

Diseño

Nicolás Pérez Ramos cod 202221292

José Miguel Alvear cod 202010602

Martín Calderón cod 202122043

1. Contexto del problema

Introducción

La solución consiste en ofrecer un sistema capaz de manejar el inventario y procesos de reserva y alquiler de vehículos para una empresa con varias sucursales. Dicha empresa tendrá un administrador principal que manejará el inventario de la empresa y que dará acceso al sistema a los administradores locales de cada sucursal, quienes a su vez son responsables de manejar el acceso de los empleados de atención y técnicos de cada sede y conocer la información adicional de estos. Además, el sistema permitirá el acceso de clientes para solicitudes de reserva a partir de una categoría de vehículos, manejará la información de estos y operará de forma tal que el sistema sea persistente a partir de archivos de extensión txt.

Indagación en el mundo del problema

Con el objetivo de estudiar el manejo de reservas en aplicaciones del mundo real y los detalles que estos incluyen, se decidió evaluar dos sistemas de renta de vehículos: Localiza y Alamo. Los aspectos principales hallados en esta indagación se presentan a continuación.

A partir del sistema de renta de vehículos Localiza, se identificaron los siguientes aspectos que puede tener la solución:

- Categorías: Todo vehículo debe pertenecer a una categoría, entre las que se identificaron:
 - para carros de tipo sedan: sedan económico, intermedio y premium
 - para SUVs: suv económico, intermedio y premium
 - para pick ups: pick up económico, intermedio y premium
- El sistema informa cual es la tarifa diaria para el tipo de vehículo.
- Una vez realizada la reserva, el sistema indica al usuario el código de reserva
- Si el cliente quiere cancelar o modificar la reserva, lo puede hacer usando el código de reserva y el número de identificación.
- La modificación de reserva permite upgrades, cambiar fecha de entrega y adicionar seguros.

Con respecto a los seguros, encontramos utilidad en utilizar los seguros que Alamo ofrece para sus vehículos y el porcentaje de la tarifa diaria que cada seguro agrega al pago del alquiler, de los que se destacan los seguros de:

- 1. protección total que cuesta aproximadamente el 100% de la tarifa diaria,
- 2. protección de neumáticos y parabrisas que cuesta aproximadamente el 11% de la tarifa diaria,

- 3.asistencia 24 horas en carretera con un costo aproximado del 20% de la tarifa diaria,
- 4. protección contra robos o daños con un costo aproximado del 35% de la tarifa diaria y
- 5. protección contra pagos deducibles que cuesta aproximadamente el 40% de la tarifa diaria.

Justificación de las propuestas del diseño de la solución

Por un lado, se identificaron características con las que debe contar nuestro diseño a pesar de no estar especificadas dentro de la descripción del problema. Entre estas se distinguió la necesidad de permitir cobros por motivos adicionales en la devolución del vehículo, por lo que, nuestra solución propone diferenciar 3 tipos de daños y una duración estimada de mantenimiento para estos: para daños leves se estimará una duración de mantenimiento de 5 días; para daños moderados, de 10 días y para daños graves, de 30 días.

Además, con respecto a la creación de usuarios, nuestro diseño requiere que los nuevos usuarios sean mayores de edad, tengan una licencia activa e ingresen un método de pago vigente, a pesar de que no sea utilizado durante dicho registro, dado que esto facilita que en el momento de reserva no se deba pedir constantemente que el usuario registre un método de pago.

Por otro lado, el enunciado no especifica al detalle el funcionamiento de la creación y modificación de reservas y alquileres, por lo que se esquematizaron los procesos de alquiler y reserva con las siguientes características:

- Las reservas solo se podrán planificar para recoger y entregar el vehículo en los horarios designados de la sede particular en la que se recoge y entrega el vehículo.
- En el momento de la modificación de una reserva, el sistema quita dicha reserva de la lista de reservas activas que tiene el vehículo para volver a evaluar en el inventario que vehículo se puede asignar a la reserva modificada, dado que cualquier mínima modificación en la reserva condiciona su disponibilidad, y en caso de que esta no se cumpla restablece la reserva sin modificaciones e informa al usuario que en caso de que la reserva no sea de utilidad la elimine.
- En el caso de que un cliente se acerque directamente a una sede a rentar un vehículo, el empleado deberá primero crear una reserva, dado que este es el único mecanismo necesario para verificar la disponibilidad de los vehículos del inventario.
- Los detalles de fechas, sedes de entrega y devolución, entre otros, forman parte de la reserva, por lo que cualquier ajuste que se desee realizar en el momento de recoger el vehículo (formalización del alquiler) se tramitará como una modificación de la reserva.
- En el momento de la creación o formalización del alquiler se reevalúa la disponibilidad del vehículo, dado que es posible que de forma desafortunada el vehículo haya sufrido alguna avería o requiera un mantenimiento no prevenido. Cuando el vehículo previamente asignado deja de estar disponible se busca en el inventario otro vehículo y pueden ocurrir dos escenarios: se encuentra otro vehículo y tanto la reserva como el alquiler no sufren cambios de tarifa, o no se encuentra otro vehículo y se cancela la reserva y alquiler, por lo que se devuelve el pago del 30% al cliente.

- En cada instancia de pago se revisa que el método de pago no caduque antes de la fecha de devolución del vehículo, dado que esto permite al sistema tramitar los pagos sin modificar el método de pago del usuario.
- Con respecto al cobro adicional por la entrega del vehículo en otra sede, este pago solo se tramita en el momento de la entrega del vehículo, dado que el cliente puede haber manifestado en la reserva que lo entregará en una sede diferente, pero devolverlo en la misma sede que recogió el vehículo.
- Una demora en la entrega del vehículo representará un pago adicional y una entrega temprana permitirá el reembolso de la diferencia que no debe pagar el usuario por entregarlo antes.
- Tras la devolución del vehículo, el sistema planificará hasta 3 posibles eventos en el siguiente orden: un evento de 24 horas para trasladar al vehículo si fue devuelto en una sede diferente a su sede original, un evento de periodo variable si el vehículo entra en mantenimiento y un evento de 24 horas para limpiar el vehículo. Solo después de estos eventos el vehículo volverá a estar disponible.

Además, con respecto a la validación de los estados de una reserva o un alquiler, es importante mencionar que para la primera se obtendrá exclusivamente a partir de la disponibilidad del vehículo. Por lo que una reserva será: activa si forma parte de la lista de reservas activas de algún vehículo, cancelada si ningún vehículo la tiene dentro de su lista de reservas activas y completada si existe un alquiler con el mismo identificador que la reserva. En adición, en cuanto al estado del alquiler, dada la simplicidad del objeto en comparación con la reserva, tiene un atributo que indicará que el alquiler está activo si el alquiler se creó correctamente y no cuando el vehículo se haya devuelto al completar el alquiler.

Descripción del proceso de persistencia

Una parte fundamental de la solución es su capacidad de que la información sea persistente. Para esto, en la solución se prioriza la simplicidad de los archivos por encima de la cantidad de los mismos, dado que esta simplicidad facilita los procesos de revisión del funcionamiento correcto del sistema y la identificación de errores en la solución. En estos archivos toda fecha es un *int* con formato **aaaammdd** y de toda hora un *int* con formato **hhmm**, ya que este formato permite la comparación correcta entre fechas y horas sin usar librerías externas.

Los archivos txt separados por punto y coma diseñados fueron los siguientes:

- **info** contiene la información principal de la empresa:
 - nombre de la compañía
 - costos por conductor adicional y traslado entre sedes de un vehículo
 - periodos de temporada alta y temporada baja
- **categoría** en el que cada línea del archivo representa una categoría con el formato:
 - identificador de la categoría; nombre de la categoría; capacidad de número de pasajeros; porcentaje a pagar de la tarifa diaria en temporada alta; porcentaje a pagar de la tarifa diaria en temporada baja; monto a cobrar por daños leves en el vehículo; monto a cobrar por daños moderados; monto a cobrar por daños graves; tarifa diaria; identificador de categoría padre o superior (si no tiene es igual a 0).

