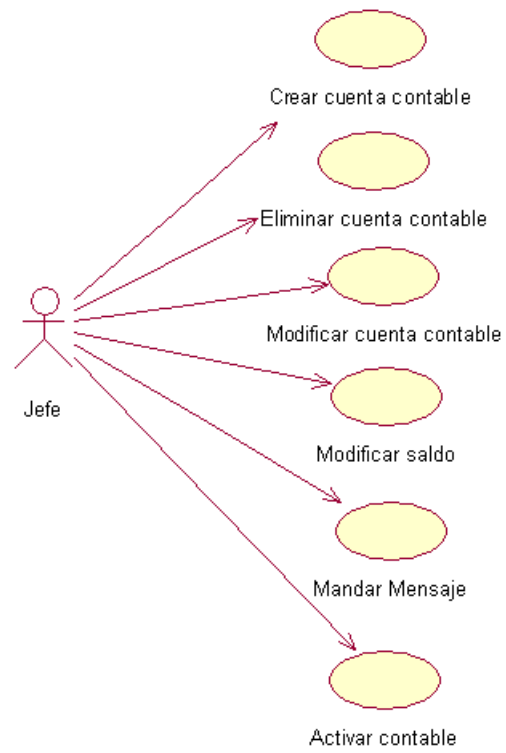


Figura 1.4: Diagrama de casos de uso para el jefe

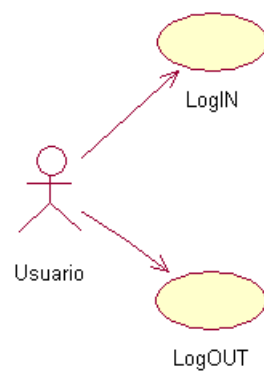


Figura 1.5: Diagrama de casos de uso para un usuario genérico (no identificado todavía)

## Capítulo 2

# Diseño

### 2.1. Diagrama de clases

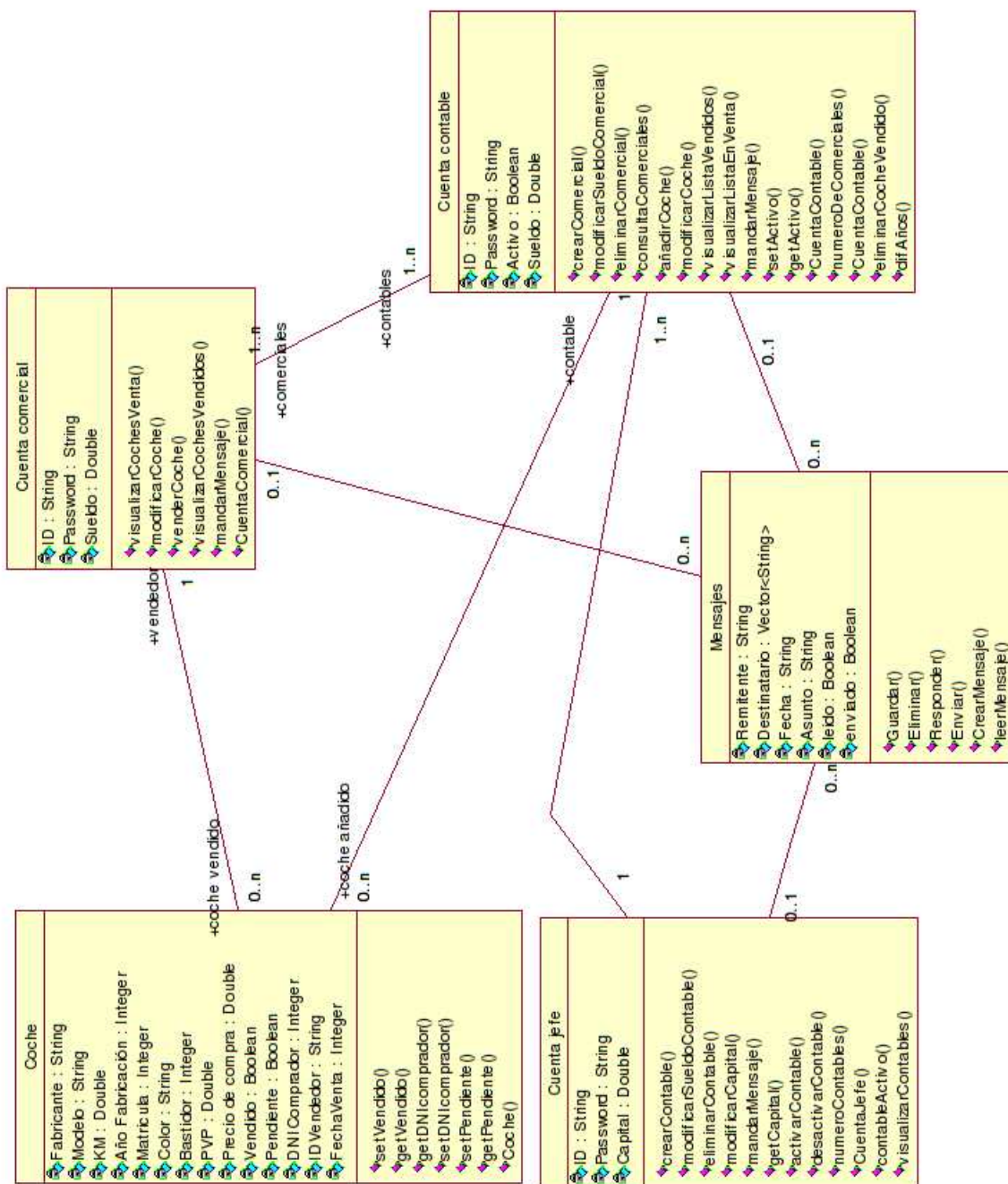


Figura 2.1: Diagrama de clases

El diagrama de clases contiene 5 clases:

1. Coche: Aunque los coches están representados en el diagrama como una clase, a la hora de implementar el sistema los coches sólo se encuentran en una base de datos y no en memoria. Es innecesario crear un objeto por cada coche del sistema porque no realizan ninguna operación.
2. Cuenta Jefe
3. Cuenta Contable
4. Cuenta Comercial
5. Mensajes

Con respecto a la relación de Cuenta Contable y Cuenta Comercial es necesario que todas las Cuentas de Contables se relacionen con todas las Cuentas de Comerciales y viceversa; para ello existe el siguiente invariante:

```
context CuentaComercial;  
self.contables == CuentaContable.allInstances();  
context CuentaContable;  
self.comerciales == CuentaComercial.allInstances();
```

## 2.2. Métodos del diagrama de clases

Función	Descripción
setVendido(Vendido : boolean) : Void	Establece el atributo de un coche como vendido pasándolo de la lista de coches en venta a la lista de coches vendidos por un comercial.
getVendido() : Boolean	Devuelve el booleano correspondiente al estado del coche.
getDNComprador() : Integer	En caso de que el coche haya sido vendido, devuelve la identificación del comprador.
setDNComprador(DNI : Integer) : Void	A la hora de la venta del coche se fija este parámetro en los datos del coche.
setPendiente(pendiente : boolean) : Void	Fija si un coche está pendiente de confirmación de compra por parte del jefe.
getPendiente() : Boolean	Devuelve un booleano indicando si un coche ha sido confirmado o no.
Coche(Fabricante : String, Modelo : String, km : Double, Año : Integer, Matrícula : Integer, Color : String, Bastidor : Integer, PVP : Double, PrecioCompra : Double, Vendido : Boolean, DNComprador : Integer, IDVendedor : String) : Coche	Corresponde a la constructora.

Cuadro 2.1: Métodos de Coche



Función	Descripción
visualizarCocheVenta() : List	Muestra la lista de coches disponibles para la venta, por cualquier comercial.
modificarCoche(mat : integer, atributo : String, valor : Object) : Void	En caso de algún error, a la hora de fijar los atributos del coche, será necesario un método que permita modificarlos.
venderCoche(dni1 : Integer, matricula : Integer) : Void	Fija la relación entre el comprador y el coche que ha sido comprado.
visualizarCocheVendidos() : List	Muestra la lista de coches vendidos correspondiente exclusivamente al comercial que ejecuta el método.
mandarMensaje(destinatario : String, remitente : String, asunto : String, contenido : String) : Mensajes	Facilita la comunicación entre los miembros del sistema. Su objetivo principal es mantener informado al jefe de todos los pormenores.

Cuadro 2.2: Métodos Cuenta Comercial

Función	Descripción
crearComercial(ID : String, Password : String, Sueldo : Double) : Void	Da de alta o activa un nuevo comercial en el sistema.
modificarSueldoComercial(ID : String, SueldoNuevo : Double) : Void	Establece el sueldo del comercial, que lo fija el contable.
eliminarComercial(ID : string) : Void	Borra del sistema los datos correspondientes a un comercial en particular.
consultaComerciales(ID : String) : List	Devuelve la lista de los comerciales con los que cuenta el sistema.
añadirCoche(Fabricante : String, Modelo : String, km : Double, Año : Integer, Matricula : Integer, Color : String, Bastidor : Integer, PVP : Double, PrecioCompra : Double, Vendido : Boolean, DNIComprador : Integer, IDVendedor : String) : Coche : Void	Introduce un nuevo coche a la lista de coches en venta.
modificarCoche(mat : Integer, atributo : String, valor : Object) : Void	Modifica el atributo que le pases por parámetro con el valor correspondiente de tipo objeto ya que dependiendo del atributo será de un tipo u otro.
visualizarListaVendidos() : List	Devuelve una lista con los coches cuyo atributo vendido está a true en la base de datos de coches.
visualizarListaEnVenta() : List	Devuelve una lista con los coches cuyo atributo pendiente está a true y vendido a false.
mandarMensaje(Destinatario : String, Remitente : String, Asunto : String, Contenido : String) : Mensajes	Facilita la comunicación entre los miembros del sistema. Su objetivo principal es mantener informado al jefe de todos los cambios realizados.
setActivo(Activo : Boolean) : Void	Establece el estado de un contable. Sólo el contable que está activo puede interactuar con el sistema.
getActivo() : Boolean	Informa sobre el estado de un determinado contable.
CuentaContable(ID : String, Password : String, Activo : Boolean, Sueldo : Double) : Cuenta contable	Constructora de Contable
numeroDeComerciales() : Integer	Devuelve el número de comerciales que componen la lista de comerciales del sistema.

Cuadro 2.3: Métodos Cuenta Contable

<b>Función</b>	<b>Descripción</b>
crearContable(ID : String, Password : String, Sueldo : Double) : Void	Da de alta un nuevo contable en el sistema.
modificarSueldoContable(ID : String, SueldoNuevo : Double) : Void	Establece el sueldo del contable, pudiendo ser fijado únicamente por el jefe.
eliminarContable(ID : String, ID2 : String) : Void	Borra del sistema el respectivo contable.
modificarCapital(capital : Double) : Void	Establece el presupuesto o capital de la empresa.
mandarMensaje(Destinatario : String, Remitente : String, Asunto : String, Contenido : String) : Mensajes	Facilita la comunicación entre los miembros del sistema.
getCapital() : Double	Devuelve el capital de la aplicación.
activarContable(ID : String) : Void	Activa un contable, que será quien pueda interactuar en el sistema.
desactivarContable(ID : String) : Void	Inactiva o deshabilita el contable que hasta el momento estaba activo.
numeroContables() : Integer	Devuelve el número de contables pertenecientes al sistema.
CuentaJefe(ID : String, Password : String, capital : Double) : Cuenta jefe	Corresponde a la constructora.
contableActivo() : String	Devuelve el contable que está activo en el momento de la consulta.
visualizarContables() : List	Muestra la lista de contables del sistema.

Cuadro 2.4: Métodos Cuenta Jefe

## 2.3. Diagrama de estados de Mensaje

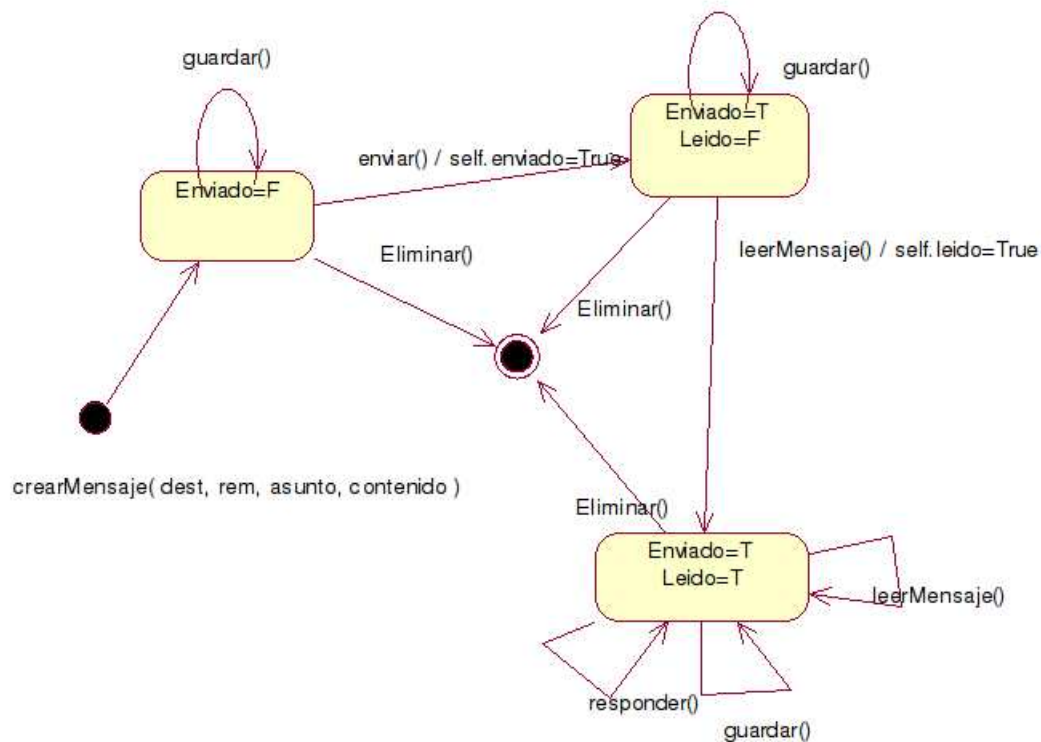


Figura 2.2: Diagrama de estados de Mensaje

La figura 2.2 simboliza cuáles son los cambios que se producen a la hora de mandar un mensaje. Una vez que se cree un mensaje (se añade dicho mensaje a la base de datos) pasaremos al estado en el que el mensaje todavía no ha sido enviado. Una vez que se envía, se cambia el atributo mandado del mensaje a cierto y pasamos al estado en el que se ha mandado el mensaje y todavía no ha sido leído. En este estado se puede guardar el mensaje y nos quedamos en el mismo estado o se puede leer un mensaje con lo que pasamos al estado en que se ha enviado y leído. En este último estado nos quedamos en el caso de que se lea el mensaje, se guarde o se responda al mensaje mandado. Se puede realizar la acción de eliminar desde todos los estados, siendo eliminado de la base de datos el mensaje.

