

Trabajo de Curso de Bases de Datos

Sistema Automatizado Rent a Car

**Autores**

Luis Miguel Lagos Xenes, llagos@ceis.cujae.edu.cu

Carlos Daniel Robaina Rivero, crobaina@ceis.cujae.edu.cu

Jean Michel Ruiz Remis, jruiz@ceis.edu.cu

**Tutor**

Dr. C. Isis Torres Pérez, itorres@ceis.cujae.edu.cu

**La Habana, 2023**

Resumen

El objetivo de este proyecto es desarrollar un sistema de gestión orientado a una empresa de alquiler de vehículos, permitiéndole gestionar su flota, contratos y clientes de manera eficaz y efectiva. Se identificó como principal problemática la necesidad de automatizar el proceso de contratación, así como la administración interna de los factores inherentes al negocio, como son el estado de los vehículos, la información de los clientes y choferes. Para resolver estos problemas, se pretende diseñar e implementar un sistema automatizado de gestión, respaldado en una base de datos relacional. Permitiendo almacenar de forma ordenada y coherente la ya mencionada información sobre contratos, turistas, autos y conductores. El sistema busca facilitar la gestión del negocio por parte del personal de la institución en sus respectivos niveles de acceso. Finalmente, se espera que la aplicación permita a los directivos de la empresa tomar decisiones más acertadas en la dirección de la misma. Esto gracias a poseer todos los datos necesarios, correctamente presentados, mejorando así su eficiencia operativa en general.

**Palabras clave**: gestión, turismo, alquiler, vehículos.

Contenido

[Introducción 1](#_Toc134962023)

[Desarrollo 3](#_Toc134962024)

[1. Requisitos Funcionales: 3](#_Toc134962025)

[2. Descripción de la Solución Propuesta 4](#_Toc134962026)

[2.1. Diseño de la Base de Datos: 4](#_Toc134962027)

[2.2. Modelo Lógico vs Modelo Físico 4](#_Toc134962028)

[2.3. Arquitectura del Proyecto 4](#_Toc134962029)

[2.4. Diagrama de Clases 5](#_Toc134962030)

[2.5. Seguridad: Roles y Privilegios 10](#_Toc134962031)

[2.6. Interfaz Gráfica 11](#_Toc134962032)

[2.7. Tratamiento de Errores 11](#_Toc134962033)

# Introducción

Este informe complementa el desarrollo de un sistema de gestión de bases de datos, que responde a la problemática del ejercicio académico cuyo enunciado es el siguiente:

*En el departamento de Contabilidad de la agencia Rent a Car de una corporación turística se desea controlar los ingresos por concepto de alquiler de autos. De cada turista se conocen su nombre y apellidos, su número de pasaporte, edad, sexo y un número de contacto. Un turista puede alquilar varios autos, y un auto puede ser alquilado por varios turistas, siempre y cuando no sea el mismo día, es decir, en un mismo día el carro es alquilado sólo por un turista. De cada auto se conoce su placa, su modelo (cada modelo es único dentro de una marca en específico), su marca y su situación (taller, alquilado, disponible).*

*El alquiler de un auto por un turista en un período de tiempo determinado constituye un contrato, del cual se conocen fecha de inicio, fecha de fin, forma de pago (efectivo, cheque o tarjeta de crédito), cuántos días de prórroga si los tiene y si se desea alquilar un chofer o no. Para saber el importe del alquiler, existe una tarifa para los días previstos en el contrato y otra tarifa especial de cobro para los días de prórroga. Se consideran incumplidores del contrato aquellos turistas que entregan el carro en fecha posterior a la convenida, considerándose esta diferencia como días de prórroga.*

*Un chofer no tiene un auto fijo para manejar, pero dentro de un contrato, el auto sólo puede ser manejado por un chofer. De cada chofer se conoce su número de identidad, su nombre y apellidos, su dirección particular y su categoría.*

El objetivo de este trabajo es desarrollar un sistema que permita gestionar un modelo de base de datos relacional, respondiendo a la situación anteriormente planteada. La aplicación que se propone será desarrollada en Java, en específico usando IntelliJ Idea como IDE, con un plugin para el trabajo con la interfaz gráfica(Jform Designer). En el caso de la Base de Datos se utiliza PostgreSQL y pgAdmin para efectuar la gestión de la misma. Adicionalmente se utilizan herramientas de modelación para realizar el diagrama entidad-relación (DER) y los diagramas de clases, como es el caso de ERECASE y draw.io. Como última herramienta se destaca Jasper Studio como aplicación utilizada para el diseño de los reportes.