- Ejemplo:
3;sedan_PREMIUM;5;1.3;0.7;175000;875000;8750000;350000;0
- **sedes** en el que cada línea del archivo representa una sede con el formato:
 - identificador de la sede; nombre de la sede; dirección de la sede;[hora de apertura entre semana, hora de cierre entre semana];[hora de apertura fin de semana , hora de cierre fin de semana]
 - Ejemplo:
 - 2;SedeSur;cra58 #2, Bogotá;[730,1830];[730,1530]
- **personal** en el que cada línea del archivo representa las credenciales del personal de la empresa, con el formato:
 - nombre usuario; contraseña usuario; identificador de sede a la que pertenece(el Administrador no tiene, por lo que es igual a 0); tipo de usuario
 - Ejemplo:
 - j.rodriguez;juan90;1;EmpleadoTecnico
- **seguros** en el que cada línea del archivo representa un seguro, con el formato:
 - identificador del seguro; porcentaje adicional a cobrar de la tarifa; descripción del seguro
 - Ejemplo:
 - 2;0.35;Protección contra robo y daños
- **licencias** en el que cada línea del archivo representa una licencia, con el formato:
 - número de licencia; fecha de expedición de licencia; fecha de vencimiento de licencia; país de expedición
 - Ejemplo:
 - 1034567891;20191222;20291222;ARGENTINA
- **clientes** en el que cada línea del archivo representa un cliente, con el formato:
 - nombre usuario; contraseña usuario; número de cédula; nombre del cliente; correo del cliente;fecha de nacimiento; país de nacionalidad; [número del método de pago, fecha de vencimiento del método de pago, marca del método de pago, nombre del titular del método de pago]; número de la licencia de conducción asociada.
 - Ejemplo:
 - l.perez;luis34;1034567890;Luis Pérez;luis.perez@email.com;3209876543;19851222;ARGENTINA;[99 99888877776666,20291222,MasterCard,Luis Pérez];1034567891
- **eventos** en el que cada línea del archivo representa un evento, con el formato:
 - identificador del evento; fecha de inicio del evento ; fecha final del evento; hora inicio del evento; hora final del evento; **descripción del evento**.
 - Las opciones de **descripción** están estandarizadas y son:
 - EnLimpieza, EnMantenimiento, EnTraslado.
 - Ejemplo:
 - 3;20231212;20231213;1200;1200;EnLimpieza
- **reservas** en el que cada línea del archivo representa una reserva, con el formato:
 - identificador de la reserva; fecha de inicio de reserva; fecha final de reserva; hora inicio de reserva; hora final de reserva; boolean que indica si la reserva se realizó en una sede; identificador de la sede de entrega del vehículo; identificador de la sede de devolución del vehículo; identificador de la

categoría de vehículo reservada; cédula del cliente que reservó; pago de la reserva (30%)

- Ejemplo:

2;20231111;20231112;1200;1300;false;1;3;9;1014903362;432000.0

- **vehiculos** en el que cada línea del archivo representa un vehículo, con el formato:
 - placa del vehículo; marca del vehículo; modelo del vehículo; color del vehículo; **tipo de transmisión del vehículo**; **estado del vehículo**; boolean que indica si el vehículo esta averiado o no; identificador de la categoría a la que pertenece el vehículo ; identificador de la sede en la que está el vehículo; identificador ; **lista de eventos**; **lista de reservas activas**; **lista de alquileres**
 - el tipo de transmisión del vehículo puede ser “manual” o “automatica”,
 - el estado del vehículo puede ser: “EnLimpieza”, “EnMantenimiento”, “EnTraslado”, “Disponible”,
 - la lista de eventos tiene exclusivamente identificadores de eventos agrupados dentro de corchetes cuadrados y separados por comas,
 - la lista de reservas activas tiene exclusivamente identificadores de reservas activas agrupados dentro de corchetes cuadrados y separados por comas y
 - la lista de alquileres tiene exclusivamente identificadores de alquileres agrupados dentro de corchetes cuadrados y separados por comas.
 - Ejemplo:
PKP464;ram;500;rojo;automatico;disponible;false;9;1;[2,3];[2];[3]

Es importante mencionar que los archivos se leerán (para la carga de datos) y en el orden presentado, dado que como se puede inferir a partir de las relaciones de ciertos objetos (como la relación de asignación reserva ←- vehículo), es necesario instanciar primero los objetos que forman parte de otros.

Resumen

Considerando la indagación del mundo del problema y el diseño de la solución, se implementarán los siguientes aspectos de dichos sistemas de renta de vehículos:

- Las categorías sedan, SUV, Pick-up a su vez están divididas en las categorías económico, intermedio y premium.
- El upgrade de un vehículo económico es un vehículo de sus mismas características, pero de tipo intermedio y el upgrade de un vehículo intermedio es uno de las mismas características de tipo premium. Después de la categoría premium no puede haber upgrade.
- El despliegue de la información de costo por día durante reserva.
- Permitir la modificación o cancelación de reserva a partir del código de reserva.
- Durante la modificación de reserva, permitir todo tipo de modificaciones.
- Permitir la asignación de conductores adicionales y seguros.
- Considerando el porcentaje de tarifa diaria de cada seguro, eximir el pago ciertas averías, teniendo:
 - absolución de pago por averías graves e inferiores en caso de haber el pagado seguro de protección total.

- absolución de pago por averías moderadas e inferiores en caso de haber pagado los seguros de protección contra robos o daños o protección contra pagos deducibles
- absolución de pago por averías leves en caso de haber pagado el seguro de protección de neumáticos y parabrisas

2. Iteración #1

2.1. Roles

A partir del Metamodelo definido en la etapa de Análisis, se identificaron los siguientes objetos con los siguientes Roles:

- Inventario: Inventario va a contener la información de los vehículos, sedes, las diferentes categorías y seguros. Considerando que se encargará de agrupar la información principal de la empresa, tendrá el estereotipo <Structurer>.
- Usuarios: Este objeto estará relacionado a los siguientes usuarios y sus características: administrador principal, administradores locales, personal de atención, personal técnico, cliente con sus datos personales(tarjeta, licencia). Teniendo en cuenta que este objeto principalmente mantiene y entrega información respecto a credenciales y datos personales, tendrá el estereotipo <Information holder>.
- Alquiler: Este último contendrá toda la información respecto a la reserva y posterior alquiler de cada vehículo, en la que se incluirá también la información de los conductores que el cliente desee agregar. Considerando que concentrará gran parte de la lógica del sistema, al ingresar al objeto Inventario para verificar la disponibilidad y características de cada vehículo para asignarlas a los clientes de la empresa, tendrá el estereotipo <Controller>.

2.2. Responsabilidades

Considerando los requerimientos funcionales descritos en el documento de análisis y los roles previamente descritos, se identifican la asignación de responsabilidades en la tabla 1.

Tabla 1. Asignación de responsabilidades en iteración #1

#	Responsabilidad	Componente
1	Crear categoría de vehículo	Inventario
2	Añadir vehículo al inventario	
3	Eliminar vehículo del inventario	
4	Actualizar sede de un vehículo	
5	Crear seguro	
6	Modificar precio seguro	
7	Modificar descripción del seguro	
8	Eliminar seguro	

9	Registrar Sede	
10	Modificar nombre de una Sede	
11	Modificar horario de atención de una Sede	
12	Modificar dirección de una Sede	
13	Obtener historial de un vehículo dado	
14	Registrar Administrador Local	Usuarios
15	Actualizar información de temporada baja o temporada alta	
16	Registrar empleado de atención	
17	Registrar empleado técnico	
18	Acceder a registro de los empleados de una sede	
19	Registrar alquiler	Alquiler
20	Modificar alquiler	
21	Actualizar estado del vehículo	Inventario
22	Registrar evento	
23	Registrar conductor adicional	Alquiler
24	Registrar cliente	Usuarios
25	Modificar datos del cliente	
26	Registrar reserva	Alquiler
27	Modificar reserva	
28	Cancelar reserva	

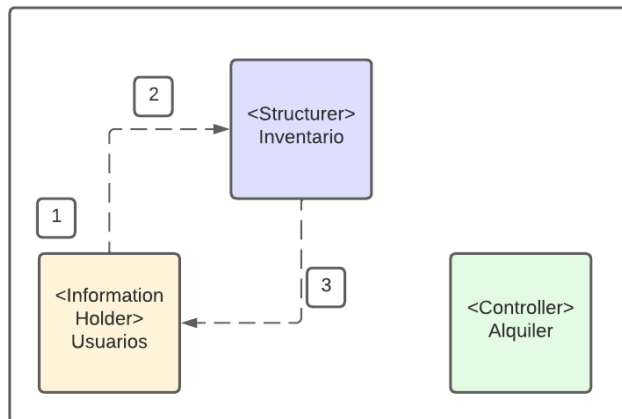
Respecto a esta asignación, es importante aclarar que a pesar de que todas estas acciones se realicen desde objetos pertenecientes al componente usuarios, las responsabilidades se asignan en función de las instancias de los componentes que se ven modificados o que son consultados. Resulta importante mencionar también que el registro de conductores adicionales solo lo podrá realizar el personal de atención en el momento de la reserva por lo que el cliente no podrá realizar dicha acción.

