**Diseño y Programación Orientada a Objetos**

**Taller 2**

**Análisis**

**REQUERIMIENTOS FUNCIONALES**

**Conductor**

Como conductor, quiero ver los pedidos de viaje en tiempo real para poder aceptarlos o rechazarlos.

Como conductor, quiero recibir la paga por mi trabajo.

Como conductor, quiero poder calificar a mi pasajero al finalizar un viaje.

**Pasajero**

Como nuevo usuario, quiero poder registrarme en la aplicación para poder hacer uso de esta.

Como pasajero, quiero poder pedir un viaje desde un punto de partida a varios destinos.

Como pasajero, quiero poder escoger mi método de pago para un viaje.

Como pasajero, quiero poder calificar a mi conductor al finalizar un viaje.

Como pasajero usuario, quiero poder escribir quejas o comentarios frente al viaje o la aplicación.

**Administrador**

Como administrador, quiero poder registrar nuevos conductores para que estos comiencen a trabajar con la aplicación.

Como administrador, quiero poder enviar el pago a los conductores.

Como administrador, quiero poder recibir el dinero que los conductores deben a final de mes.

**MODELO**

**Clases**

**Conductor**  
Requerimientos funcionales:

-Aceptar o rechazar viaje: Al conductor le deben aparecer en pantalla los pedidos de viaje en tiempo real con su tarifa, y poder escoger sí aceptarlo o rechazarlo. En caso de aceptarlo, se inicia un nuevo viaje.

-Recibir dinero: Al finalizar el ¿mes? ¿O el viaje? El conductor recibe el dinero correspondiente a su trabajo

- Calificar usuario: Al finalizar un viaje, le aparece al conductor la opción para calificar a su pasajero de 1 a 5.

Atributos:

-ubicación (Tuple<double>)  
-nombre (String)  
-calificación promedio (double)  
-número de viajes (int)  
-vehículo(s) (Vehículo)  
-cuenta bancaria (CuentaBancaria)

-documento (String)

-viajeActual (Viaje)

Métodos:

actualizarDatosBancarios()

añadirVehiculo()

calificarUsuario()

**Pasajero:**

Requerimientos funcionales:

- Pedir/cancelar un viaje. El pasajero debe poder hacer un pedido de un viaje, ingresando su ubicación de partida y los destinos que quiera. Una vez ingresada esta información, se calcula la ruta y se hace la solicitud a los conductores disponibles. En cuando un conductor acepte, se muestra la información del conductor y se inicia un nuevo viaje.

- Ingresar ubicación de salida y llegada: El usuario ingresa la información sobre el punto de salida y el destino.

-Ingresar múltiples paradas: Si el usurario lo desea puede añadir múltiples paradas al viaje.

-Calificar al conductor: Al finalizar un viaje, el pasajero debe poder calificar al conductor (1 a 5 estrellas)

-Registrarse en la app: Un nuevo usuario debe poder crear un nuevo perfil en la aplicación.

-Elegir método pago: Al solicitar un viaje, el usuario debe poder escoger su método de pago (efectivo o tarjeta).

-Escribir quejas: Al finalizar el viaje, el usuario puede escribir alguna queja o comentario que tenga al respecto del servicio o la app

Atributos:

- correo (String)

- celular (String)

- nombre (String)

- tarjeta (String)

Métodos:

calificarConductor()

pedirViaje()

cancelarViaje()

escribirQuejas()

añadirParadas()

registrarseApp()

elegirPago()

**Viaje**

Requerimientos funcionales:

Ruta: Debe calcular la ruta óptima según el tráfico actual, irregularidades en las rutas, etc.

Atributos:  
-ubicacionInicial (Tuple<double>)

-ubicacionFinal (Tuple<double>)

-distancia(double)

-tiempo(double)

-precio (double)

metodoPago (

-destinos (List<Tuple<double>>)

**Vehículo**

Requerimientos funcionales:

Atributos:

-placa (String)

-color (String)

-modelo (String)

-categoría (String)

-porcentajeGanancia (double)

**Administrador**

Requerimientos funcionales:

-Poder registrar conductores: El administrador debe tener la capacidad de poder registrar a un nuevo conductor en la aplicación. Esto agregara un usuario a la base de datos de la empresa.

-Hacer el pago a conductores: Al finalizar el mes basados en los viajes, los pagos y los descuentos. Se debe realizar el pago a los conductores según corresponde

-Descontar dinero a conductores: A final de mes se revisa el balance entre el pago total y pago en efectivo, luego se acomoda la cuenta del conductor quitándole el efectivo para que todo quede en orden

Atributos:

-conductores (List<Conductor>)

-cuentaBancaria (CuentaBancaria)

Métodos:

RegistrarConductor ()

HacerPagoConductor()

DescontarDinero()

**CuentaBancaria**

Atributos:

- nombre del titular (String)

-número de cuenta (String)

-número de documento (String)