DESCRIBIR ESTRUCTURA DEL DOCUMENTO

# Desarrollo

A continuación se comienza la descripción del proceso de solución de la problemática planteada. Comenzado por la identificación de los requisitos funcionales y a partir de ahí el desarrollo del sistema, desglosado paso por paso para su correcto entendimiento por los lectores.

## Requisitos Funcionales:

Un requisito funcional es una declaración de cómo debe comportarse un sistema. Define lo que el sistema debe hacer para satisfacer las necesidades o expectativas del usuario. Los requisitos funcionales se pueden considerar como características que el usuario detecta.

La aplicación debe garantizar los siguientes aspectos:

* Los trabajadores de la empresa deben poder iniciar sesión sesión en la aplicación usando un usuario y una contraseña previamente definidos.
* Cada trabajador debe tener un rol y un nivel de acceso asociado a ese rol, definiendo así a que funcionalidades tiene o no acceso.
* Los administradores deben poder crear y eliminar tanto usuarios como niveles de acceso a la aplicación.
* Se debe poder registrar, eliminar y consultar la información de los actores principales del negocio: clientes (turistas), autos, choferes y contratos de la empresa.
* Se debe poder gestionar además los datos de consulta o secundarios que enriquecen la información relacionada a los actores principales. En este apartado se incluyen: marcas, modelos y situación de los autos, categorías licencia de conducción de los choferes, procedencia de los turistas y métodos de pago y tarifas de los contratos.
* Se debe poder modificar la información existente con el objetivo de subsanar errores de entrada de los datos.
* Se debe poder abrir contratos nuevos y cerrar los ya creados.
* Se debe garantizar una serie de reportes que muestren el estado de la empresa. Los reportes deben poder ser exportados en PDF.

Como aclaración, se define abrir un contrato como la acción llevada a cabo cuando un cliente contrata los servicios de la empresa. El turista alquila un auto, definiendo la fecha de entrega propuesta y el método de pago. Por otra parte, se define cerrar un contrato como la acción llevada a cabo cuando un cliente deja de utilizar los servicios de la empresa. El turista devuelve el auto alquilado, marcando la fecha de entrega real del vehículo, registrando los días de prórroga si los hubiera y pagando el importe correspondiente por el servicio.

## Descripción de la Solución Propuesta

A continuación se presentan una serie de apartados que buscan describir paso a paso el proceso mediante el cual se le da solución a la problemática planteada. Se detallan las técnicas utilizadas, las decisiones tomadas y de manera general se provee la información necesaria para comprender el proceso en su totalidad.

### Diseño de la Base de Datos:

#### Identificación de Entidades y Atributos

Para indentificar las entidades y atributos corresponde realizar un análisis de cada uno de los Reportes que debe ofrecer la aplicación, así como la otención de las dependencias funcionales a partir del enunciado.

REPORTE #1:

Atributos: país, nombre del turista, número de pasaporte, cantidad de veces que el turista ha usado autos, valor total de los alquileres del turista.

En este caso se pueden definir como atributos calculables la cantidad de veces que el turista ha usado autos y el valor total de los alquileres del turista. De esta forma quedaría definida la entidad Turista con los atributos país, número de pasaporte y nombre del turista. De los cuales el número de pasaporte ejerce como identificador de los atributos del turista. Así mismo se puede identificar el país de procedencia del turista como una entidad en sí misma debido a que este atributo puede ser recogido dentro de un archivo de consulta. De esta forma quedan las siguientes dependencias funcionales:

* pasaporte -> nombreTurista, nombrePaís

A partir de las cuales aparecen las siguientes relaciones en 3ra forma normal:

* TURISTA (pasaporte, nombreTurista, nombrePaís)
* PAÍS (nombrePaís)

REPORTE #2:

Atributos: placa del auto, marca, modelo, color, cantidad de kilómetros recorridos.

