**Universidad Tecnológica de La Habana**

**José Antonio Echeverría**

****

**Facultad de Ingeniería Informática**

**Trabajo de Curso de Base de Datos**

**Control de los servicios de una base de transporte**

**Autores - Correo:**

**Javier García Hernández - garciaj1246@gmail.com**

**Richard Oropeza Garcés - richardoro2001@gmail.com**

**Herson Oramas Huguet – 45meko@gmail.com.cu**

**Tutor-Correo:**

**Ing. Juan Alejandro Baster Jiménez – jbaster@ceis.cujae.edu**

**La Habana, Abril 2023**

**Resumen**

El presente informe consiste en desarrollar un sistema de control de los servicios de una base de transporte llamada TRANSBUS, que se encarga de la transportación de turistas de una corporación turística en La Habana. Se requiere obtener distintas salidas, como un listado de carros de la base, una hoja de ruta de los carros que trabajaron en un día, un informe de discrepancias, un listado de alquileres realizados, un informe de los ingresos de la base de transporte en un mes dado y una localización de choferes. Además, se desea conocer los tres países que más actividades de turismo realizan en la corporación, determinar todas las actividades realizadas por un grupo turístico en un período de tiempo y una lista de los choferes que han trabajado con un grupo turístico dado. La solución se llevará a cabo a través de un sistema informático que permita realizar las consultas y obtener las salidas necesarias.

**Palabras clave**: TRANSBUS, Control de servicios, ERECASE, PostgreSQL, IntelliJ IDEA.

**Índice de contenido:**

[**Introducción** i](#_Toc134993222)

[**Requisitos funcionales del proyecto** ii](#_Toc134993223)

[**Descripción de la solución propuesta** iv](#_Toc134993224)

[**Diseño de la base de datos** iv](#_Toc134993225)

[**Entidades y atributos:** iv](#_Toc134993226)

[**Relaciones**: iv](#_Toc134993227)

[**DER:** v](#_Toc134993228)

[**Modelo lógico global de los datos:** v](#_Toc134993229)

[**Modelo Fisico de la Base de Datos:** x](#_Toc134993230)

[**Justificación detallada de las diferencias entre el modelo lógico y el modelo físico.** x](#_Toc134993231)

[**Descripción de la arquitectura del proyecto, describiendo los elementos de cada capa:** xii](#_Toc134993232)

[**Diagrama de clases con la correspondiente explicación, así como las consideraciones hechas al abordar la solución:** xiii](#_Toc134993233)

[**Seguridad: roles y privilegios por roles:** xiii](#_Toc134993234)

[**Pautas seguidas para el diseño de la interfaz, así como prototipo de las interfaces gráficas más importantes:** xiii](#_Toc134993235)

[**Descripción de los mecanismos empleados para el tratamiento de errores:** xiii](#_Toc134993236)

[**Conclusiones** xiv](#_Toc134993237)

[**Recomendaciones** xv](#_Toc134993238)

[**Referencias bibliográficas** xvi](#_Toc134993239)

[**Anexos** xvii](#_Toc134993240)

# **Introducción**

El turismo es considerado una actividad económica que se ha convertido en uno de los actores fundamentales del comercio internacional, y representa al mismo tiempo una de las principales fuentes de ingresos de numerosos países en desarrollo. Este crecimiento va de la mano del aumento de la diversificación y de la competencia entre los destinos. En nuestro país, esta actividad económica representa la segunda fuente de ingresos, generando miles de empleos a lo largo del territorio nacional; además de permitir la sustitución de importaciones en algunas de las aristas de la economía. Es por ello que, a partir de la necesidad de transportación de los turistas, se ha desarrollado en la capital, un sistema de control de servicios para la base de transporte TRANSBUS. El objetivo principal del proyecto es crear un sistema eficiente para el control del tráfico de la base, que permita obtener información detallada sobre los carros, las rutas y los choferes, así como generar informes precisos sobre los ingresos y los alquileres realizados.

El problema a resolver se presenta en la necesidad de contar con un sistema que permitiera una gestión más eficiente y ordenada de los servicios de transporte, tanto los planificados como los solicitados de manera directa. Con la implementación del sistema, se busca optimizar el control de los servicios ofrecidos, disminuyendo el margen de error y permitiendo una toma de decisiones más informada.

Para el desarrollo de este proyecto se utilizaron herramientas tecnológicas como PostgreSQL, un sistema de gestión de bases de datos relacional, y ERECASE, una herramienta de modelado de datos que facilita la creación de diagramas entidad-relación. Además, se empleó el entorno de desarrollo integrado (IDE) IntelliJ IDEA para la implementación del código en lenguaje Java.