## 2.4. Diagrama de estados de Coche

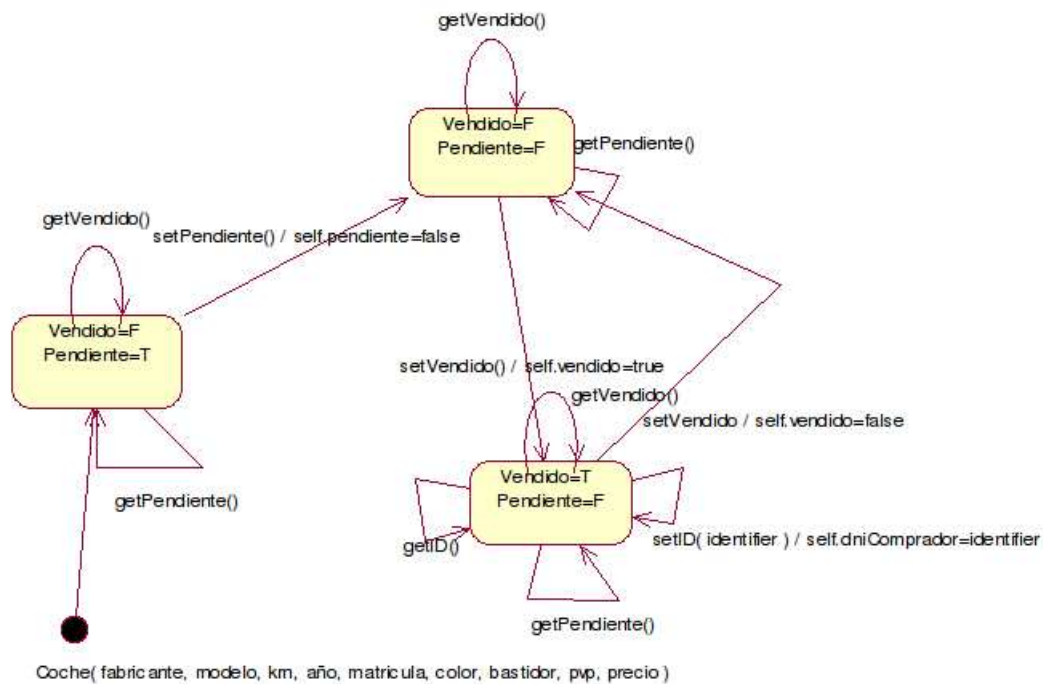


Figura 2.3: Diagrama de estados de Coche

Cuando introducimos un nuevo coche en la base de datos éste se encuentra en un estado inicial de no vendido y pendiente. Desde este estado, no se produce transición si se consulta si un coche ha sido vendido o si está pendiente de venta. Si se produce una transición que modifique el atributo pendiente pasaremos al estado de no vendido y no pendiente de añadir a la base de datos de coches en venta. Si se producen transiciones que consulten los atributos nos quedaremos en el mismo estado. Si el coche se vende, pasamos al tercer y último estado, coche vendido y no pendiente. En este estado es posible consultar tanto los atributos del coche, el ID del comprador (el cual también se podrá modificar) y no se producirá transición alguna.

## 2.5. Diagrama de estados de Contable

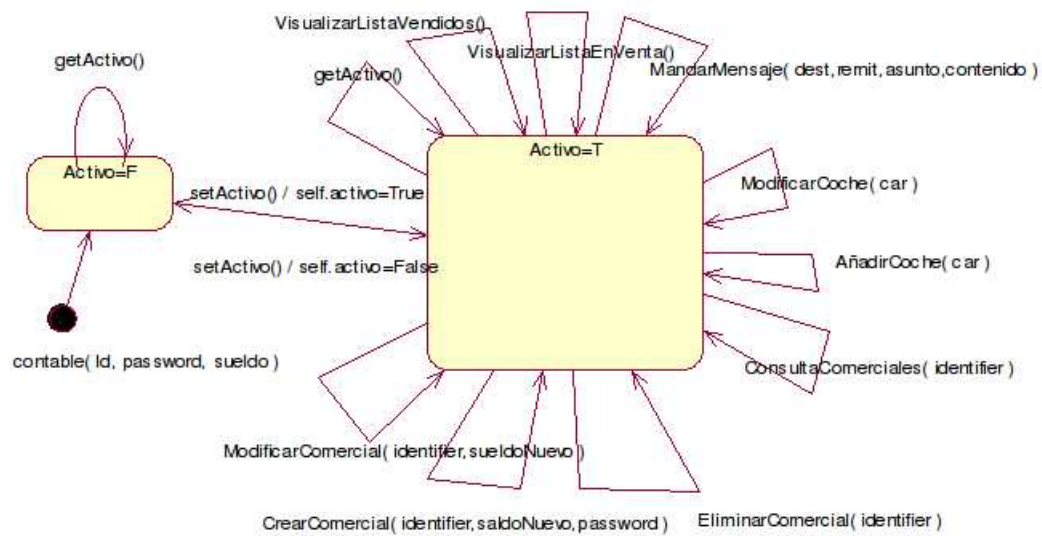


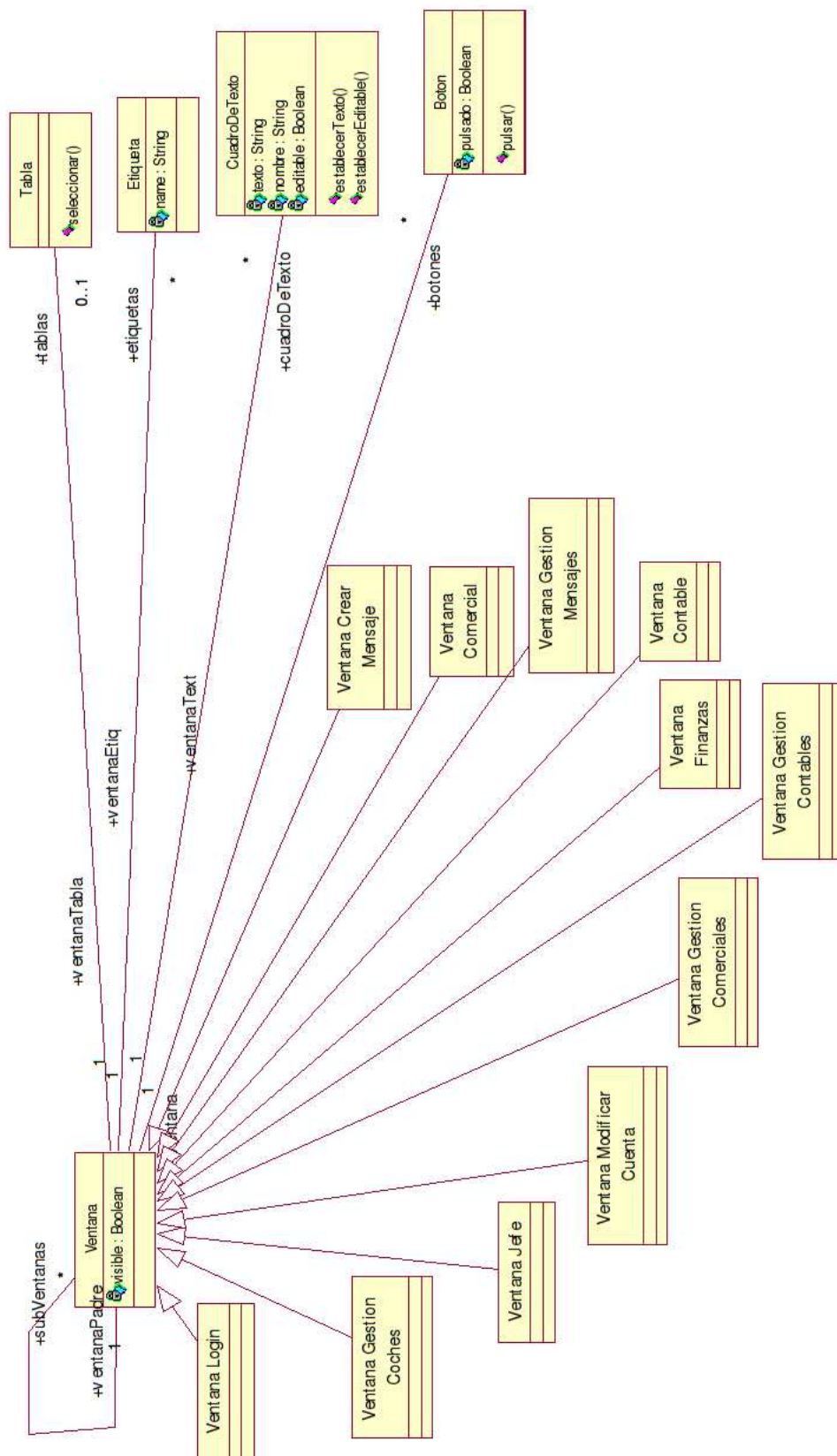
Figura 2.4: Diagrama de estados de Contable

Cuando se da de alta a un nuevo contable, éste comienza en el estado inactivo. Desde este estado no podrá realizar ninguna acción; sólo en el caso de que se active, podrá realizar sus funciones tales como modificar coche, añadir coche, etc.

## 2.6. Diagrama de clases de la interfaz

El interfaz del sistema está compuesto por ventanas. Éstas pueden contener los siguientes elementos: tabla, etiqueta, cuadro de texto y botón.

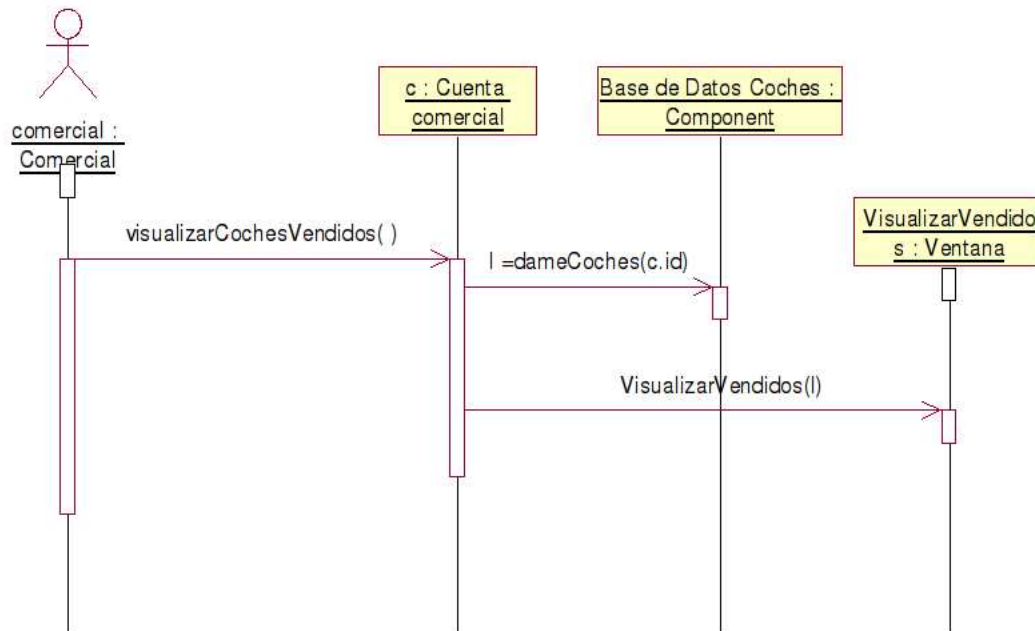
Tenemos una clase (Ventana) abstracta de la que heredarán todas las ventanas que componen la interfaz de nuestro sistema.



## 2.7. Diagramas de secuencias de sistema

### 2.7.1. Diagrama de secuencias de visualizar coches vendidos por un comercial

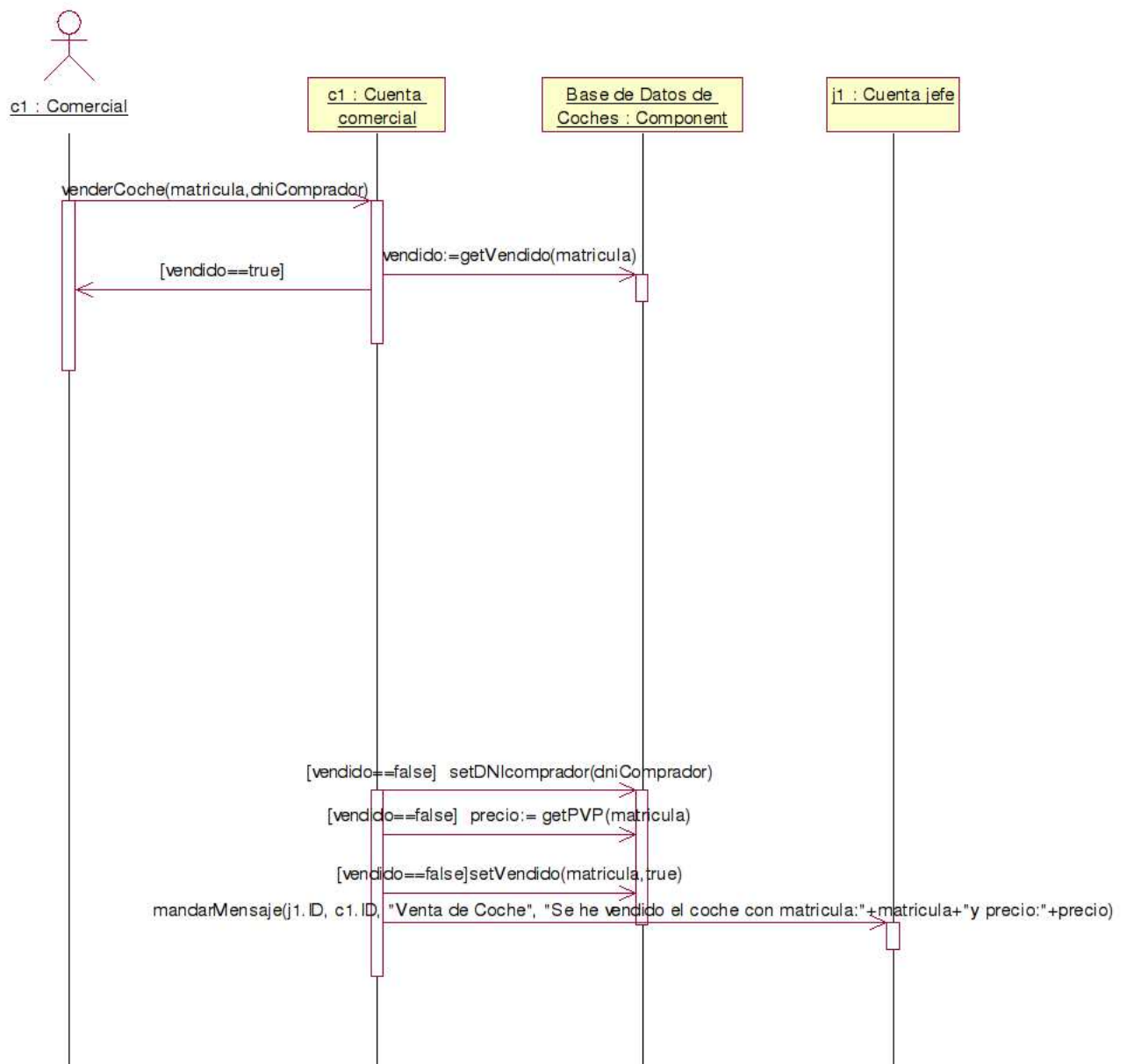
En este caso, mostramos el hecho de que un comercial visualice sus coches vendidos. Lo importante de este diagrama consiste en que cualquier comercial puede visualizar únicamente los coches que han sido vendidos por él.





