Empresa

Documento de Especificación de Requerimientos de Software (SRS)

1. Introducción

1.1 Propósito

El propósito de este documento es describir los requerimientos funcionales y no funcionales de la aplicación denominada empresa, destinada a la gestión de un sistema de gestión de viajes.

1.2 Alcance

La aplicación pretende gestionar una flota de vehículos, los conductores, los clientes y los pedidos de transporte. El sistema permitirá a los usuarios realizar operaciones relacionadas con la administración de viajes, cálculo de tarifas y manejo de información sobre vehículos y conductores. La aplicación deberá contar con un mecanismo de persistencia de datos.

1.3 Definiciones, acrónimos y abreviaturas

- **SRS**: Software Requirements Specification.
- **Cliente**: Persona que solicita un servicio de viaje.
- **Chofer**: Persona que opera un vehículo para proporcionar el servicio de viaje.
- **Vehículo**: Medio de transporte utilizado para realizar los viajes.
- **Pedido**: Solicitud de un cliente para realizar un viaje.
- **Viaje**: Servicio de transporte realizado desde un punto de origen hasta un destino.
- **Usuario**: Cliente registrado o administrador.
- **Singleton:** Patrón de diseño que restringe la creación de instancias de una clase a un solo objeto.
- **Assert:** Declaración usada en el código para comprobar suposiciones hechas por el programador.
- **DTO**: Data Transfer Object. Un objeto que lleva datos entre procesos para reducir el número de llamadas de método.
- **Serialización**: Proceso de convertir un objeto en una secuencia de bytes para su almacenamiento o transmisión.
- **Deserialización**: Proceso de reconstruir un objeto a partir de una secuencia de bytes.
- **GUI**: Graphical User Interface (Interfaz Gráfica de Usuario).
- **Excepción**: En programación, un evento que ocurre durante la ejecución de un programa y que interrumpe el flujo normal del mismo.
- **Controlador**: Clase que maneja la lógica del sistema y la interacción entre la interfaz de usuario y la lógica de negocio.
- **Constantes**: Valores fijos utilizados a lo largo del código para evitar errores y mejorar la legibilidad.
- **Mensajes**: Textos predefinidos que se muestran a los usuarios o se utilizan en la gestión de excepciones.

2. Descripción general

2.1 Perspectiva del sistema

La aplicación será del tipo standalone, se prevé una implementación en java en el modo vista controlador (MVC), esta elección se justifica en los requerimientos NO FUNCIONALES. Esta situación implica la existencia de un código que dispara la aplicación que se denominara "Main", el cuál consistirá en una única instancia del módulo "Controlador".

La aplicación cuenta con un módulo Controlador que se ocupa de administrar la relación entre el módulo módelo de negocio, el módulo de vista y el módulo de persistencia. Dichos módulos se describirán a continuación en los requerimientos funcionales.

2.2 Funciones del sistema

Las principales funciones incluyen:

- **Gestión de choferes**: Añadir, listar y verificar disponibilidad de choferes.
- **Gestión de clientes**: Registro de nuevos clientes y autenticación.
- **Gestión de vehículos**: Añadir, listar y verificar disponibilidad de vehículos.
- **Gestión de pedidos**: Crear y validar pedidos para viajes.
- **Gestión de viajes**: Crear, finalizar y calificar viajes.
- **Autenticación de usuarios**: Permitir a los usuarios (clientes y administradores) iniciar y cerrar sesión.
- **Gestión de Usuarios**: Manejo de autenticación de usuarios, registro de nuevos clientes y cierre de sesión.
- **Gestión de Conductores**: Adición de nuevos conductores al sistema.
- **Persistencia de Datos**: Lectura y escritura de datos de la empresa en un archivo binario para preservar el estado del sistema.
- **Cálculo de puntajes** para pedidos en función del tipo de vehículo.
- **Cálculo del salario** de los conductores, tanto temporales como permanentes.
- **Cálculo del costo de los viaje**s basados en diversos factores como la zona, la cantidad de pasajeros, el uso de baúl, y si se transportan mascotas.
- **Persistencia de Datos** : Capacidad para almacenar y recuperar datos en formato binario y XML de clientes, conductores, vehículos, clientes y viajes.