A continuación, se detallarán los procesos llevados a cabo para el diseño e implementación del sistema, así como los resultados obtenidos y las posibles mejoras a implementar

# **Requisitos funcionales del proyecto**

A partir de las necesidades identificadas en el proyecto, las funcionalidades que deberá brindar el sistema son:

1. Registro de carros en la base de transporte, incluyendo marca, cantidad de asientos, tipo de combustible y consumo de combustible por kilómetro recorrido.
2. Registro de los datos de cada carro, incluyendo número de flotilla, placa, y nombre de los dos choferes que lo manejan.
3. Registro de la lista de choferes “cubre-franco” de cada marca de ómnibus.
4. Registro de la hoja de ruta de los carros que trabajaron en un día, incluyendo fecha, número del carro, cantidad de kilómetros al salir de la base, placa, hora de salida y detalles de cada servicio.
5. Generación de informe de discrepancias para un mes dado, que permita comparar los kilómetros planificados y recorridos, así como el combustible planificado y consumido.
6. Registro de alquileres realizados, incluyendo número del carro, número de contrato, país, nombre del solicitante, fecha de inicio y fin, kilómetros recorridos e importe cobrado.
7. Generación de informe de ingresos de la base de transporte en un mes dado, que incluya la cantidad total de servicios, la cantidad de alquileres, ingresos por concepto de alquileres, cantidad de otros servicios planificados, ingresos por servicios planificados e ingresos totales.
8. Localización de choferes por distrito de residencia, incluyendo nombre, DNI, dirección y teléfono.
9. Identificación de los 3 países que más actividades de turismo realizan en la corporación, a partir del análisis de la transportación.
10. Búsqueda de todas las actividades que realizó un grupo turístico dado en un período de tiempo determinado.
11. Generación de lista de los choferes que han trabajado con un grupo turístico dado.

Estas funcionalidades permitirán al sistema controlar eficazmente los servicios de la base de transporte, facilitando la gestión de los recursos y la generación de informes precisos sobre las actividades realizadas.

# **Descripción de la solución propuesta**

## **Diseño de la base de datos**

### **Entidades y atributos:**

1. Brand: brand\_name, amo\_seats, fuel\_type, spending
2. Car: fleet\_number, plate
3. Driver: ID, driver\_name, address, phone
4. District: district\_code, district\_name
5. Service: service\_code, service\_name, request\_number, tour\_group\_code, country, pickup\_place, pickup\_time, pax, service\_kms, amount
6. Report: report\_code, amo\_services, amo\_rents, income\_rents, amo\_others, income\_others, income\_total
7. Contract: contract\_code, applicant\_name, start\_date, end\_date, contract\_kms, contract\_amount, contract\_country
8. Month: month\_code
9. Couple: couple\_code
10. Roadmap: date,kms,departure\_time

### **Relaciones**:

1. C-C: Un carro es manejado por una pareja de choferes y cada pareja maneja un solo carro.
2. D1-D2: Muchos choferes forman parejas
3. B-C: Una marca tiene varios carros y un carro pertenece a una marca
4. Con-C: Un carro tiene muchos contratos y un contrato pertenece a un carro
5. Discrepancy: Para cada mes y para cada carro que ha trabajado en el mes se conoce: planned\_kms, tours\_kms, difference\_kms, planned\_fuel, consumed\_fuel, dif\_spending\_fuel
6. D-D: Cada chofer vive en un distrito en particular
7. R-M: Para cada mes se realiza un informe
8. C-S: Una hoja de ruta presenta muchos servicios y los servicios son presentados por muchas hojas de rutas
9. C-R: Un carro tiene varias hojas de rutas y una hoja de ruta pertenece a un solo carro.
10. B-FC Una marca tiene muchos choferes cubre-franco y un chofer cubre-franco maneja autos de una marca en especifico

### **DER:**

Para más de detalle sobre este tema consulte el **anexo A**

### **Modelo lógico global de los datos:**

**1.Atributos**

R(month\_code, contract\_code, date, service\_code, service\_name, request\_number, tour\_group\_code, country, pickup\_place, pickup\_time, pax, service\_kms, amount, applicant\_name, start\_date, end\_date, contract\_kms, contract\_amount, contract\_country, fleet\_number, report\_code, plate, couple\_code, brand\_name, ID, kms, departure\_time, planned\_kms, tours\_kms, difference\_kms, planned\_fuel, consumed\_fuel, dif\_spending\_fuel, amo\_services, amo\_rents, income\_rents, amo\_others, income\_others, income\_total amo\_seats, fuel\_type, spending, driver\_name, address, phone, district\_code, district\_name)

