DOCUMENTO DE ANÁLISIS

El modelo del dominio se hizo bajo el estándar de un diagrama UML, por lo que sigue las reglas del mismo.

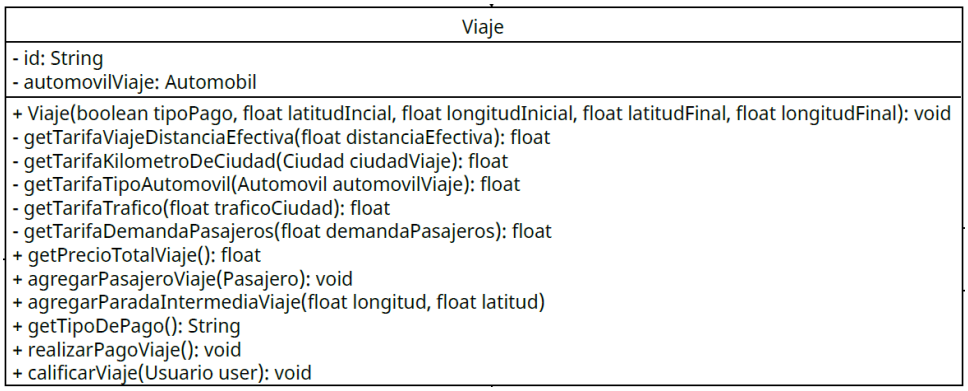
Funcionalidades:

- *Usuarios:*
    - Usuario Anónimo | Puede registrarse como usuario pasajero
    - Usuario Pasajero | Puede solicitar un viaje
    - Usuario Conductor | Cuenta tiene que ser creada desde sistema web de administrador. Este puede buscar viajes disponibles y consigo tiene registrado los vehículos que estén a su nombre.
- Viaje:
    - Se puede solicitar el precio total del viaje (Este contiene consigo Tarifa de distancia efectiva, tarifa kilómetro de la ciudad, tarifa por tipo de automóvil, tarifa por tráfico y tarifa por demanda de pasajeros).
    - Se puede agregar un pasajero al viaje en curso (Para que en el momento del pago si ambos tienen tarjeta puedan dividir el pago)
    - Se puede agregar una parada intermedia al viaje en curso.
    - Se puede realizar pago del viaje con efectivo y si se paga con tarjeta y los pasajeros tienen una registrada se puede dividir el pago entre ellos.
    - Se puede calificar el viaje tanto de parte del conductor como de los pasajeros.
- Extras:
    - Cuando el conductor no esté en un viaje recibirá puntos por mantenerse en movimiento.
    - Sistema web permite a los administradores crearUsuariosConductores, pagar saldo pendiente a conductores y saber la cantidad de conductores registrados.

Nota:

- No especificamos detalles como los porcentajes de distribución de la empresa y el conductor, la cantidad de calificaciones necesarias para establecer una calificación base, entre otros. Esto debido a que estos son muy específicos y no tienen que ser tenidos en cuenta en esta fase de diseño. Sin embargo, estos estarán presentes en la fase de implementación (que es la que precede a la fase que estamos realizando).

Glosario:



Viaje: Hace referencia a la clase Viaje. Es el elemento principal del programa. Tiene como atributos el ID de un viaje y el automóvil que le corresponde. Recibe por parámetros el tipo de pago (efectivo o tarjeta de crédito) y las coordenadas del punto de origen y punto final del viaje.

Se encarga de obtener toda la información relacionada directamente con el viaje:

-Distancia efectiva del viaje.

-Tarifa por kilómetro de la ciudad correspondiente.

-Tarifa según el tipo de automóvil.

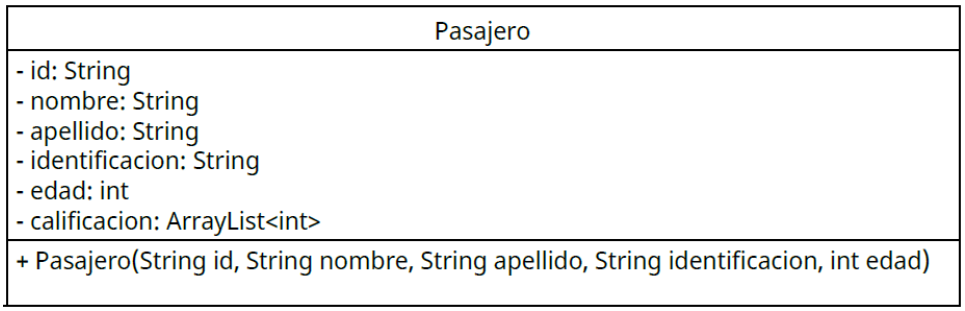
-Tarifa según las condiciones del tráfico.

