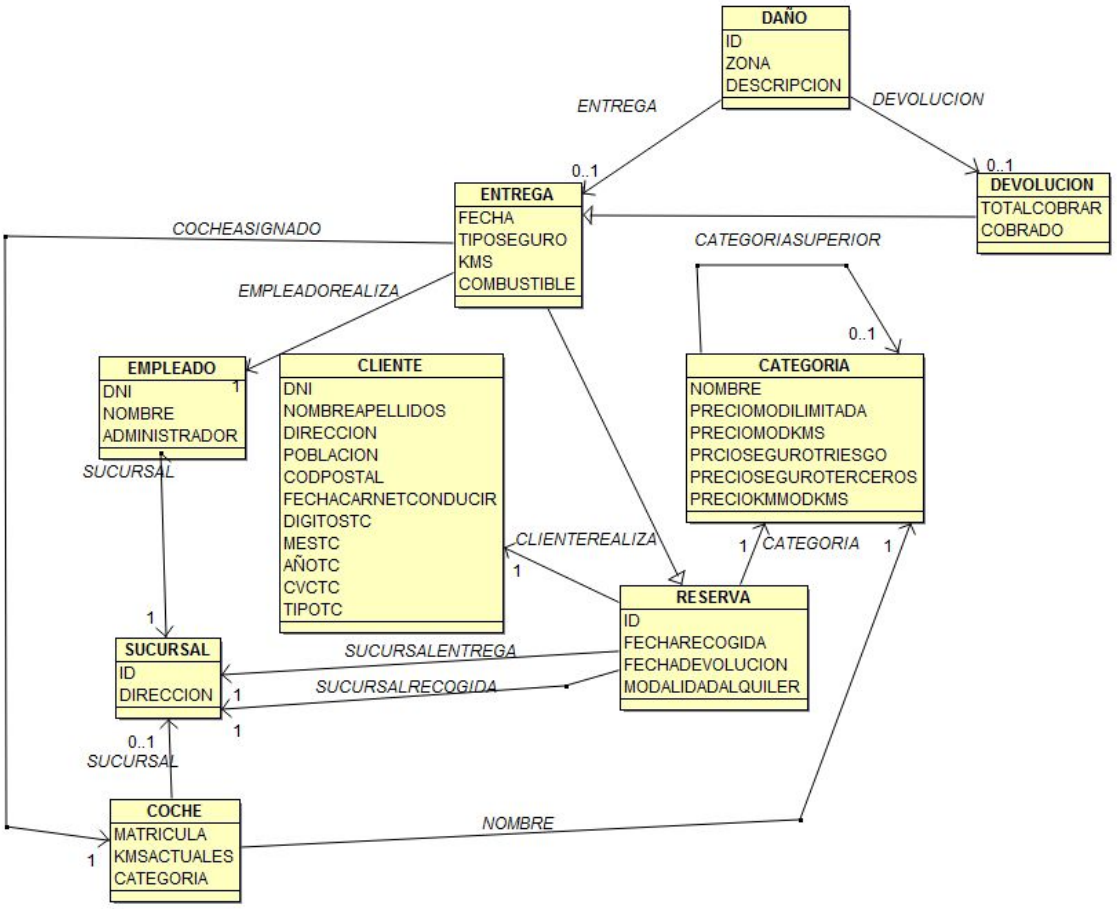
# Práctica 3 - Ingeniería Inversa

Víctor Iranzo Jiménez
Isidro Torregrosa Torralba
Adriano Vega Llobell
Clara Vidal Juan

# 1. Modelo de Datos




Para realizar el modelo de datos de la aplicación a partir del script 'alquilerdevehiculos.script' hemos procedido de la siguiente forma:

1.- Copiar el script a un editor de texto con resaltado de sintaxis SQL, quitar todas aquellas sentencias que no intervienen en la creación de tablas y formatear el script para que sea legible.
2.- Hacer en papel un diagrama de clases empleando la información proporcionada por las sentencias de creación de tablas. En esta parte hemos tenido en cuenta los siguientes puntos:

- Si una FK tiene el constraint NOT NULL la cardinalidad del elemento referenciado será 1, de lo contrario 0.
- Si una tabla tiene una FK a otra por el mismo atributo, y este atributo resulta tener el constraint PRIMARY KEY, entonces lo convertiremos en una especialización.

3.- Crear el diagrama de clases utilizando BoUML. Creemos necesario incluir la información referente al tipo y los constrains pues se trata de un modelo de datos, de modo que el tipo queda reflejado en el campo 'type', el constraint NOT NULL empleando una 'multiplicity' de 1

y finalmente el constraint PRIMARY KEY poniendo 'multiplicity' a 1 y marcando la casilla 'unique', pues las PK tienen estos dos constraints.