**2.Dependencias Funcionales**

1. brand\_name 🡪 amo\_seats, fuel\_type, spending
2. fleet\_number 🡪 plate, couple\_code, brand\_name
3. ID 🡪 driver\_name, address, phone, district\_code
4. couple\_code 🡪 ID1, ID2
5. fleet\_number, date 🡪 kms, departure\_time
6. service\_code🡪service\_name, request\_number, tour\_group\_code, country, pickup\_place, pickup\_time, pax, service\_kms, amount
7. month\_code, fleet\_number 🡪 planned\_kms, tours\_kms, difference\_kms, planned\_fuel, consumed\_fuel, dif\_spending\_fuel
8. contract\_code 🡪 applicant\_name, start\_date, end\_date, contract\_kms, contract\_amount, contract\_country, fleet\_number
9. report\_code 🡪 amo\_services, amo\_rents, income\_rents, amo\_others, income\_others, income\_total
10. district\_code 🡪 district\_name
11. month\_code 🡪 report\_code

**3. Llaves Candidatas y cierres**

Z={month\_code, contract\_code, date, service\_code}+

Z={month\_code, contract\_code, date, service\_code, service\_name, request\_number, tour\_group\_code, country, pickup\_place, pickup\_time, pax, service\_kms, amount }(6DF)

Z={month\_code, contract\_code, date, service\_code, service\_name, request\_number, tour\_group\_code, country, pickup\_place, pickup\_time, pax, service\_kms, amount, applicant\_name, start\_date, end\_date, contract\_kms, contract\_amount, contract\_country, fleet\_number }(8DF)

Z={month\_code, contract\_code, date, service\_code, service\_name, request\_number, tour\_group\_code, country, pickup\_place, pickup\_time, pax, service\_kms, amount, applicant\_name, start\_date, end\_date, contract\_kms, contract\_amount, contract\_country, fleet\_number, report\_code } (11DF)

Z={month\_code, contract\_code, date, service\_code, service\_name, request\_number, tour\_group\_code, country, pickup\_place, pickup\_time, pax, service\_kms, amount, applicant\_name, start\_date, end\_date, contract\_kms, contract\_amount, contract\_country, fleet\_number, report\_code, plate, couple\_code, brand\_name }(2DF)

Z={month\_code, contract\_code, date, service\_code, service\_name, request\_number, tour\_group\_code, country, pickup\_place, pickup\_time, pax, service\_kms, amount, applicant\_name, start\_date, end\_date, contract\_kms, contract\_amount, contract\_country, fleet\_number, report\_code, plate, couple\_code, brand\_name, ID }(4DF)

Z={month\_code, contract\_code, date, service\_code, service\_name, request\_number, tour\_group\_code, country, pickup\_place, pickup\_time, pax, service\_kms, amount, applicant\_name, start\_date, end\_date, contract\_kms, contract\_amount, contract\_country, fleet\_number, report\_code, plate, couple\_code, brand\_name, ID, kms, departure\_time }(5DF)

Z={month\_code, contract\_code, date, service\_code, service\_name, request\_number, tour\_group\_code, country, pickup\_place, pickup\_time, pax, service\_kms, amount, applicant\_name, start\_date, end\_date, contract\_kms, contract\_amount, contract\_country, fleet\_number, report\_code, plate, couple\_code, brand\_name, ID, kms, departure\_time, planned\_kms, tours\_kms, difference\_kms, planned\_fuel, consumed\_fuel, dif\_spending\_fuel }(7DF)

Z={month\_code, contract\_code, date, service\_code, service\_name, request\_number, tour\_group\_code, country, pickup\_place, pickup\_time, pax, service\_kms, amount, applicant\_name, start\_date, end\_date, contract\_kms, contract\_amount, contract\_country, fleet\_number, report\_code, plate, couple\_code, brand\_name, ID, kms, departure\_time, planned\_kms, tours\_kms, difference\_kms, planned\_fuel, consumed\_fuel, dif\_spending\_fuel, amo\_services, amo\_rents, income\_rents, amo\_others, income\_others, income\_total }(9DF)

Z={month\_code, contract\_code, date, service\_code, service\_name, request\_number, tour\_group\_code, country, pickup\_place, pickup\_time, pax, service\_kms, amount, applicant\_name, start\_date, end\_date, contract\_kms, contract\_amount, contract\_country, fleet\_number, report\_code, plate, couple\_code, brand\_name, ID, kms, departure\_time, planned\_kms, tours\_kms, difference\_kms, planned\_fuel, consumed\_fuel, dif\_spending\_fuel, amo\_services, amo\_rents, income\_rents, amo\_others, income\_others, income\_total amo\_seats, fuel\_type, spending }(1DF)