En este caso como atributo calculable puede saltar a la vista la cantidad de kilómetros recorridos, sin embargo se ha decidido no tratarlo de esta forma, sino como un atributo más inherente al auto. De esta forma queda definida la entidad Auto con los atributos: placa del auto, marca, modelo, color, cantidad de kilómetros recorridos, de los cuales la placa del auto ejerce como identificador de los atributos del auto. Los campos marca y modelo se identifican archivos de consulta y por tanto como entidades individuales, donde el modelo depende de la marca a la que pertenece.

Dependencias Funcionales:

* placa -> nombreModelo, nombreMarca, color, kilometraje
* nombreModelo -> nombreMarca

1FN:

* R(placa, nombreModelo, nombreMarca, color, kilometraje)

2FN:

* AUTO(placa, modelo, marca, color, kilometraje)

3FN:

* AUTO(placa, modelo, marca, color, kilometraje)
* MARCA(nombreMarca)
* MODELO(nombreMarca, nombreModelo)

REPORTE #3:

Atributos: nombre del turista, placa, marca, modelo, forma de pago, fecha de inicio del contrato, fecha de fin del contrato, prórroga, alquiler de chofer(Si o No), importe total.

En este caso se identifica como atributo calculable la cantidad de días de prórroga, calculado a partir de la diferencia entre la fecha de fin del contrato y la fecha de entrega del auto.

### Modelo Lógico vs Modelo Físico

### Arquitectura del Proyecto

A medida que una aplicación se va modularizando es natural la división de la misma en varias capas. En el caso específico de este sistema, se encuentra dividio en tres secciones o capas claramente separadas: Interfaz Gráfica, Servicios y Persistencia. Las dos primeras corresponden a la parte del proyecto que se desarrolla usando Java y la última está constituida por la Base de Datos como tal. A nivel de código el proyecto en Java se divide en 4 paquetes: *gui, dto, services, utils*, además de una clase Main que ejecuta la aplicación.

La Interfaz Gráfica constituye la capa más superficial de la aplicación, aquella con la que el usuario interactúa directamente. Está formada por una serie de Formularios correctamente vinculados a las clases que los manejan. Esta capa se restringe en este caso al paquete *gui* de la aplicación y a ciertas clases específicas del paquete *utils*. Esta capa del programa fue desarrollada haciendo uso de la librería gráfica Java Swing.

Por su parte la capa de Servicios está localizada en el paquete *services*, utilizando también ciertas funcionalidades provistas por el paquete *utils*. La clase ServicesLocator facilita el acceso a todos los servicios presentes en la aplicación, funcionando como una clase estática e implementando un patrón Singleton para cada servicio.

Las clases contenidas en el paquete *dto* se corresponden con el vínculo entre servicios e interfaz gráfica, permitiendo transportar la información desde una capa a otra.

Por último, el nivel más bajo del sistema lo constituye la Base de Datos. Allí se asegura la persistencia de la información así como las herramientas para manejarla, en específico las funciones que permiten la manipulación correcta y segura de la Base de Datos. Son precisamente estas funciones las que convierten este nivel del programa en una capa lógica más, ya que la Base de Datos no solo funciona como un simple almacén para guardar los datos, sino que también asegura comportamientos y mecanismos indispensables para solucionar la problemática planteada.

### Diagrama de Clases

En esta sección se analiza el Diagrama de Clases resultante del modelo de la solución y se explican las consideraciones tomadas al abordar la problemática.

