Documento de Diseño

Descomposición requisitos funcionales

EMPLEADO:

RF1. Actualizar el estado de los carros (decir que un carro necesita mantenimiento)

1. Pide la palca del vehículo que necesita mantenimiento
2. Pide las fechas en las que el vehículo va a estar en mantenimiento
3. Se busca el carro con esa placa dentro del sistema
   getIdVehiculo() [Vehículo]
4. Reportar que durante las fechas dadas, el carro no estará disponible por mantenimiento
   mantenimiento() [Vehículo]

RF2. Registrar conductores nuevos para la reserva

1. Pedir información conductor
2. Crear instancia de licencia para añadirla al conductor
   Licencia(numeroLicencia, PaisLicencia, VencimientoLicencia) [Licencia]
3. Crear instancia de conductor
   Conductor(conductorName, telefono, correo, licencia) [Conductor]
4. Agregar nuevo conductor al sistema
   addConductor(nuevoConductor) [sistemaAlquiler]

RF3. Generar reservas para un cliente en especifico

1. Cliente RF1.

RF4. Entregar carro a cliente

1. Pedir el nombre del cliente que tiene una reserva a su nombre
2. Buscar reserva a nombre del cliente
   getCliente() [Reserva]
3. Buscar vehículo vinculado a la reserva
   getIdVehiculo() [Reserva]
4. Registrar conductores adicionales si no están registrados
   RF2.
5. Buscar conductor registrado en el sistema usando su correo para añadirlo a la reserva
   getCorreo() [Conductor]

6. Agregar conductores adicionales a la reserva

   agregarConductor(conductor) [Reserva]

7. Buscar precio total de la reserva

   getPrecioBase() [Reserva]

8. Añadir precio extra por los conductores al precio total, valor que depende de la categoría del vehículo que se rentó

   getCategoria() [Reserva]

   getValorConductorAdicional() [Categoría]

   getPrecioConductores(precioBase, valorConductorAdicional) [Reserva]

9. Descontar precio abonado por el cliente, del precio total para saber cuanto debe pagar el cliente

10. Cambiar ubicación del carro a “Alquilado” ya que no se encuentra en ninguna sede

    setUbicacion() [Vehículo]

RF5. Recibir carro de cliente

1. Pedir la placa del vehículo que se va a retornar
2. Busca el vehículo en el inventario usando su id

   getIdVehiculo() [Vehículo]

3. Buscar la reserva que tiene ese vehículo vinculado

   getIdCarro() [Reserva]

4. Busca el cliente que reservó ese vehículo y que lo está retornando

   getCliente() [Reserva]

5. Se registra que la ubicación del carro es la sede en la que se entregó el carro

   setUbicación(nuevaUbicacion) [Vehículo]

6. A la sede sede en la que se encuentra el carro, se le añade un carro a su inventario

   agregarCarro(carro) [Sede]

7. Se elimina la reserva del sistema ya que esta reserva ya se terminó y no está vigente

   eliminarReserva(reserva) [sistemaAlquiler]

8. Se registra que el vehículo va a estar en limpieza, por ende no va a estar disponible por 2 días

   limpieza() [Vehiculo]

ADMINISTRADOR LOCAL:

RF1. Gestionar los empleados de la sede (agregar nuevos empleados)

1. Pedir los datos de login del nuevo empleado
2. Se crea una nueva instancia de Empleado

   Empleado(sucursal, empleadoUsername, empleadoPassword)
3. Agregar el nuevo empleado a la sede

   agregarEmpleado(empleado) [Sede]
4. Agregar el nuevo empleado a sistema

   addEmpleado(empleado) [sistemaAlquiler]

RF2. Modificar información de la sede

RF2.1 Modificar dirección de la sede

1. Se pide la nueva dirección de la sede
2. Se cambia la dirección por la nueva

   setUbicacion(direccion) [Sede]

RF2.2 Modificar inicio o fin horario de atención

1. Se pide la nueva hora
2. Se cambia la hora por la nueva hora

   setInicioHorario(hora) [Sede]

   setFinHorario(hora) [Sede]

ADMINISTRADOR GENERAL:

RF1. Registro de compra de un vehículo (agregar nuevo vehículo al inventario)

1. Se piden los datos del nuevo carro
2. Se busca la sede en la que estará el carro dentro del sistema para añadirlo a la instancia de Vehículo que se creará

   getNombre() [Sede]
3. Crear la instancia del carro

   Vehiculo(vehiculoID, marca, modelo, categoria, color,transmision, capacidad, ubicacion)
4. Agregar ese nuevo vehículo al sistema

   addVehiculo(NuevoCarro)

RF2. Dar de baja un vehículo (eliminar vehículo del inventario)