Z={month\_code, contract\_code, date, service\_code, service\_name, request\_number, tour\_group\_code, country, pickup\_place, pickup\_time, pax, service\_kms, amount, applicant\_name, start\_date, end\_date, contract\_kms, contract\_amount, contract\_country, fleet\_number, report\_code, plate, couple\_code, brand\_name, ID, kms, departure\_time, planned\_kms, tours\_kms, difference\_kms, planned\_fuel, consumed\_fuel, dif\_spending\_fuel, amo\_services, amo\_rents, income\_rents, amo\_others, income\_others, income\_total amo\_seats, fuel\_type, spending, driver\_name, address, phone, district\_code }(3DF)

Z={month\_code, contract\_code, date, service\_code, service\_name, request\_number, tour\_group\_code, country, pickup\_place, pickup\_time, pax, service\_kms, amount, applicant\_name, start\_date, end\_date, contract\_kms, contract\_amount, contract\_country, fleet\_number, report\_code, plate, couple\_code, brand\_name, ID, kms, departure\_time, planned\_kms, tours\_kms, difference\_kms, planned\_fuel, consumed\_fuel, dif\_spending\_fuel, amo\_services, amo\_rents, income\_rents, amo\_others, income\_others, income\_total amo\_seats, fuel\_type, spending, driver\_name, address, phone, district\_code, district\_name }(10DF)

**4. LLave Primaria**

Z={month\_code, contract\_code, date, service\_code}

**5. Primera Forma Normal**

R(month\_code, contract\_code, date, service\_code, service\_name, request\_number, tour\_group\_code, country, pickup\_place, pickup\_time, pax, service\_kms, amount, applicant\_name, start\_date, end\_date, contract\_kms, contract\_amount, contract\_country, fleet\_number, report\_code, plate, couple\_code, brand\_name, ID, kms, departure\_time, planned\_kms, tours\_kms, difference\_kms, planned\_fuel, consumed\_fuel, dif\_spending\_fuel, amo\_services, amo\_rents, income\_rents, amo\_others, income\_others, income\_total amo\_seats, fuel\_type, spending, driver\_name, address, phone, district\_code, district\_name)

**6. Segunda Forma Normal**

**Month** (month\_code, report\_code, amo\_services, amo\_rents, income\_rents, amo\_others, income\_others, income\_total)

**Contract** (contact\_code, applicant\_name, start\_date, end\_date, contract\_kms, contract\_amount, contract\_country, brand\_name, amo\_seats, fuel\_type, spending, fleet\_number, plate, couple\_code, ID, driver\_name, address, phone, district\_code, district\_name)

**Service** (service\_code, service\_name, request\_number, tour\_group\_code, country, pickup\_place, pickup\_time, pax, service\_kms, amount)

**R** (month\_code, contract\_code, date, service\_code, fleet\_number, kms, departure\_time, planned\_kms, tours\_kms, difference\_kms, planned\_fuel, consumed\_fuel, dif\_spending\_fuel)

**7. Tercera Forma Normal**

**Month** (month\_code, report\_code)

**Report** (report\_code, amo\_services, amo\_rents, income\_rents, amo\_others, income\_others, income\_total)

**Contract** (contact\_code, applicant\_name, start\_date, end\_date, contract\_kms, contract\_amount, contract\_country, fleet\_number)

**Car** (fleet\_number, plate, couple\_code, brand\_name)

**Brand** (brand\_name, amo\_seats, fuel\_type, spending)

**Couple** (couple\_code, ID1, ID2)

**Driver** (ID, driver\_name, address, phone, district\_code)

**District** (district\_code, district\_name)

**Service** (service\_code, service\_name, request\_number, tour\_group\_code, country, pickup\_place, pickup\_time, pax, service\_kms, amount)

**Discrepancy** (month\_code, fleet\_number, planned\_kms, tours\_kms, difference\_kms, planned\_fuel, consumed\_fuel, dif\_spending\_fuel)

**Roadmap** (date, fleet\_number, kms, departure\_time)

### **Modelo Físico de la Base de Datos:**

Para más de detalle sobre este tema consulte el **anexo B**

## **Justificación detallada de las diferencias entre el modelo lógico y el modelo físico.**

1. Representación de los datos:

El modelo lógico global de los datos se centra en la estructura y las relaciones conceptuales de la base de datos. Por ejemplo, se define la entidad "Car" con atributos como "fleet\_number", "plate", "couple\_code" y "brand\_name". Esto permite comprender las relaciones entre los vehículos, sus características y la marca a la que pertenecen.