2.3 Usuarios del sistema

- **Clientes**: Usuarios que realizan pedidos de transporte.
- **Administrador:** Usuario que tiene acceso a todas las funcionalidades del sistema. En particular gestionan los recursos de la empresa, como conductores y vehículos.
- **Conductores o Choferes**: Usuarios encargados de ejecutar los viajes.

2.4 Restricciones

- El sistema debe garantizar que no haya duplicados de choferes, vehículos o clientes.
- La gestión de viajes debe realizarse asegurando la disponibilidad tanto de choferes como de vehículos.
- Solo debe existir una instancia de la clase `Empresa`.
- El sistema debe garantizar la consistencia de los datos durante las operaciones, especialmente en la creación y finalización de viajes.
- Debe manejar adecuadamente las excepciones y mostrar mensajes de error claros a los usuarios.
- El sistema debe manejar un máximo de 10 pasajeros por pedido.
- El sueldo de los conductores no puede ser negativo.

- Los puntajes de los pedidos deben calcularse correctamente en función de las características del vehículo y las preferencias del cliente.
- El sistema debe manejar un gran volumen de datos, por lo que debe ser eficiente en el uso de recursos.
- Debe garantizar la integridad y consistencia de los datos durante el proceso de serialización y deserialización.
- La persistencia debe ser compatible con futuras versiones del sistema sin pérdida de datos.

3. Requerimientos Funcionales

Nota: Cuando en este documenrto se hace referencia a cálculos ver los mismo en la documentación anexa.

3.1 Constantes, Mensajes y Excepciones

En la presente sección se enumeran las constantes, mensajes y excepciones que se mencionan en los subsiguientes requerimientos funcionales.

3.1.1 Constantes

A continuación se listan las constantes que se mencionan en los requerimientos funcionales de la aplicación, son todas constantes de cadena y están en la forma nombre y valor:

- LOGIN = "LOGIN";
- REGISTRAR = "REGISTRAR";
- NOMBRE_USUARIO = "NOMBRE_USUARIO";
- PASSWORD = "PASSWORD";
- REG_BUTTON_REGISTRAR = "REG_BUTTON_REGISTRAR";
- REG_BUTTON_CANCELAR = "REG_BUTTON_CANCELAR";
- REG_REAL_NAME = "REG_REAL_NAME";
- REG_USSER_NAME = "REG_USSER_NAME";
- REG_PASSWORD = "REG_PASSWORD";
- REG_CONFIRM_PASSWORD = "REG_CONFIRM_PASSWORD";
- CH_CANT_HIJOS = "CH_CANT_HIJOS";
- CH_ANIO = "CH_FECHA";
- ZONA_STANDARD = "ZONA_STANDARD";
- ZONA_PELIGROSA = "ZONA_PELIGROSA";
- ZONA_SIN_ASFALTAR = "ZONA_SIN_ASFALTAR";
- CHECK_BAUL = "CHECK_BAUL";
- CHECK_MASCOTA = "CHECK_MASCOTA";
- CANT_KM = "CANT_KM";
- CANT_PAX = "CANT_PAX";
- NUEVO_PEDIDO = "NUEVO_PEDIDO";
- CALIFICAR_PAGAR = "CALIFICAR_PAGAR";
- CALIFICACION_DE_VIAJE = "CALIFICACION_DE_VIAJE";
- ADMINISTRADOR = "ADMINISTRADOR";
- NO_LOGEADO = "NO_LOGEADO";
- CERRAR_SESION_CLIENTE = "CERRAR_SESION_CLIENTE";
- LISTA_CANDIDATOS = "LISTA_CANDIDATOS";
- NUEVO_CHOFER = "NUEVO_CHOFER";
- NUEVO_VEHICULO = "NUEVO_VEHICULO";
- NUEVO_VIAJE = "NUEVO_VIAJE";
- PEDIDO_O_VIAJE_ACTUAL = "PEDIDO_O_VIAJE_ACTUAL";