-Tarifa según la demanda al momento de solicitar el viaje.

-Precio total del viaje.

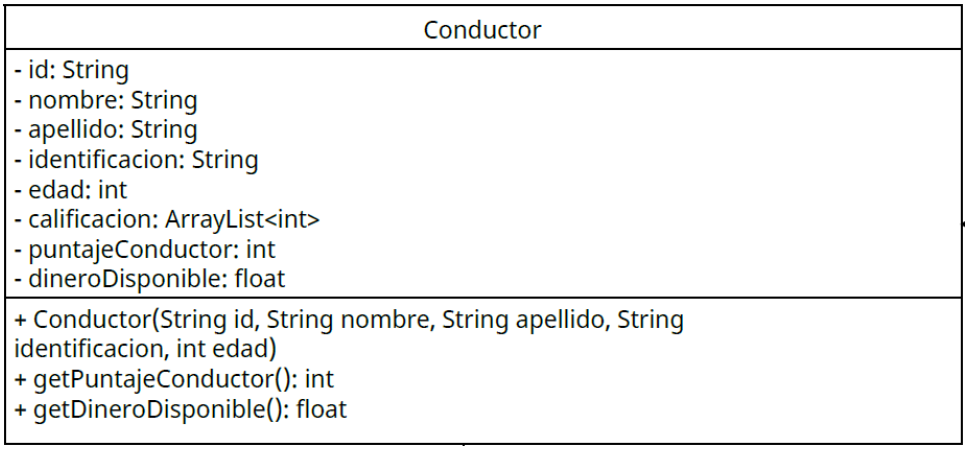
-Tipo de pago (efectivo o tarjeta de crédito).

-Calificación del viaje.



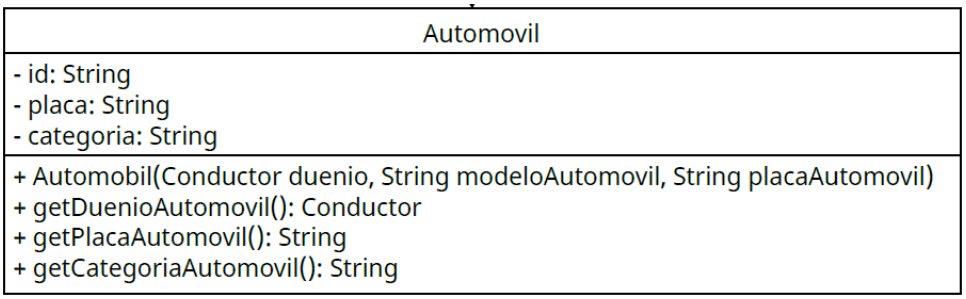
Pasajero: Hace referencia a la clase pasajero, cuyo rol en el contexto de la aplicación es solicitar los viajes. Tiene como atributos el ID del pasajero, su nombre, apellido, identificación, edad y la calificación promedio que tiene por parte de los conductores.

Para que el pasajero pueda hacer uso de la aplicación debe suministrar los datos mencionados anteriormente.



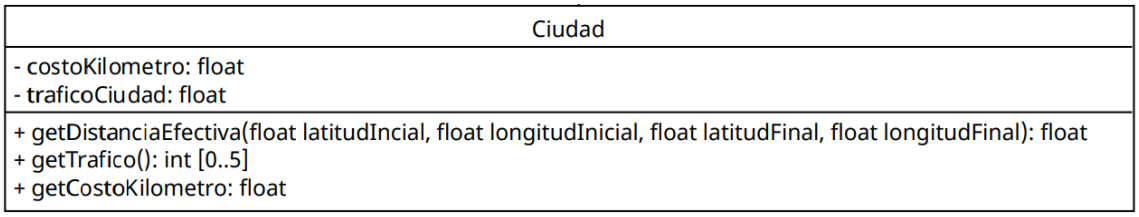
Conductor: Hace referencia a la clase conductor, cuyo rol en el contexto de la aplicación es ofrecer sus servicios y su automóvil para realizar los viajes. Tiene como atributos el ID del conductor, su nombre, apellido, identificación, edad y la calificación promedio que tiene por parte de los pasajeros. Además, tiene también un puntaje asignado, el cual depende de qué tanto circula por la ciudad cuando no está realizando un viaje. El dinero disponible hace referencia a los ingresos del conductor por los viajes realizados. Este monto puede llegar a ser negativo si los pagos son realizados en efectivo, ya que en ese caso el conductor le debe dinero a la empresa, caso contrario a que los pagos se realizaran con tarjeta de crédito, donde el monto es positivo ya que la empresa le debe dinero al conductor.