1. Se pide la placa del vehículo que se desea eliminar
2. Se busca el carro con esa placa dentro del inventario

   getIdVehiculo() [Vehículo]

3. Se elimina el vehículo del inventario general
   removeVehiculo() [sistemaAlquiler]
4. se busca la sede en la que está ubicado en vehículo
   getUbicacion() [Vehículo]
5. se elimina el vehículo del inventario de esa sede
   eliminarVehiculo() [Sede]

RF3. Configurar seguros (modificar precio de los seguros)

1. Se pide el nombre del seguro que se va a modificar
2. Se pide el nuevo precio del seguro
3. Se busca el seguro dentro del sistema usando su nombre
   getNombre() [Seguro]
4. Cambiar el precio por día por el nuevo
   setPrecioPorDia() [Seguro]

RF4. Acceder al sistema

RF4.1 Consultar información de un vehículo

1. Se pide la placa del vehículo que se desea consultar
2. Se busca el carro con esa placa dentro del inventario
   getIdVehiculo() [Vehículo]
3. Se consulta la agenda de ese vehículo
   getAgendaVehiculo() [Vehículo]
4. Se muestran los rangos de fechas de cada indisponibilidad del vehículo
   getFechaInicial() [AgendaCarro]
   getFechaFinal() [AgendaCarro]
5. Se muestra el nombre de la sede en la que se encuentra ese vehículo
   getUbicacion() [Vehículo]

RF4.2 Consultar carros en el inventario

1. Se recorre el inventario general de carros
2. Se saca la información más importante de cada carro
   getVehiculoId() [Vehículo]
   getmarca() [Vehículo]
   getmodelo() [Vehículo]

3. Se muestra al usuario esa información

CLIENTE:

RF1. Reservar vehículo

1. Se pide la información de la reserva
2. Se buscan los carros del inventario de la sede en la que se va a recoger el vehículo, que estén disponibles en las fechas y la categoría que el cliente solicita

   getVehiculos() [Sede]

   getCategoria() [Vehículo]

   ValidaDisponibilidad(fechaInicio, fechaFinal) [vehículo]
3. Se escoge un carro al azar de todos los carros disponibles encontrados
4. Calcular los días en los que el carro estará en renta

   cantidadDiasRenta(fechaInicio, fechaFinal) [sistemaAlquiler]
5. Se determina la tarifa dependiendo de la categoría del vehículo seleccionado y de la fecha

   determinarTarifa(fechaInicio, fechaFinal) [sistemaAlquiler]
6. Se crea una instancia de una reserva

   Reserva(categoriaInput, startDate, endDate, clienteNombre, placaCarro, NombreSedeRecoger, NombreSedeDevolver)
7. Se saca el valor adicional por entregar el carro en una sede diferente usando la categoría escogida

   getValorSedeDiferente() [Categoria]
8. Se saca el precio de la reserva (sin seguros ni conductores adicionales)

   getPrecio(diasRenta, tarifa, valorSedeDiferente) [Reserva]
9. Se añaden los seguros a la reserva

   getNombre() [Seguro]

   agregarSeguro(Seguro) [Reserva]
10. Sacar nuevo precio de la reserva

    getPrecioConSeguros(precio, días)
11. Agregar reserva al sistema

    addReserva(reserva) [sistemaAlquiler]
12. Sacar 30% del precio total y poner eso como el precio abonado por el cliente

    get30ptcPrecio(precioTotal) [Reserva]

    setPrecioAbonado(treintaPct) [Reserva]

RF2. Modificar reserva (datos de entrega)

RF2.1 Modificar fecha en la que se entrega el carro

1. Se pide la nueva fecha
2. Se busca en el inventario general, un carro que tenga la misma placa que el carro que se le asignó a la reserva

   getVehiculoId() [Reserva]

   getVehiculoId() [Vehículo]
3. Al carro encontrado se le cambia la fecha final de indisponibilidad en su agenda

   setFechaFinal(fecha) [AgendaCarro]
4. A la reserva se le cambia la fecha de retorno

   setFechaRetorno(fecha) [Reserva]
5. Se encuentra el nuevo precio total ya que este pudo haber cambiado y se cambia el precio de la reserva

   getPrecio() [Reserva]

   getPrecioConSeguros [Reserva]

   setPrecio(precio) [Reserva]

RF2.2 Modificar sede en la que se entrega el carro

1. Se pide la nueva sede
2. Se cambia la sede de retorno

   setIdSedeDevolver(NombreSede) [Retorno]
3. Se saca el nuevo precio de la reserva ya que este pudo haber cambiado y se cambia en la reserva

   getPrecio() [Reserva]

   getPrecioConSeguros() [Reserva]

   setPrecio(precio) [Reserva]

