

Universidad Claeh
Centro CTC



Analista Programador
Programación II

Proyecto Web
Trabajo Obligatorio

Leonardo Suárez
Angel Machado

2023

Resumen

La propuesta de este proyecto consiste en el desarrollo de una aplicación web en C# y ASP.NET para una empresa ficticia dedicada a la venta y alquiler de vehículos, denominada L.A. Motors. La aplicación brinda diversas funcionalidades, como la creación detallada de usuarios, clientes y vehículos de tres tipos diferentes, así como la capacidad de editar estos registros. El sistema principal permite la venta y alquiler de vehículos de tres categorías: autos, motos y camiones.

La aplicación se adhiere a los principios de programación orientada a objetos, destacando la aplicación de conceptos clave como la herencia y haciendo un uso significativo de listas en C#. A diferencia de proyectos convencionales, no se utiliza una base de datos en un servidor; en su lugar, se gestiona la información mediante una clase denominada Base de Datos, que contiene listas de objetos existentes y de aquellos que se crean durante el uso del programa web.

La estructura del código refleja la naturaleza jerárquica de los objetos, empleando la herencia para modelar eficientemente las relaciones entre las clases de vehículos (auto, moto y camión). Cada una de estas clases hereda métodos y atributos de la clase vehículo, pero también posee características únicas que las diferencian, permitiendo así una representación coherente de los diferentes tipos de vehículos disponibles para venta y alquiler.

La función principal de la aplicación web es la venta y el manejo de alquileres. Permite al usuario registrar datos de clientes, que se almacenan en listas correspondientes. A través de la página de ventas, se facilita la selección de un cliente y un vehículo, registrando así una venta o un alquiler con una duración específica.

El proyecto exhibe un sólido dominio de listas en C#, utilizadas para gestionar y organizar datos cruciales para el sistema, como la lista de vehículos, ventas realizadas, alquileres registrados, clientes y usuarios habilitados. Estas listas, ubicadas dentro de la clase Base de Datos, permiten el almacenamiento de datos previamente cargados y de información agregada por el usuario mediante la interfaz gráfica, funcionando como una base de datos interna.

La aplicación proporciona una interfaz intuitiva a través de ASP.NET, permitiendo a los usuarios realizar diversas operaciones, como alquilar y vender vehículos, consultar la disponibilidad y seguir el historial de transacciones. Además, permite la adición de nuevos vehículos, la creación de clientes y usuarios. Este enfoque destaca la eficacia del diseño orientado a objetos para abordar la complejidad del dominio de una rentadora de autos, proporcionando una solución completa y eficiente para L.A. Motors.

Índice

1.	Introducción.....	1
1.1.	Clases y Objetos.....	1
1.2.	Herencia.....	2
1.3.	Listas.....	2
1.4.	Control de excepciones.....	2
1.5.	ASP.NET.....	3
1.6.	HTML.....	3
1.7.	CSS.....	4
2.	Metodología de estudio.....	4
2.1.	Diagrama UML.....	5
2.2.	Clases.....	5
2.2.1.	Clase BaseDeDatos.....	6
2.2.2.	Clase Vehículo.....	7
2.2.3.	Clase Moto.....	7
2.2.4.	Clase Auto.....	8
2.2.5.	Clase Camion.....	8
2.2.6.	Clase Cliente.....	8
2.2.7.	Clase Usuario.....	10
2.2.8.	Clase Venta.....	10
2.2.9.	Clase Alquiler.....	11
2.3.	HTML.....	11
2.3.1.	Ingreso de Datos tipo string.....	12
2.3.2.	Ingreso de Datos tipo numérico.....	12
2.3.3.	Ingreso de datos tipo DateTime.....	13
2.3.4.	Ingreso de Datos tipo Url.....	14
2.3.5.	Selección de tipo Vehículo.....	14
2.3.6.	Evento Guardar.....	15
2.4.	HTML Listado de Datos.....	16
2.4.1.	Configuración de GridView.....	18
2.4.2.	Campos de GridView.....	17
2.4.3.	Configuración de botones Editar/Eliminar.....	17
2.5.	Interfaz de Usuario.....	18
2.5.1.	Login.....	18
2.5.2.	Inicio.....	19
2.5.3.	Usuarios.....	19
2.5.4.	Clientes.....	21
2.5.5.	Vehículos.....	22
2.5.6.	Ventas.....	23
2.5.7.	Alquileres.....	24
3.	Resultados y discusión.....	26
3.1.	Login.....	26
3.2.	Inicio.....	27
3.3.	Usuarios.....	28
3.4.	Clientes.....	29
3.5.	Vehículos.....	31
3.6.	Ventas.....	33
3.7.	Alquileres.....	33
4.	Conclusiones.....	35
5.	Referencias.....	36

1. Introducción

La programación orientada a objetos (POO) se revela como un paradigma esencial en el tejido del desarrollo de software. Su fundamento reside en la creación y manipulación de clases y objetos. Las clases sirven como plantillas para la generación de objetos, instancias específicas de esas clases. Este paradigma se sumerge en la fascinante noción de herencia, donde las clases pueden heredar características y comportamientos, fomentando la reutilización del código y la construcción eficiente de estructuras.

Las listas, por su parte, emergen como herramientas cruciales en la gestión de colecciones de datos, permitiendo almacenar secuencias de elementos del mismo tipo y realizar operaciones fundamentales como agregar, eliminar y ordenar.