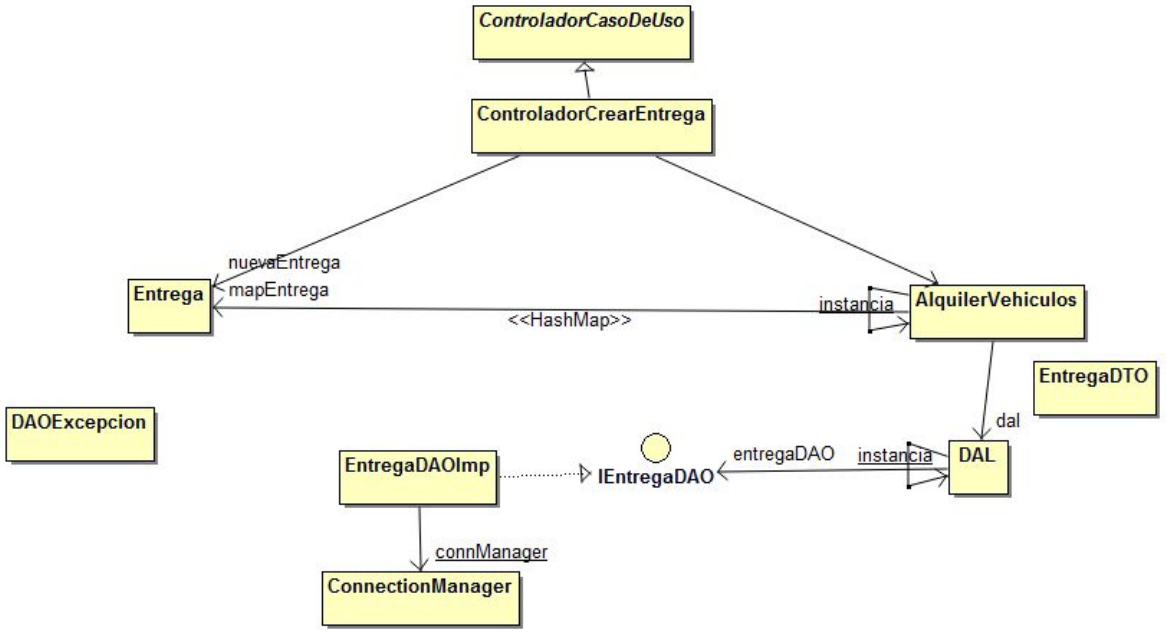
A continuación se describe lo que se ha hecho en cada tabla:

- Categoría tiene una FK a sí misma para indicar la categoría superior, el campo de la FK puede ser nulo por lo tanto la relación es de 0..1.
- Empleado tiene una FK NOT NULL a Sucursal, esto se traslada en una relación hacia sucursal desde empleado con cardinalidad 1.
- Reserva tiene FK NOT NULL a las tablas Cliente, Categoría y Sucursal, en el caso de sucursal tiene 2, sucursalDevolución y sucursalRecogida, para todas ellas hacemos relaciones como en la tabla anterior.
- En coche tenemos una relación 0..1 con sucursal, pues el campo que indica la FK puede ser nulo y luego tenemos un pequeño dilema pues tiene dos campos Nombre y Categoría, Nombre tiene una FK a categoría estableciendo una relación con cardinalidad 1, pero categoria es un atributo propio de coche, lo cual no tiene mucho sentido, con lo que recomendaríamos hacer un análisis más exhaustivo para determinar si la FK debería ser de categoría a categoría y en ese caso qué hacer con el atributo nombre. Por el momento lo hemos incluido en el diagrama sin alterar la información ofrecida por las scripts.
- Entrega resulta ser una subclase de reserva, pues su PK (id) hace referencia a la PK de Reserva (id), en cuanto a sus relaciones, ha de tener un empleado (empleadorealiza) y un coche (cocheasignado).
- Devolución tiene una FK a entrega por las PK de ambas tablas, con lo que se trata de otra herencia, en el caso anterior no compartíamos ningún atributo pero ahora fecha, kms y combustible aparecen en ambas tablas con lo que sólo es necesario indicarlos en la primera(entrega), de la misma forma la relación empleadorealiza no necesita ser indicada de nuevo.
- La última tabla creada ("Da\u00f1o") tiene dos FK nulificables a entrega y devolución, creando dos relaciones 0..1. Parecería una asociación pero como devolución es una especialización de entrega no es el caso.
- Por último, hemos de ver qué significado tienen tanto el campo "a\u00f1oTC" de la tabla clientes como el nombre de la tabla "Da\u00f1o", a simple vista parece que son strings codificadas en algún tipo de encode, con lo que empleando un conversor rápidamente encontramos su significado, añoTC y Daño, respectivamente, el conversor empleado es https://r12a.github.io/apps/conversion/

# 2. Regeneración de la información de diseño

## **Caso de Uso 1**: Entregar Vehículo Reservado

### Diagrama de Clases




Para realizar este diagrama hemos empezado viendo el código de la clase ControladorCrearEntrega, pues después del ControladorPrincipal y el ControladorCasoDeUso es el que lógicamente se encargará de crear las entregas una vez introducidos los datos necesarios para proceder a la misma. El ControladorPrincipal no lo hemos incluido en el diagrama puesto que sólo sirve para conducirnos a la ventana desde la que se inicia el caso de uso.