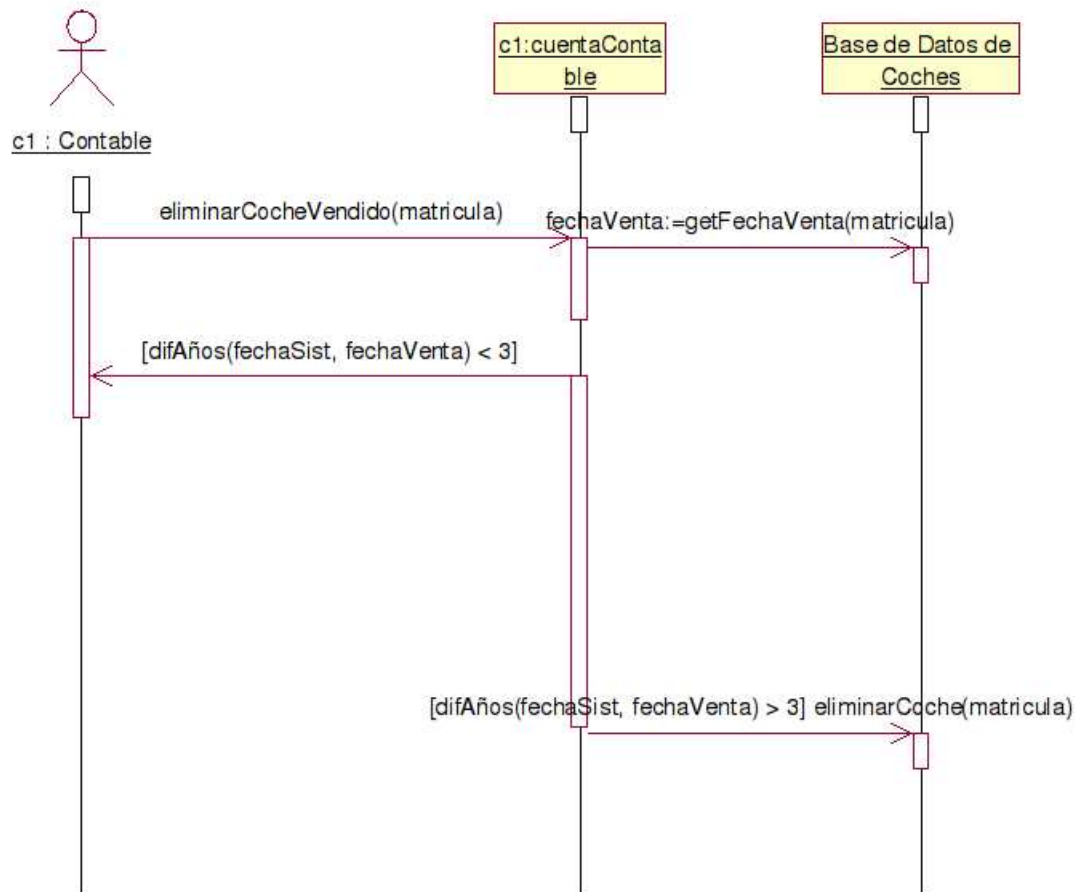
### 2.7.2. Diagrama de secuencias de vender un coche

En cuanto a vender coche, la importancia que tiene este diagrama de secuencia es mostrar el hecho de que sólo puede vender un comercial, así mismo el coche tiene que estar en la lista de coches para vender, o lo que es lo mismo, que no haya sido vendido previamente. Es relevante señalar que dicha operación informa al jefe de la venta llevada a cabo.



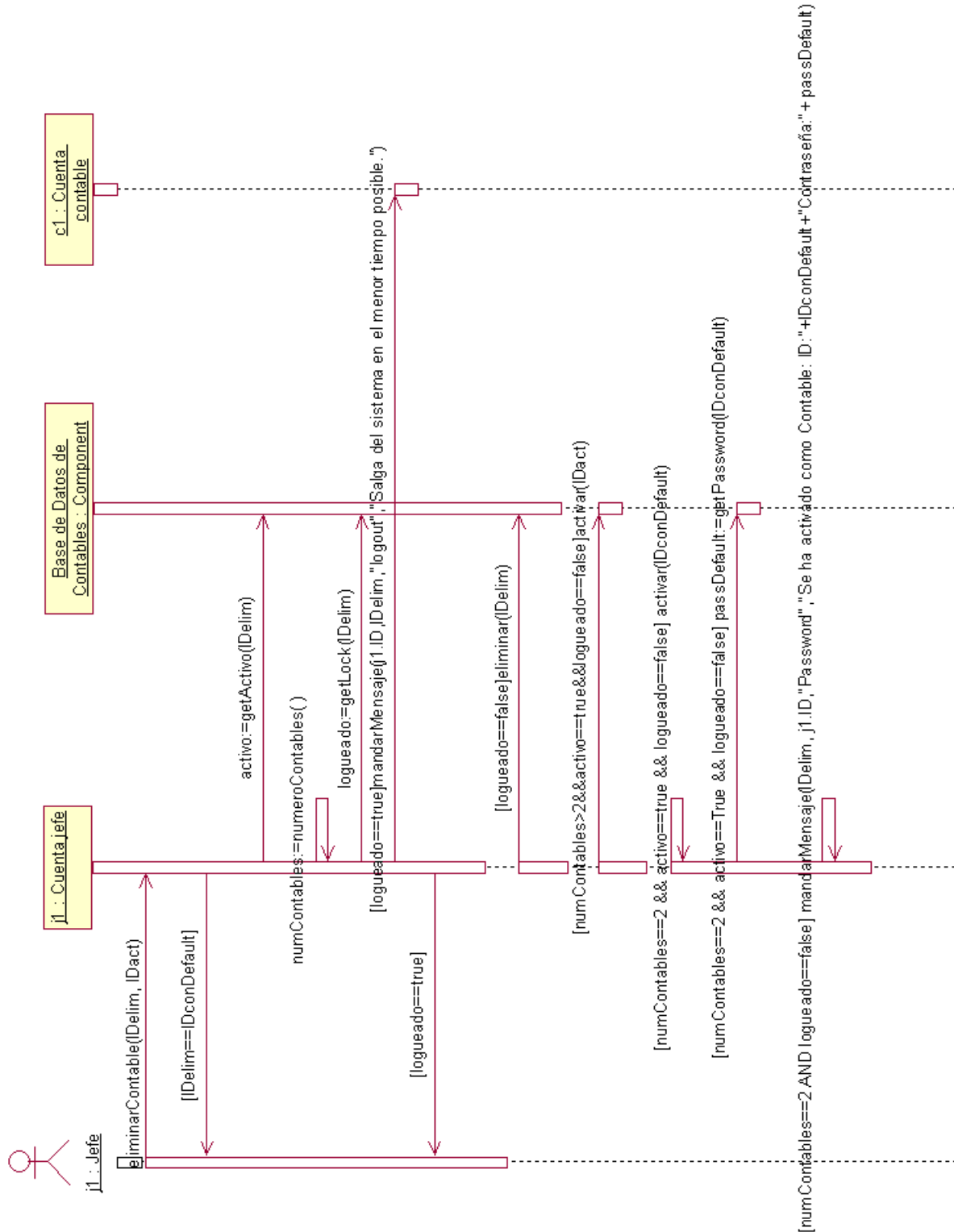
### 2.7.3. Diagrama de secuencias de eliminar un coche

Es posible que el contable activo pueda eliminar de la base de datos un coche, pero es necesario, según los requisitos que el cliente nos proporcionó que la antigüedad de la venta sea de al menos 3 años.



## 2.7.4. Diagrama de secuencias de eliminar un contable

Al eliminar un contable, hay que tener en cuenta que éste no esté logueado y que siempre exista uno en el sistema o que la cuenta de contable que se va a eliminar no sea la cuenta por defecto.



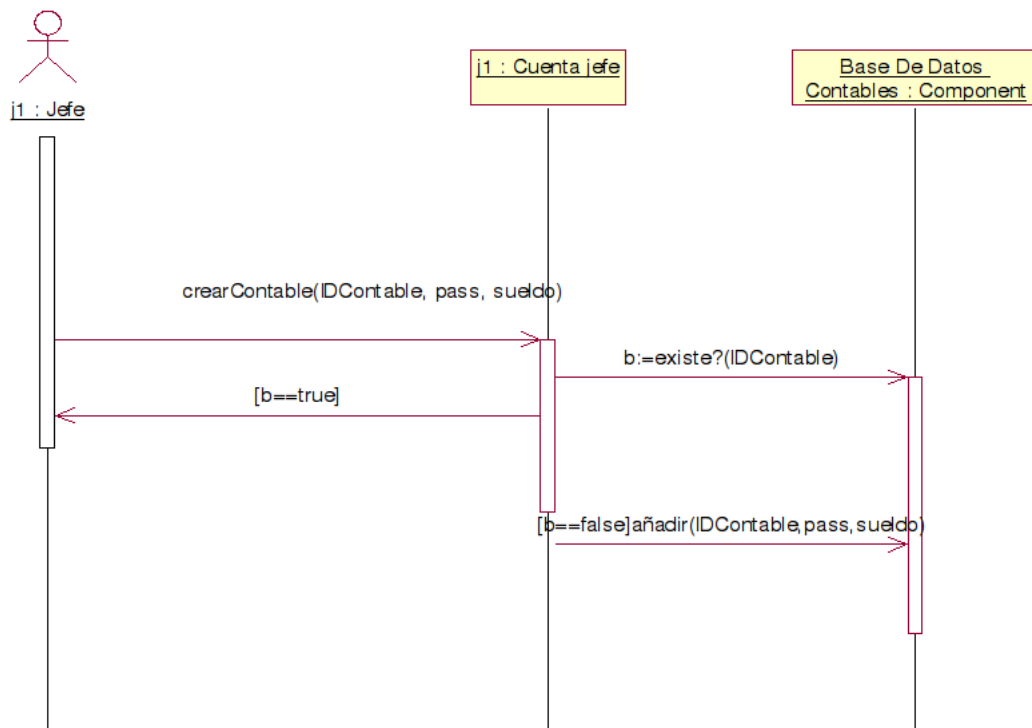
No incluimos explícitamente el diagrama de secuencias de desactivar contable por ser análogo a éste (eliminar contable).

Para mantener la coherencia del sistema, nos aseguramos de que al eliminar un comercial éste no esté logueado en el sistema. Si el comercial que se quiere eliminar no está logueado y no se trata del comercial por defecto, se lleva a cabo la eliminación del mismo.



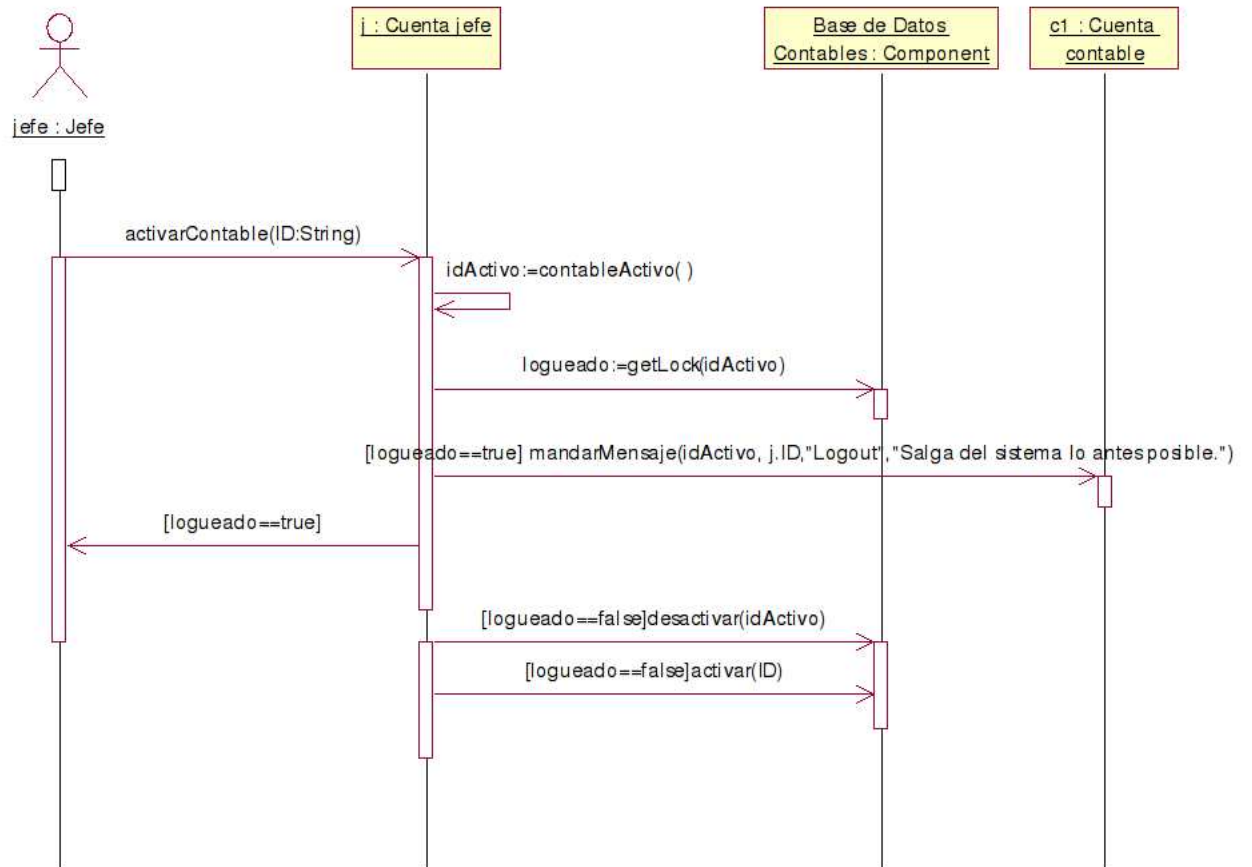
### 2.7.6. Diagrama de secuencias de crear un contable

La relevancia de este diagrama es la creación de una nueva entrada en la base de datos.



### 2.7.7. Diagrama de secuencias de activar un contable cuando hay otro contable activo

Lo importante en este caso es que, a la hora de activar un contable, se verifica si el que está activo está logueado. De ser así, es necesario que salga del sistema para poder desactivarlo y activar un nuevo contable.