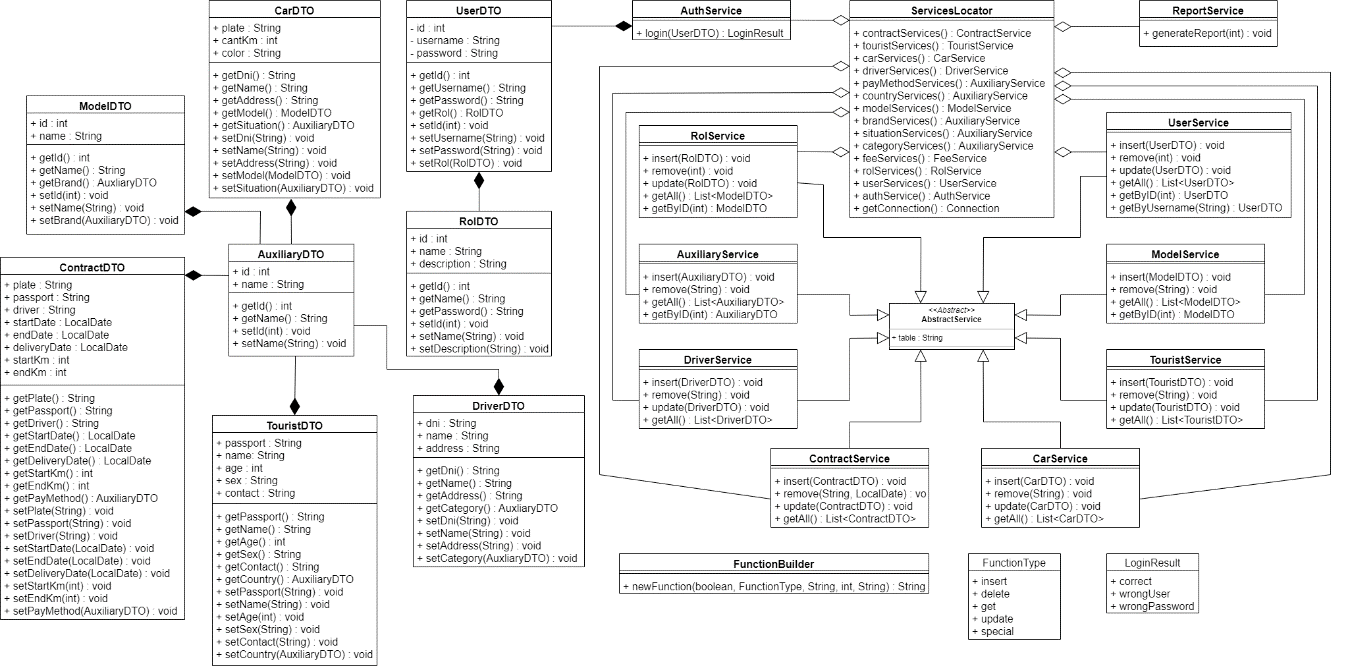


Diagrama de Clases: Panorama Global

En la imagen anterior, aunque es complicado definir los detalles, se muestra un panorama general de las clases involucradas en el proyecto y sus relaciones. De esta forma se definen tres secciones del diagrama, a la izquierda las clases relacionadas al patrón de diseño *Data Transfer Object (DTO)*, a la derecha las clases correspondientes a la capa de servicios de la aplicación y justo debajo de estas, aisladas, algunas clases de utilería.