2.3. Colaboraciones

Las colaboraciones que requieren una intervención compleja entre componentes se presenta a continuación:

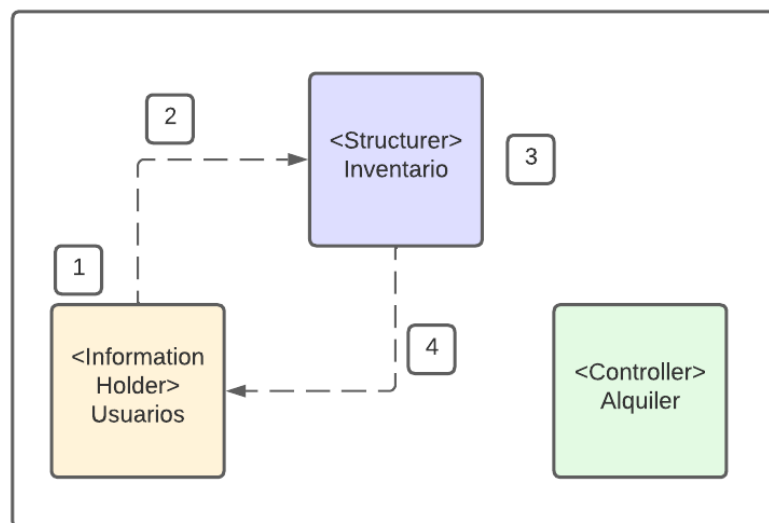
- para **Acceder a registro de los empleados de una sede** se requieren los siguientes pasos:
 1. Un usuario de perfil administrador local ingresa al sistema
 2. Se ingresa al componente inventario donde está la información de cada sede y de la sede correspondiente, se busca la lista de empleados
 3. El componente inventario retorna la lista de empleados

Figura 2. Colaboración para asignación #17



- para **Actualizar estado del vehículo** se requieren los siguientes pasos:
 1. Un usuario de perfil personal ingresa al sistema
 2. A partir del número de placa accede al vehículo
 3. Revisa el último evento para tomar las acciones necesarias:
 - a. Un vehículo puede estar en los estados: enMantenimiento, enTraslado, Disponible, enReserva, enAlquiler, enLimpieza
 - b. Todo vehículo que no tenga averías pasa a enLimpieza tras estar enAlquiler.
 - c. Todo vehículo que tenga averías pasa a enMantenimiento y después a enLimpieza. Se crea un evento que describa el mantenimiento.
 4. Se informa al personal que se actualizó el estado.

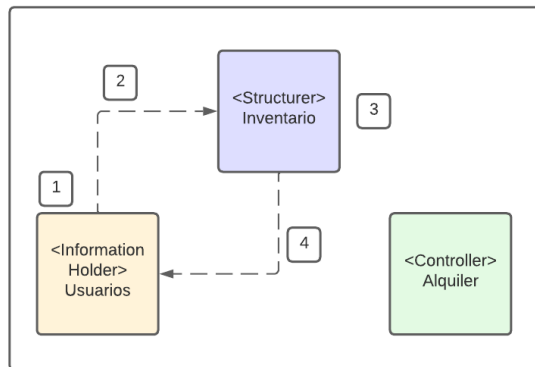
Figura 3. Colaboración para asignación #20



- para **Registrar reserva** se requieren los siguientes pasos:
 1. Un usuario de perfil cliente o personal ingresa al sistema
 2. El usuario ingresa la categoría que desea.
 3. Se busca en el inventario el primer vehículo disponible de la categoría deseada, de lo contrario se busca al primero de la categoría superior. Para saber que un vehículo está disponible se revisa:
 - Revisar que en la lista de reservasActivas las fechas deseadas para rentar no están ya reservadas para el vehículo

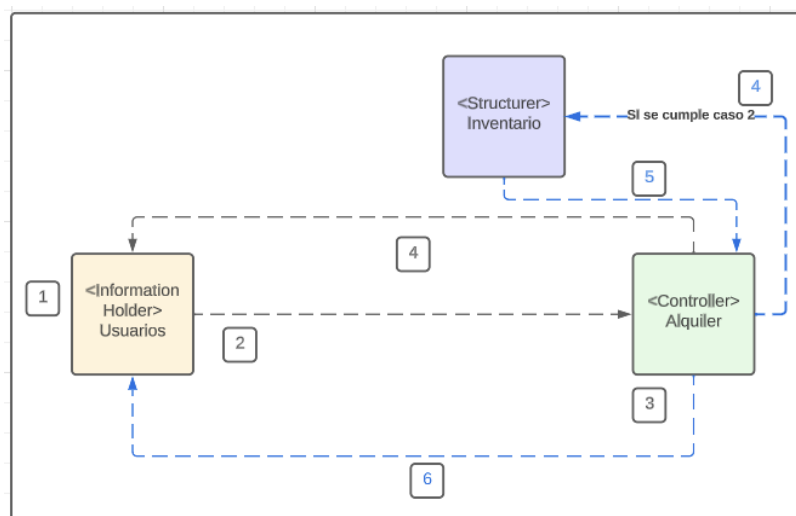
4. Se informa al usuario el costo por día y el saldo del 30% o en su defecto se informa que no hay vehículos disponibles.

Figura 4. Colaboración para asignación #25



- para **Modificar reserva** se requieren los siguientes pasos:
 1. Un usuario de perfil cliente o personal ingresa al sistema.
 2. Se busca la reserva a partir del número de reserva.
 3. El usuario ingresa la nueva categoría que desea o las nuevas fechas de recogida y entrega.
 - En caso de querer modificar el rango de fechas:
 - i. Se mira en la lista de reservaActivas que el carro ya reservado esté disponible
 - ii. Si no estará disponible para las nuevas fechas, se busca el primer vehículo disponible de la categoría deseada, de lo contrario se busca al primero de la categoría superior a partir de la lista de reservasActivas del vehículo.
 - En caso de querer hacer un upgrade:
 - i. Se revisan para la categoría solicitada los vehículos disponibles en el rango de fechas a partir de la lista de reservaActivas del vehículo.
 4. Se informa al usuario la nueva tarifa diaria y el saldo de 30% o en su defecto se permite al usuario cancelar la reserva.

