Proyecto entrega 1 – Diseño

Ana María Sánchez Mejía – 202013587

David Leonardo Manrique Lesmes – 201913129

Maria Paula Nizo Vega – 202213902

Tabla de contenido

Contexto del Problema

Antes de iniciar con el proceso de diseño se definieron las funcionalidades de alto nivel que la aplicación de la renta de automóviles debe tener con el fin de satisfacer a la hora de interactuar con la interfaz. Se realizará una aplicación que permita al usuario poder realizar de forma sencilla una reserva de un automóvil en la ciudad, asimismo debe posibilitar que los administradores o empleados de la empresa de alquiler manejen la información y tengan acceso a las modificaciones que dependerán de sus responsabilidades. A continuación, se presenta las principales características del contexto, como los requerimientos funcionales y restricciones:

Requerimientos Funcionales	Restricciones
1. Registrar un nuevo vehículo. 2. Dar un vehículo de baja. 3. Configurar información de cada sede 4. Modificar información de precios para los vehículos. 5. Configurar información de seguros. 6. Agregar un usuario al sistema. 7. Modificar un usuario al sistema. 8. Eliminar un usuario del sistema. 9. Reportar a un vehículo a mantenimiento. 10. Crear un cliente interno. 11. Revisar vehículo. 12. Crear la reserva especial. 13. Entrega de un vehículo. 14. Devolución de un vehículo. 15. Recibir el pago. 16. Registrar un nuevo conductor. 17. Bloquear la tarjeta. 18. Reserva del vehículo. 19. Pago de seguros. 20. Pago de 30%. 21. Crear usuarios.	1. Una de las reglas de dominio es que todos deben ingresar con un LogIn y una contraseña y dependiendo del usuario podrá realizar ciertas actividades en la aplicación. 2. Se asume que no existe un usuario malicioso que pueda modificar la información de la aplicación. 3. Otra regla de dominio es que la aplicación debe guardar toda la información para generar archivos log. 4. También se tiene que todo cliente debe realizar una reserva por internet, no se acepta que lleguen y creen la reserva en la sede. En las sedes solamente se atienden solicitudes extra de la reserva y se entrega el vehículo. 5. Los únicos que pueden reportar que un vehículo necesita mantenimiento son los empleados y no los clientes. 6. Se asume que todos los clientes pagan con tarjeta de crédito. 7. Los trabajadores solo pueden cambiar el estado de los vehículos que están en su sede

Figura 1. Descripción del contexto

En las siguientes figuras se ilustrarán las interacciones de forma genérica, así como los archivos de texto que describen la persistencia de las acciones que cada uno de los actores realice, asimismo, se verán algunos de los requerimientos funcionales. En cuanto a la

interacción con la interfaz, al usuario le corresponde ingresar un usuario y una contraseña para poder comenzar con el funcionamiento de la aplicación. La interfaz, por su parte, desplegara una serie de opciones dependiendo del cargo que cada uno tenga, por ejemplo: hacer reserva (Cliente), registrar precios (Administrador), entregar vehículo (Empleado de la sede), si al vehículo toca hacerle mantenimiento(empleado que revisa el vehículo), lo anterior con el fin de que el usuario haga sus operaciones correspondientes. Es la interfaz la que solicita a la empresa de renta de vehículos en realizar las acciones que marcaron los usuario