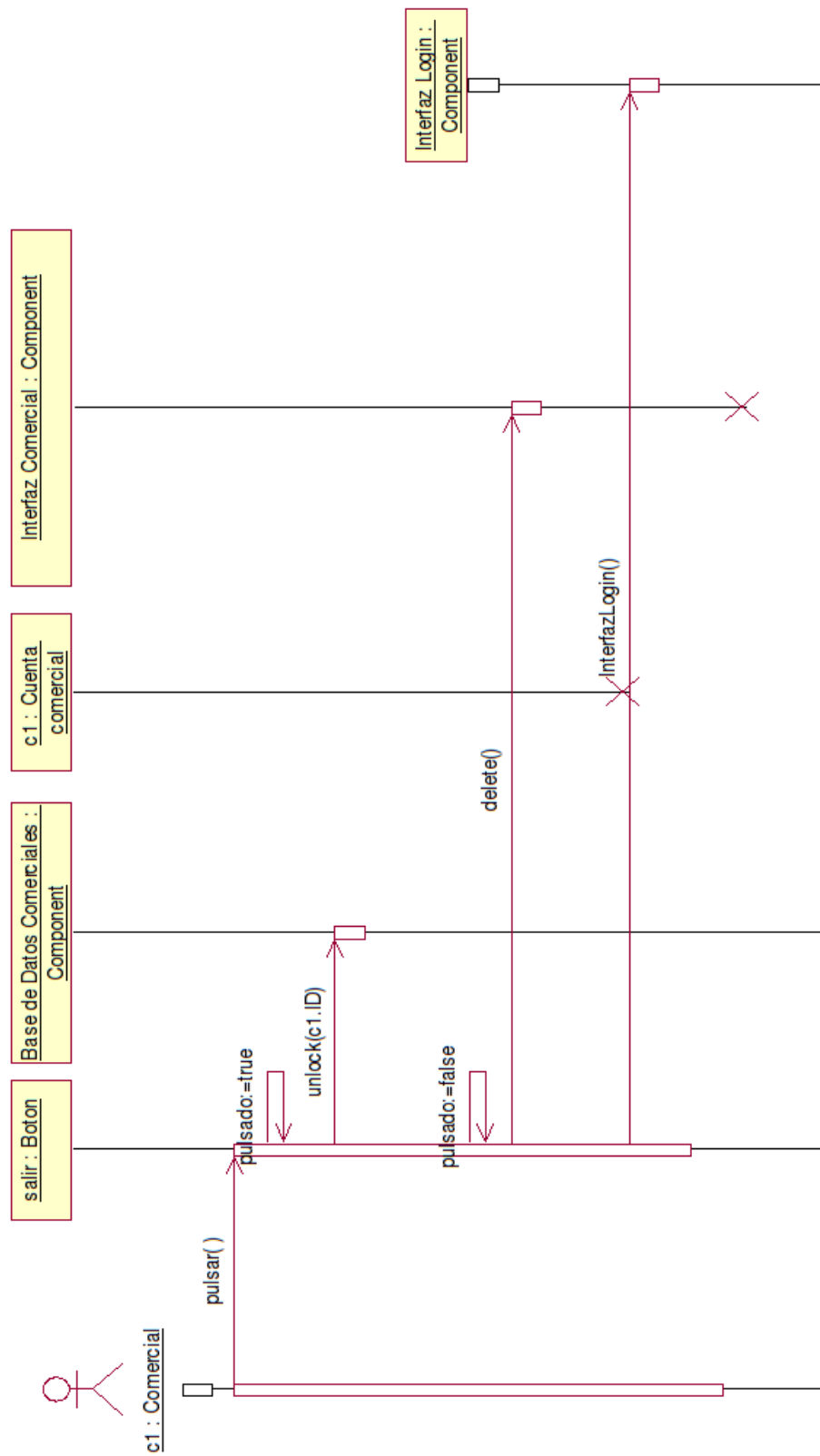


## **2.8. Diagramas de secuencias de interfaces**

### **2.8.1. Diagrama de secuencias de hacer logout**

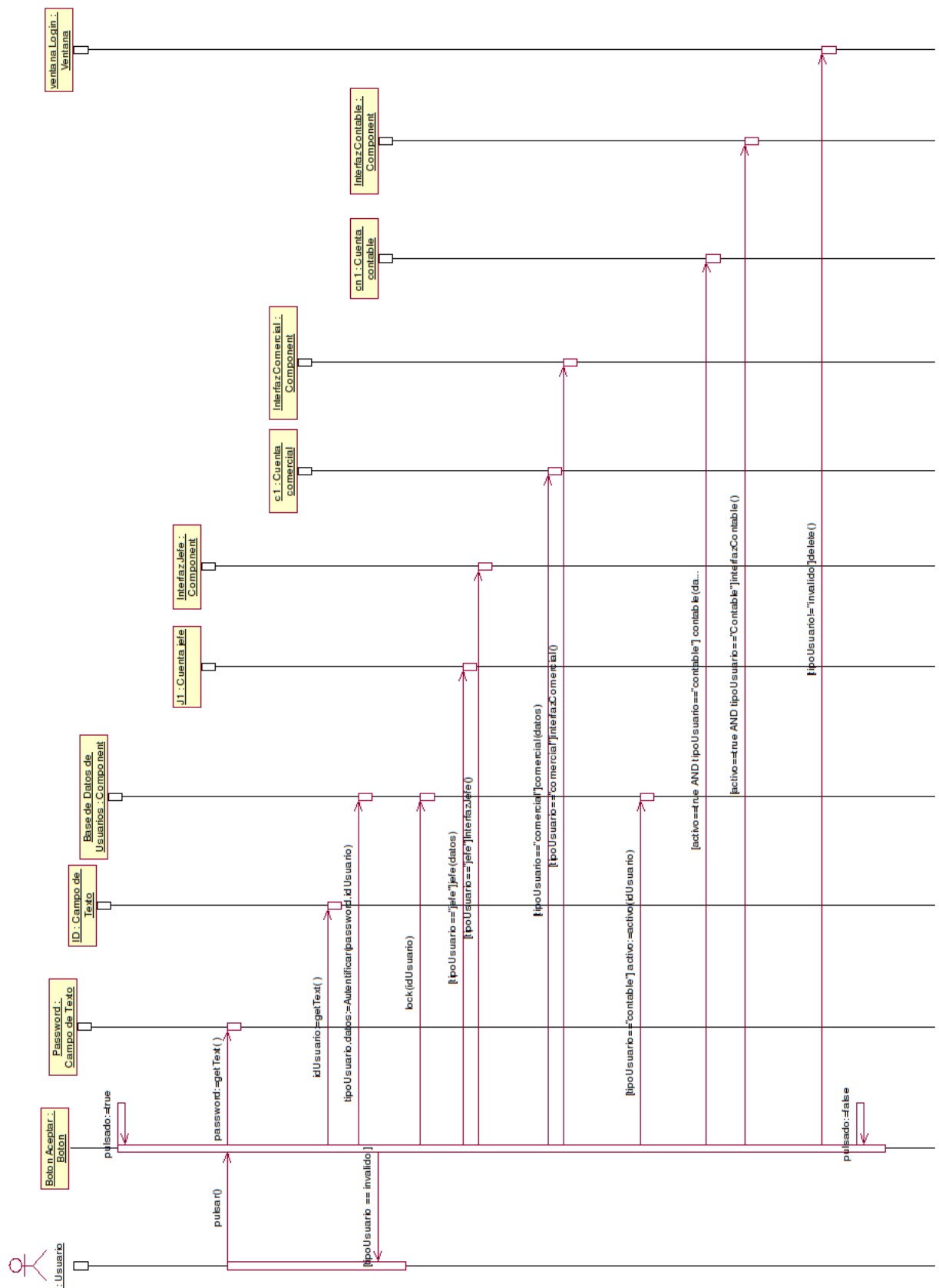
Lo importante en este diagrama es mostrar como el objeto, o lo que llamamos cuenta comercial, muere o desaparece de memoria. Sólo tendremos en memoria los objetos de las personas o cuentas que estén logueadas.





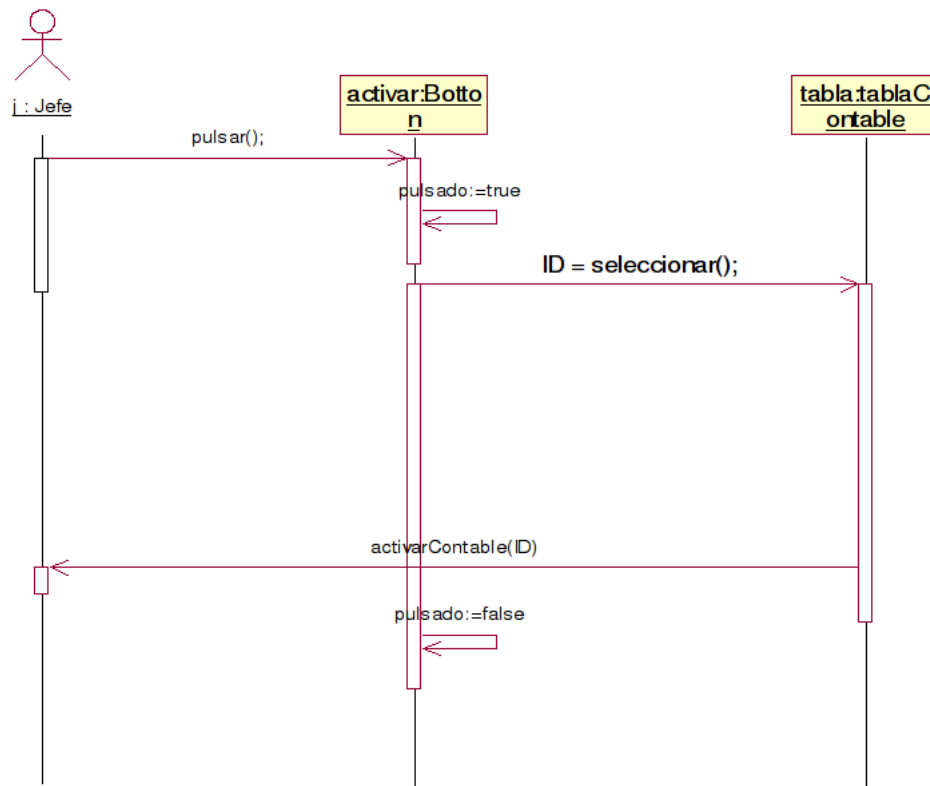
### **2.8.2. Diagrama de secuencias de hacer login**

Aquí hay que destacar la actuación del componente de la base de datos a la hora de autenticar el usuario que intenta acceder al sistema. De dar positivo el acceso, se verifica qué tipo de usuario es; si quien accede corresponde a un contable, éste tiene que estar activo. Si se cumplen estas restricciones, se crea un objeto en memoria que interactuará en el sistema. De lo contrario, se regresa a la ventana login.



### 2.8.3. Diagrama de secuencias de activar un contable

Aquí cabe destacar la interacción del jefe en el sistema al activar un contable.

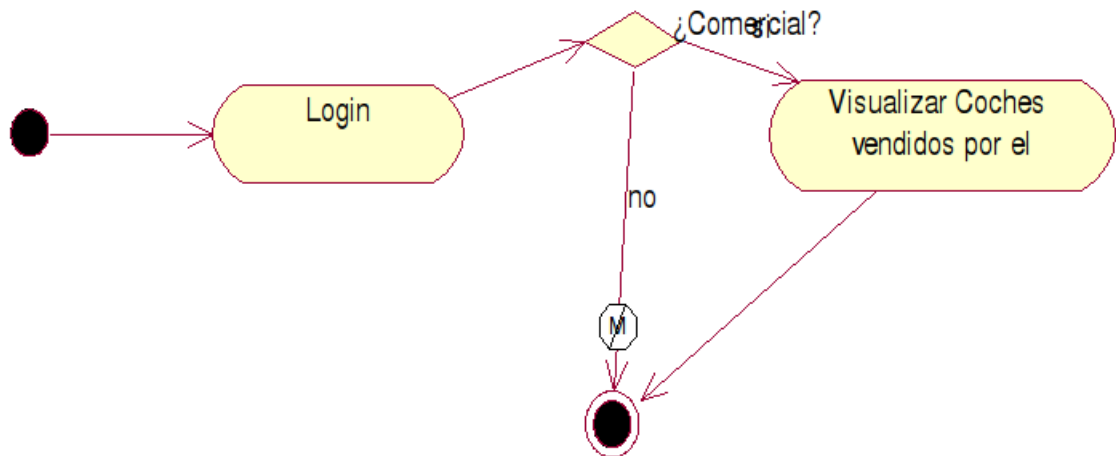


Los diagramas de secuencias que se han representado resumen lo más revelante de las operaciones en el sistema.

## 2.9. Diagrama de actividades visualizar coches vendidos por él mismo (comercial)

Para que un comercial pueda visualizar los coches vendidos por él es necesario que se haya logueado anteriormente, si el sistema lo reconoce como comercial la base de datos de coches le permitirá visualizar los coches vendidos por él comparando el identificador del vendedor (IDVendedor) con el identificador del comercial logueado.

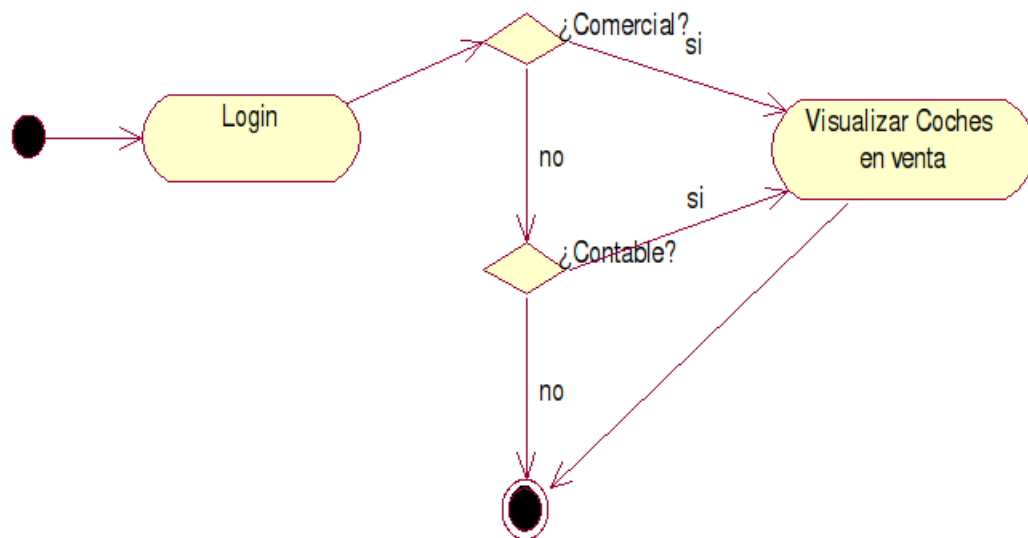
El comportamiento de la base de datos para esta operación queda explicado en el diagrama de secuencias “Visualizar Coches Vendidos”.



