

**Resumen**

El programa desarrollado es un sistema de gestión para una sucursal de una empresa de alquiler de vehículos. La aplicación permite llevar un registro detallado de los vehículos disponibles, así como de los alquileres realizados por los clientes.

Contenido

[**1.** **Introducción** 4](#_Toc150470465)

[**1.1 Excepciones y control de excepciones** 4](#_Toc150470466)

[**2.** **Metodología de estudio** 5](#_Toc150470467)

[**2.1. UML Trabajo Practico 2** 5](#_Toc150470468)

[**2.2. Clase Vehiculo** 6](#_Toc150470469)

[**2.3. Clase Detalle** 7](#_Toc150470470)

[**2.4. Clase Alquiler** 8](#_Toc150470471)

[**2.5 Clase Sucursal** 9](#_Toc150470472)

[**5. Referencias** 10](#_Toc150470473)

# **Introducción**

## **1.1 Excepciones y control de excepciones**

El control de excepciones en el lenguaje C# es fundamental para gestionar situaciones inesperadas durante la ejecución de un programa. Utiliza las palabras clave try, catch y finally para manejar acciones que podrían no completarse correctamente, controlar errores cuando sea necesario y limpiar recursos posteriormente. Las excepciones pueden ser generadas por Common Language Runtime (CLR), .NET, bibliotecas de terceros o el código de la aplicación, y se crean mediante la palabra clave throw. Es importante destacar que una excepción puede ser generada por un método distinto al llamado directamente en el código, lo que implica que el CLR busca un bloque catch adecuado en la pila de llamadas para manejar la excepción específica. En caso de no encontrar un bloque catch apropiado, el proceso finaliza y se muestra un mensaje al usuario. [1]

# **Metodología de estudio**

## **2.1. UML Trabajo Practico 2**

El sistema de gestión de alquiler de vehículos desarrollado se basa en un diagrama UML que presenta diversas clases y sus interconexiones. Entre las principales clases se encuentran: Vehiculo: Representa los vehículos disponibles para alquilar. Cada vehículo tiene atributos como número, matrícula, marca, color, kilometraje, estado, precio por día y cantidad de puertas. Está asociado con la clase Sucursal. Alquiler: Representa los contratos de alquiler entre clientes y la empresa. Contiene información como número de alquiler, precio total, datos del cliente (nombre, apellido, documento, teléfono) y una lista de detalles que incluyen vehículos alquilados y sus fechas de alquiler. Está asociado con la clase Sucursal. Detalle: Describe los detalles específicos de un alquiler, incluyendo el vehículo alquilado, la fecha de retiro y la duración del alquiler. Esta clase está compuesta por la clase Alquiler. Sucursal: Representa las sucursales de la empresa de alquiler. Cada sucursal tiene un número y una dirección, y está asociada con colecciones de vehículos y alquileres. Existen relaciones de agregación con las clases Vehiculo y Alquiler. El diagrama UML también muestra relaciones clave entre estas clases: Asociación: Indica que los objetos de una clase están relacionados con objetos de otra clase. Por ejemplo, los objetos Vehiculo y Alquiler están asociados con objetos de la clase Sucursal. Composición: Representa una relación fuerte donde una clase (por ejemplo, Detalle) es parte integral de otra clase (por ejemplo, Alquiler). Los detalles de un alquiler no pueden existir sin el alquiler asociado. Agregación: Indica una relación más débil en la que una clase (por ejemplo, Sucursal) tiene una colección de objetos de otra clase (por ejemplo, Vehiculo y Alquiler). Estos objetos pueden existir independientemente de la clase que los contiene. El diagrama UML proporciona una estructura clara para el sistema de gestión de alquiler de vehículos, mostrando cómo las clases están interrelacionadas y cómo interactúan dentro del programa. (Figura 2.1)