Figura 5. Colaboración para asignación #27



3. Iteración #2

3.1. Roles

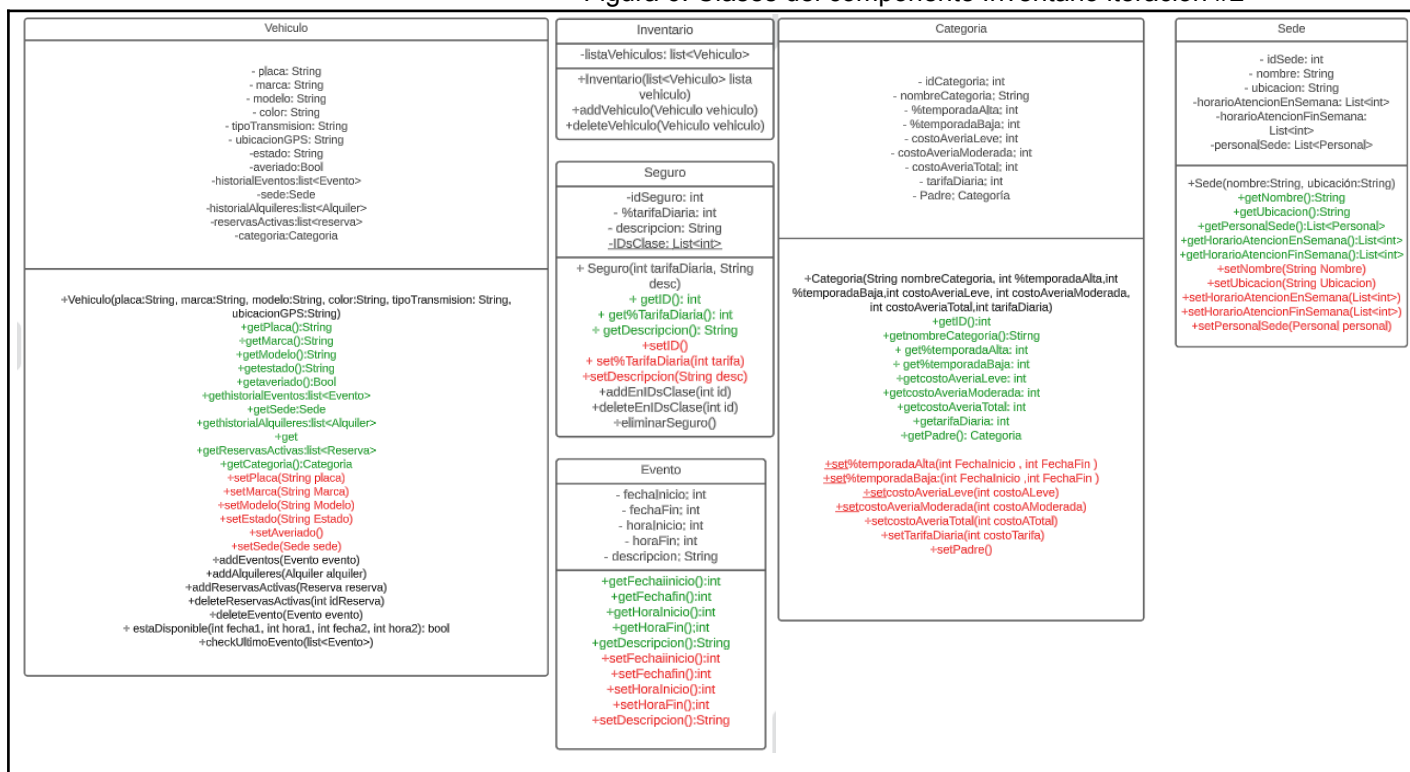
Considerando las clases definidas en el proceso de análisis, las reglas de dominio, las asignaciones, las colaboraciones y el contexto del problema se logró detallar los componentes de la solución de la siguiente manera:

Componente Inventario

-Inventario: Para este componente se diseñaron las clases: Vehículo, Inventario, Seguro, Sede, Categoría, Evento

- Vehículo contiene casi todos los atributos respecto a la información del vehículo como información de general, historial de alquileres y eventos así como el estado actual del vehículo. Esta clase cuenta con su constructor, setters(en verde) y getters(en rojo) además de métodos para:
 - añadir un nuevo evento al historial de eventos
 - añadir un nuevo alquiler al historial de alquileres
 - añadir una reserva a la lista de reservas activas
 - Eliminar una reserva de la lista de reservas activas
 - Eliminar un evento del historial de eventos
 - Calcular la disponibilidad del vehículo en un rango de fecha y hora
 - Revisar el último evento en la lista de eventos del vehículo
- Categoría contendrá la información relacionada con la categoría de los vehículos y todo lo que involucra esto según la categoría del vehículo como el nombre de la categoría, el porcentaje de sobre costo en temporada alta y de descuento en temporada baja, el costo de avería según la gravedad de esta (leve, moderada, total), la tarifa diaria a pagar de esta misma y la categoría padre en caso de que ella tenga. Esta clase cuenta con su constructor, setters(en verde) y getters(en rojo).
- Inventario contendrá un atributo que será una lista de vehículos que son todos los vehículos con los que cuenta la empresa, también contará con el método constructor y métodos para:
 - Añadir un nuevo vehículo a la lista de vehículos.
 - Eliminar un vehículo de la lista de vehículos.
- Seguro contará con toda la información sobre los seguros adicionales que ofrece la empresa y contará con un identificador para facilitar su búsqueda, con la tarifa diaria y con la descripción del seguro. Esta clase cuenta con su constructor, setters(en verde) y getters(en rojo) además de métodos para:
 - Eliminar un seguro
- Sede cuenta con la información de las distintas sedes que tendrá la empresa con un identificador de sede para facilitar su búsqueda, con el nombre y ubicación de la sede, una lista con el horario de atención entre semana y fin de semana y con una lista del personal que trabaja en la sede. Esta clase cuenta con su constructor, setters (en verde) y getters(en rojo)
- Evento contamos con la información del evento por el que acontece el vehículo como la fecha y hora de inicio, fecha y hora de fin y una descripción del evento. Esta clase cuenta con su constructor, setters (en verde) y getters(en rojo)

Figura 6. Clases del componente Inventario iteración #2



Componente Alquiler

- Alquiler: Para este componente se diseñaron las clases: Reserva, Alquiler, Conductor y pago excedente.
 - Reserva va a contener la mayor parte de los atributos que describen una reserva y su posterior alquiler, incluyendo fechas, sedes de entrega y categorías. Esta clase cuenta con su constructor, setters(en verde) y getters(en rojo) además de métodos para:
 - añadir y eliminar los IDs de reserva en una lista propia de la clase que contendrá todas las reservas activas.
 - calcular duración de la renta
 - estimar el pago del alquiler

- Alquiler contendrá la información adicional tras la confirmación de la reserva y el acercamiento del cliente a retirar el vehículo. Esta clase cuenta con su constructor, setters(en verde) y getters(en rojo) y métodos para:
 - añadir y eliminar los IDs de alquiler (iguales al id de reserva) en una lista propia de la clase que contendrá todas los alquileres activos.
 - crear el recibo que contenga el detalle de todos los pagos hechos al retornar vehículo
- El conductor tendrá la información de contacto de los conductores, incluyendo su licencia. Considerando que no será necesaria modificar la información de los conductores esta clase sólo tendrá un método que será el constructor.
- Pago excedente además de incluir el motivo de un pago adicional y la tarifa, contiene la información de las fechas de temporada alta y baja de la empresa y el costo adicional por conductor que serán obtenidos y seteados a partir de los métodos de la clase. Los métodos estáticos se deben a que el cambio de fechas o costo por conductor no dependerá de cada instancia de objeto sino de los valores que el administrador configure en el sistema.