Cuando todo va bien vemos que el controlador anterior crea una nueva instancia de la clase Entrega y a continuación obtiene la instancia de AlquilerDeVehículos y llama al método crearEntrega de la misma.

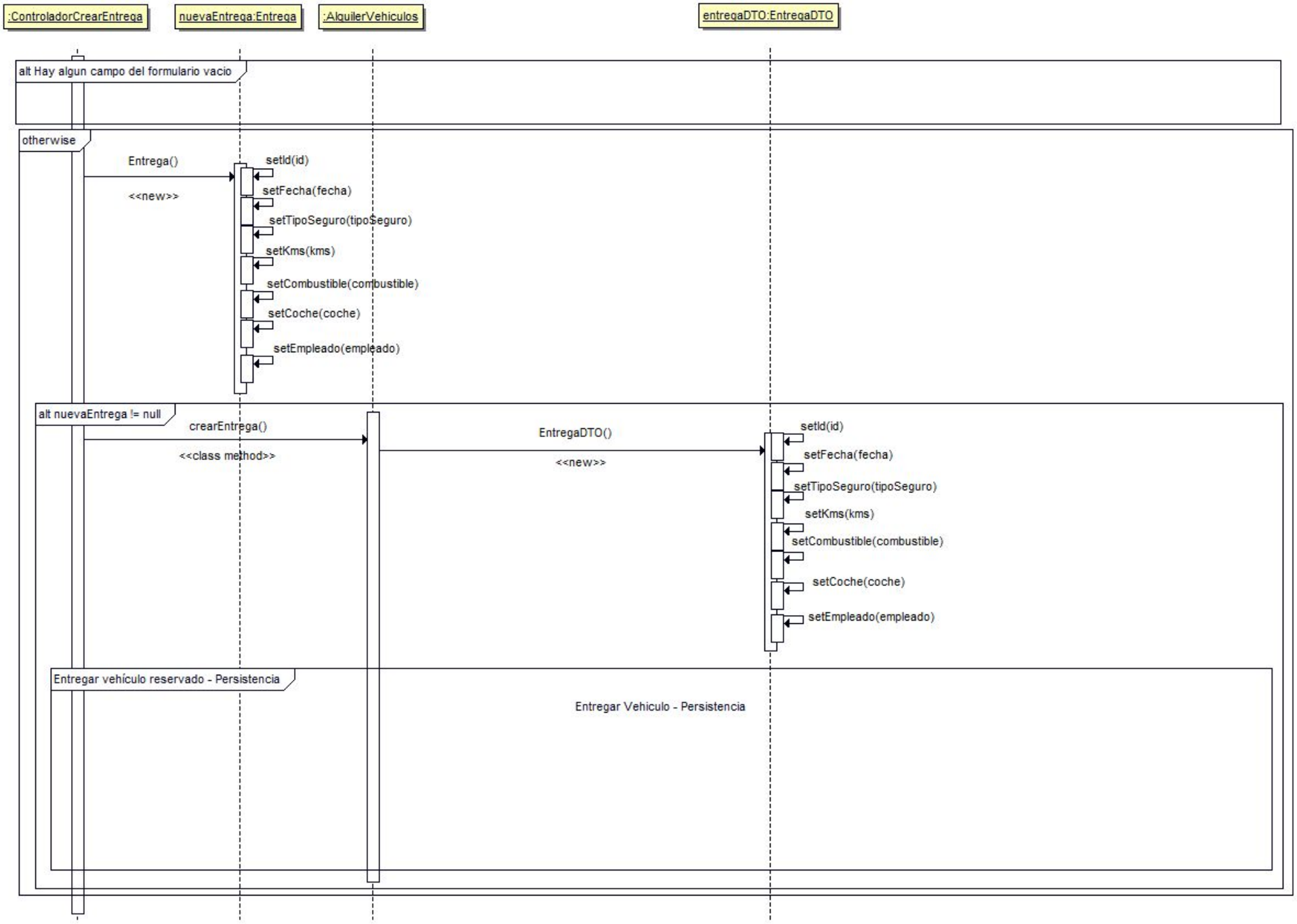
El método crear entrega hace uso de EntregaDTO para crear una nueva instancia para a continuación pasársela al DAL el cual creará una entrega en EntregaDAO.