Esta clase se encarga de obtener la información del conductor: sus datos personales, su puntaje promedio y el dinero disponible.



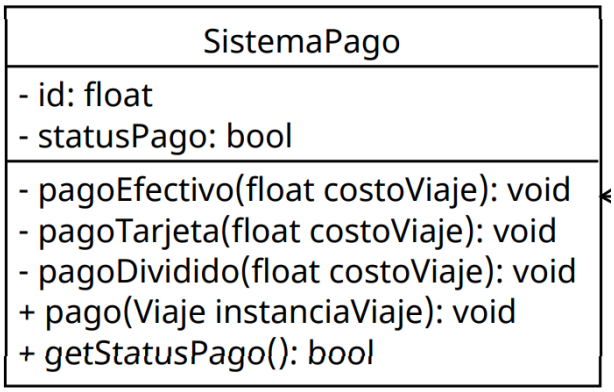
Automóvil: Hace referencia a la clase automóvil. El rol que desempeña es asociar uno o más automóviles con características específicas a los pasajeros. Los métodos son el ID del automóvil, la placa y la categoría a la que pertenece (A, B o C) dependiendo de qué tan lujoso es.

Esta clase se encarga de obtener la información del automóvil, recibe por parámetros el nombre del dueño del automóvil, el modelo y la placa. De esta clase se pueden obtener los datos mencionados anteriormente en caso de que se requiera su consulta.



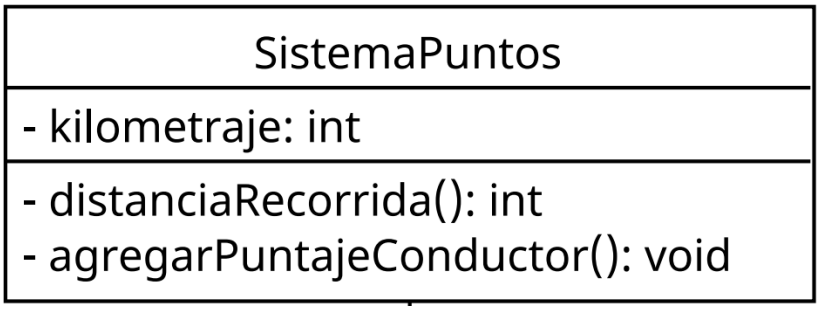
Ciudad: Hace referencia a la clase ciudad. El rol que desempeña es proveer información de la ciudad donde se realiza un viaje, ya que esta influye en el costo dependiendo de ciertas características. Sus atributos son el costo por kilómetro y el nivel del tráfico de esa ciudad.

Dada la ciudad, se requieren las coordenadas iniciales y finales del viaje para obtener la distancia efectiva. También se puede consultar el nivel del tráfico y el costo por kilómetro correspondiente a esa ciudad.

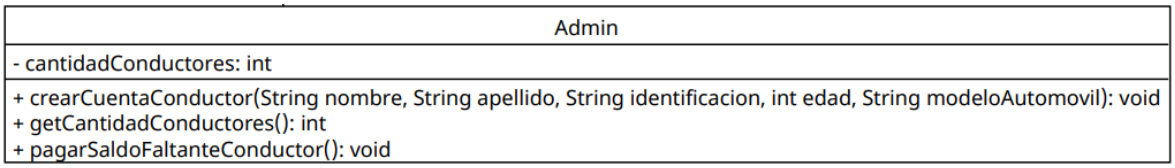


Sistema Pago: Hace referencia a la clase SistemaPago, su rol es determinar las condiciones en las que uno o varios pasajeros realizan el pago de un viaje. Los atributos son el ID del pago y el “status”, el cual puede ser en efectivo o con tarjeta de crédito.

Esta clase se encarga de obtener la información sobre la forma como se realiza el pago. Si el método utilizado es tarjeta de crédito, en caso de que existan varios pasajeros el viaje puede ser compartido. Esto no es posible para pago en efectivo.