Figura 7. Clases del componente Alquiler iteración #2



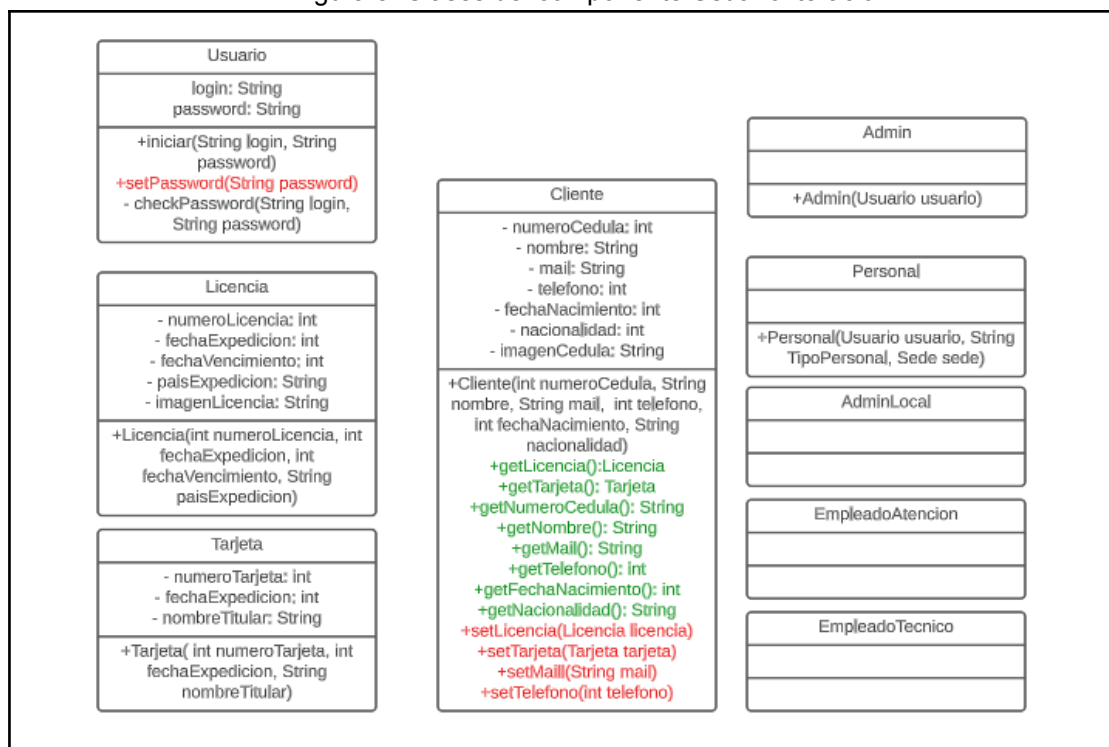
Componente Usuario

Para este componente se diseñaron las clases: Cliente, Licencia, Tarjeta, Admin, Personal, AdminLocal, EmpleadoAtencion, EmpleadoTecnico.

- Cliente va a contener toda la información personal de los clientes: datos del documento de identidad, datos de contacto e incluirá una imagen del documento de identidad. Contendrá métodos para:
 - registrarse como cliente en el sistema
 - modificar datos personales, de licencia o de tarjeta de crédito
 - realizar, modificar o cancelar una reserva

- Licencia va a contener toda la información de la licencia de conducción tanto de los clientes registrados como de los conductores adicionales que se registren al momento del alquiler e incluirá una imagen de la licencia.
- Tarjeta va a contener toda la información de la tarjeta de crédito con la que el cliente registrado vaya a realizar los pagos.
- Admin va a contener los métodos que realizará el Administrador general de la empresa, que incluyen:
 - registrar y eliminar administradores locales
 - añadir y eliminar vehículos del inventario
 - crear, configurar y eliminar seguros para clientes
 - registrar y modificar sedes
 - trasladar un vehículo entre sedes
 - obtener el historial de un vehículo
- Personal va a contener las subclases AdminLocal, EmpleadoAtencion, EmpleadoTecnico
- AdminLocal va a contener los métodos que realizará el administrador de una sede de la empresa, que incluirá registrar, modificar y observar la lista de empleados
- EmpleadoAtencion va a contener los métodos que realizarán los empleados que se encargan de la atención al cliente, que incluirá realizar la entrega y devolución de vehículos con todos los requerimientos adicionales de la función
- EmpleadoTecnico va a contener los métodos que realizarán los empleados encargados del mantenimiento de los vehículos, que incluirá actualizar el estado de un vehículo y funciones adicionales relacionadas

Figura 8. Clases del componente Usuario iteración #2



3.2 Responsabilidades

En esta segunda iteración se repartieron las responsabilidades para las diferentes clases.

Tabla 2. Asignación de responsabilidades en iteración #2

#	Responsabilidad	Clase
1	Crear categoría de vehículo	Categoría
2	Añadir vehículo al inventario	Inventario
3	Eliminar vehículo del inventario	Inventario
4	Actualizar sede de un vehículo	Vehículo
5	Crear seguro	Seguro
6	Modificar precio seguro	Seguro
7	Modificar descripción del seguro	Seguro
8	Eliminar seguro	Inventario
9	Registrar Sede	Sede
10	Modificar nombre de una Sede	Sede
11	Modificar horario de atención de una Sede	Sede
12	Modificar dirección de una Sede	Sede
13	Registrar Administrador Local	Usuarios
14	Actualizar información de temporada baja o temporada alta	Admin
15	Obtener historial de un vehículo	Admin
16	Registrar empleado de atención	AdminLocal
17	Registrar empleado técnico	AdminLocal
18	Acceder a registro de los empleados de una sede	AdminLocal
19	Registrar alquiler	Alquiler
20	Modificar alquiler	Cliente
21	Actualizar estado del vehículo	Vehículo
22	Registrar evento	Vehículo
23	Registrar conductor adicional y seguros	Alquiler
24	Registrar cliente	Usuarios
25	Modificar datos del cliente	Cliente
26	Registrar reserva	Reserva
27	Modificar reserva	Reserva
28	Cancelar reserva	Reserva

3.3.Colaboraciones

Con respecto a los componentes del sistema:

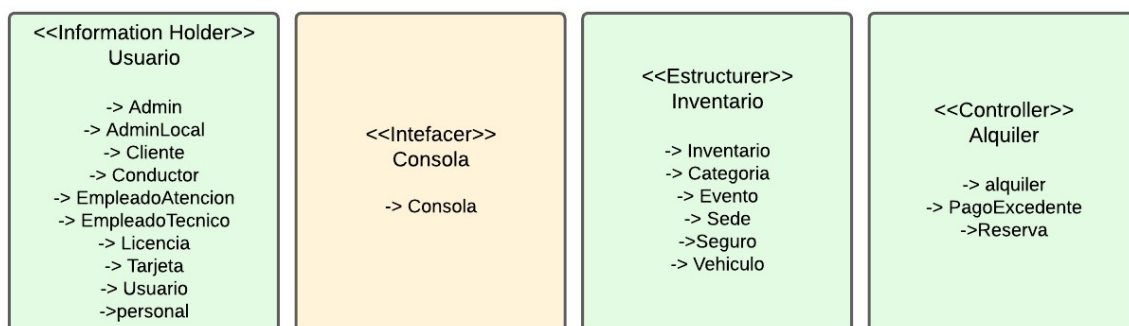
- **Alquiler** centrará la mayoría de la lógica, teniendo en cuenta que los métodos de la clase Alquiler y Reserva acceden al componente **Inventario** para poder tramitar solicitudes de rentas.
- **Usuarios** principalmente almacenará usuarios, más la lógica de las operaciones será sencilla en la medida que a cada tipo de usuario se le permitirá acceder a diferentes clases y usar los métodos propios de las clases.
- **Inventario** permitirá la persistencia del sistema a partir de las clases que mantendrán el historial de eventos o alquileres vinculados a los diferentes vehículos y como estos se relacionan en el tiempo con la clase alquiler.

4. Iteración #3

4.1. Roles

Considerando la implementación de una nueva clase Consola encargada de mostrar el menú principal (ver diagrama de clases), los roles del diseño de cada componente acompañados de las clases que forman parte de cada uno se presentan a continuación.

Figura 8. Roles iteración #3



4.2 Responsabilidades

Durante la implementación del diseño, se reconoció que el diseño no contaba con ciertos requerimientos funcionales, por lo que la redefinición de las responsabilidades se presenta a continuación.