## 2.10. Diagrama de actividades de visualizar coches en venta

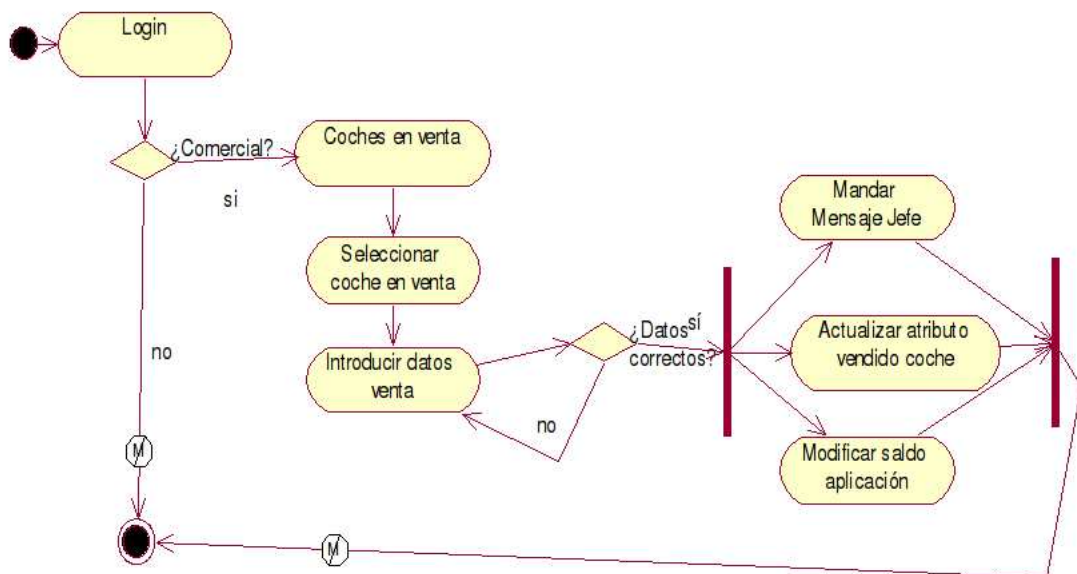
Para poder visualizar los coches que están en venta en el sistema, el usuario ha de loguearse primero para comprobar que se trata de un comercial o de un contable activo.

El jefe no tendrá pues, acceso directo a la base de datos de coches, pero ha sido informado mediante mensajes de todas las compras de coches que se han realizado por los contables y las ventas de coches que han realizado los comerciales de la empresa. Adicionalmente, el jefe siempre tiene la posibilidad de crearse él mismo una cuenta de contable y activarse a sí mismo.



## 2.11. Diagrama de actividades de vender coche

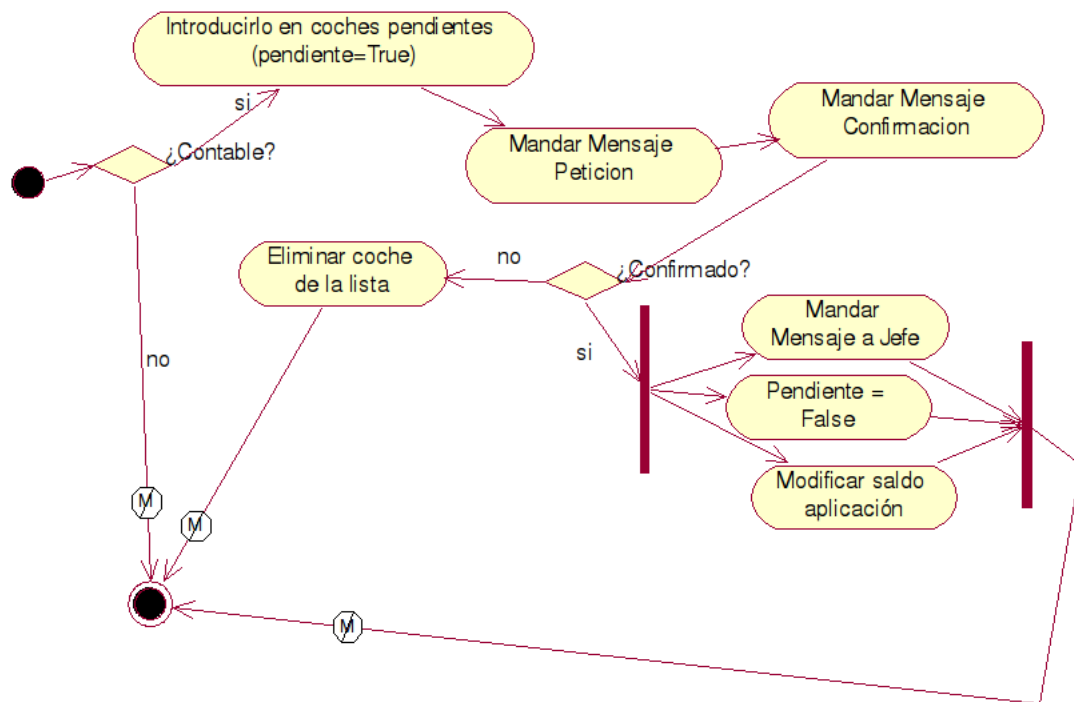
Los comerciales son los únicos empleados de la empresa con potestad para vender los coches a la venta del concesionario. Para ello, después de haberse logueado al igual que para poder efectuar cualquier operación que su cuenta ofrece, el comercial pasa a visualizar los coches que se ofrecen a la venta y que le proporciona la base de datos de coches. En ese momento podrá seleccionar el coche en el cual el cliente está interesado y realizar la venta. Esta operación es notificada al jefe vía mensaje.



## 2.12. Diagrama de actividades de añadir coche en lista de ventas

El contable es el único en la empresa que puede añadir coches a la lista de venta del concesionario. Para ello, éste introduce los datos del nuevo coche en la base de datos de los coches poniendo su atributo “pendiente” a verdadero de manera que queda pendiente de confirmar por el jefe. A este nivel del proceso, la compra del coche aún no se ha realizado.

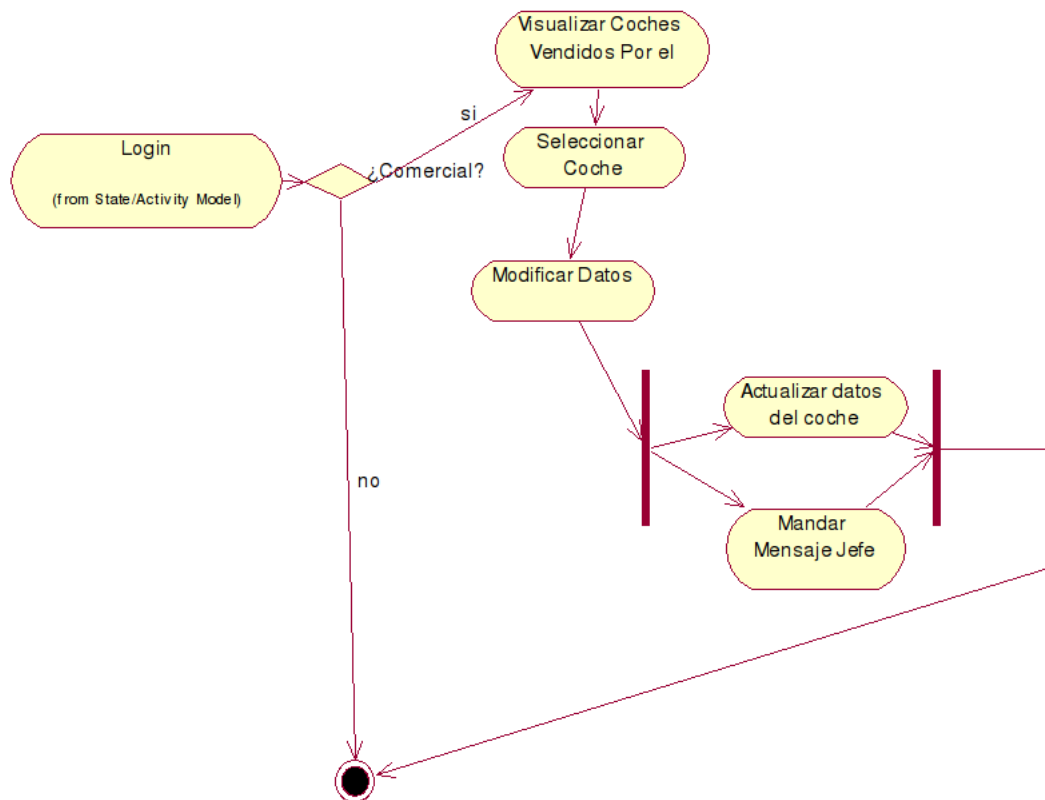
Un vez introducido el nuevo coche en la base de datos, el contable le envía un mensaje al jefe para que éste confirme o no la compra del coche. Si el jefe envía un mensaje de confirmación, el contable inicia la compra, que una vez realizada queda reflejada en la base de datos, poniendo el atributo “pendiente” a falso.





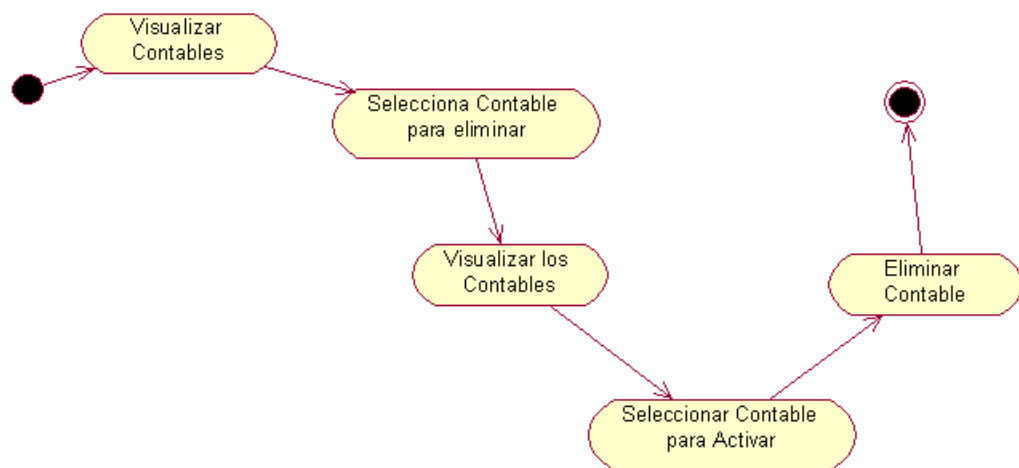
## 2.13. Diagrama de actividades de modificar datos de los coches vendidos por un comercial

Al comercial se le permite modificar los datos referentes a un coche que haya vendido. Para ello, al igual que ha sido explicado en el diagrama de actividades anterior: “Visualizar coches vendidos por él” es necesario autenticar a este comercial para que se asegure el hecho de que solamente pueda ver las ventas que éste comercial realiza.



## 2.14. Diagrama de actividades de eliminar el contable activo

El primer paso para eliminar un contable será mostrar la lista de contables. Tras seleccionar un contable para eliminar (que en este caso será el activo), habrá que seleccionar qué otro contable queremos activar. Después el sistema elimina el contable seleccionado (que en este caso era el activo).



## 2.15. Diagrama de componentes

Para llevar a cabo todo nuestro sistema necesitamos una serie de componentes:

### 2.15.1. Bases de Datos

- Base de Datos de Coches, donde se almacenan los coches del sistema con sus atributos correspondientes. Cualquier modificación de cualquiera de sus atributos debe ser actualizada en esta base de datos.
- Base de Datos de Comerciales, en ella se almacenan todos los comerciales del sistema, identificados por su ID. Cualquier modificación de ID o password de los comerciales al igual que cualquier dada de alta de un nuevo comercial debe ser actualizada en esta Base de Datos.
- Base de Datos de Contables, se almacenan todos los contables del sistema del mismo modo que los comerciales en la Base de Datos de Comerciales.
- Base de Datos de Usuarios, en esta Base de Datos están todos los usuarios del sistema. Existe esta Base de Datos, además de la de los comerciales y contables, para la operación de login, puesto que antes de que el usuario haga login no se conoce su puesto en la empresa y es necesario autenticar su ID y contraseña.  
Al igual que en las bases de contables y comerciales cualquier modificación debe ser actualizada en ésta, ya que ésta base de datos al fin y al cabo es una copia exacta de las dos anteriores.
- Base de Datos de Mensajes, se almacenan todos los mensajes indicando su remitente y destinatario, de forma que cada usuario solo podrá visualizar aquellos en los que él sea remitente o destinatario.

### 2.15.2. Interfaces

- Interfaz de login, es la primera que aparece al encender el sistema para que el usuario pueda loguearse, autenticarse y dependiendo de su puesto en la empresa darle la opción de realizar unas operaciones u otras.
- Interfaz de Jefe, se muestra cuando el sistema ha identificado al usuario como el Jefe del sistema.
- Interfaz de Contable, aparece cuando el usuario ha sido identificado como contable activo. En el caso de que no esté activo ni siquiera podrá loguearse.
- Interfaz de Comercial, figura una vez que el usuario ha sido autenticado como comercial.

### 2.15.3. Gestores

- Gestor de Jefe, Gestor de Comercial, Gestor de Contable, Gestor de Mensajes, Gestor de Coches: son, en realidad, los componentes que se encargan de gestionar las bases de datos, realizando las operaciones que se permiten en cada una de las interfaces.

### 2.15.4. Expendedor

- Expendedor, es el componente que se encarga de gestionar las transacciones entre el pago y las operaciones venta y compra realizadas por los comerciales o el contable respectivamente.

### 2.15.5. Pago

- Pago, se efectuará solo pago por tarjeta.