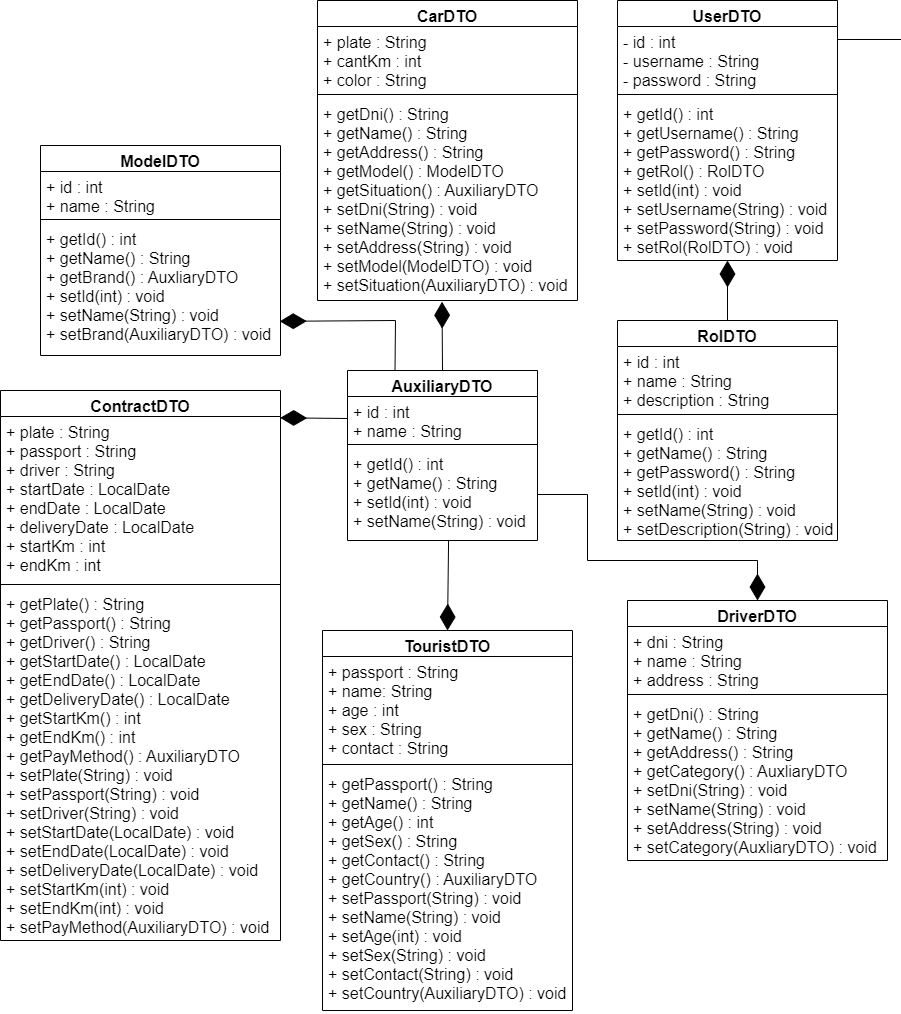


Diagrama de Clases: DTOs

Con las clases DTO presentadas se traslada el modelo de la Base de Datos a la apliacación en cuestión. Sin embargo, como se puede apreciar no se implementó una clase DTO para cada tabla. En su lugar, las tablas brand(marca), situation(situación), category(categoría), pay\_method(Método de pago) y country(país) se vinculan a la misma clase DTO: AuxiliaryDTO. La razón de esta desición se debe a que estas tablas, al ser tablas de consulta con poca información, comparten exactamente los mismos campos, id y nombre. De esta forma es posible representar la estuctura de la Base de Datos con un menor número de clases sin sacrificar información.

También es posible apreciar en el Diagrama como varias clases DTO contienen en su interior otros DTO. Este anidamiento se implementa con el objetivo de facilitar la representación de la información en la aplicación haciendo todas las consultas necesarias de una sola vez.