Tabla 3. Asignación de responsabilidades en iteración #3

#	Responsabilidad	Componente	Clase
1	Monitorear un vehículo	inventario	Inventario
2	Crear categoría		Inventario
3	Añadir vehículo al inventario		Inventario
4	Eliminar vehículo del inventario		Inventario
5	Obtener historial de un vehículo		Vehículo
6	Trasladar vehículo		Vehículo

7	Crear un seguro		Inventario
8	Modificar un seguro		Inventario
9	Eliminar un seguro		Inventario
10	Registrar una sede		Inventario
11	Modificar una sede		Inventario
12	Registrar un admin local		Inventario
13	Actualizar información de un admin local		Inventario
14	Actualizar costo por conductor adicional		Inventario
15	Actualizar costo por traslado de un vehículo entre sedes tras alquiler		Inventario
16	Actualizar el periodo de temporada alta		Inventario
17	Actualizar el periodo de temporada baja		Inventario
18	Actualizar estado de un vehículo		Evento
19	Registrar evento para un vehículo		Evento
20	Registrar empleado en una sede	usuario	personal
21	Actualizar información de un empleado		personal
22	Obtener registro de empleados de una sede		personal
23	Crear un cliente		Cliente
24	Modificar datos de un cliente		Cliente
25	Registrar una reserva	Alquiler	Reserva
26	Modificar una reserva		Reserva
27	Cancelar una reserva		Reserva
28	Registrar un alquiler		Alquiler
29	Registrar conductores y seguros al alquiler		Alquiler
30	Completar un alquiler		Alquiler

4.3.Colaboraciones

Considerando que los diagramas de secuencia describirán los procesos de reserva y alquiler, se presentan las colaboraciones principales entre los requerimientos funcionales 1 y 24.

- para **acceder al sistema** se requieren los siguientes pasos:
 1. El usuario ingresa al sistema.
 2. Se verifica si el usuario es administrador, personal o cliente.
 3. Si el usuario pertenece a alguno de estos perfiles, ingresa al menú correspondiente, caso contrario se muestra un mensaje.
- para **monitorear un vehículo** se requieren los siguientes pasos:
 1. Un usuario de perfil administrador ingresa al sistema.
 2. El usuario ingresa la placa del vehículo.

3. Si se encuentra el vehículo, se busca en la lista de eventos si hay algún evento en la fecha actual.
 4. Si no tiene ningún evento en la fecha actual, se busca en la lista de reservas activas si está en reserva.
 5. Se presenta al usuario un resumen del estado actual del vehículo.
- para **obtener el historial de un vehículo** se requieren los siguientes pasos:
 1. Un usuario de perfil administrador ingresa al sistema.
 2. El usuario ingresa la placa del vehículo.
 3. Si se encuentra el vehículo, se solicita el log del mismo.
 - para **trasladar un vehículo** se requieren los siguientes pasos:
 1. Un usuario de perfil administrador ingresa al sistema.
 2. El usuario ingresa la placa del vehículo.
 3. Si se encuentra el vehículo, el usuario ingresa la sede a la que desee trasladar el vehículo.
 4. Si la sede ingresada es diferente de la sede actual del vehículo se intenta crear un evento de 24h desde la fecha actual para trasladar el vehículo.
 5. Si no se logra crear el evento, dado que se encuentra en alquiler, en limpieza o en mantenimiento, se notifica al administrador que lo intenté en otra ocasión. Caso contrario se crea el evento y se añade a la lista de eventos del inventario.
 - para **registrar un administrador local** se requieren los siguientes pasos:
 1. Un usuario de perfil administrador ingresa al sistema.
 2. El usuario ingresa un nuevo login para el nuevo administrador local.
 3. Se verifica que el nombre de ese login no exista.
 - ★ Si ya existe se solicita que reintente.
 4. El usuario ingresa una clave e intenta asignarle una sede.
 5. Si no logra asignar una sede, porque ya existe un admin local, informa al usuario. Caso contrario, lo asigna a la sede, lo añade a la lista de credenciales e informa que se completó el registro.
 - para **actualizar el estado de un vehículo o registrar un evento para un vehículo** se requieren los siguientes pasos:
 1. Un usuario de perfil personal técnico ingresa al sistema.
 2. El usuario ingresa la placa del vehículo.
 3. Si se encuentra el vehículo, se pregunta si el vehículo está en mantenimiento o en limpieza.
 4. Se registran las fechas y horas de inicio y final del evento.
 - Para actualizar el estado la fecha de inicio del evento debe ser la fecha actual.
 - Para registrar un evento, la fecha de inicio del evento debe ser superior a la fecha actual.
 5. Se crea el evento, este se asigna al vehículo.
 6. El evento se agrega a la lista de eventos del inventario.
 - para **registrar un empleado** se requieren los siguientes pasos:

1. Un usuario de perfil administrador local ingresa al sistema.
 2. El usuario ingresa un nuevo login para el nuevo empleado.
 3. Se verifica que el nombre de ese login no exista.
 - ★ Si ya existe se solicita que reintente.
 4. El usuario ingresa una clave y el tipo de usuario del nuevo empleado.
 5. Se agrega el empleado a la sede y a la lista de credenciales.
- para **actualizar la información de un empleado** se requieren los siguientes pasos:
1. Un usuario de perfil administrador local ingresa al sistema.
 2. El usuario ingresa el login de un empleado de su sede.
 3. Si lo encuentra, el usuario modifica los campos requeridos.
 - En caso de modificar la sede en la que trabaja el empleado, quita al empleado de la lista de empleados de la sede anterior y lo asigna a la lista de empleados de la nueva sede.
- para **registrar un cliente** se requieren los siguientes pasos:
1. El usuario ingresa su número de cédula.
 - ★ Si el cliente ya existe, le permite cambiar la cédula solicitando antes ingresar su fecha de nacimiento como método de verificación.
 2. El usuario ingresa su nombre, correo, teléfono y fecha de nacimiento
 - Si el usuario es menor de edad, se informa y se cancela el proceso de registro.
 3. El usuario ingresa un nombre de login
 - Si ya existe alguien con ese login, se solicita que lo intente nuevamente
 4. El usuario ingresa una contraseña y registra su licencia (se agrega la licencia a la lista de licencias en la clase usuario).
 - Si la licencia no es válida y no desea agregar otra licencia, se cancela el proceso de registro.
 5. El usuario registra su método de pago
 - Si el método de pago no es válido y no desea agregar otro método de pago, se cancela el proceso de registro
 6. El sistema agrega el nombre de login a la lista de logins, la cédula a la lista de cédulas y al cliente a la lista de clientes.
- para **actualizar un cliente** se requieren los siguientes pasos:
1. El usuario de perfil cliente ingresa al sistema.
 2. El cliente realiza las modificaciones correspondiente
 - Si ingresa nuevos datos de su licencia
 - a. Si la licencia es válida, se agrega la licencia a la lista de licencias en la clase usuario.
 - b. Si no es válida, no se actualiza la licencia.
 - Si ingresa un nuevo método de pago
 - A. Si el método es válido, actualiza el método de pago del cliente.
 - B. Si no es válido, no actualiza el método de pago.

4.4. Diagrama de Secuencia

El conjunto de diagramas de secuencia que representan las secuencias necesarias para cumplir cada una de las responsabilidades se encuentra en la carpeta de la entrega 2 bajo el nombre “diagramasecuencia.pdf”.

4.5. Diagrama de Clases

El diagrama de clases se encuentra en la carpeta de la entrega 2 bajo el nombre “diagramaclases.pdf”.

Es importante mencionar que en este diagrama se realizaron los siguientes cambios con respecto al metamodelo presentado en el documento de análisis:

- Cambiar la relación entre la clase Licencia y Conductor
- Modificar los tipos de datos de atributos que tendrán valores numéricos asignados
- Agregar una clase Consola que se encarga de la mayoría de las interacciones con el usuario.
- Agregar un atributo de estado a la clase Alquiler, en el que cada instancia será true una vez se crea el alquiler y se realiza el cobro del 70% + pagos por seguros o conductores adicionales y será false una vez se completa el alquiler y se realizan los cobros relacionados a averías, entrega con demora ó entrega en una sede diferente a la sede en la que se recogió el vehículo.

Se recomienda verificar la sección 1. Contexto de problema donde exponemos la indagación realizada en el mundo del problema y la justificación de las propuestas del diseño de la solución, que fueron implementadas en su totalidad.

5. Iteración #4

La creación de la interfaz gráfica de usuario presentó la oportunidad de conocer formas diversas de crear la lógica de dicha interfaz. Anteriormente utilizamos la clase “Consola” para justamente correr el programa basado en consola. Durante este proceso de aprendizaje tomamos la decisión de hacer uso de **JTabbedPane**, paneles que permiten una creación ágil de menús que incorpora el Controlador con la vista. Considerando lo anterior, decidimos crear la GUI en un solo paquete que involucra tanto a la Vista como al Controlador en el paquete *Vista*. En dicho paquete, fue necesaria el diseño y la implementación de las siguientes clases:

VentanaMain: Esta clase es la que se encarga de cargar la información de la aplicación, y crea una ventana para que el usuario inicie sesión, ingresando su nombre de usuario y contraseña, o para que se registre como cliente.