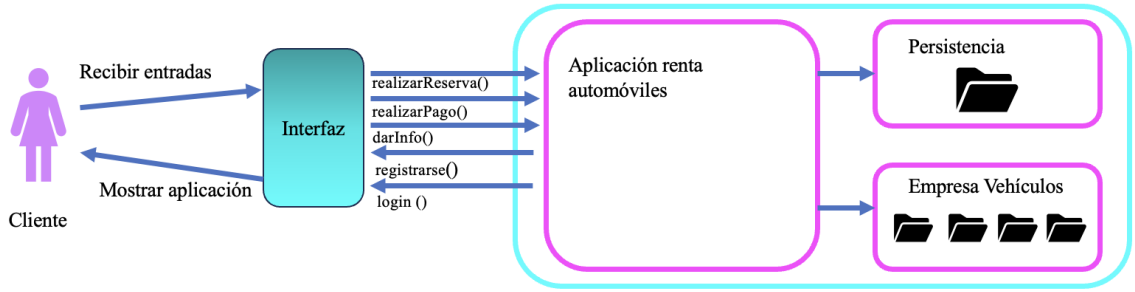

Figura 2. Descripción del contexto cliente

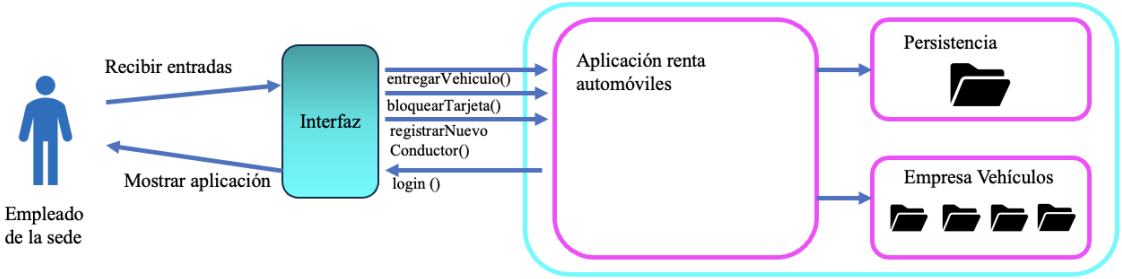

Figura 3. Descripción del contexto empleado de la sede

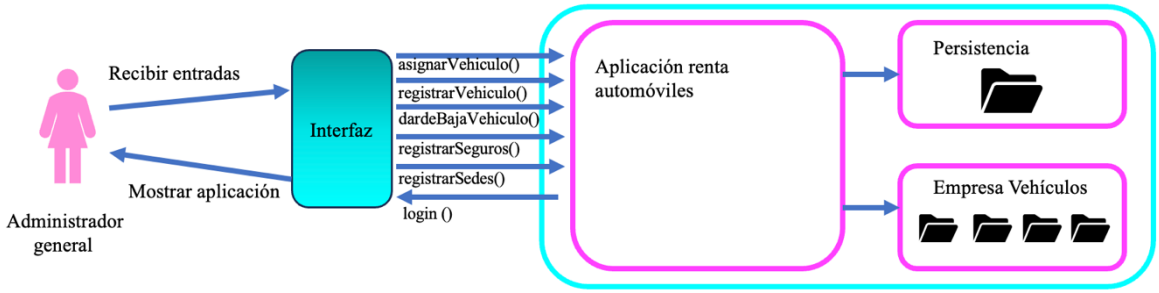

Figura 4. Descripción del contexto administrador general

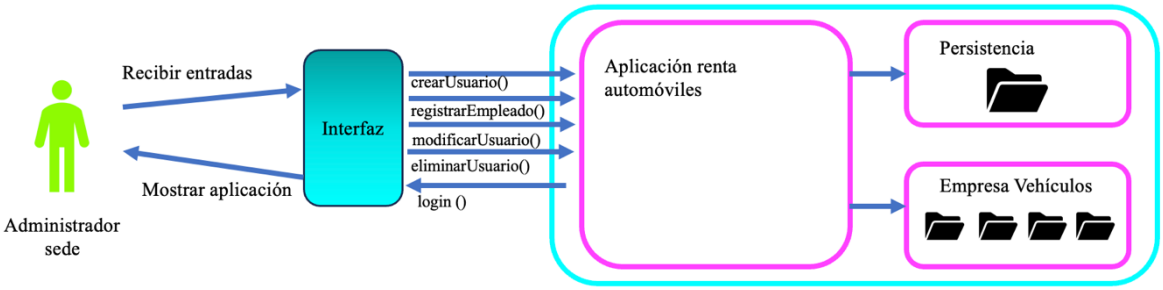

Figura 5. Descripción del contexto administrador sede

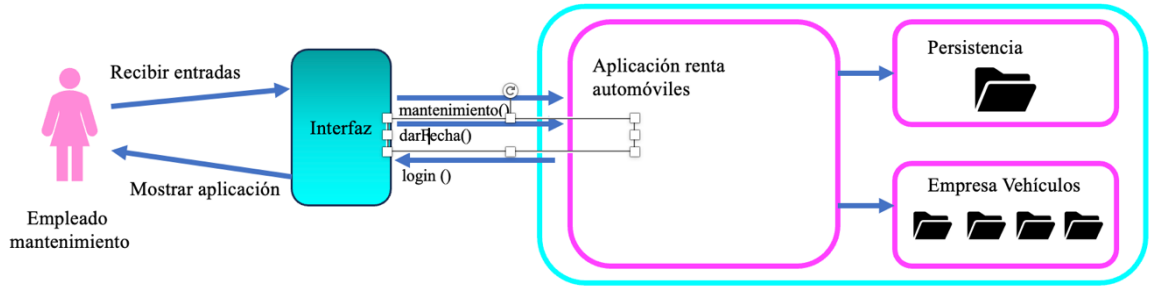

Figura 6. Descripción del contexto empleado mantenimiento

Diagramas

Diagrama de clases de diseño

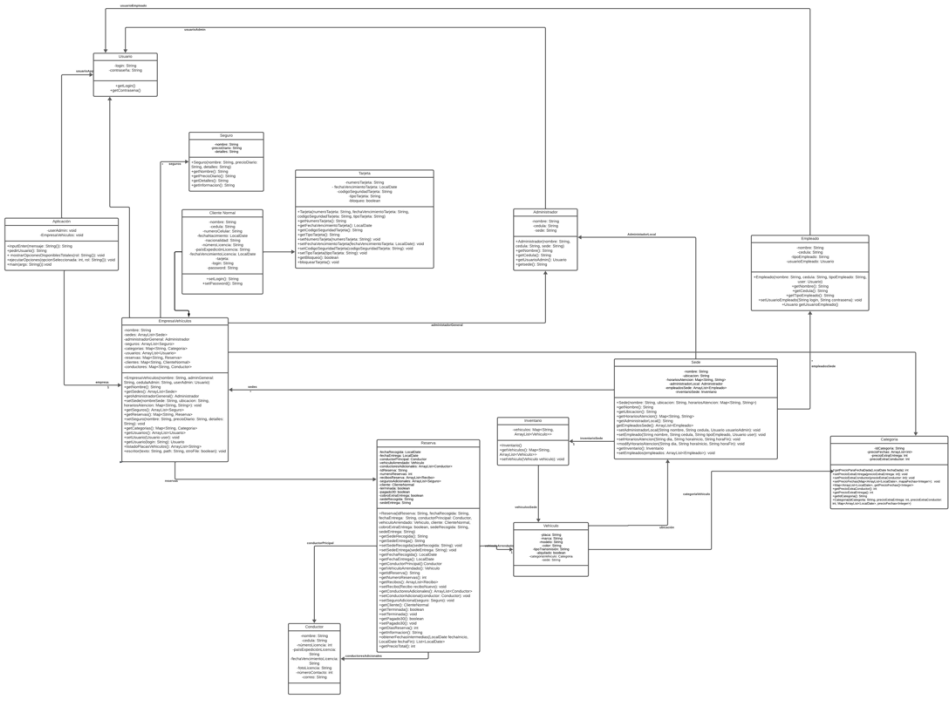