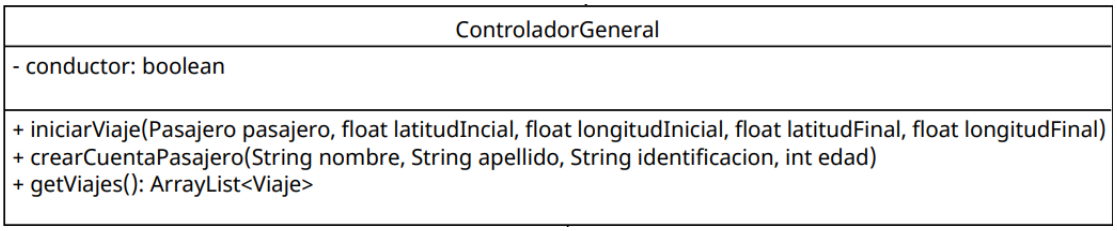


Sistema Puntos: Este elemento del modelo cumple el rol de registrar la información relacionada al sistema de puntos implementado, en el cual se premia a los conductores por transitar constantemente cuando no están haciendo un viaje. Tiene como atributos el kilometraje recorrido mientras no está en un viaje. Sus métodos permiten obtener esa distancia recorrida y agregarle el puntaje al conductor correspondiente.

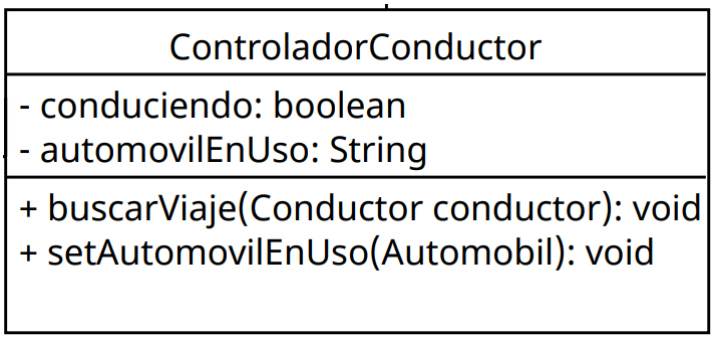


Admin: Este elemento fue creado para representar el rol de administrador en la aplicación. Es el único con la facultad para registrar varios conductores y administrar los pagos de estos.

Tiene como atributo la cantidad de conductores correspondiente al administrador. Esta clase permite crear una cuenta de un conductor dados sus datos personales y la información de su automóvil. También permite consultar la cantidad de conductores registrados y administrar los pagos.

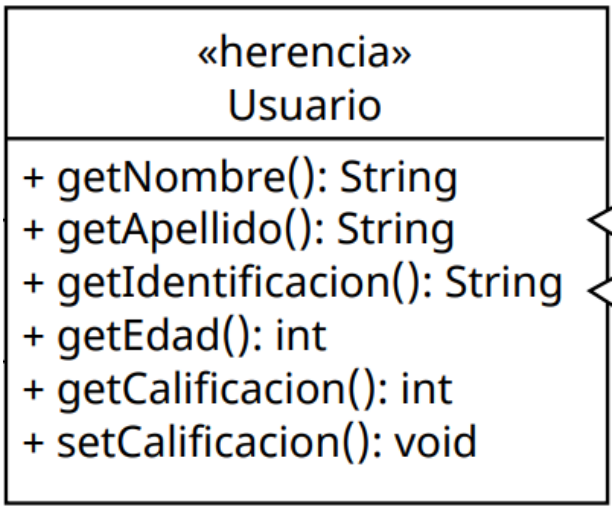


Controlador General: Este elemento fue creado con el objetivo de que sirva de intermediario con otros elementos del modelo. Tiene como atributo el “booleano” conductor, el cual indica si se está tratando con un conductor, dependiendo de esto se ejecutan o no las diferentes funcionalidades como iniciar viaje, crear cuenta de pasajero y obtener los viajes.



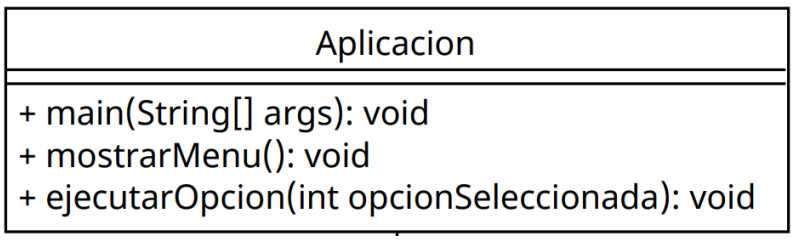
Controlador Conductor: Este elemento cumple una funcionalidad análoga a la del elemento anterior. La diferencia es los elementos en los que están siendo intermediarios.

En este caso se tiene como atributos el "booleano" conduciendo y el automóvil que está en uso. Este elemento permite buscar un viaje dado su conductor y establecer si el automóvil está en uso.