VentanaAdmin: Esta clase es la encargada de mostrar la ventana con la que interactúa el administrador principal de la empresa. Contiene los menús para que pueda administrar los vehículos, las categorías de vehículos, las sedes de la empresa, los seguros disponibles, el personal de las sedes, los periodos de temporada alta/baja, y la visualización de alto nivel para una sede.

VentanaAdminLocal: Esta clase es la encargada de mostrar la ventana con la que interactúan los administradores locales de cada sede. Contiene el menú para poder registrar nuevo personal o actualizar el personal existente, y acceder a la lista del personal.

VentanaAtencion: Esta clase es la encargada de crear la ventana con la que interactúa el personal de atención al cliente. El empleado de atención puede buscar un cliente, crear un alquiler para un cliente en la sede, agregar seguros al alquiler y completarlo al recibir el vehículo del cliente.

VentanaEmpleadoTecnico: Esta clase es la encargada de crear la ventana con la que interactúa el personal de mantenimiento de los vehículos. El menú le permite actualizar el estado de un vehículo estando en mantenimiento o lavado.

VentanaCliente: Esta clase es la encargada de crear la ventana con la que interactúan los clientes. En ella un cliente puede crear, modificar y cancelar reservas; cambiar su contraseña; y cambiar sus datos de contacto, licencia y pago.

VentanaRegistro: Esta clase crea una nueva ventana que se abre al apretar el botón de registrarse. En ella un cliente en varios pasos va ingresando toda la información necesaria para poder crear su usuario en la aplicación y poder hacer uso de los servicios.

CardsPanels: Esta clase es una clase que crea paneles utilizando “CardLayout” que permite crear o editar objetos siguiendo varios pasos, dependiendo de la información ingresada por el usuario.

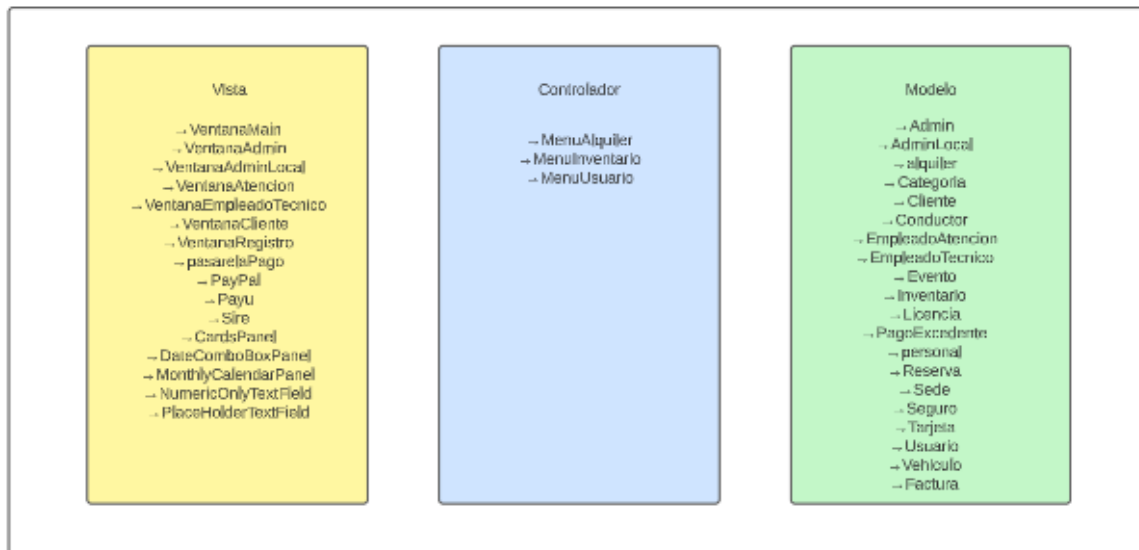
DateComboBox: Esta clase crea un JComboBox especializado para elegir las fechas necesarias en diferentes requerimientos.

MonthlyCalendarPanel: Esta clase es utilizada para facilitar la visualización de alto nivel de una sede, mostrando el calendario de cómo se va a rellenar.

NumericOnlyTextField: Esta clase se utiliza para crear un campo de texto especializado donde solo se permite al usuario ingresar dígitos numéricos.

PlaceholderTextField: Esta clase se utiliza para crear un campo de texto especializado donde, hasta que el usuario escriba algo, se mostrará un texto “placeholder” que servirá como ejemplo para lo que tiene que ingresar el usuario.

Figura 9. Resumen de la implementación



6. Iteración #5 (Entrega 3)

6.1. Emisión de facturas en pdf

Para esta parte creamos una clase llamada facturas que como parámetro necesitará un alquiler que creará la factura en la carpeta de facturas esta clase cuenta con un método constructor y con un método facturas encargado de crear las facturas en este necesitamos acceder a información de la empresa ya que en la parte superior podemos encontrar el nombre de la empresa seguido e Id de factura que será el mismo Id de alquiler luego más abajo encontraremos la fecha de emisión de la factura seguido de la dirección de la sede donde se hace la entrega del vehículo, más abajo podemos encontrar algunos datos personales del cliente como su nombre, telefono y direccion de e-mail, por último encontramos algunos datos de la reserva como el monto a pagar, la duración de la renta y la categoría del vehículo que se dio en alquiler, al final de la página encontramos la firma del encargado de la sede responsable de emitir la factura. Un ejemplo del diseño de la factura se presenta a continuación.


Figura 10. Ejemplo del diseño de la factura

RUII	
Id Factura: 1	

Fecha: 2023-12-07	
Direccion: Cra. 7 #20-2 a 20-109 Bogota, Cundinamarca	

Datos Cliente:	
Nombre: Nicolas	
Telefono: 312493238	
Mail: nperezramos0@gmail.com	

Description:	Monto:
Alquiler vehiculo tipo:	0.0
sedan_SUPREME	
por 5 días.	

6.2. Características de los vehículos

Resulta relevante resaltar que una buena abstracción de los conceptos **Vehículo**, **Categoría** y **Seguro** nos permitió cumplir de antemano con la inclusión de las características solicitadas para la última entrega, considerando que:

- el tipo de vehículo desde un principio fue un atributo obligatorio para la creación de una nueva categoría (véase figura 10), que a su vez es un atributo obligatorio para la creación de un nuevo vehículo (véase figura 11). Es importante mencionar que nuestra implementación permite que cada vehículo solo pueda pertenecer a una categoría, que especifica tanto el tipo de vehículo como el nivel de lujo del mismo.

Figura 10. Formulario para creación de categorías

Ingrese la información de la nueva categoría	
Tipo de vehículo:	<input type="text" value="Ej: PickUp"/>
Nivel de lujo del vehículo:	<input type="text" value="Premium"/>
Capacidad:	<input type="text" value="1"/>
% a pagar por Temporada Alta:	<input type="text" value="0.1"/>
% a pagar por Temporada Baja:	<input type="text" value="0.1"/>
Costo por avería leve (COP):	<input type="text"/>
Costo por avería moderada:	<input type="text"/>
Costo por avería total:	<input type="text"/>
Tarifa diaria:	<input type="text"/>
Categoría superior/padre:	<input type="text" value="0: Ninguna"/>
<input type="button" value="Registrar Categoría"/>	

Figura 11. Formulario para el registro de vehículos

Ingrese la información del nuevo vehículo	
Placa:	<input type="text" value="Ej: ABC123"/>
Marca:	<input type="text" value="Ej: Chevrolet"/>
Modelo:	<input type="text" value="Ej: Aveo"/>
Color:	<input type="text" value="Ej: Azul"/>
Transmisión:	<input type="radio"/> Manual <input type="radio"/> Automática
Categoría:	<input type="text" value="3: sedan_PREMIUM"/>
Sedes:	<input type="text" value="1: SedeChia"/>
<input type="button" value="Registrar vehículo"/>	