Figura 7. Diagrama de diseño final

Diagrama de clases de alto nivel

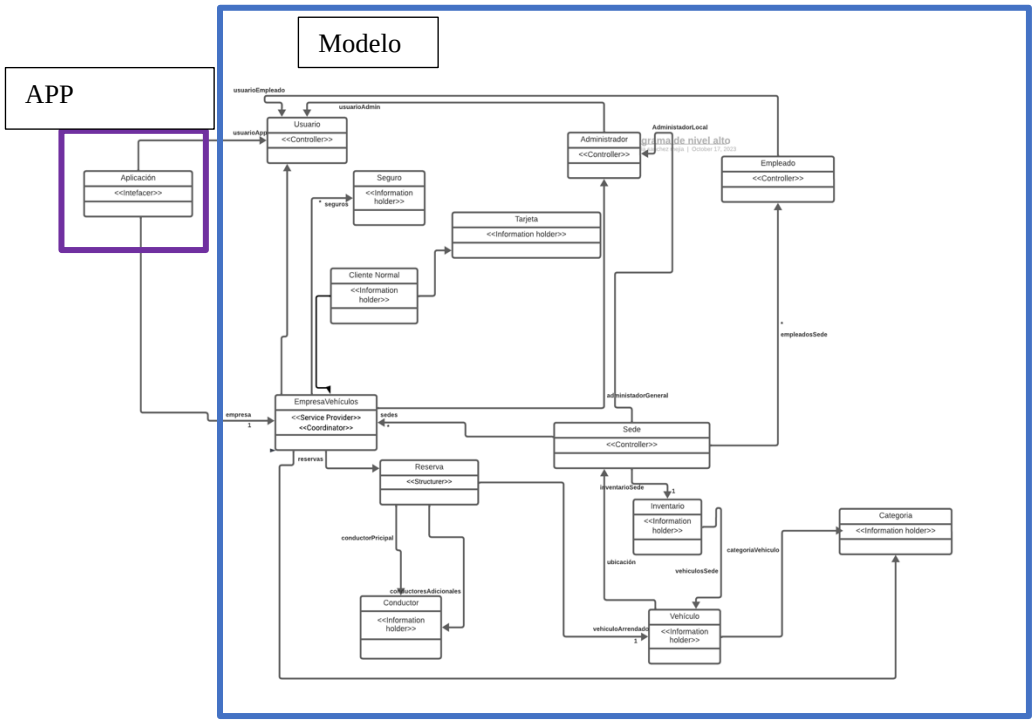

Figura 8. Diagrama de alto nivel

Diagramas de secuencia

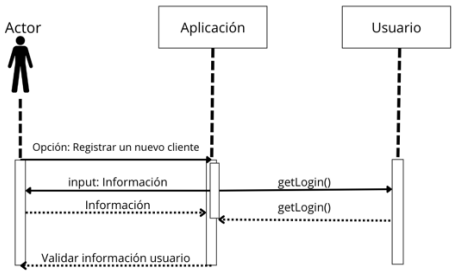

Figura 9. Crear usuario

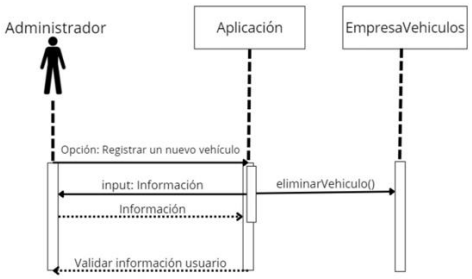

Figura 10. Eliminar vehículo

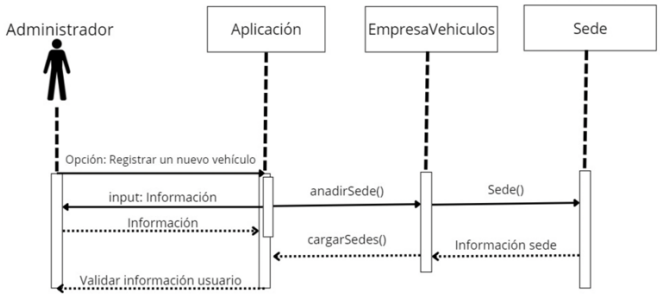

Figura 11. Crear sede

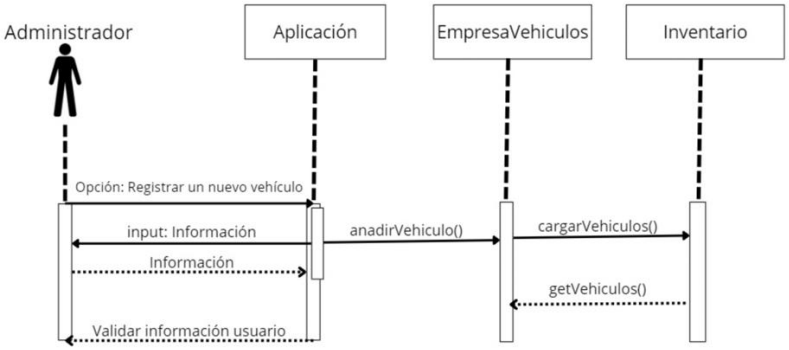

Figura 12. Crear vehículo

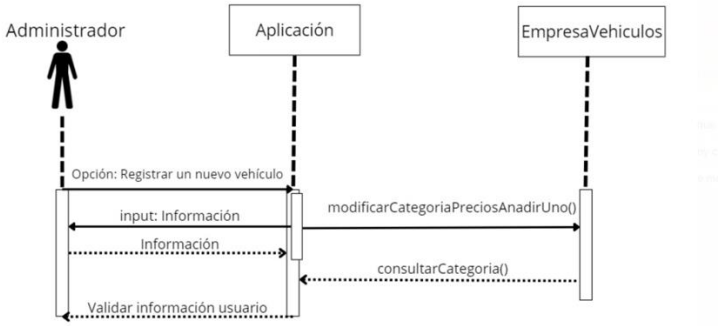

Figura 12. Modificar precios de los vehículos

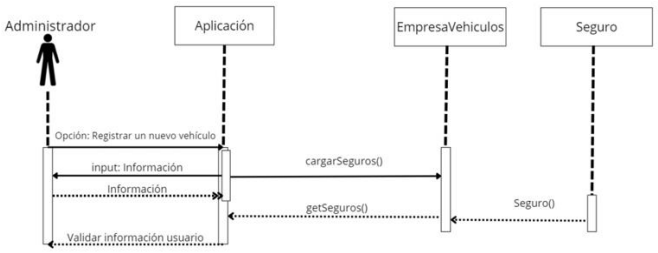

Figura 13. Configurar información de seguros