- DETALLE_VIAJE_CLIENTE = "DETALLE_VIAJE_CLIENTE";
- LISTA_VIAJES_CLIENTE = "LISTA_VIAJES_CLIENTE";
- MOTO = "MOTO";
- AUTO = "AUTO";
- COMBI = "COMBI";
- PERMANENTE = "PERMANENTE";
- TEMPORARIO = "TEMPORARIO";
- CERRAR_SESION_ADMIN = "CERRAR_SESION_ADMIN";
- PANEL_CLIENTE = "PANEL_CLIENTE";
- DNI_CHOFER = "DNI_CHOFER";
- NOMBRE_CHOFER = "NOMBRE_CHOFER";
- PATENTE = "PATENTE";
- CANTIDAD_PLAZAS = "CANTIDAD_PLAZAS";
- LISTA_PEDIDOS_PENDIENTES = "LISTA_PEDIDOS_PENDIENTES";
- LISTA_CHOFERES_LIBRES = "LISTA_CHOFERES_LIBRES";
- LISTA_VEHICULOS_DISPONIBLES = "LISTA_VEHICULOS_DISPONIBLES";
- LISTA_VEHICULOS_TOTALES = "LISTA_VEHICULOS_TOTALES";
- LISTA_VIAJES_DE_CHOFER = "LISTA_VIAJES_DE_CHOFER";
- CALIFICACION_CHOFER = "CALIFICACION_CHOFER";
- TOTAL_SUELDOS_A_PAGAR = "TOTAL_SUELDOS_A_PAGAR";
- SUELDO_DE_CHOFER = "SUELDO_DE_CHOFER";
- LISTADO_DE_CLIENTES = "LISTADO_DE_CLIENTES";
- LISTA_VIAJES_HISTORICOS = "LISTA_VIAJES_HISTORICOS";
- VALOR_VIAJE = "VALOR_VIAJE";
- CHECK_VEHICULO_ACEPTA_MASCOTA "CHECK_VEHICULO_ACEPTA_MASCOTA";

Además se prevé un formato para las fechas : dd MM yyyy.

3.1.2 Mensajes

Los mensajes son informaciones al usuario se enumeran indicando la acción que dispara el mensaje y el mensaje:

- USUARIO_REPETIDO("Usuario repetido")
- PASS_ERRONEO("Password incorrecto")
- USUARIO_DESCONOCIDO("Usuario inexistente")
- PARAMETROS_NULOS("Algun parametro requerido es null")
- PASS_NO_COINCIDE("La contrasena y su confirmacion no coinciden")
- SIN_VEHICULO_PARA_PEDIDO("Ningun vehiculo puede satisfacer el pedido")
- PEDIDO_INEXISTENTE("El Pedido no figura en la lista")
- CHOFER_NO_DISPONIBLE("El chofer no esta disponoble")
- VEHICULO_NO_DISPONIBLE("El vehiculo no esta disponible")
- VEHICULO_NO_VALIDO("El vehiculo no puede atender del pedido")
- VIAJE_INEXISTENTE("El viaje no esta regitrado")
- VIAJE_YA_FINALIZADO("El viaje ya figura como finalizado")
- CLIENTE_CON_VIAJE_PENDIENTE("Cliente con viaje pendiente")
- VEHICULO_YA_REGISTRADO("Vehiculo ya Registrado CHOFER_YA_REGISTRADO("Chofer Ya Registrado")
- CHOFER_SIN_VIAJES("El chofer no tiene ningun viaje realizado");

3.1.3 Excepciones

Se establecen las siguientes excepciones que son mencionadas en los requerimientos funcionales, las mismas son disparadas cuando por alguna circunstancia sebe informarse a los niveles superiores de la aplicación que ha ocurrido un evento inesperado. El tipo de excepción es el que nos indica el evento inesperado:

- **Excepción de Chofer No Disponible (`ChoferNoDisponibleException`)**:
  - Debe lanzarse cuando se intenta asignar un chofer que no está disponible para un viaje.
  - Debe contener información sobre el chofer que no está disponible.
- **Excepción de Chofer Repetido (`ChoferRepetidoException`)**;
  - Debe lanzarse cuando se intenta registrar un chofer que ya existe en el sistema.
  - Debe proporcionar detalles sobre el DNI del chofer duplicado y la instancia existente del chofer.
- **Excepción de Cliente con Viaje Pendiente (`ClienteConViajePendienteException`)**:
  - Debe lanzarse cuando un cliente intenta hacer un nuevo pedido mientras tiene un viaje pendiente.
  - Debe proporcionar un mensaje claro indicando que el cliente ya tiene un viaje pendiente.
- **Excepción de Contraseña Errónea (`PasswordErroneaException`)**:
  - Debe lanzarse cuando un usuario introduce una contraseña incorrecta al intentar iniciar sesión.
  - Debe contener el nombre de usuario y la contraseña introducida.
- **Excepción de Pedido Inexistente (`PedidoInexistenteException`):**
  - Debe lanzarse cuando se intenta acceder o modificar un pedido que no existe en el sistema.
  - Debe contener una referencia al pedido en cuestión.
- **Excepción de Falta de Vehículo para Pedido (`SinVehiculoParaPedidoException`)**:
  - Debe lanzarse cuando no hay vehículos disponibles para un pedido específico.
  - Debe proporcionar detalles sobre el pedido para el cual no hay vehículos disponibles
- **Excepción de Usuario No Existente (`UsuarioNoExisteException`)**:
  - Debe lanzarse cuando se intenta iniciar sesión con un nombre de usuario que no está registrado en el sistema.
  - Debe proporcionar el nombre de usuario introducido.
- **Excepción de Usuario Repetido (`UsuarioYaExisteException`)**:
  - Debe lanzarse cuando se intenta registrar un nombre de usuario que ya está en uso.
  - Debe contener el nombre de usuario introducido.
- **Excepción de Vehículo No Disponible (`VehiculoNoDisponibleException`)**:
  - Debe lanzarse cuando se intenta asignar un vehículo que no está disponible para un viaje.
  - Debe contener información sobre el vehículo en cuestión.
- **Excepción de Vehículo No Válido (`VehiculoNoValidoException`)**:
  - Debe lanzarse cuando un vehículo no cumple con los requisitos para un pedido específico.

Debe contener referencias tanto al vehículo como al pedido.

- **Excepción de Vehículo Repetido (`VehiculoRepetidoException`)**:
  - Debe lanzarse cuando se intenta registrar un vehículo que ya está en el sistema.
  - Debe proporcionar la patente del vehículo y una referencia a la instancia existente del vehículo.
- **Excepción de Viaje Inexistente (`ViajeInexistenteException`)**:
  - Debe lanzarse cuando se intenta acceder a un viaje que no existe.

    - Debe contener una referencia al viaje en cuestión.
- **Excepción de Viaje Ya Finalizado (`ViajeYaFinalizadoException`)**:
    - Debe lanzarse cuando se intenta finalizar un viaje que ya ha sido finalizado.
    - Debe contener una referencia al viaje en cuestión.

3.2 Enumeración de los requerimientos funcionales

3.2.1 Módulo Controlador

- **Gestión de Sesiones:**
    - **Login**: Permitir que los usuarios inicien sesión utilizando un nombre de usuario y contraseña.
    - **Logout**: Permitir a los usuarios cerrar sesión y guardar el estado actual de la empresa.
- **Registro de Clientes**:
    - Permitir a los nuevos clientes registrarse en el sistema, verificando que el nombre de usuario no esté duplicado y que las contraseñas coincidan.
- **Gestión de Pedidos**:
    - Permitir a los clientes crear nuevos pedidos especificando la cantidad de pasajeros, si llevan mascotas, si necesitan baúl y la zona.
    - Validar la disponibilidad de vehículos para el pedido y gestionar las excepciones si no hay vehículos disponibles.
- **Gestión de Conductores**:
    - Permitir la adición de nuevos conductores, ya sean temporales o permanentes.
    - Validar que el conductor no esté ya registrado en el sistema.
- **Gestión de Vehículos**:
    - Permitir la adición de nuevos vehículos como motos, autos o combis, validando que el vehículo no esté ya registrado en el sistema.
- **Gestión de Viajes**:
    - Permitir la creación de un nuevo viaje asignando un pedido a un conductor y un vehículo disponibles.
    - Gestionar la finalización de viajes, permitiendo a los clientes calificar y pagar el viaje.
- **Persistencia de Datos**:
    - Leer el estado de la empresa desde un archivo binario al iniciar la aplicación.
    - Guardar el estado de la empresa en un archivo binario al realizar operaciones críticas como logout.