RF3. Registrarse en el sistema

1. Se piden los datos del cliente
2. Se hace una instancia de Licencia con los datos de la licencia dados

   Licencia(numeroLicencia, PaisLicencia, VencimientoLicencia) [Licencia]
3. Se hace una instancia de MedioPago con los datos dados
   MedioPago(numeroTarjeta, tipoTarjeta, VencimientoTarjeta) [MedioPago]
4. Se hace la instancia de Cliente

   Cliente(telefono, clienteName, correo, licencia, medioPago, login, contrasena) [Cliente]

5. Se hace la instancia de Conductor

   Conductor(clienteName, telefono, correo, licencia) [Conductor]

6. Se agrega el nuevo cliente y conductor creado al sistema

   addCliente(cliente) [sistemaAlquiler]

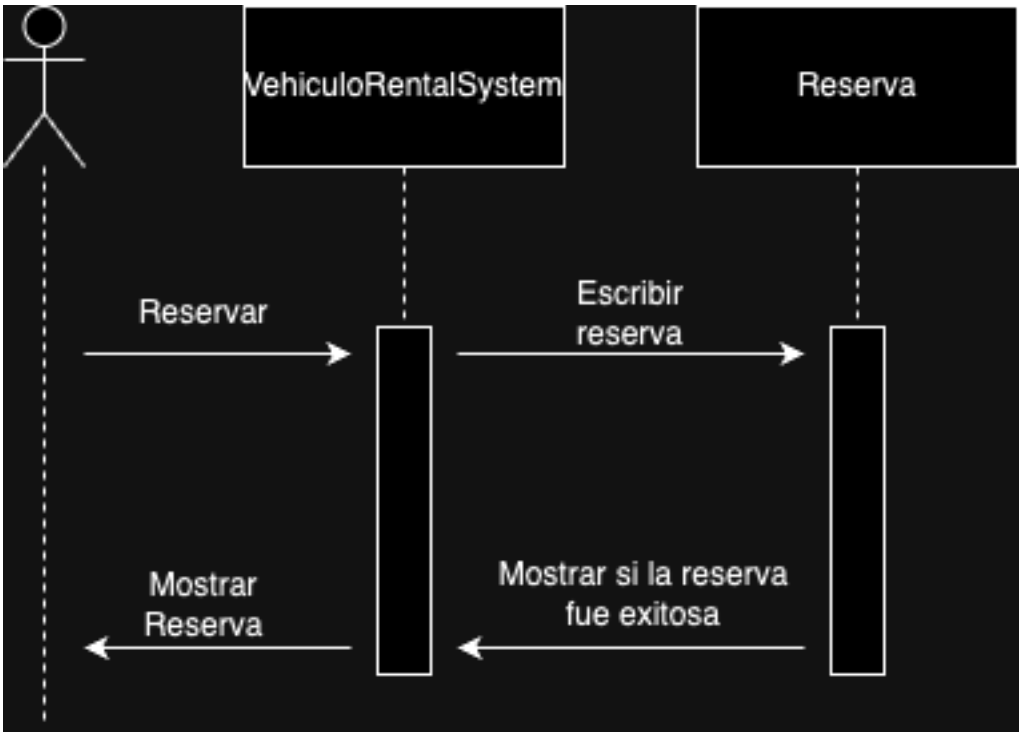
   addConductor(conductor) [sistemaAlquiler]