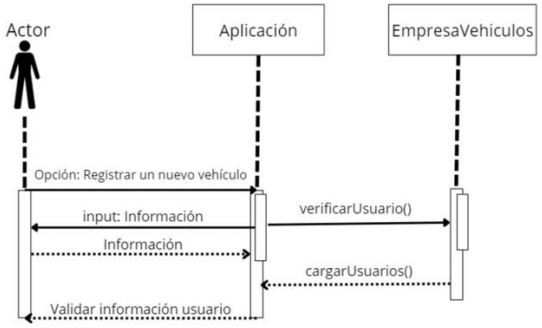

Figura 14. Modificar un usuario al sistema

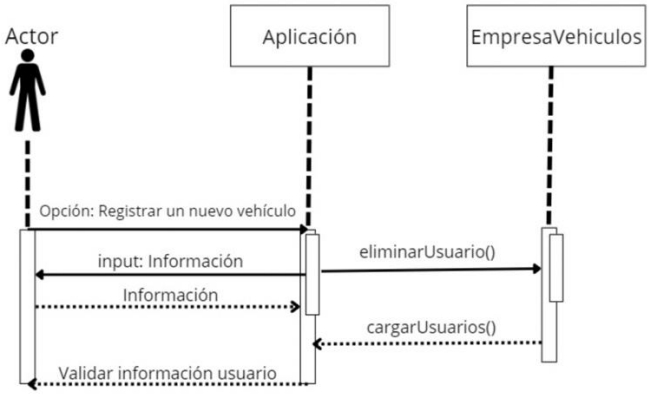

Figura 15. Eliminar un usuario del sistema

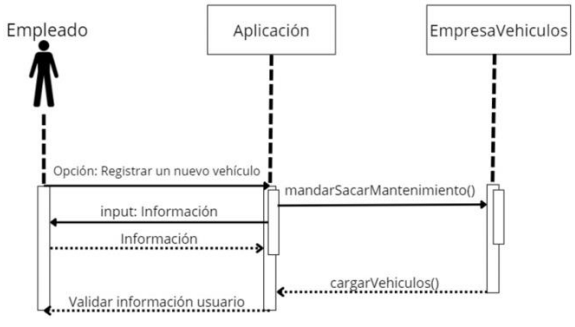

Figura 16. Reportar a un vehículo a mantenimiento

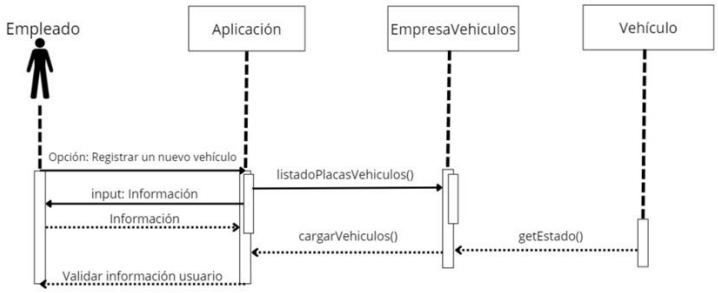

Figura 17. Reportar a un vehículo a mantenimiento

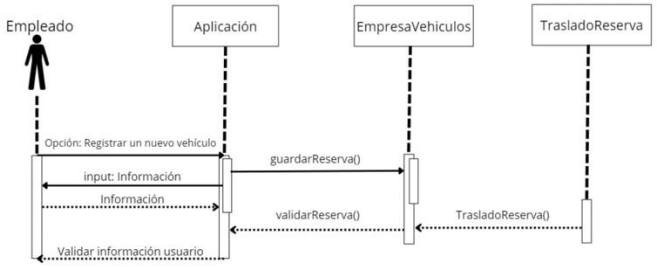

Figura 18. Reportar a un vehículo a mantenimiento

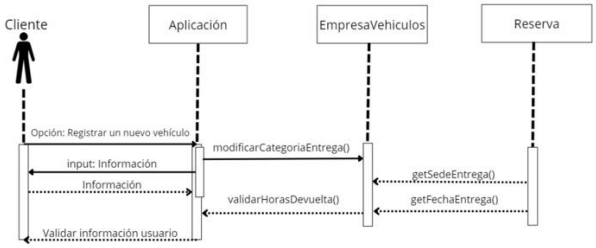

Figura 18. Entrega de un vehículo

Métodos

Vehículo

Constructor:

El constructor de la clase Vehículo toma varios parámetros y se utiliza para crear una instancia de un vehículo con valores iniciales para los atributos. Los parámetros son la placa, la marca, el modelo, el color, el tipo de transmisión, la categoría del vehículo y la sede a la que está asignado.

Métodos Get:

- getSede(): Devuelve la sede a la que está asignado el vehículo.
- getCategoria(): Devuelve la categoría del vehículo.

- getMarca(): Devuelve la marca del vehículo.
- getPlaca(): Devuelve la placa del vehículo.
- getModelo(): Devuelve el modelo del vehículo.
- getColor(): Devuelve el color del vehículo.
- getTipoTransmicion(): Devuelve el tipo de transmisión del vehículo.
- getAlquilado(): Devuelve un valor booleano que indica si el vehículo está alquilado o no.

Usuario

Constructor:

El constructor de la clase Usuario toma tres parámetros: el nombre de usuario (login), la contraseña (contrasena), y el rol del usuario (rol). Se utiliza para crear una instancia de un usuario con valores iniciales para estos atributos.

Métodos Get:

- getLogin(): Devuelve el nombre de usuario del usuario.
- getContrasena(): Devuelve la contraseña del usuario.
- getRol(): Devuelve el rol del usuario en el sistema.

TraladoReserva

Constructor:

El constructor de la clase TrasladoReserva toma varios parámetros y se utiliza para crear una instancia de un traslado con valores iniciales para los atributos. Los parámetros son la placa del vehículo, la fecha en formato de cadena, la sede de salida, la sede de llegada y el conductor responsable. El constructor realiza la conversión de la fecha de cadena a un objeto LocalDate para almacenarla adecuadamente.

Métodos Get:

- getPlacaVehiculo(): Devuelve la placa del vehículo utilizado en el traslado.
- getFecha(): Devuelve la fecha del traslado en formato LocalDate.
- getSedeSalida(): Devuelve la sede de salida.
- getSedeLlegada(): Devuelve la sede de llegada.
- getConductor(): Devuelve el objeto ClienteEspecial que representa el conductor o responsable del traslado.

Tarjeta

Constructor:

El constructor de la clase Tarjeta toma cuatro parámetros y se utiliza para crear una instancia de una tarjeta con valores iniciales para los atributos. Los parámetros son el

número de tarjeta, la fecha de vencimiento en formato de cadena, el código de seguridad y el tipo de tarjeta.

Métodos Get:

- getFechaVencimientoTarjeta(): Devuelve la fecha de vencimiento de la tarjeta en formato LocalDate.
- getNumeroTarjeta():Devuelve el número de la tarjeta.
- getCodigoSeguridadTarjeta(): Devuelve el código de seguridad de la tarjeta.
- getTipoTarjeta(): Devuelve el tipo de tarjeta.
- getBloqueo(): Devuelve un valor booleano que indica si la tarjeta está bloqueada o no.

Métodos Set:

- setNumeroTarjeta(String numeroTarjeta): Permite establecer el número de la tarjeta.
- setFechaVencimientoTarjeta(LocalDate fechaVencimientoTarjeta): Permite establecer la fecha de vencimiento de la tarjeta en formato LocalDate.
- setCodigoSeguridadTarjeta(String codigoSeguridadTarjeta): Permite establecer el código de seguridad de la tarjeta.
- setTipoTarjeta(String tipoTarjeta): Permite establecer el tipo de tarjeta.

Método bloquearTarjeta():

Este método se utiliza para cambiar el estado de bloqueo de la tarjeta a true, lo que indica que la tarjeta está bloqueada.

Seguro

Constructor:

El constructor de la clase Seguro toma tres parámetros y se utiliza para crear una instancia de un seguro con valores iniciales para estos atributos. Los parámetros son el nombre del seguro, el precio diario y los detalles.

Métodos Get:

- getNombre(): Devuelve el nombre del seguro.
- getPrecioDiario(): Devuelve el precio diario del seguro.
- getDetalles(): Devuelve los detalles del seguro.

Método getInformacion():

Este método concatena la información del seguro en una cadena de texto formateada y la devuelve. Proporciona una descripción más detallada del seguro, incluyendo el nombre, el precio diario y los detalles.

Sede

Constructor:

El constructor de la clase Sede toma tres parámetros: el nombre de la sede, la ubicación y los horarios de atención en forma de mapa. Se utiliza para crear una instancia de una sede con valores iniciales para estos atributos. Además, inicializa la lista de empleados y el inventario de la sede.

Métodos Get:

- getNombre(): Devuelve el nombre de la sede.
- getUbicacion(): Devuelve la ubicación de la sede.
- getHorariosAtencion(): Devuelve el mapa de horarios de atención.
- getAdministradorLocal(): Devuelve el nombre del administrador de la sede.
- getEmpleadosSede(): Devuelve la lista de empleados de la sede.
- getInventario(): Devuelve el objeto Inventario de la sede.