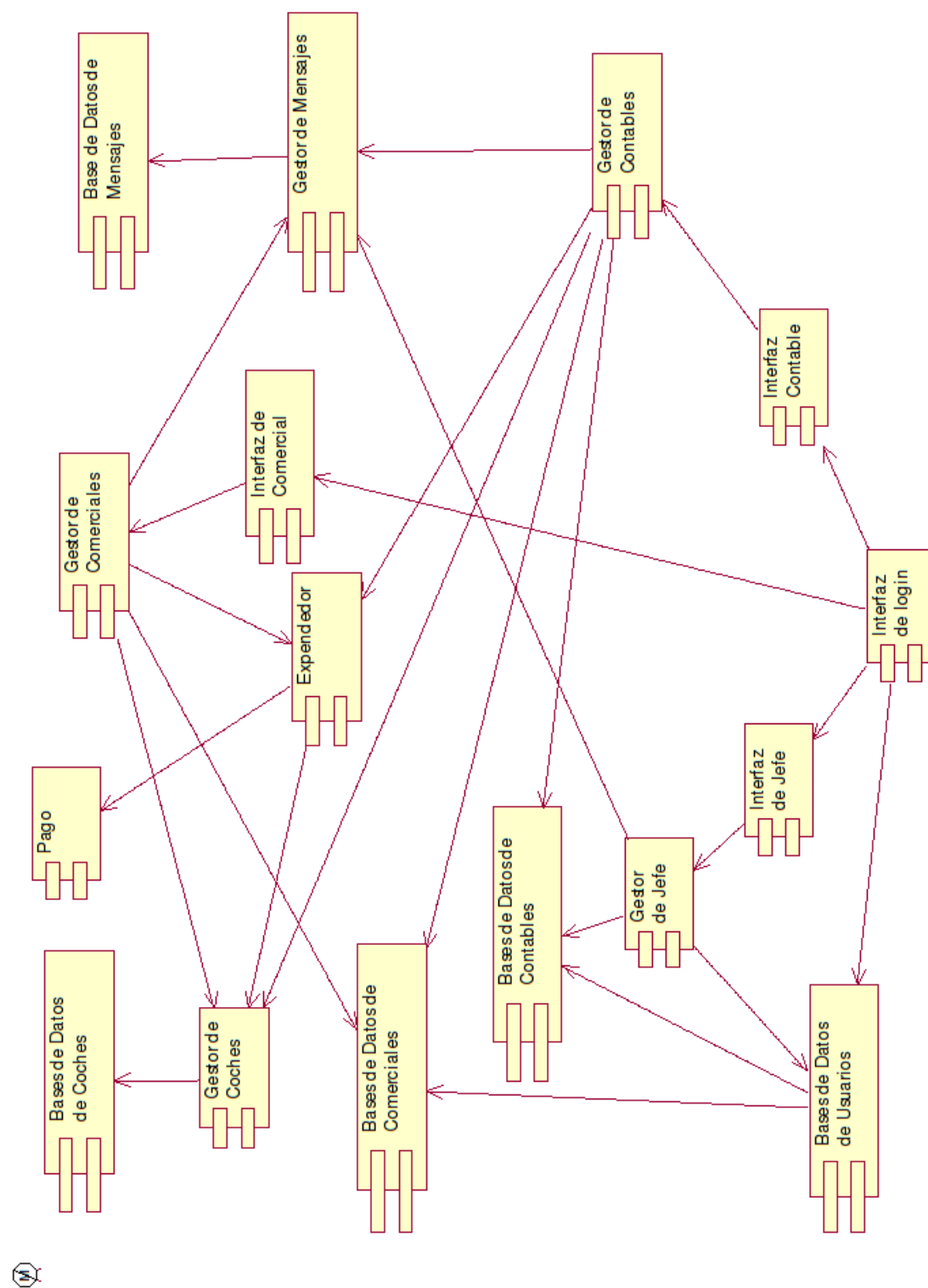


Figura 2.5: Diagrama de componentes

## Métodos que ofrecen los componentes

Función	Descripción
getLock(id: String)	Devuelve un booleano que indica si el usuario está logueado
desactivar(id: String )	Cambia el atributo del contable “activo ” a false
activar(id: String)	Cambia el atributo del contable “activo ” a true
existe?(id: String)	Comprueba la existencia de un identificador en la base de datos devolviendo un booleano
añadir(IDAux: String,pass: String,sueldo: int)	Añade un comercial en la base de datos
getActivo(IDelim: String)	Devuelve un booleano indicando si un contable está activo
eliminar(id: String)	Elimina a un contable de la base de datos
lock(id: String)	Establece que el usuario está logueado
unlock(id: String)	Establece que el usuario ha salido del sistema y puede gestionarse su cuenta
getPassword(IDconDefault: String)	Obtiene el password del contable de un identificador dado
getActivos()	Ejecutará una sentencia SQL que devolverá la lista de contables activos. El único objetivo de este método es ser utilizado en las pruebas

Cuadro 2.5: Base de Datos de los Contables

Función	Descripción
getLock(idAux: String)	Devuelve un booleano indicando si el usuario está logueado
eliminar(ID1: String)	Elimina a un comercial de la base de datos
lock(id: String)	Establece que el usuario está logueado
unlock(id: String)	Establece que el usuario ha salido del sistema y puede gestionarse su cuenta
getPassword(IDcomDefault: String)	Obtiene el password del contable de un identificador dado
existe?(id: String)	Comprueba la existencia de un identificador en la base de datos devolviendo un booleano

Cuadro 2.6: Base de Datos de los Comerciales

Función	Descripción
getFechaVenta(mat: integer)	Obtiene la fecha de venta de un coche determinado por su matrícula
eliminarCoche(mat: integer)	Elimina a un coche de la base de datos
getVendido(mat: integer)	Obtiene si un coche, determinado por su matrícula, está vendido
setDNIcomprador(dniComp: integer)	Modifica el dni del comprador del coche
setVendido(mat: integer,b: bool)	Modifica el atributo “vendido” del coche
dameCoches(id: String)	Devuelve la lista de los coches que ha vendido un comercial determinado por su identificador
getPVP(mat:integer)	Obtiene el precio del coche

Cuadro 2.7: Base de Datos de los Coches

<b>Función</b>	<b>Descripción</b>
Autenticar(s1: string, s2: string)	Función que devuelve el tipo de usuario(comercial, contable, jefe o inválido) y los datos del mismo
lock(id: string)	Determina que el usuario está logueado
unlock(id: String)	Determina que el usuario ha salido del sistema y puede gestionarse su cuenta
activo(id: String)	Comprueba si el usuario está activo en caso de que sea contable
existe?(id: String)	Comprueba la existencia de un identificador en la base de datos devolviendo un booleano

Cuadro 2.8: Base de Datos de los Usuarios

## 2.16. Pruebas

### 2.16.1. Pruebas de Regresión y Sistema

Cada vez que desarrollemos una nueva versión de la aplicación, será necesario ejecutar las pruebas de regresión para comprobar que determinados cambios en el código no han estropeado nada que antes funcionara.

De aquí en adelante explicaremos los valores que esperamos para determinadas acciones, así como en ciertos casos algunos ejemplos. Desarrollaríamos una batería de pruebas para que se probaran muchos casos muy diversos, tanto casos extremos como casos normales.

Hemos alternado entre ejemplos de resultados obtenidos/esperados y expresiones OCL que expresan el resultado esperado mediante pre y post condiciones.

#### ■ Jefe

##### 1. Añadir un contable a la base de datos

- El valor esperado del número de contables es el número de contables que había anteriormente más uno.
  - Correcto:
    - ◇ @pre: numContables == 4;
    - ◇ @post: numContables == 5;
  - Ejemplo incorrecto:
    - ◇ @pre: numContables == 4;
    - ◇ @post: numContables == 4;
- El contable añadido figura como una entrada en la base de datos pudiéndose visualizar sus atributos.
  - Correcto:
    - ◇ @pre: !BBDD → existe?("12345678");
    - ◇ @post: BBDD → existe?("12345678");
  - Ejemplo incorrecto:
    - ◇ @pre: !BBDD → existe?("12345678");
    - ◇ @post: !BBDD → existe?("12345678");
- Observar que se crea como inactivo.
  - Correcto:
    - ◇ @post: !BBDD → getActivo("12345678");
  - Ejemplo incorrecto:
    - ◇ @post: BBDD → getActivo("12345678");

##### 2. Al añadir un contable comprobar que el número de contables activos es uno, y no es el propio contable añadido.

- En la base sólo figura una entrada de cuenta contable con el atributo activo igual a true y no es el que acabamos de añadir.
  - inv: BBDD → getActivos().size() == 1;

##### 3. Eliminar un contable que no esté en el sistema en el momento de la acción.

- Al eliminar contable el número de contables disminuye en uno.
  - Correcto:
    - ◇ @pre: numContables == 4;
    - ◇ @post: numContables == 3;
  - Ejemplo incorrecto:
    - ◇ @pre: numContables == 4;
    - ◇ @post: numContables == 4;

- Los datos del contable eliminado no figure en la base de datos.
    - Correcto:
      - ◊ @pre: BBDD  $\rightarrow$  existe?("12345678");
      - ◊ @post: !BBDD  $\rightarrow$  existe?("12345678");
    - Ejemplo incorrecto:
      - ◊ @pre: BBDD  $\rightarrow$  existe?("12345678");
      - ◊ @pre: BBDD  $\rightarrow$  existe?("12345678");
4. Eliminar un contable que se encuentra dentro del sistema en el momento de la acción (solamente puede ocurrir si es el contable activo).
- Comprobar que no es eliminado.
    - Correcto:
      - ◊ @pre: numContables == 4;
      - ◊ @post: numContables == 4;
    - Ejemplo incorrecto:
      - ◊ @pre: numContables == 4;
      - ◊ @post: numContables == 3;
  - Los datos del contable siguen figurando en la base de datos.
    - Correcto:
      - ◊ @pre: BBDD  $\rightarrow$  existe?("12345678");
      - ◊ @post: BBDD  $\rightarrow$  existe?("12345678");
    - Ejemplo incorrecto:
      - ◊ @pre: BBDD  $\rightarrow$  existe?("12345678");
      - ◊ @post: !BBDD  $\rightarrow$  existe?("12345678");
  - El número de contables no disminuye.
    - Análogo al caso "Comprobar que no es eliminado".
  - El contable sigue pudiendo realizar operaciones normalmente.
5. Eliminar el último contable del sistema, sin contar con la cuenta por defecto. Satisface los requisitos SISTEMA2 y CONT2.
- El número de contables en el sistema es uno, que corresponderá a la cuenta por defecto (defaultContable).
    - @pre: numContables == 1;
    - @post: numContables == 1;
  - Se le administra al jefe los datos relativos a esta cuenta (ID y Password por defecto) mediante un mensaje informativo.
  - Esta cuenta debe poseer el atributo activo a true en la base de datos.
    - @pre: !BBDD  $\rightarrow$  getActivo("defaultContable");
    - @post: BBDD  $\rightarrow$  getActivo("defaultContable");
  - El contable tiene que haber sido eliminado.
    - @pre: BBDD  $\rightarrow$  existe?("12345678");
    - @post: !BBDD  $\rightarrow$  existe?("12345678");
6. Eliminar un contable activo (El número de contable en la base de datos es mayor que dos). Satisface el requisito CONT1.
- Al eliminar contable el número de contables disminuye en uno.
    - Análogo a eliminar un contable no activo.
  - Los datos del contable eliminado no figure en la base de datos.
    - @pre: BBDD  $\rightarrow$  existe?("12345678");
    - @post: !BBDD  $\rightarrow$  existe?("12345678");
  - Observar que el número de contables activos en el sistema es uno. El nuevo contable habrá sido el elegido por el jefe.



- @post: BBDD  $\rightarrow$  getActivos().size() == 1;
- 7. Modificar datos relativos a un contable (Ejemplo: sueldo).
  - Comprobar que los datos modificados se actualizan en la base de datos inmediatamente.
    - Ejemplo correcto (sueldo):
      - ◊ @post: BBDD  $\rightarrow$  sueldo("12345678")  $\neq$  BBDD  $\rightarrow$  sueldo("12345678")@pre;
    - Ejemplo incorrecto (sueldo):
      - ◊ @post: BBDD  $\rightarrow$  sueldo("12345678") == BBDD  $\rightarrow$  sueldo("12345678")@pre;
- 8. Insertar dinero en la aplicación
  - Ingresar una cantidad de dinero y comprobar que el contable puede ver esta cifra.
    - @post: capital > capital@pre;
  - El contable podrá usar este dinero en las operaciones que así lo requieran.
  - Se puede ver el capital de la aplicación actualizado.
- 9. Activar y desactivar contables
  - El número de contables en la base de datos permanece sin modificar.
    - @post: numContables == numContables@pre;
  - El contable que se desactiva tiene el atributo correspondiente a false y el que se activa, tiene su atributo a true.
    - @pre: BBDD  $\rightarrow$  getActivo("12345678") && !BBDD  $\rightarrow$  getActivo("87654321");
    - @post: !BBDD  $\rightarrow$  getActivo("12345678") && BBDD  $\rightarrow$  getActivo("87654321");

## ■ Contable

Todas las pruebas relativas a añadir, eliminar y modificar comercial, son análogas a las realizadas por el jefe

1. Cerciorarse de que cuando el jefe da el visto bueno a la adquisición de un coche, éste se añade a la lista de coches en venta
  - El número de coches de la base de datos es el que había anteriormente más uno
  - Se tiene que ver el coche añadido (así como sus atributos) en la base de datos
  - Comprobar que ese coche se pueda vender
2. Modificar un atributo de un coche en la lista de ventas
  - El coche debe seguir estando en la base de datos
  - El número de coches de la base debe permanecer invariante
  - Una vez cambiado el atributo, debe verse la modificación en la base de datos
3. Visualizar los coches de la lista de ventas
  - El número de coches de la base de datos debe permanecer sin variación
  - Sólo se deben ver aquellos coches cuyo atributo de vendido sea false y pendiente sea false también
4. Eliminar un coche que haya sido vendido en un período inferior a tres años
  - El coche no se puede eliminar de la base de datos correspondiente
5. Eliminar un coche que haya sido vendido en un período superior a tres años
  - El número de coches de la base de datos se reduce en uno
  - No se puede acceder a los atributos de este coche en la base de datos
6. Acceder con un contable inactivo
  - No se puede acceder
7. Eliminar el último comercial. Satisface el requisito SISTEMA1
  - El número de comerciales es uno, que corresponderá a la cuenta por defecto del comercial (defaultComercial)
    - @pre: numComerciales == 1;
    - @post: numComerciales == 1;

- Se le administra al contable activo los datos relativos a esta cuenta (ID y Password por defecto) mediante un mensaje informativo
  - El comercial tiene que haber sido eliminado
    - @pre: BBDD  $\rightarrow$  existe?("12345678");
    - @post: !BBDD  $\rightarrow$  existe?("12345678");
8. Eliminar un comercial que no sea el último y que no esté identificado en el sistema en el momento de la acción
- Comprobar que el número de comerciales se ha disminuido en 1
    - @post: numComerciales == numComerciales@pre - 1;
  - Comprobar que el comercial eliminado ha sido el esperado
    - @pre: BBDD  $\rightarrow$  existe?("12345678");
    - @post: !BBDD  $\rightarrow$  existe?("12345678");
9. Eliminar un comercial que no sea el último y que esté identificado en el sistema en el momento de la acción
- Comprobar que no es eliminado
    - @post: numComerciales == numComerciales@pre;
    - @post: BBDD  $\rightarrow$  existe?("12345678");

#### ■ Comercial

1. Comprobar que se ha vendido un coche
- El número de coches cuyo atributo vendido es false es uno menos del que había anteriormente
  - Debe seguir viéndose en la base de datos hasta que haya transcurrido tres años, en cuyo caso el contable podrá eliminarlo

#### ■ Servicio de Mensajería

1. Mandar un mensaje
- Comprobar que el remitente y el destinatario son usuarios que están actualmente dados de alta en el sistema
  - Comprobar que el mensaje se encuentre en la bandeja de entrada del destinatario
  - Comprobar que el mensaje puede ser leído por el destinatario
  - Comprobar que al destinatario se le muestra el remitente del mensaje y el asunto
  - El mensaje debe estar en la bandeja de salida del remitente
2. Guardar un mensaje
- Comprobar que el mensaje guardado no se envía
  - El mensaje se encuentra entre los borradores del creador del mensaje
3. Eliminar un mensaje
- El número de mensajes que le quedan al usuario es uno menos de los que había antes

#### ■ General

- Intentar acceder con una cuenta, identificador, que no se encuentre en la base de datos
  - No se permite el acceso