3.2.2 Módulo modelo de negocios

Este módulo lanzado por el módulo controlodor ha de interactuar con el modelo de datos, la vista y la persistencia, a través del controlador. Son sus requerimientos:

- **Agregar Chofer**:
    - **Descripción**: Añade un nuevo chofer a la lista de choferes disponibles.
    - **Entrada**: Objeto `Chofer`.
    - **Salida**: Chofer añadido a los mapas y listas correspondientes.
    - **Excepciones**: `ChoferRepetidoException` si el chofer ya existe.

- **Obtener Choferes**:
    - **Descripción**: Devuelve el mapa de todos los choferes registrados.
    - **Entrada**: Ninguna.
    - **Salida**: `HashMap<String, Chofer>`.

- **Agregar Cliente**:
    - **Descripción**: Registra un nuevo cliente.

    - **Entrada**: Usuario, contraseña y nombre real.
    - **Salida**: Cliente añadido al mapa de clientes.
    - **Excepciones**: `UsuarioYaExisteException` si el usuario ya existe.

- **Autenticación de Cliente**:
    - **Descripción**: Permite a un cliente iniciar sesión en el sistema.
    - **Entrada**: Nombre de usuario y contraseña.
    - **Salida**: Objeto `Usuario` si la autenticación es exitosa.
    - **Excepciones**: `UsuarioNoExisteException` o `PasswordErroneaException`.

- **Agregar Vehículo**:
    - **Descripción**: Añade un nuevo vehículo a la lista de vehículos disponibles.
    - **Entrada**: Objeto `Vehiculo`.
    - **Salida**: Vehículo añadido al mapa y lista correspondientes.
    - **Excepciones**: `VehiculoRepetidoException` si el vehículo ya existe.
    -
- **Obtener Vehículos**:
    - **Descripción**: Devuelve el mapa de todos los vehículos registrados.
    - **Entrada**: Ninguna.
    - **Salida**: HashMap<String, Vehiculo>`.

- **Crear Pedido**:
    - **Descripción**: Crea un nuevo pedido para un cliente.
    - **Entrada**: Objeto `Pedido`.
    - **Salida**: Pedido añadido al mapa de pedidos.
    - **Excepciones**: `SinVehiculoParaPedidoException` si no hay vehículos disponibles.
    -
- **Crear Viaje**:
    - **Descripción**: Asigna un chofer y un vehículo a un pedido para crear un viaje.
    - **Entrada**: Objeto `Pedido`, objeto `Chofer`, objeto `Vehiculo`.
    - **Salida**: Viaje creado y chofer/vehículo marcados como ocupados.
    - **Excepciones**: `PedidoInexistenteException`, `ChoferNoDisponibleException`, `VehiculoNoDisponibleException`,`VehiculoNoValidoException`,`ClienteConViajePendienteException`.

- **Finalizar Viaje**:
    - **Descripción**: Finaliza un viaje y lo marca como completado.
    - **Entrada**: Objeto `Viaje` y calificación.
    - **Salida**: Viaje añadido a la lista de viajes terminados y chofer/vehículo marcados como desocupados.
    - **Excepciones**: `ViajeInexistenteException`, `ViajeYaFinalizadoException`.

3.2.3 Módulo módelo de datos

- **Gestión de Usuarios**:
    - El sistema debe permitir la autenticación de usuarios mediante un nombre de usuario y contraseña.
    - El administrador debe ser un Singleton, es decir, solo debe existir una instancia de la clase `Administrador`.

- **Gestión de Vehículos**:
    - El sistema debe soportar diferentes tipos de vehículos: `Auto`, `Combi`, `Moto`.