Métodos Set:

- setAdministradorLocal(String nombre, String cedula, Usuario usuarioAdmin): Permite establecer el administrador local de la sede, creando un nuevo objeto Administrador.
- setEmpleado(String nombre, String cedula, String tipoEmpleado, Usuario user): Permite agregar un empleado a la lista de empleados de la sede.
- setHorariosAtencion(String dia, String horaInicio, String horaFin): Permite establecer los horarios de atención para un día de la semana en el mapa de horarios.
- modifyHorarioAtencion(String dia, String horaInicio, String horaFin): Permite modificar los horarios de atención para un día de la semana en el mapa de horarios.
- setEmpleados(ArrayList<Empleado> empleados): Permite establecer la lista de empleados de la sede.

Reserva

Constructor:

El constructor de la clase Reserva toma múltiples parámetros y se utiliza para crear una instancia de una reserva con valores iniciales para estos atributos. Los parámetros incluyen el identificador de la reserva, las fechas de recogida y entrega, el conductor principal, el vehículo arrendado, el cliente, indicadores de cobro extra y las sedes de recogida y entrega.

Métodos Get:

Varias funciones get permiten acceder a los atributos de la reserva, como las fechas, conductores, vehículos, etc.

Métodos Set:

Las funciones setConductorAdicional, setSeguroAdicional, setSedeRecogida, y setSedeEntrega permiten agregar conductores adicionales, seguros adicionales y actualizar las sedes de recogida y entrega.

Métodos de Cálculo de Precio:

getPrecioTotal: Calcula el precio total de la reserva, teniendo en cuenta la duración, conductores adicionales, seguros y cargos extra.

Métodos de Información:

getInformacion: Proporciona información detallada sobre la reserva, incluyendo su identificador, fechas y cliente.

Método Estático para Obtener Fechas Intermedias:

obtenerFechasIntermedias: Un método estático que devuelve una lista de fechas intermedias entre dos fechas dadas.

El código representa una clase que modela una reserva de vehículo. Almacena información relevante sobre la reserva, incluyendo las fechas, los conductores, el vehículo, los seguros y otros detalles. Además, ofrece métodos para calcular el precio total de la reserva y proporcionar información detallada sobre la misma.

Recibo

Constructor:

Recibo(): Este es el constructor de la clase Recibo, que inicializa un objeto recibo. En este caso, el constructor no recibe ningún parámetro y no realiza ninguna acción especial al ser invocado.

Métodos Get:

- getIdReserva(): Un método de acceso que devuelve el valor del atributo idReserva. Devuelve un entero que representa el identificador de la reserva asociada al recibo.
- getTipoObjeto(): Un método de acceso que devuelve el valor del atributo tipoObjeto. Devuelve una cadena de texto que representa el tipo de objeto relacionado con el recibo.
- getPrecio(): Un método de acceso que devuelve el valor del atributo precio. Devuelve un entero que representa el precio asociado al recibo.

Métodos Set:

- setIdReserva(int idReserva): Un método de modificación que establece el valor del atributo idReserva con el valor pasado como parámetro.

- setTipoObjeto(String tipoObjeto): Un método de modificación que establece el valor del atributo tipoObjeto con la cadena de texto pasada como parámetro.
- setPrecio(int precio): Un método de modificación que establece el valor del atributo precio con el entero pasado como parámetro.

Inventario

Constructor:

Inventario(): Este es el constructor de la clase Inventario, que inicializa un objeto inventario. En este caso, el constructor inicializa la variable vehiculos como un nuevo HashMap vacío.

Métodos:

- getVehiculos(): Un método de acceso que devuelve el mapa vehiculos que contiene la información sobre los vehículos organizados por categoría.
- setVehiculo(Vehiculo vehiculo): Un método que agrega un vehículo al inventario. Toma un objeto de la clase Vehiculo como parámetro y lo agrega al mapa vehículos en la lista correspondiente a su categoría. Si no existe una lista para la categoría, se crea una nueva lista y se agrega el vehículo. Si ya existe una lista para la categoría, se agrega el vehículo a esa lista.

EmpresaVehiculo

Constructor:

Este constructor crea una instancia de la clase EmpresaVehiculos y la inicializa con los siguientes campos:

nombre: Se le asigna el valor proporcionado como el nombre de la empresa.

administradorGeneral: Se crea una instancia de la clase Administrador y se asigna como el administrador general de la empresa.

sedes: Se inicializa como una lista vacía, presumiblemente para almacenar ubicaciones o sucursales.

categorias: Se inicializa como un mapa vacío, posiblemente utilizado para almacenar categorías.

seguros: Se inicializa como una lista vacía, posiblemente utilizada para almacenar tipos de seguros.

usuarios: Se inicializa como una lista vacía, posiblemente utilizada para almacenar usuarios.

reservas: Se inicializa como un mapa vacío, probablemente para almacenar información sobre reservas.

clientes: Se inicializa como un mapa vacío, posiblemente utilizado para almacenar datos de clientes.

conductores: Se inicializa como un mapa vacío, presumiblemente para almacenar información sobre conductores.

En conjunto, este constructor establece los valores iniciales de los campos esenciales de la clase EmpresaVehiculos para su uso posterior en la gestión de una empresa relacionada con vehículos.