Por otro lado, el modelo físico de la base de datos se refiere a la implementación concreta del modelo lógico utilizando un sistema de gestión de bases de datos específico. Por ejemplo, se define el tipo de datos de los atributos, como "int" para "fleet\_number" y "chracter varying" para "plate". Además, se pueden establecer restricciones como la longitud máxima de caracteres para los atributos "plate" y "brand\_name".

1. Nivel de abstracción: En el modelo lógico global de los datos, se enfatiza la comprensión conceptual de la estructura de la base de datos. Por ejemplo, se definen entidades como "Car" y "Brand" y se establecen relaciones entre ellas, como "brand\_name" en la entidad "Car" que se relaciona con la entidad "Brand".

En contraste, el modelo físico de la base de datos se preocupa por los detalles técnicos y físicos de cómo se almacenan y acceden los datos. Por ejemplo, se pueden crear indices en los atributos "brand\_name" de las tablas "Car" y "Brand" para mejorar la velocidad de búsqueda y consulta de registros.

1. Diseño y optimización: En el modelo lógico global de los datos, se realiza el diseño lógico de la base de datos, considerando las entidades, atributos y relaciones. Por ejemplo, se define la entidad "Service" con atributos como "service\_code", "service\_name" y "amount", y se establece una relación entre "Service" y "Tourist\_Group" mediante el atributo "tour\_group\_code".

En el modelo físico de la base de datos, se aplican técnicas de optimización para mejorar el rendimiento y la eficiencia. Por ejemplo, se puede utilizar la partición de tablas para dividir grandes conjuntos de datos en fragmentos más pequeños y manejables, lo que agiliza las consultas y las operaciones de mantenimiento, un ejemplo de esto sería agrupar los servicios por fechas.

1. Flexibilidad y adaptabilidad: El modelo lógico global de los datos proporciona una representación abstracta de la base de datos, lo que permite realizar cambios en la estructura sin afectar la lógica del sistema. Por ejemplo, se puede agregar una nueva entidad como "Tourist\_Group" con su respectivo atributo "tour\_group\_code" sin alterar las reglas de negocio existentes.

En contraste, el modelo físico de la base de datos está estrechamente vinculado al sistema de gestión de bases de datos utilizado y su estructura física. Por ejemplo, si se decide cambiar el sistema de gestión de bases de datos, es posible que sea necesario ajustar el modelo físico y convertir las sentencias de consulta existentes al lenguaje específico del nuevo sistema.

## **Descripción de la arquitectura del proyecto, describiendo los elementos de cada capa:**

Se sigue una arquitectura que consta de tres capas principales: la capa de base de datos, la capa de servicios y la capa de interfaz de usuario. A continuación, se describen los elementos de cada capa:

Capa de Base de Datos:

Esta capa se encarga del almacenamiento persistente de los datos y proporciona los mecanismos para su manipulación. Los elementos clave en esta capa son las tablas y los esquemas definidos según el modelo físico de la base de datos. Cada tabla representa una entidad del modelo lógico global de los datos y contiene columnas que representan los atributos de esa entidad. También se definen claves primarias, claves foráneas y otras restricciones para garantizar la integridad de los datos. En esta capa, se realizan las operaciones de consulta, inserción, actualización y eliminación de datos.

Capa de Servicios:

La capa de servicios actúa como un intermediario entre la capa de base de datos y la capa de interfaz de usuario. Aquí se implementa la lógica del negocio y se definen los servicios o componentes que exponen las funcionalidades del sistema. Estos servicios se encargan de recibir las solicitudes de la interfaz de usuario, procesarlas utilizando la lógica de negocio correspondiente y comunicarse con la capa de base de datos para obtener o manipular los datos necesarios. Los servicios pueden abarcar diversas funcionalidades como cálculos, validaciones, procesamiento de datos y coordinación de operaciones complejas.

Capa de Interfaz de Usuario:

La capa de interfaz de usuario se ocupa de la presentación y la interacción con el sistema. Los elementos en esta capa incluyen la interfaz gráfica, los formularios, los botones y otros componentes visuales que permiten a los usuarios interactuar con la aplicación. Esta capa se comunica con la capa de servicios para enviar solicitudes y recibir respuestas, mostrando los resultados y proporcionando una experiencia de usuario intuitiva y amigable.