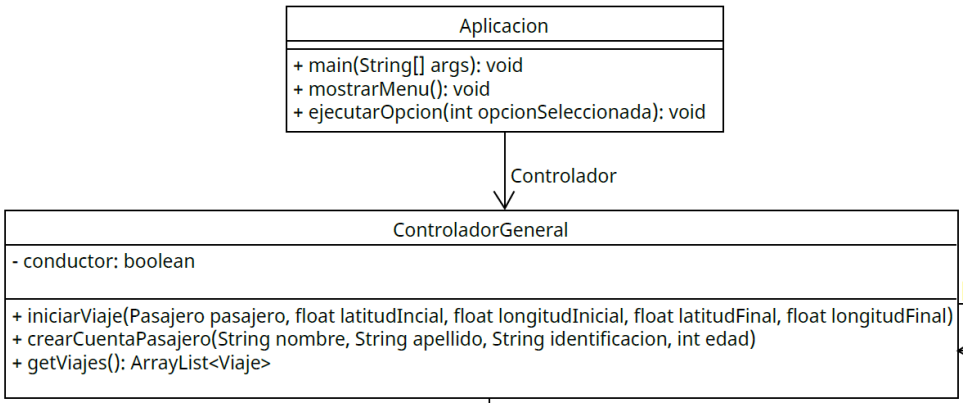
<<**Herencia**>> **Usuario:** Este elemento representa la relación de **herencia** en la cual los elementos "pasajero" y "conductor" heredan de "Usuario" debido a sus similitudes.

Permite obtener los datos personales de los usuarios de la aplicación (tanto conductores como pasajeros), así como sus calificaciones.

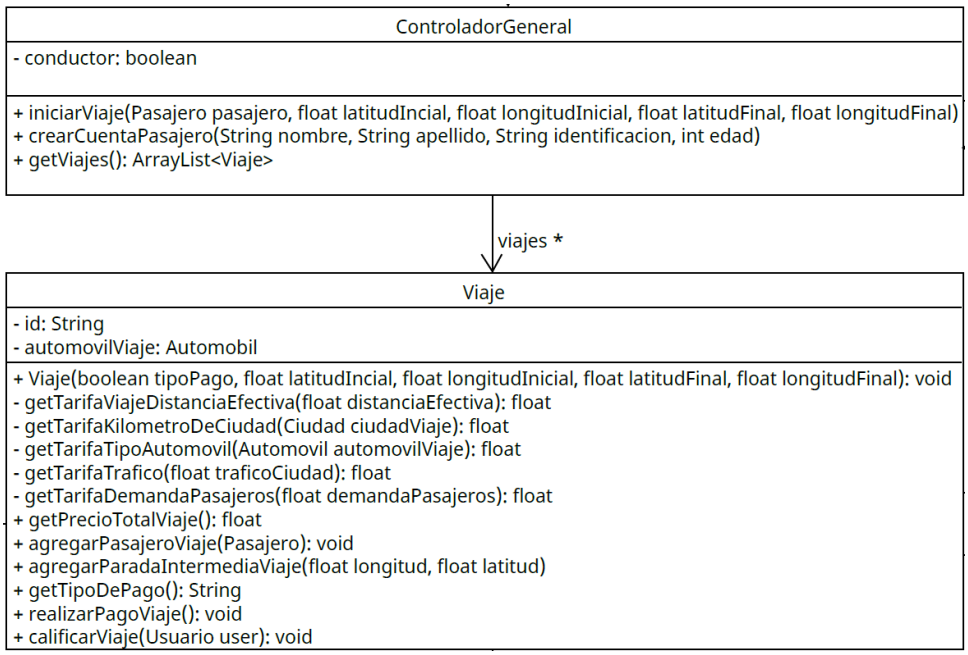


Aplicación: Este elemento cumple el rol de "consola", ya que se encarga de ejecutar las funcionalidades de las diferentes opciones que ofrece el modelo.

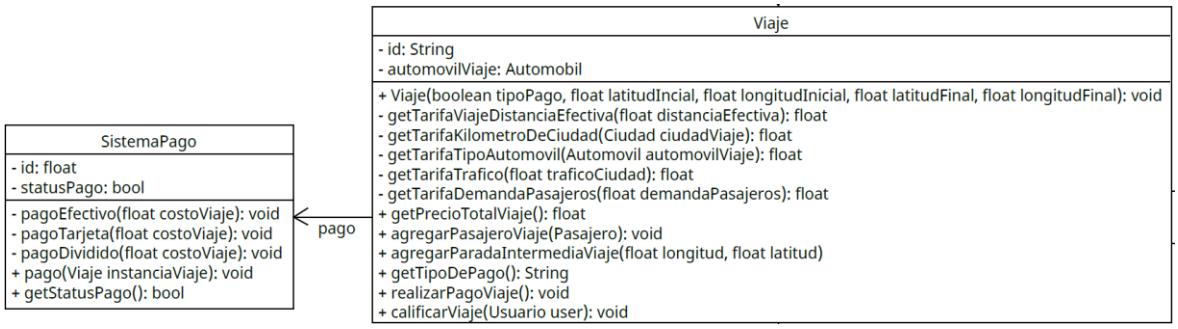
Relaciones entre elementos UML:



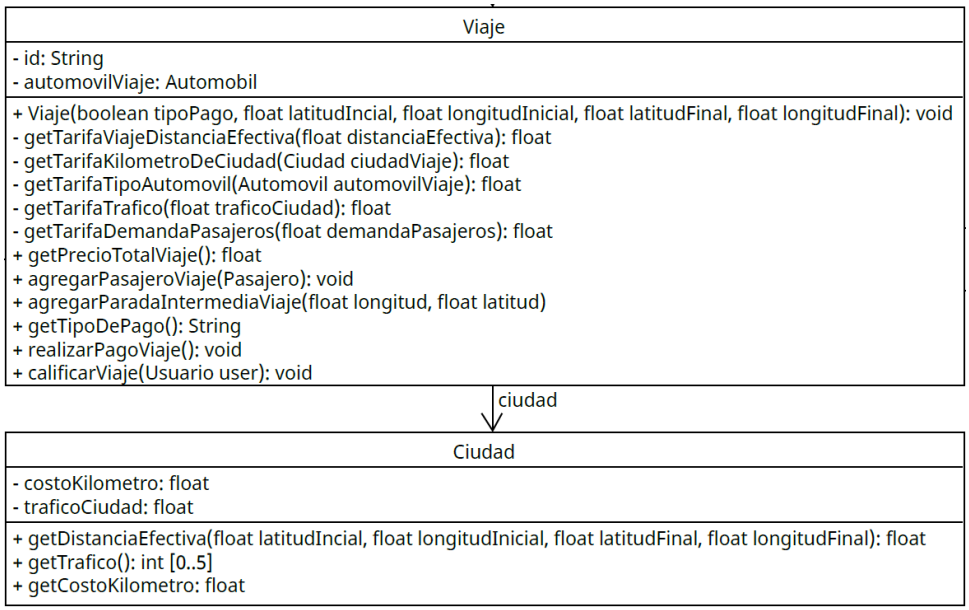
Esta es una relación de asociación ya que la aplicación llama al controlador para poder ejecutar el programa, pero cada uno es independiente.



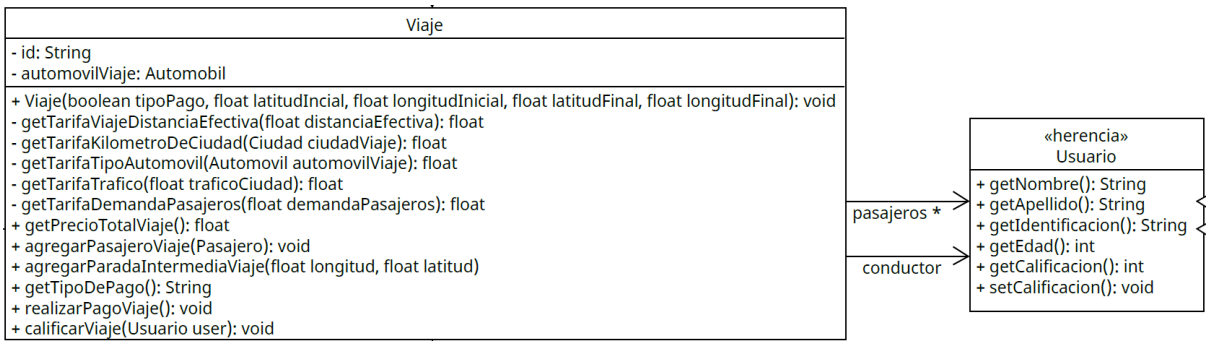
Esta es una relación de asociación ya que el controlador se encarga de crear todas las instancias y la clase viaje de ejecutar todas las funciones relacionadas con el viaje.



