# TRABAJO PRÁCTICO N°3

## TEMA ELEGIDO:

El programa consta de una **Concesionaria** que posee Autos, Camionetas y Motocicletas. Éstas tienen su Alta - Baja - Modificación y su parte de filtrado con exportación de reporte en formato .csv, y también su parte de estadísticas dentro del formulario REPORTES con exportación .txt

## IMPLEMENTACIONES TP3:

**Clase 10 - Excepciones**: Implementada en los setters de cada atributo, se generaron 2 clases personalizadas, una de archivo y otra de atributo, y además fueron utilizadas en las vistas de los formularios para ser capturados y comunicar al usuario posibles errores.

**Clase 11 - Pruebas Unitarias**: Implementada en un proyecto aparte, se realizaron 2 test, uno que indica que el objeto creado se espere una excepción personalizada ya que se le pasaron al constructor atributos inválidos, y otro método con la carga del archivo en formato JSON de una lista y que esta no sea null.

**Clase 12 - Tipos Genéricos**: Implementada en clase Archivos, para leer y escribir archivos con el tipo de dato especificada en T, y luego en clase Filtrar, clase Estadística, utilizada para los reportes, mismo funcionamiento con dato del tipo T donde T sea de la clase Vehículos.

**Clase 13 - Interfaz:** Implementada entre 2 clases (AUTO - CAMIONETA), las cuales se le asignan propiedades como cantidad de puertas y marca de vehículo.

**Clase 14 - Archivos**: Implementada a la hora de cargar los datos de las listas y guardar datos, además de la exportación de los filtros y estadísticas en .csv o .txt (Lista Autos y Camionetas en formato JSON, Lista Motocicletas en formato XML).

**IMPLEMENTACIONES TP4:**

**Métodos de Extensión**: Clase estática llamada **Extensiones Métodos**, la cual posee 3 métodos estáticos los cuales se llaman **MaxColor()**, éste retorna el color que más abunda entre la cantidad de las listas de Autos, Camionetas, Motocicletas. Usada para los reportes estadísticos.

**SQL:** Implementada a la hora de insertar datos en las listas de Autos, Camionetas, Motocicletas,modificar un vehículo, baja lógica cambiandole el estado en 0 y traerme el número máximo de id de cada tipo de vehículo, éste para cuando el usuario esté esperando la carga de las listas de datos igualmente puede dar de alta un vehículo.

**Eventos y Delegados:** Implementó el delegado **Financiar** que retorna un string y no recibe parámetros, creo **Evento** en cada clase de vehículo con el tipo de delegado dicho, en los constructores le asignó el método obligatorio abstract **GetCaracteristicasFinanciacion()** que éste calcula la financiación del vehículo llamando a los métodos de la clase Financiación, crea un texto con las características y lo retorna para ser utilizado en el momento en el que el usuario hace doble click en el dataGrid y vé los detalles del vehículo.

**Task:** Aplicado al iniciar el programa, muestra un gif donde indica o mejor dicho simula carga de datos del programa con un sleep de 15 segundos, se deshabilita el botón de reportes pero sigue habilitado el alta de un vehículo. Una vez que termina, indica por mensaje que se completó la carga de datos correctamente.

**Expresiones LAMBDA:** Utilizadas prácticamente en todo el flujo del programa donde se necesitaba buscar, filtrar u ordenar listas.

**ESTADÍSTICAS - REPORTES**

Fueron implementadas las estadísticas pedidas que fueron faltantes en la entrega del TP3, se creó un formulario donde se indican los porcentajes entre distintas características de los vehículos y cuanto son por sobre el total.
Además, se sigue teniendo la opción de exportarlo en este caso en formato .txt
Se encuentra en el botón celeste **“VER ESTADÍSTICAS”** en el formulario de REPORTES.