Métodos Principales:

A continuación, se describen algunos de los métodos más relevantes de la clase:

- EmpresaVehiculos: Constructor de la clase que inicializa la empresa de vehículos con un nombre, un administrador general y otras estructuras de datos vacías.
- setSede: Permite agregar una nueva sede a la empresa. Recibe como parámetros el nombre de la sede, la ubicación y un mapa con los horarios de atención.
- setSeguro: Permite agregar un nuevo seguro a la empresa. Recibe el nombre del seguro, el precio diario y detalles. También actualiza un archivo de texto con la información de los seguros.
- setUsuario: Agrega un nuevo usuario al sistema y lo almacena en la lista de usuarios. También actualiza un archivo de texto con la información de los usuarios.
- verificarUsuario: Comprueba si un usuario con un nombre de usuario y contraseña proporcionados existe y devuelve su rol.
- eliminarVehiculo: Elimina un vehículo del sistema y actualiza el archivo de vehículos correspondiente.

- anadirVehiculo: Permite agregar un vehículo a una sede específica y actualiza el archivo de vehículos.

- anadirSede: Agrega una nueva sede a la empresa junto con su administrador y usuarios correspondientes. También actualiza el archivo de sedes.

- eliminarSede: Elimina una sede de la empresa y actualiza el archivo de sedes.

- consultarCategoria: Consulta una categoría de vehículos por su identificador.

- modificarCategoriaPreciosAnadirUno: Modifica el precio de una categoría para un rango de fechas específico.

- modificarCategoriaConductor y modificarCategoriaEntrega: Estos métodos permiten modificar los precios adicionales por conductor y entrega para una categoría de vehículos.

- eliminarSeguro y agregarSeguro: Estos métodos se utilizan para eliminar y agregar seguros a la empresa.

- agregarUsuario: Agrega un nuevo usuario a una sede específica y actualiza el archivo de usuarios.

- eliminarUsuario: Elimina un usuario de la empresa y actualiza los archivos correspondientes.

- validarReserva y guardarReserva: Estos métodos se utilizan para validar y guardar reservas en el sistema.

- Varios métodos para cargar información desde archivos de texto, como cargarVehiculos, cargarSeguros, cargarCategorias, cargarSedes, cargarUsuarios, y cargarClientes.

Es importante destacar que esta explicación se basa en el código proporcionado, y la implementación real de cada método puede variar según los detalles en las clases relacionadas. La clase EmpresaVehiculos es un componente clave para gestionar todos los aspectos de una empresa de alquiler de vehículos y facilitar su funcionamiento.

Empleado

Constructor:

El constructor de la clase permite crear una instancia de Empleado y establecer los valores iniciales de los atributos nombre, cedula, tipoEmpleado, y usuarioEmpleado.

Métodos get:

- getNombre(): Este método devuelve el nombre del empleado.
- getCedula(): Devuelve la cédula del empleado.
- getTipoEmpleado(): Devuelve el tipo de empleado.
- getUsuarioEmpleado(): Devuelve el objeto Usuario asociado a este empleado.

Métodos set:

- setUsuarioEmpleado(String login, String contrasena): Este método permite establecer un nuevo objeto Usuario para el empleado. Se le proporciona un nombre de usuario (login), una contraseña (contrasena) y el tipo de empleado. Este método crea un nuevo objeto Usuario con estos valores y lo asigna al atributo usuarioEmpleado.

Conductor

Constructor:Conductor: El constructor de la clase permite crear una instancia de Conductor y establecer los valores iniciales de los atributos utilizando los parámetros proporcionados. La fecha de vencimiento de la licencia se convierte de un formato de cadena a un objeto LocalDate.

Métodos get:

getNombre(), getCedula(), getNumeroLicencia(), getPaisExpedicionLicencia(), getFechaVencimientoLicencia(), getFotoLicencia(), getNumeroContacto(), y getCorreo(): Estos métodos permiten acceder a los valores de los atributos del conductor.

ClienteNormal

Constructor:

El constructor de la clase permite crear una instancia de Conductor y establecer los valores iniciales de los atributos utilizando los parámetros proporcionados. La fecha de vencimiento de la licencia se convierte de un formato de cadena a un objeto LocalDate.

Métodos get:

getNombre(), getCedula(), getNumeroLicencia(), getPaisExpedicionLicencia(), getFechaVencimientoLicencia(), getFotoLicencia(), getNumeroContacto(), y getCorreo(): Estos métodos permiten acceder a los valores de los atributos del conductor.

ClienteEspecial

Constructor:

El constructor permite crear una instancia de ClienteEspecial y establecer los valores iniciales de los atributos utilizando los parámetros proporcionados.

Métodos get:

getNombre() y getCedula(): Estos métodos son requeridos por la interfaz Cliente y permiten acceder a los valores de los atributos del cliente.

Categoría

Métodos:

Constructor Categoria: El constructor permite crear una instancia de Categoria y establecer los valores iniciales de los atributos utilizando los parámetros proporcionados.

Métodos get:

getIdCategoria(), getPrecioExtraEntrega(), getPrecioExtraConductor(), y getPrecioFechas(): Estos métodos permiten acceder a los valores de los atributos de la categoría.

getPrecioParaFechaDada(LocalDate fechaDada): Este método permite obtener el precio para una fecha específica dentro del período definido en precioFechas. Recorre las fechas y compara si la fecha proporcionada está dentro del rango de fechas, y si lo está, devuelve el precio correspondiente.