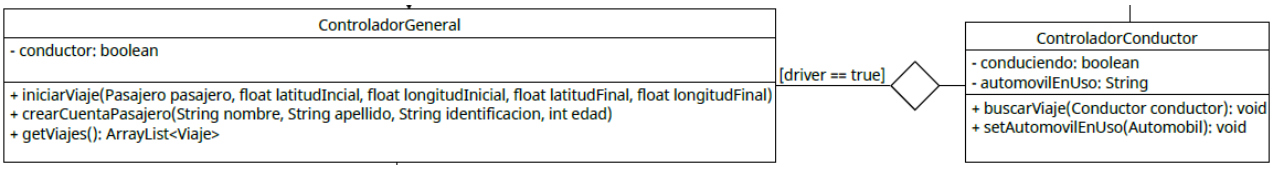
Esta es una relación de asociación ya que el viaje tiene como instancia calcular los costos y todo lo relacionado con el pago de los viajes, de lo cual se encarga esta clase se encarga de estos procesos de forma independiente.



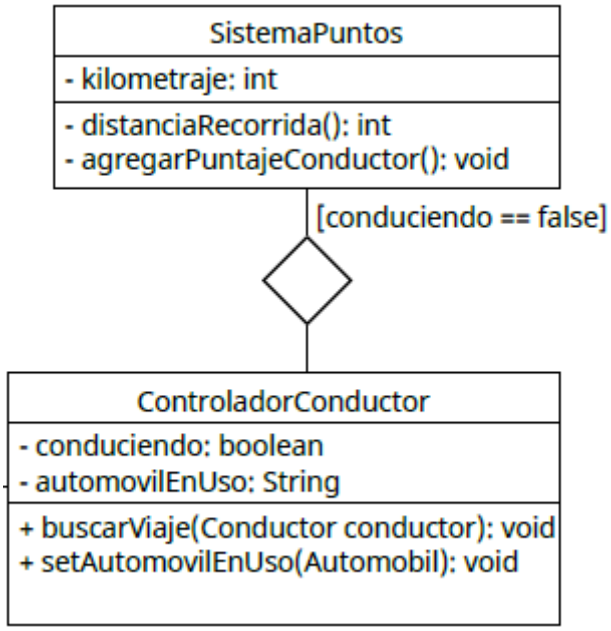
Esta es una relación de asociación ya que el viaje tiene influencia por la ciudad por lo cual la clase Ciudad se encarga de tener todos los factores que afectan el costo del viaje de forma independiente.



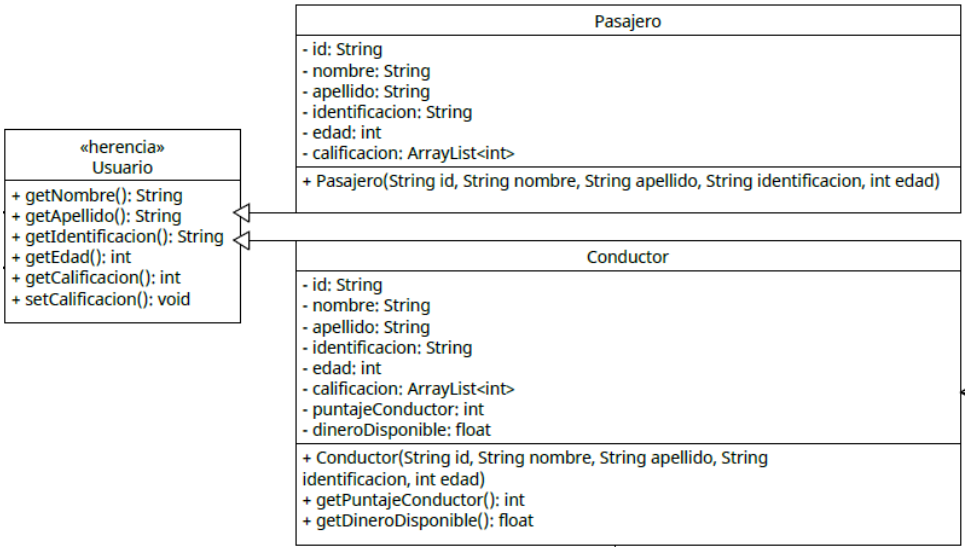
Esta es una relación de asociación ya que el viaje tiene a los usuarios que son los que usan la aplicación de tal manera que la información de estos que serían los pasajeros y conductores es independiente al viaje como tal, pero es importante tener almacenados estos datos por lo cual las clases se relacionan de esta manera.