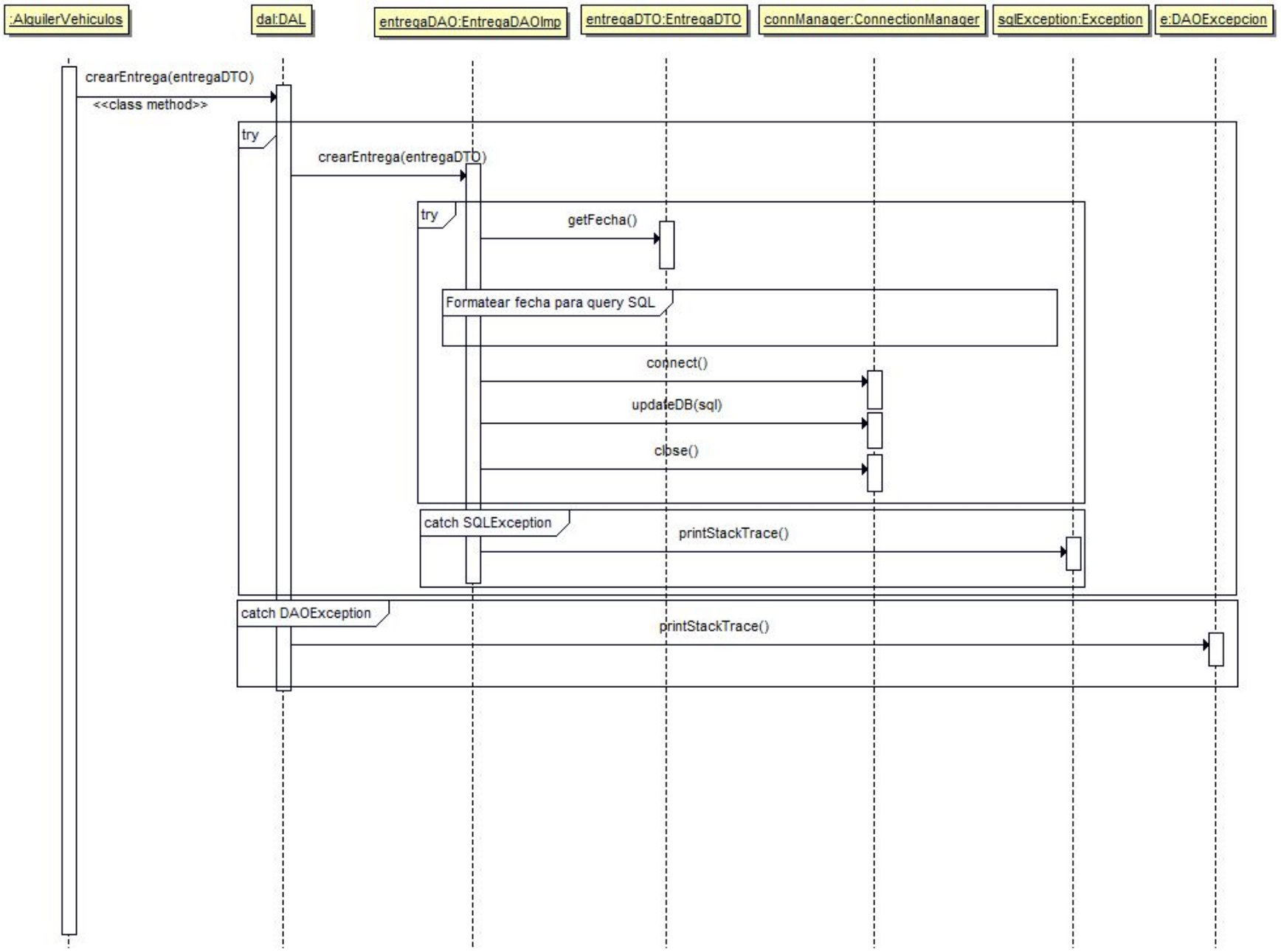
La implementación de EntregaDAO finalmente hace uso del ConnectionManager para insertar la nueva entrega en base de datos.

# Diagrama de Secuencia

Para poder realizar la entrega de un vehículo realizado, una vez el empleado ha seleccionado un vehículo e introducido los datos de la reserva, ocurre:

1. Se comprueban que todos los campos del formulario tengan un valor introducido.
2. En caso de que ningún campo esté vacío, se obtienen los valores y se procede a la creación de la entrega, llamando a su constructor.
    a. Dentro del constructor de Entrega, se hace uso de los métodos set para asignar cada atributo.
3. A continuación, se comprueba si se ha creado la entrega correctamente (no está con valor nulo). En caso afirmativo, comienza el proceso de persistir el valor.
4. Para persistir la entrega, se realiza una llamada a crearEntrega() de Alquiler de Vehículos, el *Business Controller* de la aplicación.
5. Esta llamada desencadena la creación de un *Data Transfer Object* (EntregaDTO)
    a. El constructor de EntregaDTO también hace uso de sus métodos set para la asignación de sus atributos.
6. Una vez construido el DTO, se le pasa como argumento a crearEntrega() de la clase DAL (*Data Access Layer*).
7. Dentro de este método, encontramos que intenta (try) pasarle el entregaDTO a la clase EntregaDAO mediante crearEntrega()
    a. En caso de error, se captura la excepción DAOException y se imprime la traza de error a consola.
8. En el método del DAO (Data Access Object), se obtiene la fecha del DTO y se formatea para realizar la consulta SQL.
9. A continuación, se intenta hacer tres llamadas diferentes al ConnectionManager, para ejecutar la consulta SQL:
    a. Open() para abrir la conexión con la base de datos
    b. updateDB() para ejecutar la consulta.
    c. Close() para cerrar la conexión
        i. En caso de que fallen cualquiera de estas llamadas, se captura una SQLException y se imprime su traza de error.