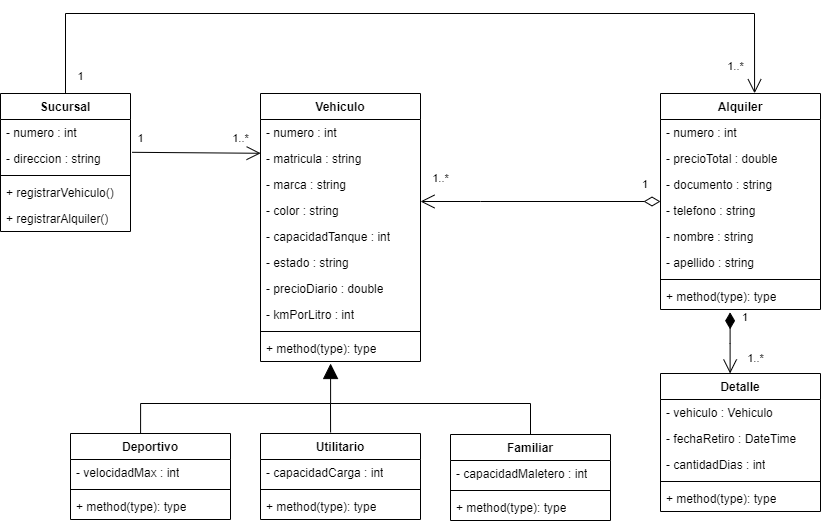


Figura 2.1: Diagrama UML.

## **2.2. Clase Vehiculo**

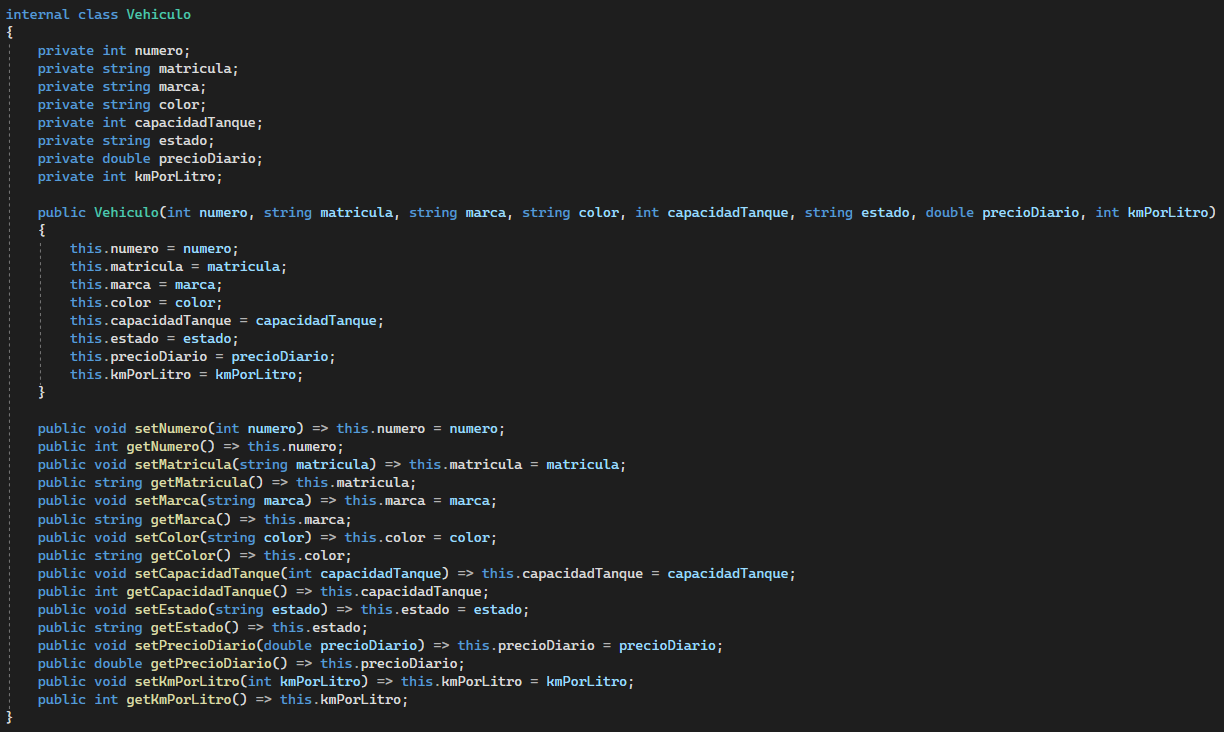
La clase Vehiculo representa un vehículo en la automotora. Cada vehículo tiene atributos como número, matrícula, marca, color, kilometraje, estado, precio por día y cantidad de puertas. Estos atributos describen las características del vehículo, como su identificación única, detalles físicos y su disponibilidad para alquiler. Esta clase es esencial para llevar un registro de los vehículos disponibles y sus propiedades (Figura 2.2).

Figura 2.2: Clase Vehiculo con sus atributos y métodos.

## **2.3. Clase Detalle**

La clase Detalle está diseñada para representar los detalles específicos de un alquiler asociado a un vehículo en particular. Almacena información como el vehículo involucrado, la fecha de retiro y la cantidad de días del alquiler. En otras palabras, un Detalle guarda los pormenores de un alquiler específico, lo que facilita el seguimiento de cada transacción de alquiler (Figura 2.3).