## **Diagrama de clases con la correspondiente explicación, así como las consideraciones hechas al abordar la solución:**

El diagrama UML presenta las siguientes clases:

1. Months\_DTO: Esta clase representa un mes y contiene los atributos "month\_code" y "report\_code". Estos atributos permiten identificar de manera única un mes y relacionarlo con su correspondiente código de informe.
2. Report\_DTO: Esta clase representa un informe y almacena información relacionada con los servicios realizados, el alquiler de vehículos, los ingresos generados, entre otros. Los atributos incluyen "report\_code", "amo\_services", "amo\_rents", "income\_rents", "amo\_others", "income\_others" e "income\_total". Estos atributos proporcionan datos específicos sobre el informe y permiten realizar cálculos basados en los ingresos y los servicios prestados.
3. Contract\_DTO: Esta clase representa un contrato y contiene información relevante sobre el solicitante, las fechas de inicio y finalización, los kilómetros contratados, el monto del contrato, el país y el número de flota. Los atributos incluyen "contract\_code", "applicant\_name", "start\_date", "end\_date", "contract\_kms", "contract\_amount", "contract\_country" y "fleet\_number". Estos atributos ayudan a identificar y gestionar los contratos dentro del sistema.
4. Car\_DTO: Esta clase representa un carro y almacena detalles como el número de flota, la placa, el código de pareja y el nombre de la marca. Los atributos son "fleet\_number", "plate", "couple\_code" y "brand\_name". Estos atributos permiten identificar y gestionar los automóviles dentro del sistema.
5. Brand\_DTO: Esta clase representa una marca de carro y contiene información sobre el número de asientos, el tipo de combustible y los gastos asociados. Los atributos incluyen "brand\_name", "amo\_seats", "fuel\_type" y "spending". Estos atributos proporcionan detalles sobre las características y especificaciones de la marca del carro.
6. Couple\_DTO: Esta clase representa una pareja y almacena el código de pareja, así como los dos identificadores asociados a ella. Los atributos son "couple\_code", "ID1" e "ID2". Estos atributos permiten identificar y gestionar las parejas dentro del sistema.
7. Driver\_DTO: Esta clase representa un conductor y contiene detalles como el identificador, el nombre, la dirección, el teléfono y el código de distrito. Los atributos son "id\_driver", "driver\_name", "address", "phone" y "district\_code". Estos atributos proporcionan información personal y de contacto del conductor.
8. District\_DTO: Esta clase representa un distrito y almacena el código de distrito y el nombre del distrito. Los atributos son "district\_code" y "district\_name". Estos atributos permiten identificar y gestionar los distritos dentro del sistema.
9. Service\_DTO: Esta clase representa un servicio y contiene información como el código de servicio, el nombre del servicio, el número de solicitud, el código del grupo turístico, el país, el lugar de recogida, la hora de recogida, la cantidad de pasajeros, los kilómetros del servicio y el monto. Los atributos incluyen "service\_code", "service\_name", "request\_number", "tour\_group\_code", "country", "pickup\_place", "pickup\_time", "pax", "service\_kms" y "amount". Estos atributos proporcionan detalles sobre el servicio realizado, como la ubicación, la fecha y hora de recogida, la cantidad de pasajeros y el monto asociado al servicio.
10. Discrepancy\_DTO: Esta clase representa una discrepancia y almacena información relacionada con los kilómetros planificados, los kilómetros recorridos, la diferencia de kilómetros, el combustible planificado, el combustible consumido y la diferencia de gastos de combustible. Los atributos son "month\_code", "fleet\_number", "planned\_kms", "tours\_kms", "difference\_kms", "planned\_fuel", "consumed\_fuel" y "dif\_spending\_fuel". Estos atributos permiten realizar un seguimiento de las discrepancias entre los datos planificados y los datos reales del vehículo.
11. Roadmap\_DTO: Esta clase representa una hoja de ruta y almacena información sobre la fecha, el número de flota, los kilómetros recorridos y la hora de salida. Los atributos son "date", "fleet\_number", "kms" y "departure\_time". Estos atributos permiten registrar y gestionar las hojas de ruta de los vehículos.
12. Free\_Cover\_DTO: Esta clase representa una cobertura gratuita y almacena información sobre el conductor y la marca del automóvil asociada. Los atributos son "id\_driver" y "brand\_name". Estos atributos permiten gestionar la asignación de coberturas gratuitas a conductores específicos.
13. Fuel\_DTO: Esta clase representa el tipo de combustible y almacena información sobre el tipo de combustible utilizado. El atributo es "fuel\_type". Este atributo permite gestionar los diferentes tipos de combustible utilizados en los vehículos.
14. Tourist\_Group\_DTO: Esta clase representa un grupo turístico y almacena información sobre el código del grupo turístico. El atributo es "tour\_group\_code". Este atributo permite identificar y gestionar los diferentes grupos turísticos en el sistema.
15. Country\_DTO: Esta clase representa un país y almacena información sobre el nombre del país. El atributo es "country\_name". Este atributo permite identificar y gestionar los países asociados a los servicios prestados.