:ControladorCrearEntrega
nuevaEntrega:Entrega
:AlquilerVehiculos
entregaDTO:EntregaDTO
alt Hay algun campo del formulario vacio
otherwise
Entrega()
<<new>>
setId(id)
setFecha(fecha)
setTipoSeguro(tipoSeguro)
setKms(kms)
setCombustible(combustible)
setCoche(coche)
setEmpleado(empleado)
alt nuevaEntrega != null
crearEntrega()
<<class method>>
EntregaDTO()
<<new>>
setId(id)
setFecha(fecha)
setTipoSeguro(tipoSeguro)
setKms(kms)
setCombustible(combustible)
setCoche(coche)
setEmpleado(empleado)
Entregar vehículo reservado - Persistencia
Entregar Vehiculo - Persistencia

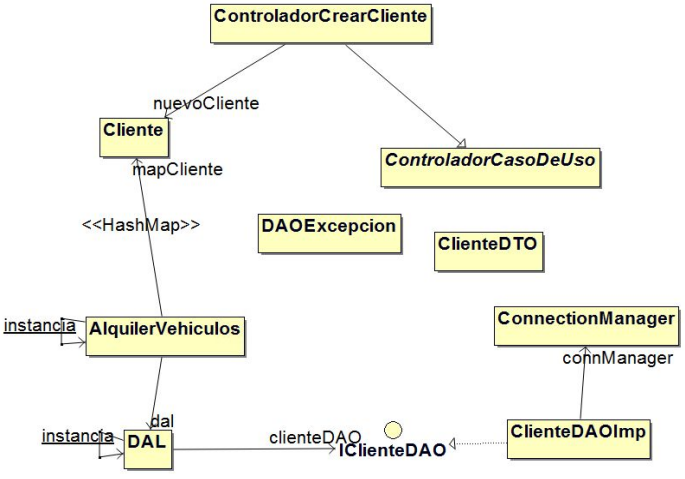

:AlquilerVehiculos
dal:DAL
entregaDAO:EntregaDAOImp
entregaDTO:EntregaDTO
connManager:ConnectionManager
sqlException:Exception
e:DAOExcepcion
crearEntrega(entregaDTO)
<<class method>>
try
crearEntrega(entregaDTO)
try
getFecha()
Formatear fecha para query SQL
connect()
updateDB(sql)
close()
catch SQLException
printStackTrace()
catch DAOException
printStackTrace()

# **Caso de Uso 2**: Crear Cliente

## Diagrama de Clases

En la creación de un cliente participan las siguientes clases:

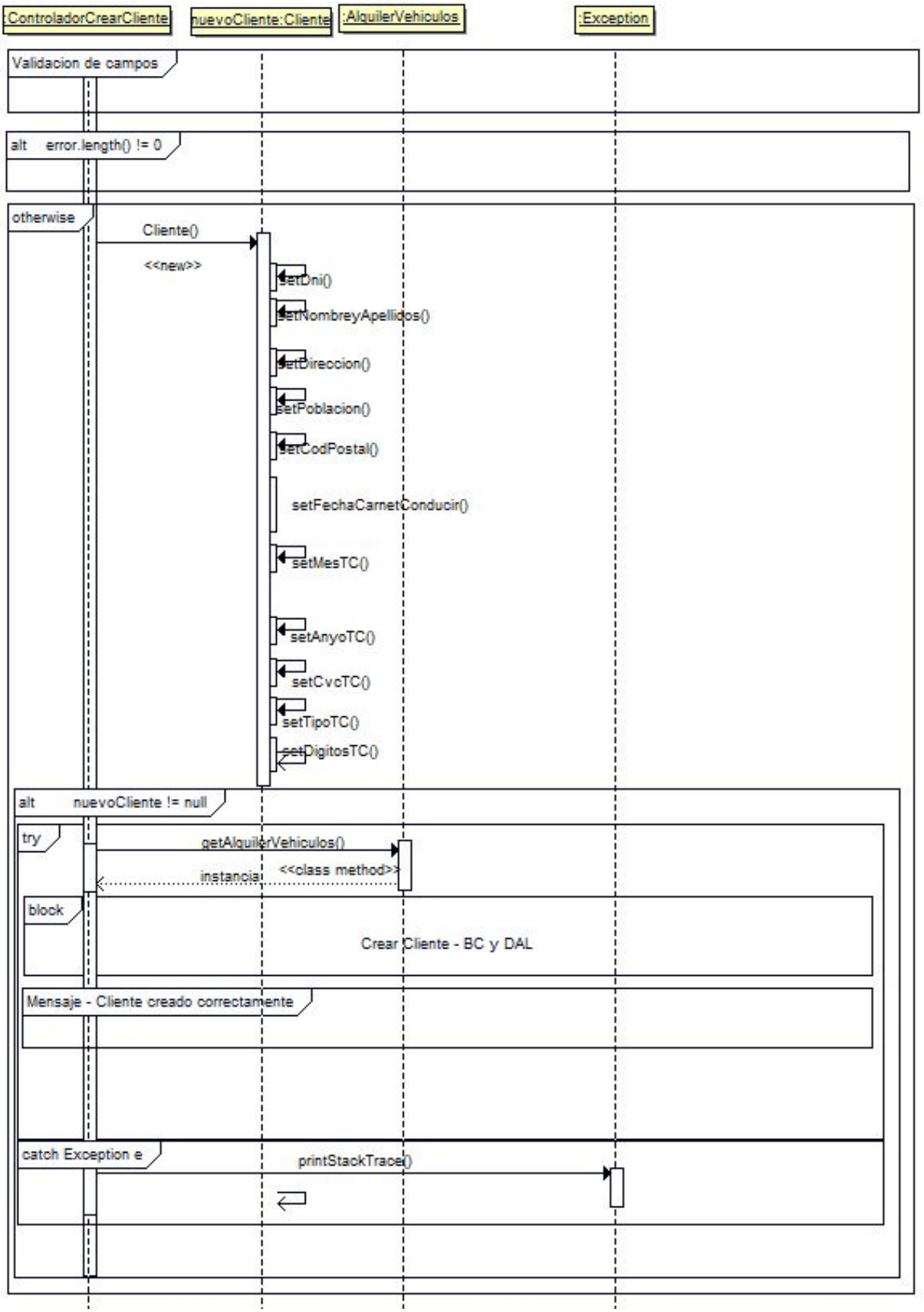
1. ControladorCasoDeUso: Controlador encargado de comunicarse con el con el controlador principal y alternar las distintas pantallas de la aplicación según corresponda.
2. ControladorCrearCliente: Clase de la capa de presentación encargada de capturar las interacciones del usuario en la ventana Crear Cliente, y de empezar el proceso de crear un nuevo cliente cuando éste pulse el botón aceptar.
3. Cliente: Clase de la capa de lógica encargada de crear un cliente a partir de los datos introducidos en su constructor.
4. AlquilerVehículos: Business controller de la aplicación. Contiene un HashMap de los modelos así como una referencia a la capa de persistencia (Dal).
5. ClienteDTO: Clase encargada de gestionar aquellos datos de la clase de lógica que nos interesa que persistan. En este caso, el cliente
6. DAL: Proporciona acceso simplificado a los datos almacenados en la capa de persistencia.
7. IClienteDAO: Interfaz que define los métodos de utilizados por el dal para consultar e insertar en la base de datos.
8. clienteDAOImp: Implementación de los métodos de la interfaz IClienteDAO, encargada de formular las consultas SQL necesarias para comunicarse con la base de datos a partir
9. ConnectionManager: Clase encargada de establecer la conexión con la base de datos y atacarla.
10. DAOException: Clase que extiende de Exception y se encarga de manejar las excepciones lanzadas en el DAO.