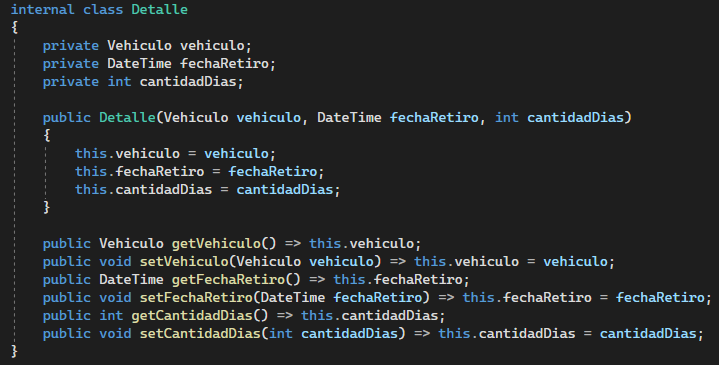


Figura 2.3: Clase Detalles con sus atributos y métodos.

## **2.4. Clase Alquiler**

La clase Alquiler modela un contrato de alquiler en la automotora. Contiene detalles sobre el número de alquiler, el precio total, la información del cliente (como documento, teléfono, nombre y apellido) y una lista de detalles asociados al alquiler. Esta clase permite gestionar y organizar los contratos de alquiler, proporcionando una estructura para almacenar información relevante sobre cada transacción (Figura 2.4).

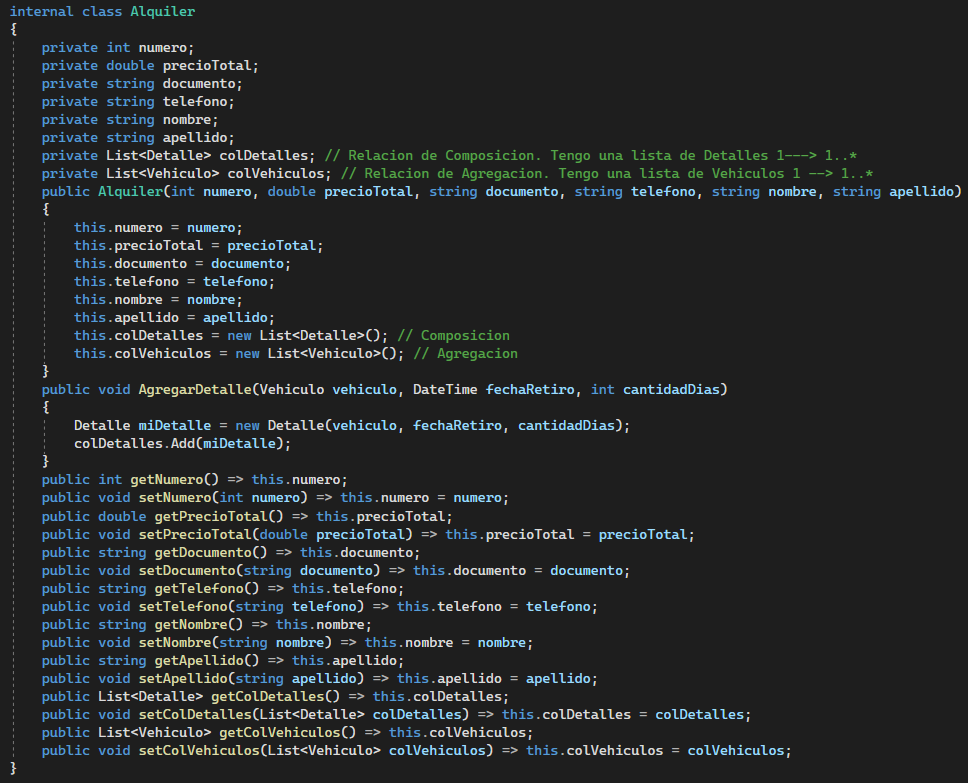
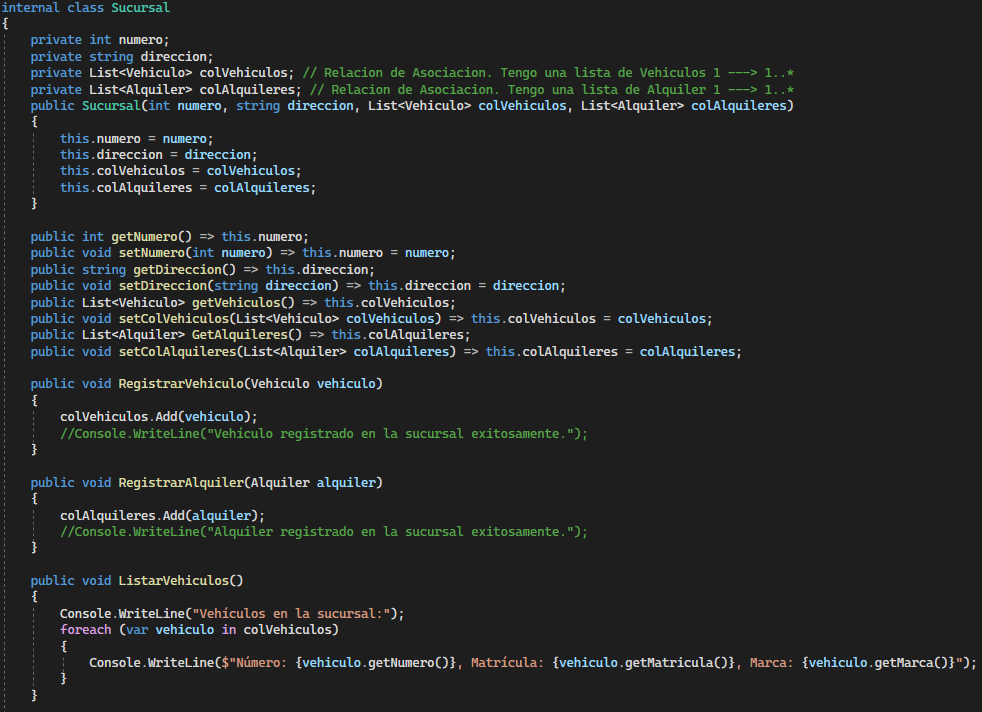