Las consideraciones realizadas al abordar la solución en este proyecto incluyen:

1. Identificación y modelado adecuado de las entidades relevantes: Se ha realizado un análisis exhaustivo de los datos necesarios para el proyecto y se han identificado las entidades clave que representan los elementos del dominio. Se ha asegurado que las clases y sus atributos reflejen con precisión la estructura y los atributos de los datos en el contexto del proyecto.
2. Relaciones y asociaciones entre las clases: Se han establecido relaciones y asociaciones adecuadas entre las clases para reflejar las relaciones lógicas entre los elementos del dominio. Se han utilizado relaciones como asociaciones y agregaciones para representar las conexiones entre las entidades, lo que permite una representación precisa de la estructura de los datos.
3. Tipos de datos y restricciones: Se han tenido en cuenta los tipos de datos apropiados para los atributos de cada clase, asegurando que sean coherentes con los requisitos del proyecto. Además, se han aplicado restricciones y validaciones para garantizar la integridad de los datos y prevenir errores o inconsistencias.
4. Abstracción y encapsulamiento: Se ha aplicado el principio de abstracción al identificar las clases y sus atributos esenciales para representar los elementos clave del sistema. Además, se ha utilizado el encapsulamiento para ocultar los detalles internos de las clases y proporcionar interfaces claras para acceder y manipular los datos.
5. Nomenclatura clara y consistente: Se ha utilizado una nomenclatura adecuada y coherente para los nombres de clases y atributos, siguiendo convenciones de nomenclatura estándar. Esto facilita la comprensión y el mantenimiento del código, así como la comunicación efectiva entre los miembros del equipo.
6. Representación visual clara: El diagrama UML proporciona una representación visual clara de las clases y sus relaciones, lo que facilita la comprensión de la estructura del sistema. Se han utilizado los diferentes elementos de UML, como las clases, los atributos y las relaciones, de manera efectiva para transmitir la información de manera concisa y precisa.

Para más de detalle sobre este tema consulte el **Anexo C**

## **Seguridad: roles y privilegios por roles:**

Seguidad y Roles

La seguridad es un aspecto fundamental en cualquier proyecto de software, y en este informe se describe cómo se aborda la seguridad a través de la implementación de roles y privilegios por roles. Estos roles y privilegios proporcionan un control de acceso adecuado y aseguran que los usuarios tengan los permisos necesarios para realizar sus tareas dentro del sistema. A continuación, se detalla su implementación.

Roles:

* Administrador: Tiene los privilegios más elevados y acceso completo a todas las funcionalidades del sistema. Puede administrar usuarios, realizar cambios en la estructura de la base de datos y ejecutar operaciones críticas.
* Soporte: Tiene acceso limitado a las funcionalidades que se relacionan con los servicios que brinda la empresa, tanto de alquiler como de transporte.
* Gerente de soporte: Tiene acceso a los permisos de soporte, más la administración de las discrepancias, choferes e informe de los ingresos.

Es importante destacar que se aplican principios de "menos privilegios", lo que significa que a los usuarios se les asignan los privilegios mínimos necesarios para realizar sus tareas, reduciendo así el riesgo de accesos no autorizados y aumentando la seguridad global del sistema.

La asignación de roles y privilegios se realiza durante el proceso de creación de usuarios en el sistema. Cada usuario se asigna a un rol específico, lo que determina los privilegios que se le otorgan.

El método de seguridad implementado para el sistema de roles es la autenticación. Se utiliza un mecanismo de autenticación sólido, como contraseñas seguras de un mínimo de 8 caracteres, con mayúsculas, minúsculas y caracteres especiales.

En conclusión, la implementación de roles y privilegios por roles en este proyecto garantiza un control de acceso adecuado y una seguridad sólida. Los roles permiten asignar permisos y responsabilidades específicas a los usuarios, mientras que los privilegios definen las acciones que cada rol puede realizar en el sistema. Esto ayuda a proteger la integridad de los datos y a prevenir accesos no autorizados, asegurando que cada usuario tenga los privilegios necesarios para desarrollar sus tareas y cumplir con sus funciones sin comprometer la seguridad de la información.