Métodos set:

PrecioExtraEntrega(), setPrecioExtraConductor(), y setPrecioFechas(): Estos métodos permiten actualizar los valores de los atributos de la categoría.

Administrador

Métodos:

Constructor Administrador: El constructor permite crear una instancia de Administrador y establecer los valores iniciales de los atributos utilizando los parámetros proporcionados.

Métodos get:

getNombre(), getCedula(), getUsuarioAdmin(), y getSede(): Estos métodos permiten acceder a los valores de los atributos del administrador.

Decisiones de Diseño

- Se tomó la decisión de que la clase Aplicación sea un <<Interfacer>> para las opciones del menú de los usuarios. Esta clase solamente tiene la función de ejecutar los procesos de autenticación, inicio de sesión y registro en la plataforma. Se tomó la decisión de almacenar los usuarios como una lista. Esto para reducir la complejidad de búsqueda para la autenticación del usuario.
- Se tomó la decisión de que la clase empresaVehiculos sea de estereotipo <<Service Provider>> y que Administrador sean<<Controller>> para que se encarguen de las funciones complejas del sistema. En particular, Administrador se encarga de definir los precios, registrar los empleados, crear una persistencia. Es decir, su rol se basa en la interacción con la información real y la carga de esta al sistema.
- Se tomó la decisión de que la clase Sede almacene un inventario de los carros que tiene cada una de las sedes. La clase Sede actúa como <<Information Holder>> ya que solamente conserva la información de los vehículos y actualiza el inventario de los carros que se rentan, si toca o no mandarlos a reparación.
- Adicionalmente, se tomó la decisión de almacenar los tipos de vehículos en la clase clasificación la cual permitirá que el cliente pueda escoger dentro de las diferentes categorías (Suv, Automóvil, Camioneta, etc). Esto con el objetivo de organizar la información para su búsqueda, inserción y eliminación, puesto que son pocos datos en toda ocasión.

- Se tomó la decisión de definir la clase Aplicación como un <<Coordinator>> ya que a través de la interacción por consola de los menús, se delega a las otras clases que están dentro del paquete de la lógica que realicen sus funciones.

Roles

Asignar roles a las clases es una parte importante del diseño de un sistema. Los roles definen cómo cada clase se relaciona con otras clases y qué responsabilidades tiene en el sistema.

- Control de contraseñas – Usuario
- Control de usuarios – Usuario
- Carga y lectura de archivos de datos – Empresa Vehículos
- Representa a un usuario del sistema. – Usuario
- Almacenamiento de la información de vehículos – Inventario
- Realiza una reserva de un vehículo – Reserva
- Almacenamiento de la información de un nuevo vehículo – Vehículo
- Almacenamiento de la información de una tarjeta – Tarjeta
- Almacenamiento de la información de los seguros – Seguro
- Almacenamiento de la información de los inventarios – Sede
- Almacenamiento de la información de los clientes – Clientes

Responsabilidades

Vehículo:

Responsabilidades:

Almacenar y proporcionar información detallada sobre el vehículo, incluyendo placa, marca, modelo, y más.

Mantener un registro del estado de alquiler del vehículo.

Proporcionar datos esenciales para la gestión de flotas de vehículos.

Usuario:

Responsabilidades:

Almacenar credenciales de acceso, incluyendo nombre de usuario y contraseña.

Identificar el rol del usuario en el sistema (administrador, empleado, cliente).

Facilitar la autenticación y el control de acceso.

TrasladoReserva:

Responsabilidades:

Almacenar información relevante sobre el traslado, como placa del vehículo, fecha, y sedes de salida y llegada.

Facilitar la administración de traslados relacionados con reservas.

Tarjeta:

Responsabilidades:

Almacenar detalles de la tarjeta, incluyendo número, fecha de vencimiento y tipo.

Gestionar el estado de bloqueo de la tarjeta para pagos seguros.

Seguro:

Responsabilidades:

Almacenar información sobre el seguro, incluyendo nombre, precio y detalles adicionales.

Proporcionar datos necesarios para la cobertura de seguros.

Sede:

Responsabilidades:

Almacenar información de la sede, como ubicación, horarios de atención y personal.

Facilitar la gestión y operación de las distintas sedes.

Reserva:

Responsabilidades:

Almacenar información completa sobre la reserva, incluyendo fechas, vehículos, conductores y detalles de cobro.

Calcular el precio total de la reserva en función de los elementos seleccionados.

Recibo:

Responsabilidades:

Almacenar información sobre el recibo, incluyendo el ID de la reserva, tipo de objeto y precio.

Registrar transacciones y pagos relacionados con reservas.

Inventario:

Responsabilidades:

Almacenar información detallada sobre vehículos clasificados por categoría.

Permitir la adición y gestión de vehículos en el inventario.

EmpresaVehiculo:

Responsabilidades:

Coordinar y gestionar sedes, seguros, usuarios, reservas y más.

Facilitar el acceso y la gestión de todos los componentes del sistema.

Realizar funciones de autenticación y control de acceso en el sistema.

Empleado:

Responsabilidades:

Almacenar información personal y roles de los empleados.

Proporcionar acceso a detalles individuales de los empleados.

Facilitar la creación y gestión de usuarios asociados a los empleados.

Conductor, ClienteNormal, ClienteEspecial:

Responsabilidades:

Almacenar detalles específicos de conductores y clientes, como licencias, contactos y datos personales.

Proporcionar acceso a los atributos individuales de conductores y clientes.

Contribuir a la gestión de conductores y clientes en el sistema.