### 2.16.2. Pruebas de la interfaz gráfica

#### ■ Ventana Login

- Comprobar que se puede escribir en ambos cuadros de texto
- Comprobar que se pueden pulsar ambos botones, pero nunca los dos a la vez
- Comprobar que únicamente esté activo el botón login en el caso en el que ambos campos estén rellenos

- Comprobar que la conexión con el componente respectivo (base de datos) funciona correctamente; es decir, verifica la existencia del usuario y después de cumplir esto que la contraseña corresponda al usuario. Posteriormente deberá abrirse la interfaz correspondiente al tipo de usuario: para el contable la interfaz contable, para comercial la interfaz comercial y para el jefe su respectiva interfaz
  - Si pulsamos el botón “salir” se abandonará la aplicación (este botón tendrá la misma funcionalidad para todas las ventanas).
- Interfaz Usuario
    - Al pulsar el botón “atrás” desaparecerá esta ventana y volveremos a la ventana de login
    - Verificar que cada botón despliega la ventana correspondiente al texto escrito en él
- Interfaz Comercial
    - Al hacer clic sobre el botón “Lista de coches” deberá aparecer una ventana solamente con la lista de los coches que ha vendido el propio comercial
- Ventana Gestión (Genérico)
    - Comprobar que los botones que modifican los datos de la base de datos no estarán activos hasta que no se haya seleccionado una fila de la tabla
    - El botón “Eliminar” borrará el elemento seleccionado de la tabla
    - Comprobar que todos los elementos correspondientes a la ventana de gestión de la base de datos aparecen en la tabla
    - Al pulsar el botón “Añadir” aparecerá una nueva fila en la tabla
    - Al pulsar el botón “Aceptar” se guardarán los cambios realizados en la tabla en la base de datos siempre y cuando sea información válida, si no, se mantendrán los elementos con la configuración que tenían anteriormente
    - Todas las filas de la tabla pueden ser modificadas directamente al escribir sobre los campos de cada una de ellas
- Ventana de Gestión de Contables
    - Sólo habrá uno de los contables con el checkBox “Activo” de la columna correspondiente de la tabla
    - Al pulsar el botón “Activar” se quedará seleccionada su checkBox “Activo” y ya no estará seleccionado el checkBox del anterior contable que lo tenía deseleccionado
    - Al pulsar el botón “Desactivar” se deselecciona el checkBox “Activo” del contable
- Ventana Modificar Cuenta
    - No estará activo el botón guardar hasta que no se haya escrito en los campos de texto “New-Password” y “RepitaPassword” y además haya el mismo texto
    - Al pulsar Guardar se modificará el campo Password del usuario correspondiente en la base de datos
- Interfaz de Mensajes
    - Comprobar que se garantiza la privacidad, es decir, que un usuario no tiene acceso a los mensajes de otros usuarios
- Ventana de Envío de Mensajes
    - No se permite enviar el mensaje hasta que no se haya puesto el destinatario
    - En el campo de texto del emisor debe venir escrito el del usuario propio
    - Al pulsar el botón guardar debe guardarse el mensaje en la bandeja de borradores

### 2.16.3. Pruebas específicas

1. Usabilidad: para desarrollar las pruebas de usabilidad, lo ideal sería entregar una versión de evaluación de nuestra aplicación al cliente, y comprobar que las herramientas son intuitivas y que son usables en todos los aspectos y situaciones.
2. Carga (rendimiento): comprobamos el correcto funcionamiento del sistema cuando múltiples usuarios se conectan a la vez e interactúan entre sí y con la base de datos.
3. Aceptación: le proporcionaremos las pruebas de regresión y sistema que hemos especificado anteriormente al cliente, para que él (o un grupo independiente que lo represente) pueda ejecutarlas y validar que el sistema realmente cumple su propósito.
4. Sistema: las pruebas de regresión pueden ser utilizadas en este sentido, para comprobar que el sistema en realidad tiene la reacción esperada en cada caso.
5. Integración:
  - **Componente de bases de datos**  
Podríamos diseñar una batería de pruebas para realizarlas sobre este componente y asegurarnos que la información que hay en la base de datos se corresponde con lo esperado.
  - **Componente de pago**  
Si utilizáramos el componente de pago de PayPal existe una manera de crear cuentas para pruebas, y así desarrollar nuestras propias pruebas utilizando dichas cuentas. Para entrar y crear cuentas de prueba hay que iniciar sesión en <http://developer.paypal.com> con la dirección de correo “grupoisucm@hotmail.com” y contraseña “grupoisucm”. Existe la opción tanto de crear la cuenta de prueba de comprador como de vendedor.  
  
Gracias a estas pruebas podríamos comprobar si realmente las transacciones se han efectuado correctamente, tanto si son ingresos (venta de coches) como pagos (compra de coches).
6. Regresión: Como hemos comentado anteriormente, desarrollaríamos una batería de pruebas para los casos anteriormente citados, con muchos tipos de casos diferentes, tanto casos extremos como casos medios (ya que normalmente serán los más habituales que se darán en el sistema).

Nos hemos documentado acerca de la API que podríamos utilizar para escribir nuestras pruebas y hemos encontrado que QTestLib (<http://doc.trolltech.com/latest/qtestlib-manual.html>) satisface nuestras necesidades.

Esta API nos permite crear baterías de pruebas, y todas ellas especificando el valor esperado de una determinada operación o llamada a método.

## 2.17. Apariencia de la interfaz

### 2.17.1. Interfaz Comercial

Cuando un usuario se autentifica como comercial, aparecería la siguiente interfaz . En ella, el usuario podrá acceder a la lista de coches y a podrá ver su cuenta de mensajes. Desde esta pantalla se podrá modificar su cuenta y si lo desea, abandonar su sesión.

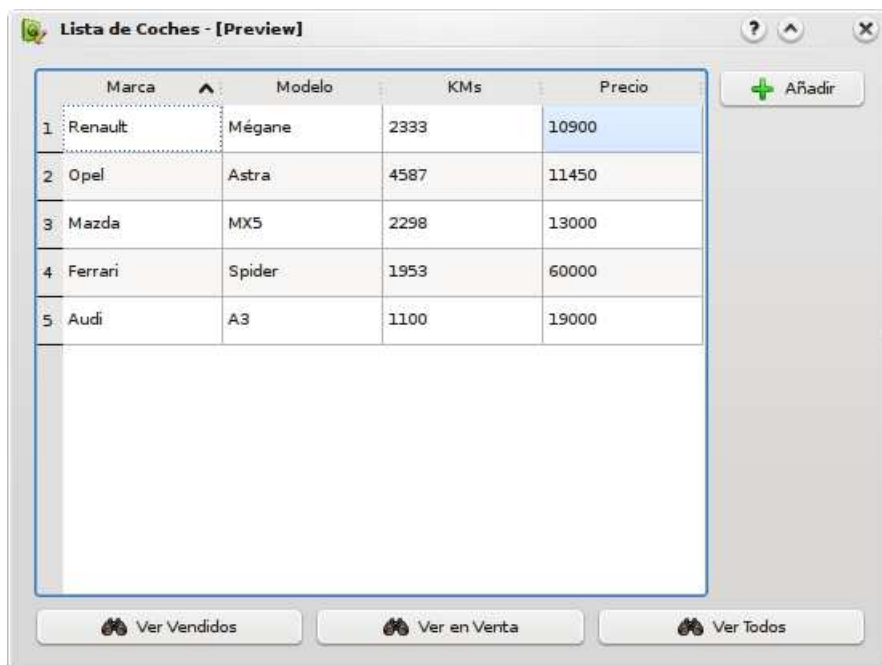


### 2.17.2. Interfaz Base de Datos de Coche

Tras pulsar el botón de lista de coches (ver interfaz comercial), aparecerá la ventana siguiente. En ella se mostrará, en principio, todos los coches. En ella se podrán modificar los datos relativos a un coche sin más que hacer doble clic sobre uno de los campos de la tabla.

En caso de que se quiera añadir una nueva entrada, se pulsará el botón añadir, se añadirá una nueva fila y el usuario tendrá que ir modificando cada uno de los campos.

También se podrá visualizar los coches que hayan sido vendidos por él mismo pulsando “Ver Vendidos”, ver los automóviles que están actualmente a la venta (“Ver en Venta”) y, por último, ver tanto los coches que están en venta como los vendidos por él pulsando “Ver todos”.



### 2.17.3. Interfaz Contable

Cuando un usuario se autentifica como contable, aparecería la siguiente interfaz . En ella, el usuario podrá acceder a gestionar comerciales y coches. A esta ventana sólo podrá acceder el contable que está activo actualmente.

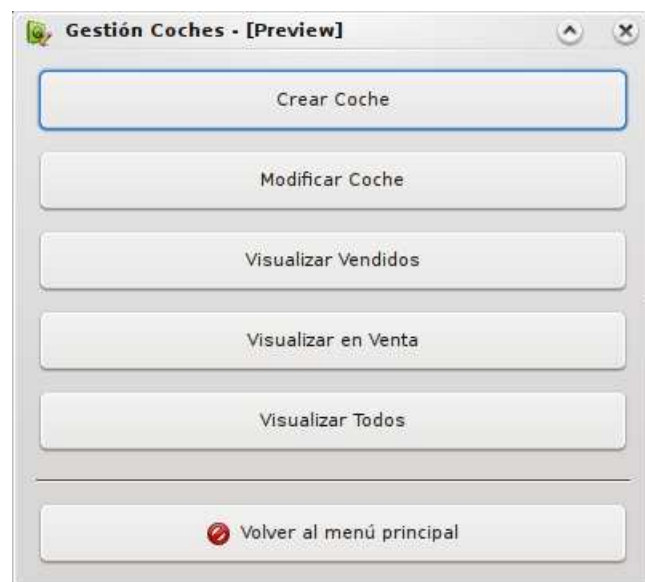
Podrá acceder a su cuenta de correo, modificar su cuenta y abandonar la sesión actual.



#### 2.17.4. Interfaz de Gestión de Coches

Una vez que el usuario pulse el botón Gestión de coches (ver interfaz Contable) aparecerá la siguiente ventana.

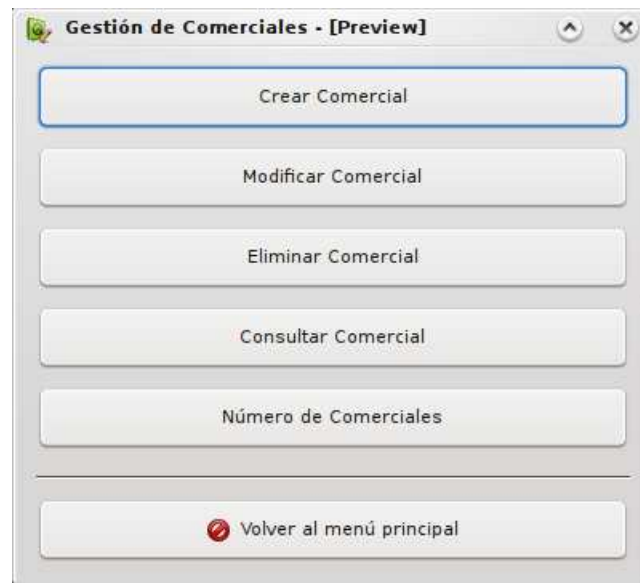
Desde ella se podrá crear y modificar un coche. Asimismo, se podrán visualizar los coches que hayan sido vendidos, que estén actualmente en venta y ver todos los coches (tanto vendidos como en venta). Y pulsando sobre “Volver al menú principal”, retornará a la Interfaz de Contable.



#### 2.17.5. Interfaz Gestión de Contables

Esta ventana es de acceso único para las cuentas de contables, además que sólo tendrá acceso el contable activo. Desde ésta ventana se puede crear, modificar, eliminar, consultar y saber el número de comerciales. Al pulsar cada uno de los botones que se muestran posteriormente irán apareciendo las interfaces oportunas.

El botón “Volver al menú principal” vuelve a la ventana principal de la cuenta de este usuario.



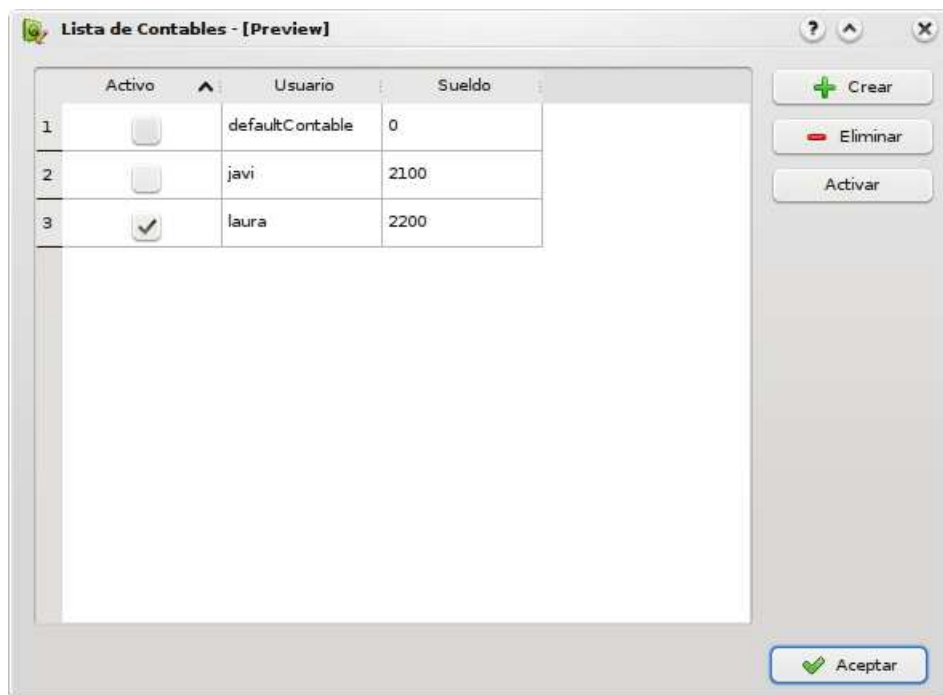
### 2.17.6. Interfaz Jefe

Este es el menú para elegir las diferentes opciones de las cuáles dispone el jefe. Es la primera ventana que se mostraría cuando al identificarse el jefe. El botón “Cerrar Sesión” abandonaría la cuenta de jefe y se volvería a Mostrar la ventana de login.



### 2.17.7. Interfaz Lista Contables

A esta ventana se llegará pulsando la opción “Gestión Contable” que se muestra en la ventana de Jefe anterior. En su parte central mostrará todos los contables con su correspondiente información. El jefe podrá crear un nuevo contable, modificar o eliminar uno de ellos. Cuando se pulse el botón “Aceptar” se guardarán todos los cambios realizados.



### 2.17.8. Interfaz Login

Esta ventana es la primera que se mostrará al iniciar la aplicación. En ésta el usuario se identificará. En el caso en el que sea un usuario válido se mostrará la ventana correspondiente con el tipo de cuenta de este usuario. Aunque un contable tenga su cuenta y quiera introducirse en el sistema si no está activo no podrá hacerlo. Si en vez de pulsar “Aceptar” se pulsa el botón salir el cliente abandonaría la aplicación.