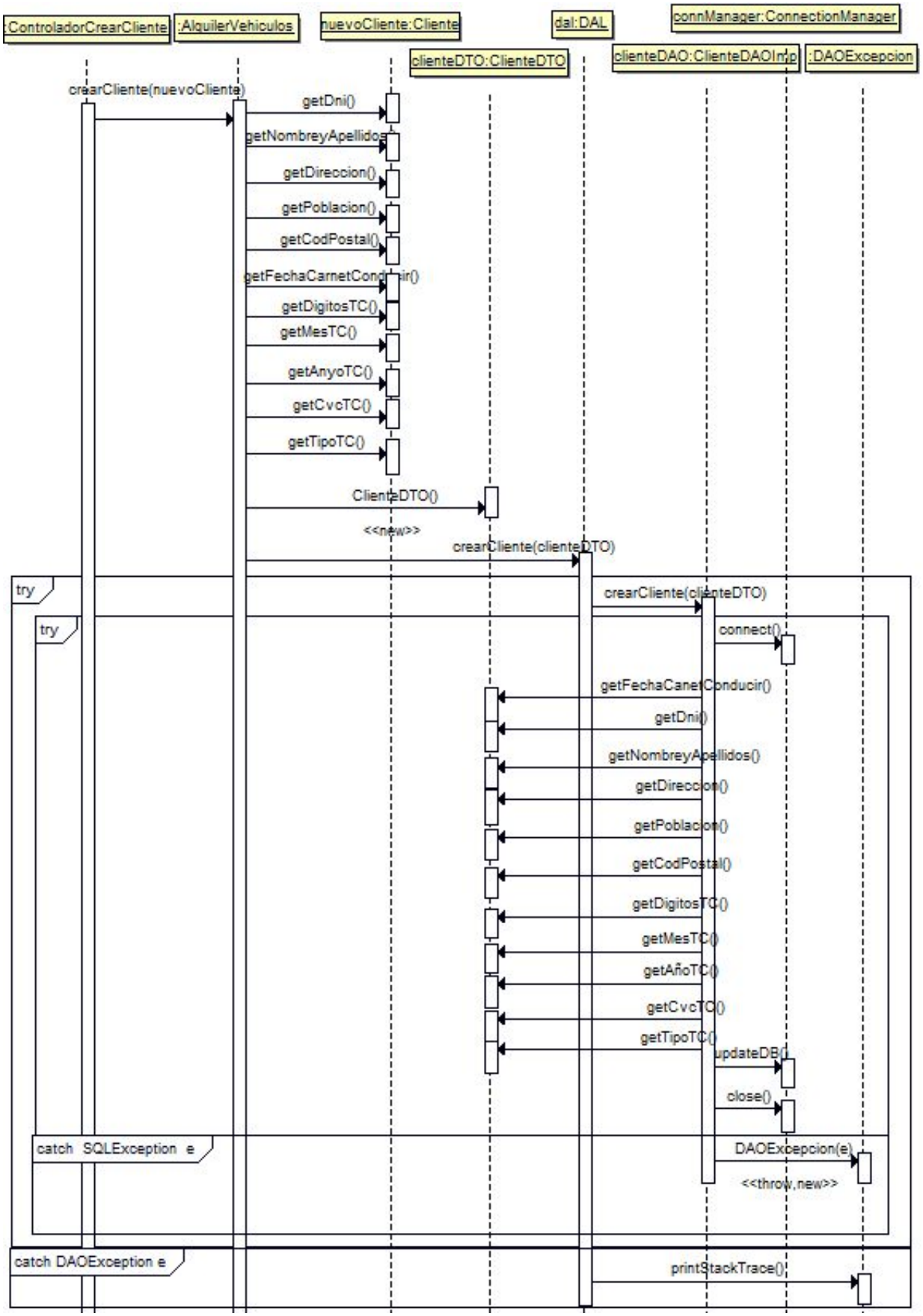
# Diagrama de secuencia

Para crear a un nuevo cliente se sigue la siguiente secuencia de instrucciones:

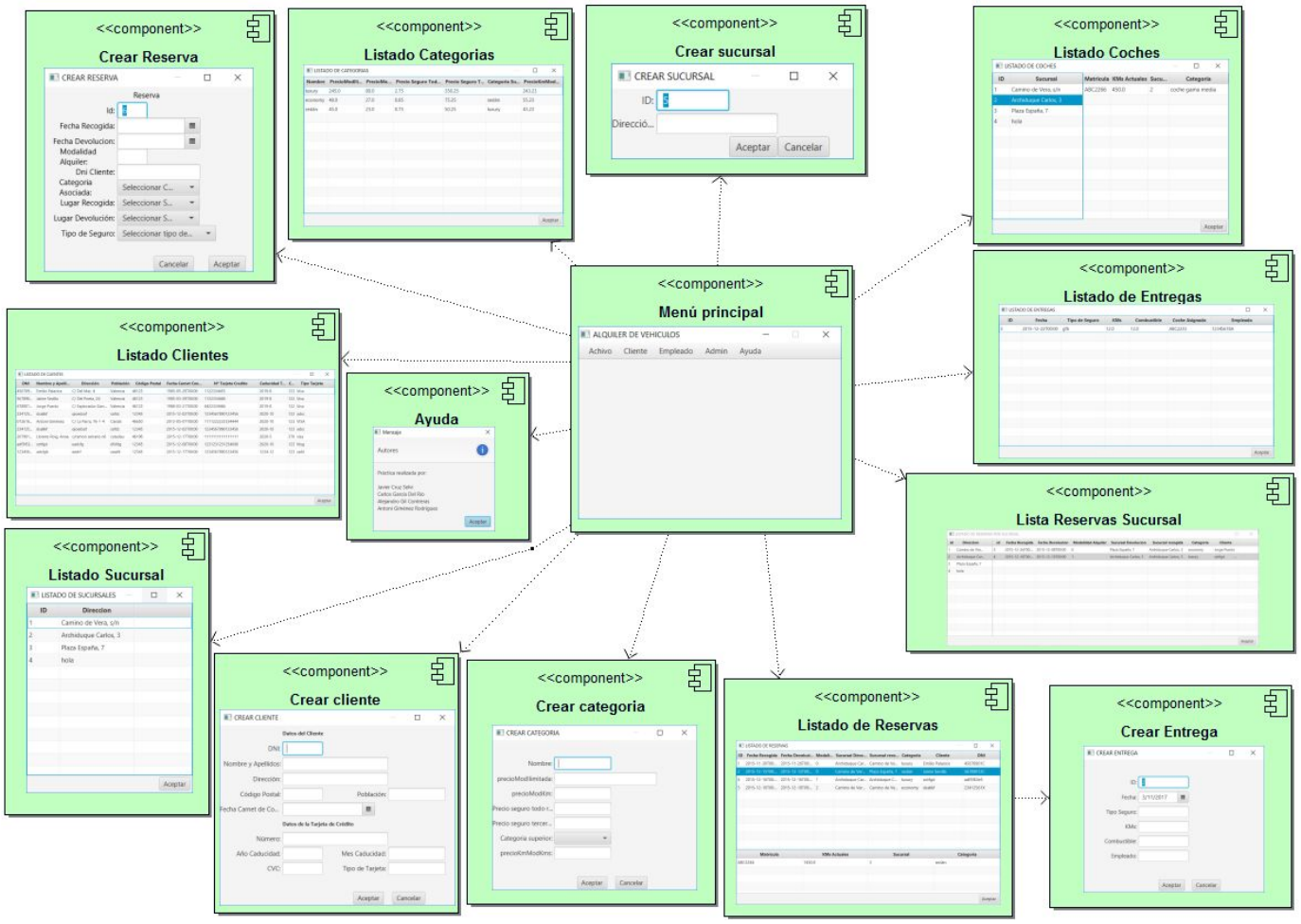
1. Desde el ControladorCrearCliente se comprueba que ninguno de los campos a introducir por el cliente esté vacío.
2. En caso de que ningún campo esté vacío se obtienen sus valores y se pasa a la creación de un cliente, llamando al constructor de Cliente y pasándole como argumento los valores obtenidos.
    a. El constructor hace uso de los métodos set para inicializar los atributos del Cliente
3. En el ControladorCrear cliente se comprueba que este nuevo objeto cliente se haya creado correctamente, es decir, que sea distinto de null. En ese caso comienza el proceso de persistir el valor:
4. Desde ControladorCliente se obtiene una instancia de AlquilerVehículos, nuestro Business Controller, y se ejecuta su método CrearCliente(), pasándole como argumento el cliente previamente creado.
5. Se obtiene el valor de los atributos a través de los getters de cliente y se pasan como argumento del nuevo ClienteDTO que se crea con los datos que queremos persistir.
6. El nuevo objeto ClienteDTO se pasa como argumento al método crearCliente() del dal.
7. El dal intenta pasarle el ClienteDTO a ClienteDAO, mediante crearCliente()
    a. En caso de error, se captura la excepción DAOException y se imprime la traza de error a consola
8. En ClienteDAOImp se intenta con un try establecer la conexión con el ConnectionManager, obtener el valor de los atributos de CienteDTO usando los getters y pasárselo en forma de consulta en el método upadate(). Seguidamente se cierra la conexión (close())
    a. En caso de que fallen cualquiera de estas llamadas, se captura una SQLException y lanza una DAOException.