## **Pautas seguidas para el diseño de la interfaz, así como prototipo de las interfaces gráficas más importantes:**

## **Descripción de los mecanismos empleados para el tratamiento de errores:**

# **Conclusiones**

# **Recomendaciones**

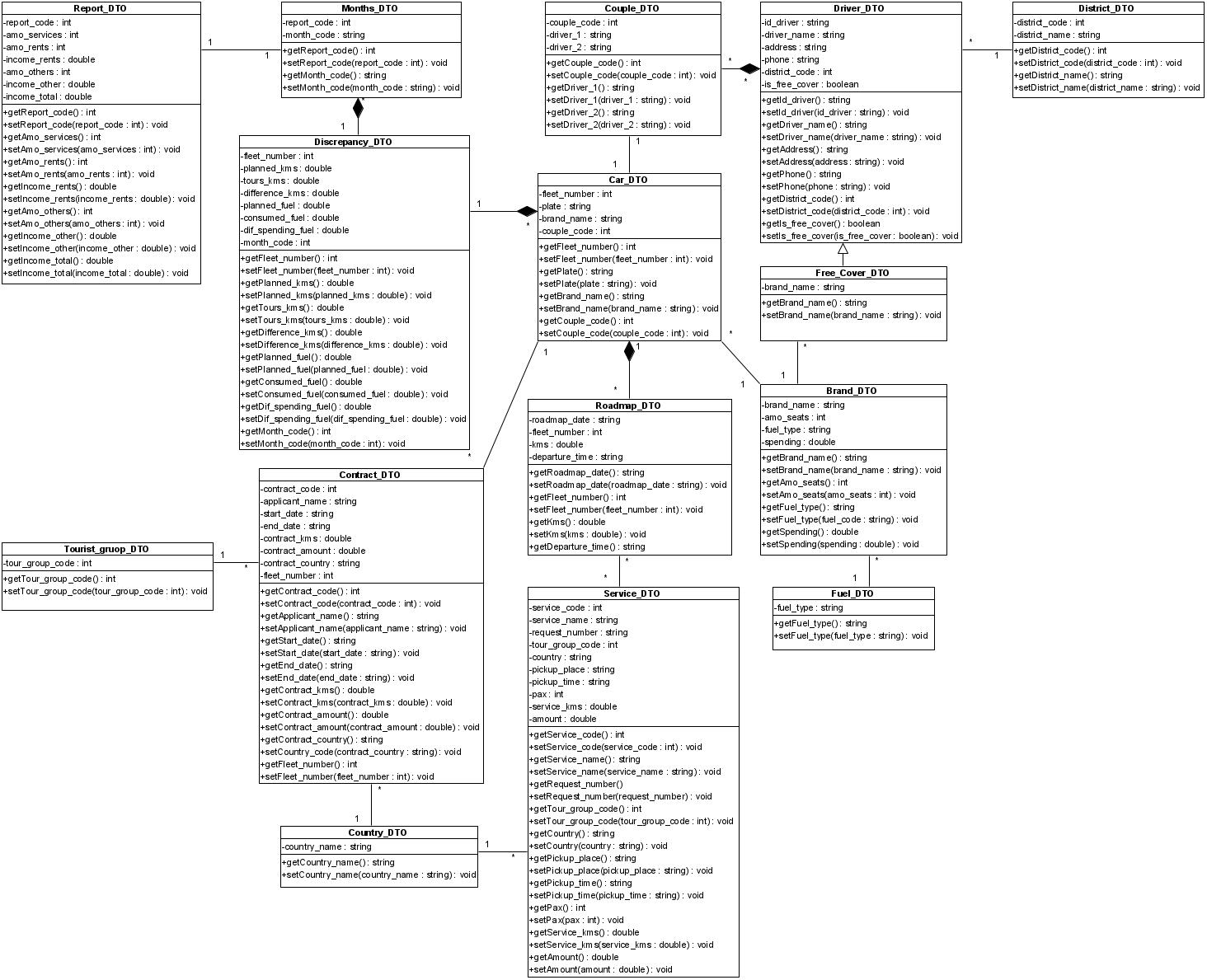
1. Sistema de seguimiento de mantenimiento de los vehículos: implementar en el sistema un registro del mantenimiento de los vehículos, incluyendo las fechas en que se realizaron las revisiones, cambios de aceite, reparaciones y otros mantenimientos, facilitaría el control de los costos de mantenimiento y la prolongación de la vida útil de los vehículos.
2. Integración con aplicaciones de análisis de datos: incorporar la funcionalidad de análisis de datos permitiría obtener información valiosa sobre el rendimiento de los vehículos, el comportamiento de los conductores, las rutas más utilizadas y otros datos relevantes, lo que permitiría tomar decisiones informadas para mejorar la eficiencia del sistema de transporte.
3. Añadir una funcionalidad de análisis y estadísticas, para que los administradores puedan obtener información detallada sobre el rendimiento de los conductores, el consumo de combustible, la rentabilidad de los servicios y otros aspectos clave del negocio.

# **Referencias bibliográficas**

# **Anexos**

**Anexo A:**

**Anexo B:**

**Anexo C:**