### 2.17.9. Interfaz Modificar Cuenta

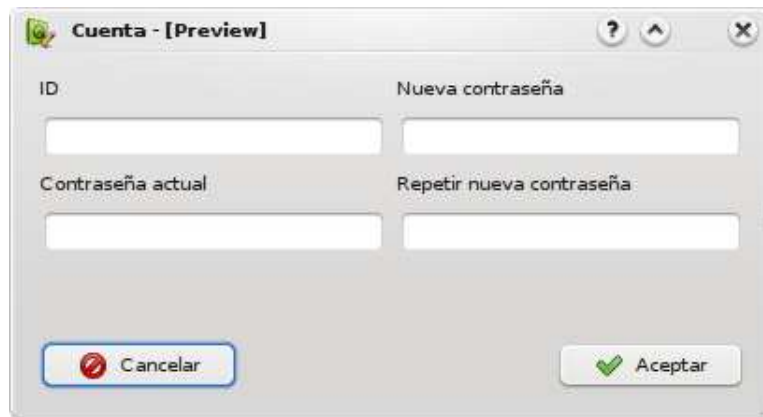
Este diálogo sirve para cambiar la contraseña del usuario. Inicialmente en el campo de texto de “ID” aparecerá el identificador del usuario y este campo no se podrá modificar. Se tendrá que rellenar el campo “Contraseña actual” con la contraseña actual, y los campos “Nueva Contraseña” y “Repetir nueva contraseña”.

Al pulsar el botón “Aceptar” se comprobará si el campo “Contraseña actual” coincide con la contraseña asociada al identificador del usuario actual. Si es así se comprobará si los otros dos campos están rellenos y además conciden. Únicamente en este caso se realizará el cambio oportuno de contraseña en la base de datos. El identificador de usuario no podrá ser modificado en ningún caso.

Si se pulsa el botón “Cancelar” ningún cambio tendrá efecto y el sistema quedará en el mismo estado que estaba anteriormente.

En cualquier caso (pulsar “Aceptar” o “Cancelar”) volveríamos a la ventana que mostró esta interfaz.





**Cuenta - [Preview]**

ID

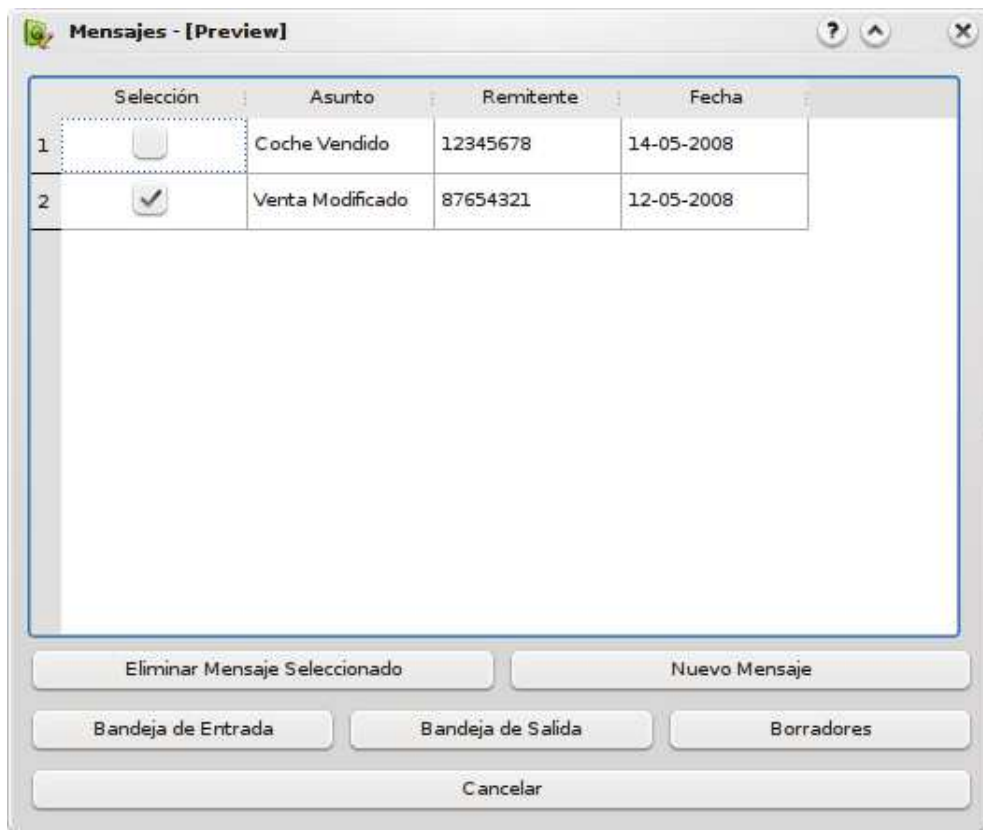
Nueva contraseña

Contraseña actual

Repetir nueva contraseña

### 2.17.10. Interfaz principal mensajes

Este diálogo es mostrado tras pulsar el botón “Mensajes” desde cualquier menú principal de cualquier tipo de usuario. Desde aquí se podrán visualizar todos los mensajes, tanto los recibidos como los enviados y guardados. Se podrán eliminar los mensajes que hayan sido seleccionados. Haciendo doble clic sobre una fila, se mostrará el cuerpo del mensaje en una ventana separada.

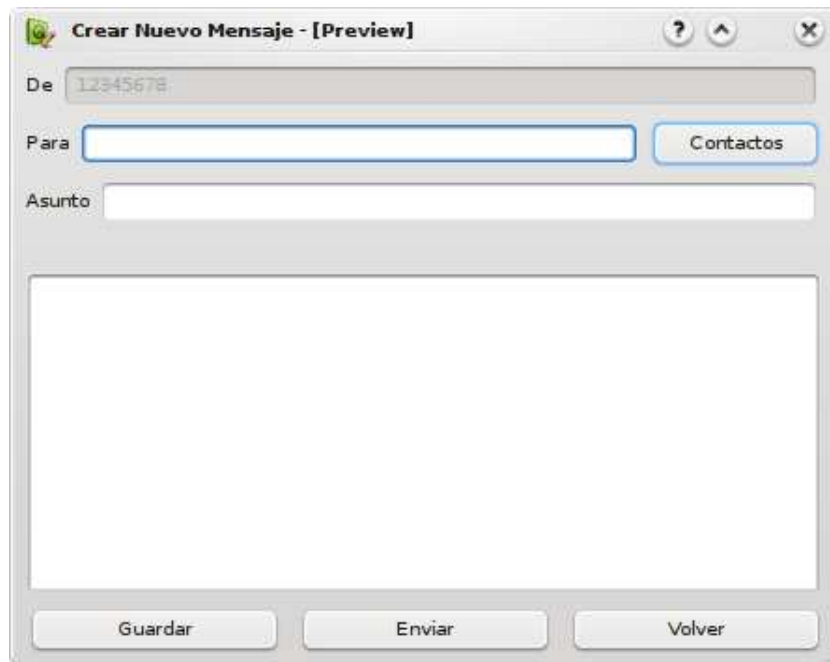


**Mensajes - [Preview]**

	Selección	Asunto	Remitente	Fecha
1	<input type="checkbox"/>	Coche Vendido	12345678	14-05-2008
2	<input checked="" type="checkbox"/>	Venta Modificado	87654321	12-05-2008

### 2.17.11. Interfaz nuevo mensaje

Esta interfaz se mostrará al pulsar el botón “Nuevo” en la interfaz principal de mensajes. Se mostrará el remitente (el identificador del usuario) en un cuadro de texto que no podrá ser editado. Podrá seleccionar el/los destinatarios del mensaje pulsando sobre el botón “Contactos” o escribirlos él mismo, y rellenar el asunto y cuerpo del mensaje.



## 2.18. Invariantes de la Interfaz

A continuación se muestran los invariantes más relevantes de las interfaces de nuestro sistema.

### 2.18.1. Interfaz principal mensajes

Context `VentanaMensaje` inv:

```
self.titulo="Ventana Mensajes"

self.botones->size() = 7
self.botones->exist(b:Boton| b.texto="Eliminar");
self.botones->exist(b:Boton| b.texto="Redactar");
self.botones->exist(b:Boton| b.texto="Bandeja de Entrada");
self.botones->exist(b:Boton| b.texto="Bandeja de Salida");
self.botones->exist(b:Boton| b.texto="Borradores");
self.botones->exist(b:Boton| b.texto="Salir");
self.botones->exist(b:Boton| b.texto="Atrás");

self.tablas->size()=1;
self.tablas.checkbox=false;

self.botones->select(b Boton| (b:Boton.pulsado=(true)))>->size()=1.
```

### 2.18.2. Interfaz login

Context `VentanaLogin`

```
getBoton(aux:String): Boton
body:
self.botones->select(n| n.Nombre=aux)

inv:
(self.getBoton("Login").activo == true) implies self.cuadroDetexto->forAll(c:cuadroDetexto |
c.texto <> "")
```

### 2.18.3. Interfaz ventana gestión contable

Context VentanaGestionContable

```
listaBotonesModificadores(): set [botones]
body:
self.botones->collect(b| b.nombre<>"crear" and b.nombre<> "numContables")

inv:
self.tablas->size()==1;
self.tablas.checkbox=false;
self.botones->select(b Boton|(b:Boton.pulsado=(true)))->size()==1.
self.listaBotonesModificadores->forall(b| b.activo=true) implies (self.tabla.seleccionada()<> 0 )
```

### 2.18.4. Interfaz mandar mensajes

Context VentanaRedactar

```
getBotton(aux:String): Boton
body:
self.botones->select(n| n.Nombre=aux)

inv:
self.getBotton("Enviar").activo=true implies self.CuadroDeTextoDestinatario.texto <> "".
```

### 2.18.5. Interfaz modificar cuenta

```
self.getBotton("Aceptar").activo=true implies self.CuadroDeTextoNuevoPassword.texto <> "" and
(self.CuadroDeTextoNuevoPassword.texto=self.CuadroDeTextoRepetirPassword.texto ).
```

## 2.19. Investigación de los componentes utilizados en el sistema

### 2.19.1. Propuestas para el componente de pago

1. La primera propuesta es SWIFT Message Suite. Es el paquete más completo de la aplicación SWIFT MESSAGE y, aunque su precio sea elevado (9,612€), ofrece diversos servicios para el análisis y la validación del componente y para comprobar su funcionamiento correcto. Para más información y detalles de las componentes de la aplicación se puede consultar la página web siguiente:  
[http://www.paymentcomponents.com/swift/message\\_suite/components\\_suite.html](http://www.paymentcomponents.com/swift/message_suite/components_suite.html)
2. Como segunda propuesta recomendamos los servicios que ofrece PayPal cuyos pagos son gestionados vía mail. Esta aplicación no tiene ningún coste inicial pero al servicio de PayPal le corresponde un pequeño porcentaje de las transacciones que se realicen a través de su aplicación (De 1.9 % a 2.9 % + \$0.30 USD). También nos ofrece la posibilidad de realizar pruebas de integración muy fiables para su evaluación. Para más información se puede consultar la web:  
[https://www.paypal.com/us/cgi-bin/webscr?cmd=\\_email-payments-overview-outside](https://www.paypal.com/us/cgi-bin/webscr?cmd=_email-payments-overview-outside)

### 2.19.2. Propuestas para el componente de bases de datos

1. Para este componente hemos decidido ofrecer el servicio que nos ofrece la aplicación MySQL. El servicio ofrecido es uno de los más fiables y seguros que existen actualmente, con un alto grado de adaptabilidad. Además tiene gran importancia la oferta de paquetes determinados para usarlos junto a aplicaciones de pago online. El precio del paquete más completo es 479 euros. Para obtener más detalles consultar:  
<http://www.mysql.com/>
2. La segunda opción es EnterpriseDB, una aplicación muy completa basada en PostgreSQL. Su coste es de 995\$ (mejor opción escogida detallada en

<http://www.enterprisedb.com/shop.do?cID=10013&pID=10015>). Abarca la mayor parte de los casos de uso más comunes y es notablemente simple de instalar y usar. Para más información consultar: <http://www.enterprisedb.com/products/index.do> Solamente: 995\$  
(<http://www.enterprisedb.com/shop.do?cID=10013&pID=10015>)

## 2.20. Implementación

Toda nuestra información estará acumulada en una base de datos. La base de datos tendrá cinco tablas, una para cada tipo de información diferente: coches, contables, comerciales, mensajes y usuarios. La base de datos estará controlada por varios gestores, éstos se implementarán mediante un patrón singleton.

### 2.20.1. Información sobre las tablas

#### 1. Coches

Esta tabla tendrá varias columnas: fabricante del coche, modelo, kilometraje, año de fabricación, matrícula del coche, color, número de bastidor, precio con el que fue comprado el coche por el concesionario, precio de venta, fecha de venta, DNI del comprador, si el coche ha sido vendido o no y por qué comercial ha sido vendido.

#### 2. Mensajes

Hemos decidido que esta tabla tenga los siguientes campos: destinatario, remitente, asunto, mensaje, fecha y si ha sido leído o no.

#### 3. Usuarios

Esta tabla contendrá seis columnas: nombre, contraseña, tipo de usuario, sueldo, activo y capital. En esta tabla estarán todos los usuarios introducidos, tanto comerciales como contables y el jefe. El tipo de usuario podrá ser “comercial”, “contable” o “jefe”. Activo será siempre falso en usuarios que no sean de tipo “contable”, y únicamente habrá un tipo “contable” que tenga este campo a cierto. Capital será 0 en todos los tipos de usuario, excepto en “jefe”, que es donde se almacenará el capital de la aplicación.

Por defecto la base de datos contendrá tres cuentas: “defaultComercial”, “defaultContable” y “cuentaJefe”. La contraseña de “cuentaJefe” será “cuentaJefe” por defecto. Por defecto las cuentas “defaultComercial” y “defaultContable” tendrán las contraseñas “defaultComercial” y “defaultContable” respectivamente. Se sugerirá encarecidamente al jefe que cambie las contraseña cuando el sistema sea instalado.

Las contraseñas estarán encriptada en MD5 en la tabla por motivos de seguridad. Siempre que éstas se requieran (tanto para login como para ser modificadas), primero se encriptarán en MD5 y luego se compararán con este campo. Todas las llamadas a la base de datos se realizarán utilizando el lenguaje SQL.

### 2.20.2. Lenguaje utilizado

Utilizaremos el lenguaje C++ con el toolkit Qt (framework para desarrollar aplicaciones multi-plataforma) para desarrollar nuestra aplicación. Existen dos posibilidades principalmente que podríamos exponer al cliente:

- Desarrollar la aplicación como software libre con una licencia no restrictiva (como GPL o LGPL), por lo que el coste de utilizar las librerías de Qt sería 0 €. GPL o LGPL en este sentido significa proporcionar el código al usuario final (en este caso nuestro cliente), no únicamente el ejecutable.
- Desarrollar la aplicación como código cerrado (no entregamos el código fuente a nuestro cliente), con lo que tendríamos que adquirir una licencia de Qt, que asciende a la cifra estimada de 600 €.