:ControladorCrearCliente
nuevoCliente:Cliente
:AlquilerVehiculos
:Exception
Validacion de campos
alt error.length() != 0
otherwise
Cliente()
<<new>>
setDni()
setNombreyApellidos()
setDireccion()
setPoblacion()
setCodPostal()
setFechaCarnetConducir()
setMesTC()
setAnyoTC()
setCvcTC()
setTipoTC()
setDigitosTC()
alt nuevoCliente != null
try
getAlquilerVehiculos()
<<class method>>
instancia
block
Crear Cliente - BC y DAL
Mensaje - Cliente creado correctamente
catch Exception e
printStackTrace()

**Bloque: Crear Cliente - BC y DAL**



# 3. Mapa de la IGU




La navegación en la aplicación es como sigue: cuando ésta se inicia, comenzamos en la ventana del Menú principal. Desde aquí podemos navegar al resto de ventanas a través de las opciones del menú.

**Archivo**

- ➔ Salir: cerramos la aplicación.

**Cliente**

- ➔ Crear Reserva: navegamos a la ventana Crear Reserva.
- ➔ Crear Cliente: navegamos a la ventana Crear Cliente.

**Empleado**

- ➔ Crear Cliente: navegamos a la ventana Crear Cliente.
- ➔ Crear Sucursal: vamos a la ventana Crear Sucursal.
- ➔ Crear Categoria: vamos a la ventana Crear Categoria.
- ➔ Listar Entregas: vamos a la ventana Listado de Entregas.

- ➔ Entregar coche reservado: vamos a la ventana Listado de Reservas. Al seleccionar una sucursal con coches reservados y uno de los coches que se listen en la tabla de abajo, navegaremos al formulario Crear Entrega, donde haremos ésta efectiva.
- ➔ Listar Reservas Sucursal: nos lleva a la ventana Lista Reservas Sucursal. Al seleccionar una de las sucursales en la tabla de la izquierda, se nos actualizará la tabla de la derecha con las reservas de ésta.
- ➔ Listar Coches: navegamos a la ventana Listado Coches. Al seleccionar una de las sucursales en la tabla de la izquierda, se nos actualizará la tabla de la derecha con los coches asociados a ésta.
- ➔ Listar Categorias: vamos a la vista Listado Categorías.
- ➔ Listar Clientes: navegamos al componente Listado Clientes.

**Admin**

- ➔ Listar sucursales: navegamos a la ventana Listado Sucursal.

**Ayuda**

- ➔ Acerca de: vamos a la ventana Ayuda, un mensaje con información de los autores de la Aplicación.