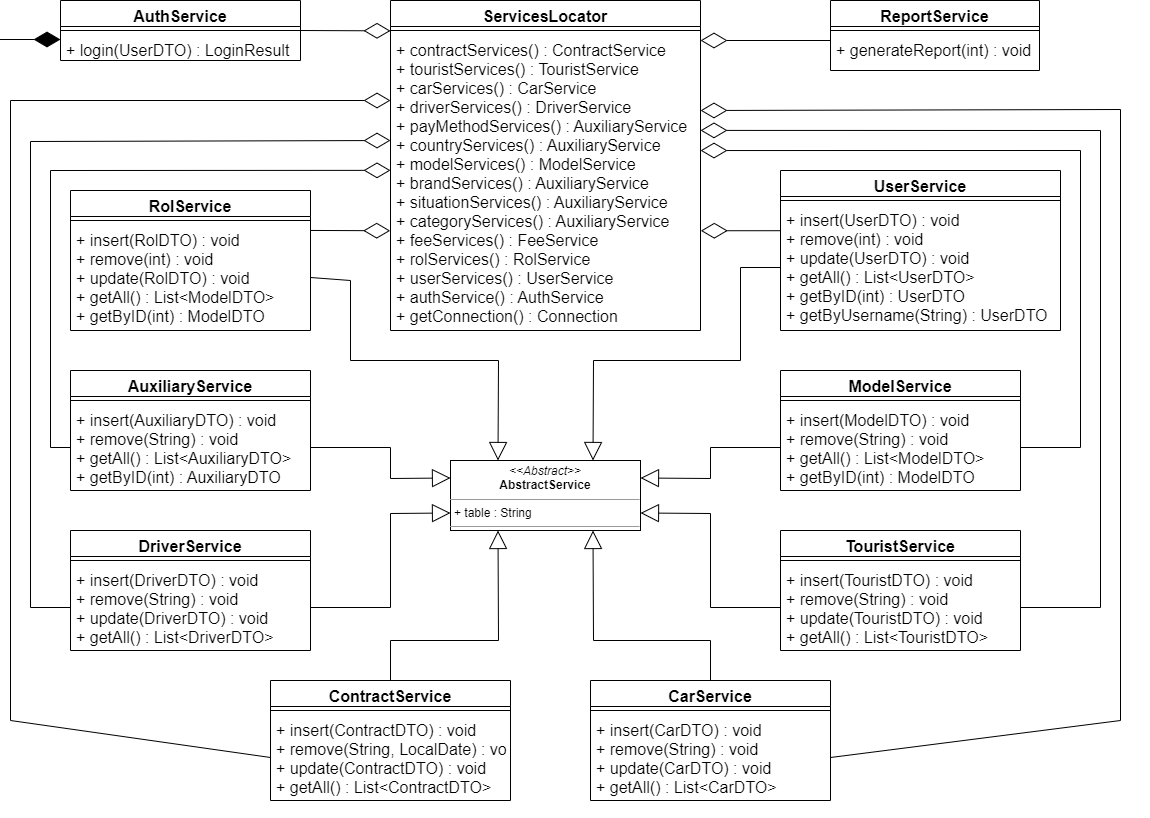


Diagrama de Clases: Services

En esta otra sección del diagrama se aprecia la estructura de la capa de Servicios de la aplicación. Teniendo como centro la clase ServicesLocator que funciona como facilitador de acceso a todos los servicios que provee la aplicación. En este caso se puede ver como se extiende la decisión tomada para los DTO. En lugar de existir una clase de servicios para cada tabla, aparece la clase AuxiliaryService que provee los métodos necesarios para trabajar con las tablas antes mencionadas. Sin embargo, la entidad ServicesLocator si presenta un método que ofrece servicios para cada tabla por separado, trabajando con distintas instancias de AuxiliaryService apuntando a distintas tablas en la Base de Datos.

Como se muestra además en el Diagrama las clases de servicio mencionadas heredan de la clase abstracta AbstractService. Esta clase almacena el nombre de la Base de Datos a la que debe acceder el serivicio, sirviendo este parámetro para diferenciar las distintas instancias de AuxiliaryService.

Además se pueden encontrar dos clases de servicio adicionales. La primera, ReportsService, ofrece la interfaz para realizar la consultas relacionadas a los reportes de la aplicación. La segunda, AuthService, asegura la posibilidad de la autenticación en la aplicación, así como mantener el control de la sesión actual.

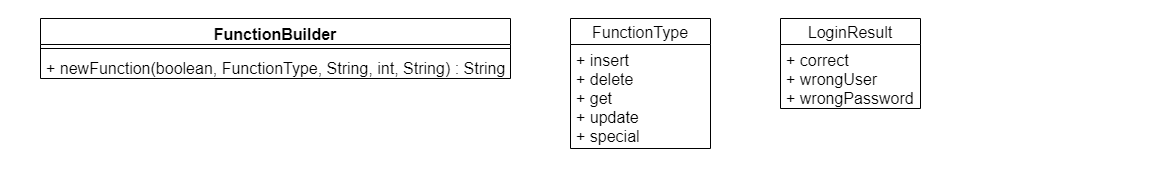


Diagrama de Clases: Otros

Por último se pueden analizar 3 clases. La primera, FunctionBuilder, responde una implementación del patrón de diseño *Builder*. Esta clase presenta un método el cual, a partir de ciertos parámetros, construye una cadena de caracteres que se corresponde con una función de la Base de Datos construida con el lenguaje plpgsql. Esta capacidad permite reducir en gran medida los errores cometidos al trabajar con cadenas de texto independientes, además de permitir un código más legible y reutilizable. Es importante destacar que esto solo es posible cuando se ha definido un estándar de nombrado para las funciones de la Base de Datos.

Las clases restantes son *enums* que funcionan como útiles en otros procesos de la aplicación. FunctionType corresponde precisamente al tipo de función que debe construir el FunctionBuilder ya mencionado. Por su lado LoginResult representa los valores posibles que pueden ser obtenidos de un intento de autenticación en la aplicación, permitiendo a la interfaz gráfica responder ante cada uno de ellos.

### Seguridad: Roles y Privilegios

En este apartado se analiza de manera general la seguridad de la aplicación, tratando temas como encriptación de información delicada y roles de acceso al sistema.

