Taller 2 DPOO

Documento de análisis

Wyo Hann Chu Méndez - 202015066 - w.chu

David Burgos Mendez - 201818326 - d.burgos

Juan Daniel Sepúlveda - 202113067 - j.sepulvedao

Casos de uso:

Conductor:

- Registrarme datos en la aplicación:

Para que el conductor pueda trabajar en la aplicación es necesario que se registre con toda su información. En este caso se le pide al conductor sus datos, estos se guardan a la espera de que el usuario administrativo los analice y cree el objeto Conductor.

- Aceptar viaje y visualizar la información

Cuando el conductor está listo para realizar servicios este tiene que aceptar un viaje y ver sus detalles. Cuando el usuario crea un viaje se notifica al conductor, en caso de que lo acepte se le añadirá el conductor al Objeto Trip, el conductor puede ver los diferentes datos del viaje como el costo o las paradas para poder realizar el servicio.

- Recibir un pago

Al momento de terminar el servicio es necesario que se le realice el pago al conductor. Al cerrar un Trip y terminarlo es necesario que se calcule el ingreso al conductor y a la empresa Ub*r, para esto se revisa el coso del objeto Trip y con el atributo: tipo_carro del conductor y se realiza el pago al conductor.

- Acumular puntos

Cuando el conductor no está prestando un servicio y se está moviendo este puede conseguir puntos los cuales se traducen en bonificaciones económicas. Si en el objeto conductor no está en un Trip y la variable moviéndose esta en True (o 1) se le van a ir añadiendo puntos al parámetro puntos.

Pasajero:

- Registrarse en la aplicación

Para que el pasajero pueda pedir viajes es necesario que se registre en la aplicación, como no necesita la revisión de un administrativo el usuario solo tiene que enviar sus datos y se creara el objeto Pasajero con toda su información.

- Crear un viaje

Después de crear el objeto Pasajero este puede crear viajes, estos viajes necesitarán el punto de inicio y las paradas, pero el resto de los atributos (como distancia o costo) se calcularán al momento de crear el objeto.

Administrativo:

- Agregan conductor

Cuando un conductor se registra en la aplicación el administrativo puede revisar estos datos para ver si cumple con todas las especificaciones del trabajo, cuando se confirma que el conductor está en la capacidad de trabajar en Ub*r el administrativo crea un objeto Conductor con base en la información dada.

- Revisar Viajes / Pagos

Después de cada viaje y cada transacción se añadirá un elemento a la lista de pagos y viajes de la clase Registro, el usuario administrativo puede entrar ver estos datos pidiéndole las listas a Registro.

Descripción de clases:

- La clase de aplicación se encarga de tener una lista de usuarios que pueden ser administrativos, conductores o pasajeros. Además, esta se encarga de la parte que interactúa con el usuario.

- La clase usuario contiene la información básica de conductores, administrativos y usuarios.

- La clase de Pasajero es una hija de la clase Usuario. Esta se encarga de crear los viajes e interactuar con el pago de los mismos, tiene una relación de “viaje” con la clase Trip que puede tomar un valor entre 0 y 1 para simbolizar un viaje en curso. Además, tiene una relación con la clase Pago llamada “pagos” que se encarga de almacenar los pasajeros que van a pagar el viaje.

- La clase Conductor es una hija de la clase Usuario. Esta se encarga de conectar a un conductor con un viaje y de guardar su carro, el tipo del mismo, la puntuación del conductor, los puntos o bonos que puede adquirir, un bool para saber si se está moviendo (para las bonificaciones) y una distribución del pago basado en los porcentajes que el conductor le debe dar a Ub*r al final del mes. Además, está sujeto a recibir un pago desde la clase Trip una vez se haya terminado el viaje, esta se guardará en la variable p_distribucion que se va sumando conforme más viajes se van haciendo.

- La clase Admin es una hija de la clase Usuario. Esta se encarga de manejar los datos guardados en la clase registro, misma que contiene la información sobre los conductores, viajes y pagos hechos a Ub*r. Esa tiene una relación con registro, ya que, los admins pueden ver la información en cualquier momento.

- La clase Trip se encarga de procesar toda la información de un viaje que el pasajero le brindará. Acá se guardará el inicio del viaje y una lista de strings que son las posibles paradas que el usuario puede tener. La información de trip es pasada al conductor que está en espera de recibir un viaje. El pago se calcula en trip (solo el costo) y se pasará esta información a la clase Pago donde se hará el resto de la operación. Además, se tendrá un id del viaje que será guardado para que pueda ser visto por un admin luego.

- La clase Pago se encarga de calcular la transacción del pago una vez se sabe la cantidad que el Pasajero debe pagar y cuántos pagarán. Esta tiene una relación con Conductor, ya que, después de calcular el pago se le da este monto al conductor y otras relaciones con las clases de Efectivo y Crédito que se encargan de calcular el monto basadas en la forma de pago que el Pasajero haya elegido. Además, el pago cuenta con un id único para que su almacenamiento en registro sea más fácil.

- La clase Registro se encarga de guardar toda la información de los conductores, de los viajes realizados y de los pagos realizados. Acá los administradores podrán acceder a la información global de todas las transacciones realizadas en la aplicación.

- La clase Crédito se encarga de hacer la transacción del pago en caso de que el Pasajero haya puesto el tipo de pago como “crédito”. Además, acá se guardará la información del intermediario en caso de que el administrativo o el usuario lo necesite. También se

harán los cálculos en caso de que el pago se haga con más de una tarjeta de crédito si hay varios pasajeros que pagan el mismo viaje.

- La clase de efectivo se encarga de verificar que el pago se haya hecho en efectivo.