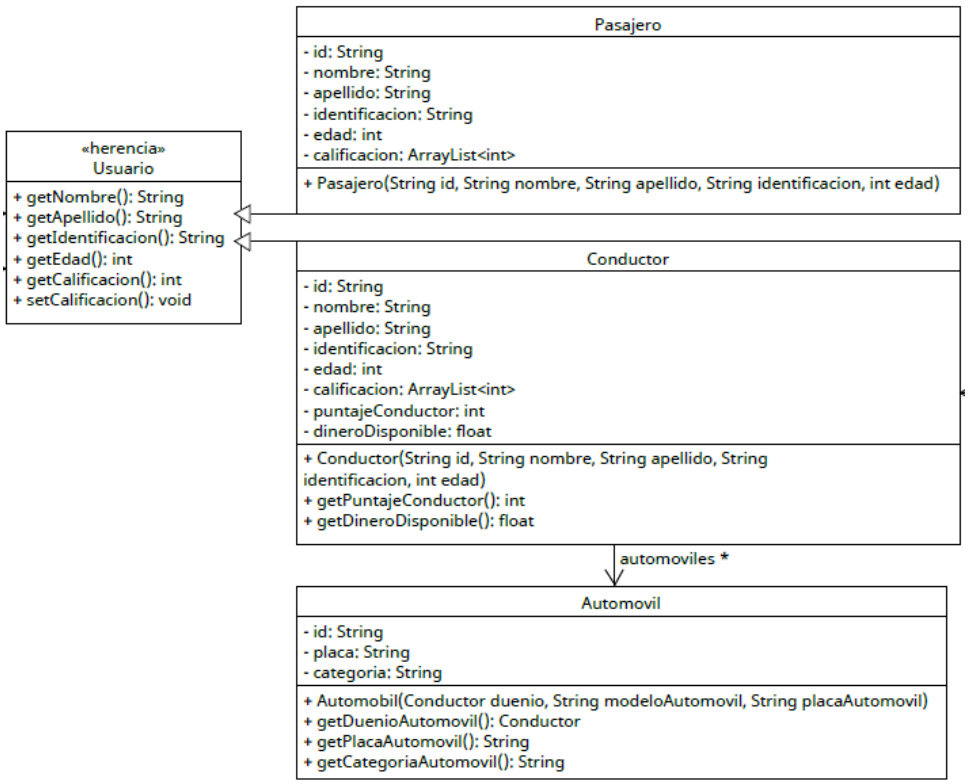
Esta es una relación de agregación ya que su relación es débil y cada una es independiente, en este caso si se cumple la condición de que el conductor está trabajando o activo (==true) es que se llama a la clase del controladorConductor.



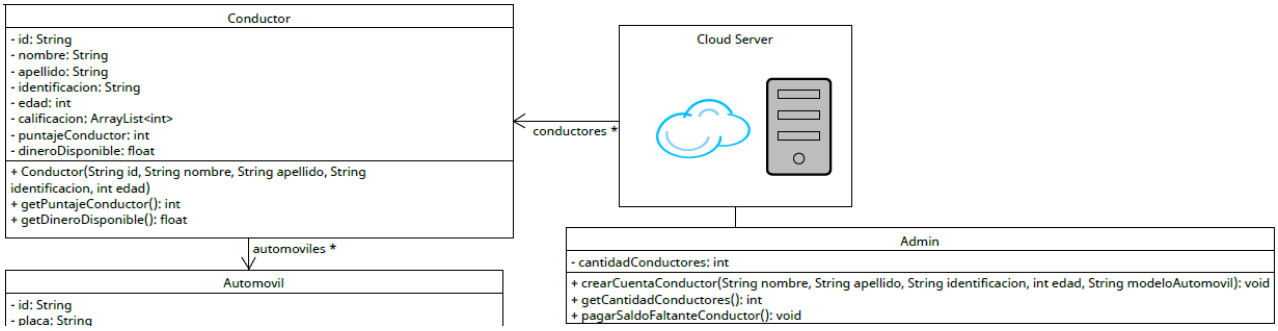
Esta es una relación de agregación ya que si se cumple la condición de que el conductor no está en un viaje (conduciendo == False) se llama a la clase Sistema de puntos la cual se encarga de calcular de forma independiente los puntos que gana el conductor cuando no está en un viaje.



Es una relación de herencia ya que las clases de pasajero y conductor heredan de la clase usuario al igual que todos sus métodos, mientras que las clases conductor y pasajero tienen sus propios atributos y datos, pero los métodos los hereda de la clase Usuario.



Es una relación de asociación ya que el conductor se relaciona con el automóvil que tiene, y este a su vez tiene ciertas características que van a influir en el precio del viaje de tal manera que son dos clases independientes pero el automóvil genera sus propios métodos para relacionar un automóvil de ciertas características con un conductor.



Es una relación de asociación ya que el cloud server sirve como un dashboard es un sistema web que soporta la aplicación desde el cual solo administradores pueden entrar, en donde se puede realizar la creación de cuentas de conductores, realizar pagos de saldo faltante de parte de la compañía hacia el conductor y saber la cantidad de conductores registrados.