En primer lugar se han definido varios roles con distintos niveles de acceso a la información. De esta manera cada usuario que se crea, además de su nombre y contraseña, se vincula a uno de estos roles asegurando que solo pueda acceder a aquella información a la que tenga permisos. Los roles en cuestión son: administrador, jefe de área, trabajador y visitante.

1. **Visitante**: Acceso a los Reportes presentados por la aplicación, así como a consultar la información de las distintas tablas. Solo Lectura. Este rol está dedicado a usuarios trabajadores de la empresa que no están vinculados directamente con el aquiler pero que necesitan acceso a la información expuesta. En esta categería se podría incluir el departamento de Recursos Humanos y Contabilidad, así como los directivos de la empresa.
2. **Trabajador**: Acceso a abrir y cerrar Contratos, registrar y modificar Turistas, Países, Autos, Marcas, Modelos, Situaciónes de los Autos y Categorías de Licencia. Este rol está dedicado a usuarios que tratan directamente con los clientes. La aplicación está destinada principalmente a automatizar su trabajo por lo que ellos serán los usuarios más comunes.
3. **Jefe de Área**: Acceso a registrar y modificar Métodos de Pago, modificar las tarifas, contratar y despedir Conductores, modificar y eliminar Contratos, así como eliminar información de cualquier tabla antes mencionada. Este rol está dedicado a usuarios con altos cargos relacionados directamente al alquiler de los autos. Garantizando la posibilidad de eliminar cualquier tipo de información sensible bajo su responsabilidad.
4. **Administrador**: Acceso total a la aplicación, lo que incluye creación, modificación y eliminación de usuarios y roles. Este rol está dedicado a aquellos usuarios designados como administradores del sistema, teniendo acceso de escritura en toda la aplicación y teniendo la tarea de administrar el acceso a la misma.

Cada uno de los roles mencionados incluye los permisos de los roles anteriores.

Las contraseñas de cada uno de los usuarios se han identificado como un tipo de información sensible y que necesario que sea protegida mediante encriptación. Para dar cumplimiento a esta tarea se utilizó una librería de seguridad para Java que ofrece la capacidad de encriptar información. Asegurando de esta forma que la transferencia de contraseñas entre las distintas capa de la aplicación se haga de forma segura, así como que estas contraseñas se guarden protegidas en la Base de Datos.

### Interfaz Gráfica

### Tratamiento de Errores

En esta sección se analiza el tratamiendo de los errores que pudieran ocurrir durante la ejecución del sistema propuesto. Para comenzar se debe dividir la estrategia en dos partes: intentar evitar los errores antes de que ocurran y responder correctamente a aquellos que no sean evitables.

En primer lugar siempre se debe intentar evadir el problema antes de que ocurra. Esta previsión, en el caso que se presenta, comienza en la interfaz de usuario. Se hace necesario validar cualquier dato de entrada antes de que el mismo llegué siquiera a la capa de servicios. Este simple detalle asegura un trabajo más eficiente de las capas subyacentes, además de una preocupación menos para usuarios y desarrolladores. También se utilizó como método para prevenir excepciones la evasión del trabajo con cadenas de texto siempre que fuera posible, ya que estas pueden ser altamente propensas a errores de escritura. Esta medida se extiende desde el diseño de la interfaz, evitando presentarle al usuario campos de texto si estos pueden ser sustituidos por ComboBox u otros tipos de selectores. Hasta el propio código fuente de la aplicación, mediante la implementación de la clase FunctionBuilder que sustituye el engorroso trabajo con cadenas de texto para hacer las peticiones a la Base de Datos.

Por otra parte, existen errores que no pueden ser evitados en primera instancia, o que sería muy costoso o complicado hacerlo. Se puede analizar el ejemplo de intentar borrar un dato de una tabla que constituye una llave foránea en otra. Perfectamente podría implementarse un sistema de validación para no permitir intentar dicha eliminación, si se comprueba que esa llave efectivamente está presente en otras tablas. Sin embargo, esta funcionalidad ya la proveé el Sistema Gestor de Base de Datos por lo que intentar replicarla en la capa de servicio sería un desperdicio de tiempo y esfuerzo. En su lugar basta con capturar la excepción lanzada por la Base de Datos y advertir al usuario de que no se pudo completar su solicitud. De esta forma se concreta el tratamiendo de los errores post-ocurrencia, mediante la captura de excepciones, haciendo uso del sistema de excepciones provisto por Java.