    - Cada vehículo debe tener un método para calcular el puntaje del pedido (`getPuntajePedido`).
    - El sistema debe asegurarse de que los autos no acepten más de 4 pasajeros, y las combis entre 5 y 10 pasajeros.

- **Gestión de Conductores**:
    - El sistema debe permitir la gestión de conductores permanentes y temporales.
    - Debe calcular el sueldo bruto de un conductor permanente en función de su antigüedad y cantidad de hijos.
    - Debe calcular el sueldo neto de todos los conductores aplicando una deducción del 14%.

- **Gestión de Pedidos**:
    - El sistema debe permitir la creación de pedidos con información sobre el cliente, cantidad de pasajeros, si se transporta mascota o baúl, kilometraje y zona.
    - Debe validar que la cantidad de pasajeros esté entre 1 y 10.
    - Debe permitir la finalización de viajes y calcular una calificación para el chofer.

- **Cálculo del Costo de Viajes**:
    - El sistema debe calcular el costo del viaje basado en la zona, cantidad de pasajeros, uso de baúl y transporte de mascotas.
    - Debe permitir modificar la tarifa base de un viaje.

3.2.3 Módulo persistencia

- **Serialización Binaria (PersistenciaBIN):**
    - El sistema debe permitir la serialización de objetos en un formato binario para su almacenamiento en disco.
    - Debe ser capaz de deserializar estos objetos de vuelta a su forma original.
- **Serialización XML (PersistenciaXML):**
    - El sistema debe permitir la serialización de objetos en formato XML, facilitando su lectura y portabilidad.
    - Debe ser capaz de deserializar objetos desde un archivo XML.
- **Conversión de Objetos Empresa (UtilPersistencia):**
    - Debe existir una funcionalidad para convertir un objeto `Empresa` en un `EmpresaDTO` (Data Transfer Object) para facilitar su almacenamiento.
    - El sistema debe permitir restaurar un objeto `Empresa` desde un `EmpresaDTO`.
- **Gestión de Excepciones:**
    - El sistema debe manejar adecuadamente las excepciones durante los procesos de serialización y deserialización, incluyendo situaciones como `IOException` y `ClassNotFoundException`.
    - Debe proporcionar mensajes de error claros y manejables cuando ocurran problemas durante la persistencia.

3.2.4 Módulo de Vista

El modulo vista administra la ventana que ve el usuario, a elección de éste la ventana hará visible un panel : de Login, Registro, Cliente o Adminisgrador. Al iniciarse la aplicación por defecto se inicia con el panel de Login.

Nota: Una descripción detallada del comportamiento de la ventana acompaña el presente documento en el Anexo Ventana.

Los requerimientos de los diferentes paneles son:

- **Panel de Login**
  - Está pensado para dar acceso a la aplicación luego de proveer un nombre de usuario y una password.
  - Si el usuario existe y la clave es correcta se da acceso (al panel cliente o administrador según corresponda).
  - Si la clave no es correcta se dispara el mensaje ***PASS_ERRONEO.***
  - Si el usuario no existe el mensaje será ***USUARIO_DESCONOCIDO***
  - Debe existir un opción que nos permita acceder al panel de registro para ingresar nuevos clientes.
- **Registro**
  - Debe existie una opción para cancelar sin hacer nada y vuelve al panel de login.
  - Debe existir una opción para registrar un cliente y al optar por ella pueden ocurrir dos escenarios:
    - Si el nombre de usuario está disponible, y la contraseña y su confirmación coinciden, se registra el Cliente y volvemos al panel de Login.
    - Si el usuario ya existe, ya sea otro cliente o el administrador, se lanza una ventana emergente con el mensaje correspondiente a Mensajes.***USUARIO_REPETIDO***
    - Si la contraseña y su confirmación no coinciden, se lanza una ventana emergente con el mensaje correspondiente a Mensajes.***PASS_NO_COINCIDE***
- **Cliente**
  - Debe identificar al cliente elegido.
  - Debe permitir el cierre de la sección y volver al panel de Login.
  - Dentro del panel del cliente se deben tratar las siguientes situaciones:
  - No hay pedidos entonces se tiene que poder generar uno nuevo.
    - Un pedido nuevo tiene que tener una cantidad de pasajeros >0 y <10 y la distancia deberá ser >0.
      - Si la empresa tiene registrado al menos un vehículo que pueda satisfacer el viaje (por cantidad de pasajeros, uso o no de baúl y aceptación o no de mascota) se genera un nuevo pedido.
      - Si la empresa no posee un Vehículo que pueda satisfacer el viaje (por cantidad de pasajeros, uso o no de baúl y aceptación o no de mascota) se emite el mensaje correspondiente: ***SIN_VEHICULO_PARA_PEDID.***
  - Si hay un pedido sin vehículo y chofer asignado, el cliente solo puede ver.
  - Si hay un pedido asignado a un viaje, el cliente deberá calificarlo.
    - Para pagar el vieje la calificación del mismo deberá ser >0 y <5.
    - Cuando elejimos pagar el viaje se da por terminado y se almacena en el histórico.
- **Administrador**