- tanto la prima de seguro del vehículo como el cobro de otros seguros fue relacionado con cada categoría a partir del porcentaje de la tarifa diaria que cada seguro debe cobrar. A partir de esto nuestra aplicación propone el cobro del 100% de la tarifa diaria para el motivo “prima de seguro del vehículo”, y tanto con este seguro como con el resto se multiplica la *tarifa diaria* **por** el *porcentaje de cobro* de la tarifa diaria **por** el *número de días de alquiler* para obtener el valor a cobrar bajo el motivo de **seguros** (para mayor detalle respecto a los seguros implementados en la solución, véase la iteración #1 del documento)

6.3. Pagos con Tarjeta de Crédito

Considerando la flexibilidad requerida para la posibilidad de implementar nuevas pasarelas de pago en la aplicación, se tomaron las siguientes decisiones:

1. Se creó la clase abstracta **pasarelaPago** a partir de la cual se pueden realizar las implementaciones concretas de pasarelas de pago nuevas. Dicha clase abstracta propone el despliegue de un JFrame para representar a la pasarela de pagos y en dicha clase se tuvieron en cuenta los siguientes aspectos:
 - a. que su constructor reciba por parámetros al cliente, el motivo de pago, el monto a pagar, el ID de la transacción y el path del archivo en el que se registran las transacciones.

- b. Se estableció `tarjetaValida()`, método que evalúa la validez de los parámetros que se ingresan en la pasarela de pago para realizar una transferencia, tales como: número y titular de la tarjeta, cvc de la tarjeta y fecha de expiración de la tarjeta.
 - c. Se implementó el método `setTarjeta()` que crea el objeto **Tarjeta** una vez la información de transacción es válida.
 - d. Se desarrolló el método `completarTransferencia()` que intenta la transacción pertinente a partir de un objeto **Tarjeta**. El resultado anterior se registrará como un booleano en el atributo *transferenciaCompletada*. Es pertinente mencionar que esta implementación al ser simulada, propone que todo pago con tarjeta de marca **Discover** rebote, para de esta forma experimentar el rebote de pagos.
 - e. Se plantearon los métodos `crearEntrada()` y `agregarEntrada(Object nuevaEntrada)` para el registro de las transacciones en el archivo pertinente.
2. Las implementaciones concretas **PayPal**, **Payu**, **Sire** heredan el comportamiento de **pasarelaPago**, y cada implementación concreta en su constructor despliega el JFrame dentro del que el usuario ingresa los datos necesarios para realizar la transferencia correspondiente. Una vez obtienen los datos del cliente, cada pasarela pasa los datos obtenidos por el método `tarjetaValida()`, tras verificar que sea válida crea la tarjeta con `setTarjeta()` e intenta la transferencia con `completarTransferencia()`. Es importante mencionar que si el usuario se abstiene de completar el proceso de entrega de datos o la transferencia rebota y el usuario no ingresa un nuevo método de pago, esto se reflejará en el booleano *transferenciaCompletada* con un valor de **false**. Caso contrario el booleano será **true**.
3. Se decidió que las clases de las pasarelas de pago se guarden en el package *vista* y los archivos de registro de pagos en la carpeta *registroPagos*.
4. En todos los procesos que involucran el pago de un monto (creación de reserva, modificación de reserva, creación de alquiler, cierre de alquiler) el usuario puede elegir la pasarela de pago. Para esto se hizo uso de método **forName()** que permite la carga dinámica de las pasarelas, cuyos nombres (acompañados por la ruta del archivo en el que registran los pagos) se encuentran en el archivo **config.txt** en la carpeta *registroPagos*.

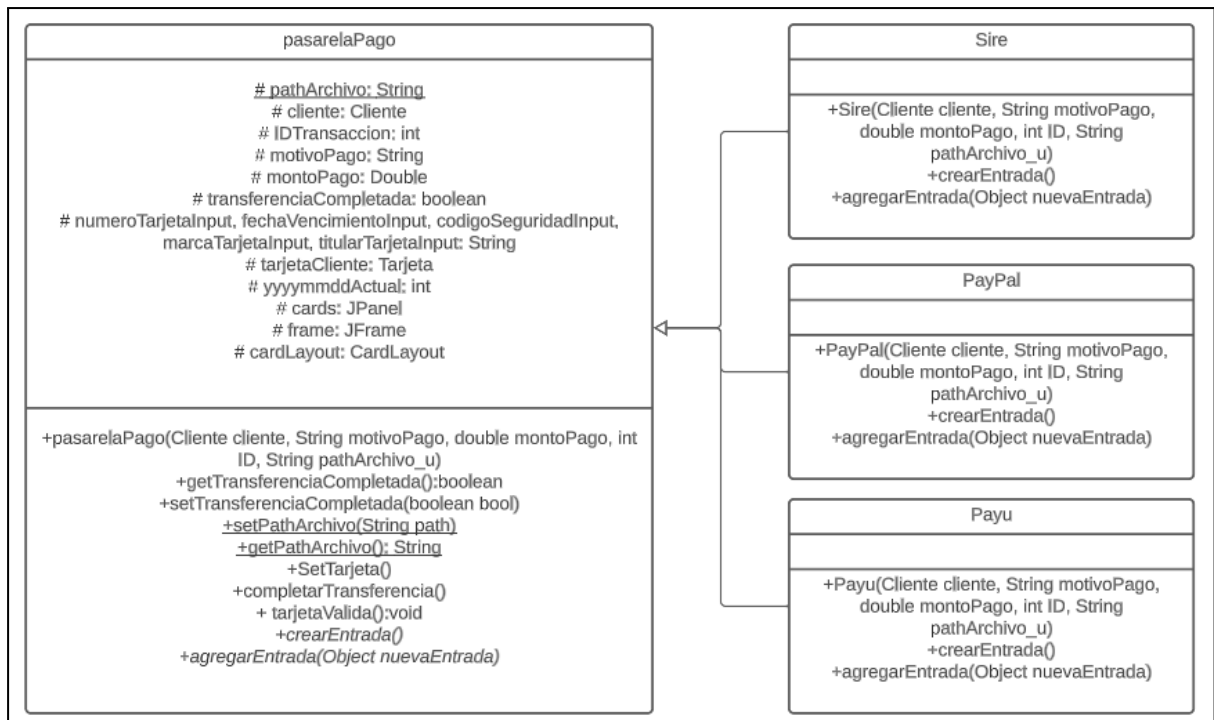
Figura 12. Contenido del archivo config.txt iteración #6

```
registroPagos > E config.txt
1 PayPal->registroPayPal.log
2 Payu->registroPayu.txt
3 Sire->registroSire.json
```

5. Una vez la pasarela se cierra, la lógica de la aplicación toma las decisiones pertinentes a partir del booleano *transferenciaCompletada*.
 6. Se creó un archivo **readme** en la carpeta *registroPagos* en el cual se resumen los pasos a seguir para implementar una nueva pasarela.

El resultado de esta nueva implementación se puede visualizar en el diagrama de clases y resumen en la figura 13.

Figura 13. Resumen implementación pasarelas de pago iteración #6



6.4. Nueva aplicación para clientes

Para esta nueva aplicación, se identificó la posibilidad de invocar métodos ya implementados en la clase `ventanaCliente` y `ventanaMain`, además de modificar los métodos relacionados con la creación de la reserva, para que dicho RF reciba por parámetro un booleano a partir del cual pueda aplicar un 10% al pago de reserva cuando dicho booleano es **True**.

A partir de lo anterior, se planteó la creación de dos nuevas clases: `ventanaMain2` y `ventanaCliente2`, donde la primera permite la autenticación exclusiva de clientes o la creación de nuevos perfiles de usuario y la segunda permite tanto la creación de reservas como la consulta de disponibilidad de una categoría dada en un rango de fechas dado para la sede elegida.

Las clases descritas anteriormente se agruparon en el paquete *vista2* y se resumen en el siguiente diagrama.

Figura 14. Resumen implementación vista2 iteración #6

<<Vista>>
vista2

-> ventanaMain2
->ventanaCliente2