Figura 2.4: Clase Alquiler con sus atributos, métodos y relaciones.

## **2.5 Clase Sucursal**

La clase Sucursal representa una ubicación específica de la automotora. Tiene atributos como número, dirección, una lista de vehículos y una lista de alquileres. Esta clase actúa como un contenedor para organizar y gestionar los vehículos disponibles y los contratos de alquiler en una sucursal particular. Permite realizar operaciones como registrar vehículos y alquileres, así como listarlos cuando sea necesario (Figura 2.5 y 2.6).



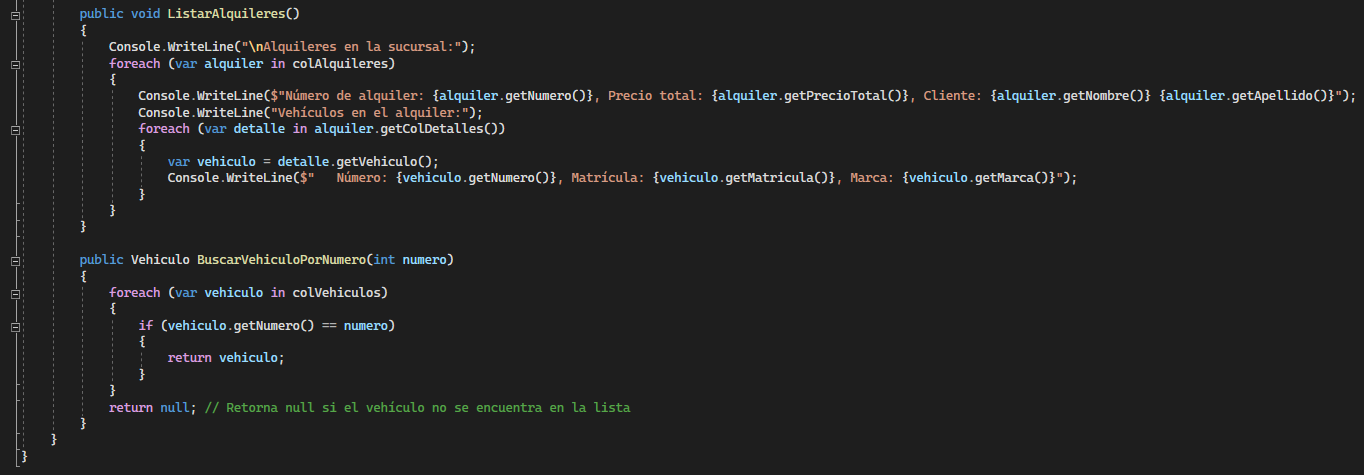
Figura 2.5: Clase Sucursal con sus atributos, métodos y relaciones.

Figura 2.6: Continuación de la figura 2.4 del código de la Clase Sucursal.

# **5. Referencias**

[1] https://learn.microsoft.com/es-es/dotnet/csharp/fundamentals/exceptions/