  El panel de administrador nos permite realizar 4 acciones bien diferenciadas:
  - Visualización de listas e información de choferes.
  - Registro de un nuevo chofer
  - Registro de un nuevo Vehículo
  - Gestión de Pedidos (generado por parte de un Cliente)
  - **Visualización de Información:**
    - mostrará la lista de choferes registrados, si se selecciona uno entonces:
      - Se visualizarán los viajes históricos de dicho Chofer (objetos de tipo Viaje) , la calificación del chofer seleccionado y su sueldo. Si no hay un chofer elegido entonces los campos permanecerán vacios.
    - Se visualizarán los clentes registrados.
    - Se visualizarán los vehículos registrados.
    - Se visualizarán el histórico de viajes de la empresa.
    - Se visualizará el toal de los sueldos a pagar.

- **Registro de un nuevo Chofer**
  - opción que permita deerminar la opción de chofer temporario o permanente.
  - Se debe asegurar el ingres de un DNI y un Nombre para aceptar un nuevo chofer.
  - se debe completar la cantidad de hijos >=0 y el aniol >1900 y <3000
  - al intentar el registro de un chofer si el DNI ya existe se debe informar mediante el mensaje: ***CHOFER_YA_REGISTRADO.*** En caso conterio se da de alta el chofer.
- **Registro de un nuevo Vehículo**
  - Si se desea indicar el tipo de vehiculo (Moto, Auto, Combi) y ingresar la Patente de vehículo. En el caso de un auto o una combi de be indicarse la cantidad de plazas (1-4 en un auto y 5-10 en una combi).
  - Al registrar u vehículo si la patente no pertenece a un vehículo registrado, se da de alta el vehículo. En caso contrario se emite el mensaje: ***VEHICULO_YA_REGISTRADO***
- **Gestión de Pedidos**
  - Visualiza la lista de pedidos pendientes.
  - Visualiza la lista de choferes libres (no están realizando ningún viaje).
  - Visualiza la lisa de vehículos disponibles la cual permanece vacía hasta que se seleccione un pedido pendiente, en ese momento se llena con los vehículos disponibles que pueden satisfacer el pedido y no estén realizando un viaje (esta lista podría continuar vacía).
  - Si se elije es posible elegir un pedido, un chofer y un vehículo en simultaneo, será posible registras un nuevo viaje. Esto se realizará seleccionando un opción.

4 Requerimientos no funcionales

Lenguaje de Programación: Nuestra empresa desarrolla solo aplicaciones Java, el comitente acepta la restricciones resultantes.
Rendimiento: Debe manejar eficientemente las operaciones con grandes cantidades de datos, utilizando estructuras de datos como `HashMap` y `ArrayList`.
Seguridad: Debe garantizarse que las operaciones de autenticación y manejo de contraseñas sean seguras.
Mantenibilidad: El código debe estar bien estructurado para permitir futuras modificaciones y expansiones.
Escalabilidad: Debe manejar eficientemente el crecimiento en la cantidad de datos almacenados sin deteriorar el rendimiento.
Compatibilidad: El sistema debe ser compatible con diferentes plataformas y entornos de ejecución que soporten Java.

5. Apendices

N/A