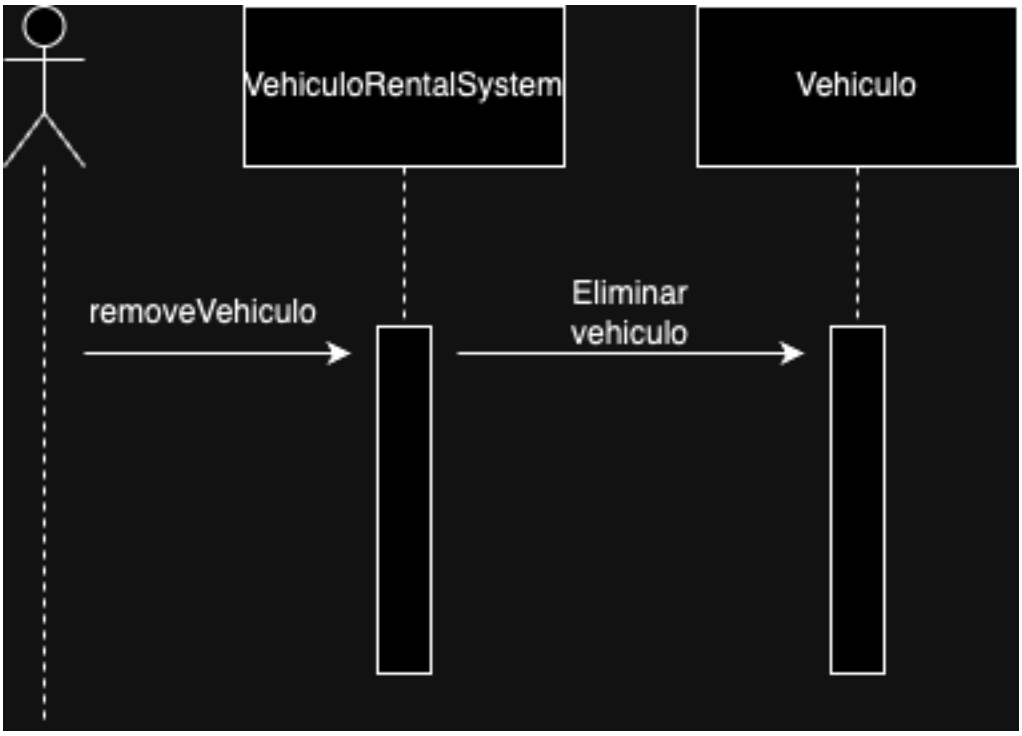
Responsabilidades

A continuación se encuentra una tabla la cual contiene las responsabilidades y cual es el componente el cual debe asumir cada una.

#	Responsabilidad	Componente
1	Actualizar el estado de los carros	Vehículo
2	Consultar información de un vehículo	
3	Registro de compra de un vehículo	VehiculoRentalSystem
4	Entregar carro a cliente	
5	Recibir carro de cliente	
6	Dar de baja un vehículo	
7	Consultar carros en el inventario	Sede
8	Gestionar los empleados de la sede	
9	Modificar información de la sede	
10	Configurar seguros	Seguro
11	Generar reservas para un cliente en especifico	Reserva
12	Registrar conductores nuevos para la reserva	
13	Reservar vehículo	
14	Modificar reserva	

Diagramas de Secuencia


VehiculoRentalSystem
Reserva
Reservar
Escribir
reserva
Mostrar
Reserva
Mostrar si la reserva
fue exitosa


VehiculoRentalSystem
Vehiculo
removeVehiculo
Eliminar
vehiculo

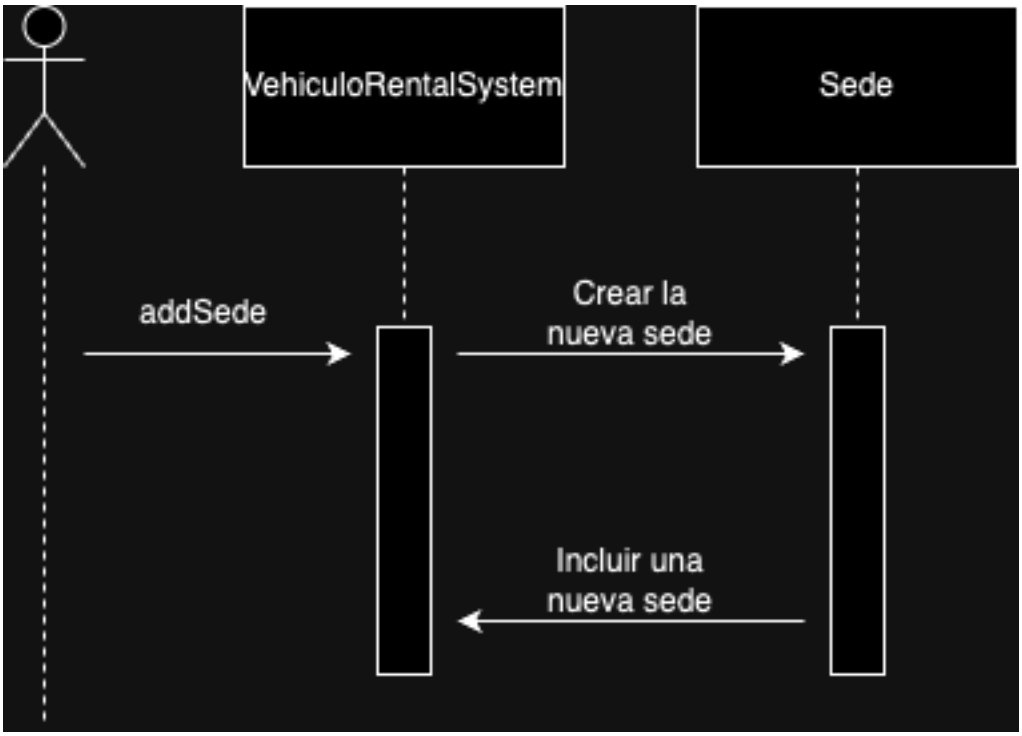

VehiculoRentalSystem
Sede
addSede
Crear la
nueva sede
Incluir una
nueva sede

Diagrama de Clases