Aunque en este proyecto no se hicieron uso directo de los conceptos de agregación y composición, estos son elementos esenciales en la programación orientada a objetos, abriendo la puerta a relaciones más complejas entre objetos.

En el ámbito del desarrollo en C#, el control de excepciones se revela como un componente esencial para manejar situaciones imprevistas y fortalecer la robustez de los programas. La detección y el manejo de excepciones se llevan a cabo mediante la estructura try-catch.

En este proyecto, se integran diversos lenguajes de programación, destacando HTML, CSS y Javascript dentro del marco web ASP.NET, utilizado como plantilla para la aplicación web. HTML, un lenguaje de etiquetas, constituye la base donde se estructura el contenido en diversas formas. CSS entra en escena para dotar de diseño, color y funcionalidades a estas etiquetas, mientras que Javascript se encarga de la funcionalidad predeterminada de las mismas.

Dentro de ASP.NET, se hace uso de las bibliotecas de Bootstrap, enriqueciendo los lenguajes mencionados con componentes y estilos que mejoran la interfaz gráfica de la aplicación. Bootstrap facilita la creación de una interfaz atractiva y responsiva, ofreciendo herramientas que optimizan la experiencia del usuario en la aplicación web.

1.1. Clases y Objetos

Las clases y los objetos constituyen los pilares fundamentales de la Programación Orientada a Objetos (POO), y mantienen una estrecha interrelación. Los objetos son instancias creadas a partir de clases, y estas últimas funcionan como modelos o moldes para la creación de objetos. Comúnmente, a los objetos se les denomina instancia de plantilla.

Ahora, piensa en personas concretas: tú, tu madre, el presidente de tu país, figuras históricas como Alan Turing o Nelson Mandela, entre otros ejemplos. Estos individuos particulares son ejemplos de objetos. Cada uno de ellos se ajusta a los atributos definidos por la plantilla de la clase "persona", como el nombre, la edad, la estatura, y así sucesivamente.[1]

1.2. Herencia

La herencia se erige como uno de los conceptos esenciales en la Programación Orientada a Objetos, permitiendo la creación de una clase secundaria que puede reutilizar, extender o modificar el comportamiento de una clase primaria. La clase cuyos miembros son heredados se denomina "clase base", mientras que aquella que hereda los miembros de la clase base se conoce como "clase derivada".

Podemos ilustrar esta relación con una analogía. Imagina la palabra "persona". Cuando nos referimos a una "persona", no estamos haciendo referencia a una persona específica, sino que estamos abstrayendo la noción general de una entidad humana. Esta noción generalizada es lo que denominamos una clase.

Es importante destacar que en C# y .NET, se admite únicamente la herencia única, lo que significa que una clase sólo puede heredar de una única clase. Sin embargo, la herencia se muestra como un proceso transitivo, lo que habilita la construcción de una jerarquía de herencia para un conjunto de tipos. En otras palabras, un tipo D puede heredar del tipo C, el cual a su vez hereda del tipo B, y así sucesivamente, hasta llegar a la clase base A. Gracias a esta transitividad, los miembros del tipo A se encuentran disponibles para el tipo D.

No obstante, es importante destacar que no todos los miembros de una clase base son heredados por las clases derivadas. Los siguientes miembros no se heredan:

- Constructores estáticos, que tienen la responsabilidad de inicializar los datos estáticos de una clase.
- Constructores de instancias, que se invocan para crear una nueva instancia de la clase. Cada clase debe definir sus propios constructores según sus necesidades.
- Finalizadores, los cuales son llamados por el recolector de elementos no utilizados en tiempo de ejecución para destruir instancias de una clase.[2]

1.3. Listas

Las listas en C# forman parte de los tipos de datos nativos en el framework de .NET y nos permiten almacenar secuencias de variables. Por ejemplo, si deseamos gestionar un conjunto de nombres de personas, podemos utilizar una variable de tipo `List<string>`, que se corresponde con una lista de cadenas de texto en la sintaxis de C#.

```
List<string> personNames;
```

Esta estructura `List` en C# es fuertemente tipada, lo que significa que, como se muestra en el ejemplo, al crear una lista, debemos especificar el tipo de objetos que contendrá (ya sea cadenas de texto, números u otro tipo definido por el usuario).[3]

1.4. Control de excepciones

Las características de control de excepciones del lenguaje C# le ayudan a afrontar cualquier situación inesperada o excepcional que se produce cuando se ejecuta un programa. El control de excepciones usa las palabras clave "try", "catch" y "finally" para intentar realizar acciones que pueden no completarse correctamente, para controlar errores cuando decide que es razonable hacerlo y para limpiar recursos más adelante.

En muchos casos, una excepción la puede no producir un método al que el código ha llamado directamente, sino otro método más bajo en la pila de llamadas. Cuando se genera una excepción, CLR desenreda la pila, busca un método con un bloque "catch" para el tipo de excepción específico y ejecuta el primer bloque "catch" que encuentra. Si no encuentra ningún bloque "catch" adecuado en cualquier parte de la pila de llamadas, finalizará el proceso y mostrará un mensaje al usuario.[4]

1.5. ASP.NET

ASP.NET es un marco web gratuito para crear excelentes sitios web y aplicaciones web con HTML, CSS y JavaScript. También puede crear API web y usar tecnologías en tiempo real como Web Sockets.

ASP.NET ofrece tres marcos para crear aplicaciones web: Web Forms, ASP.NET MVC y ASP.NET Web Pages. Los tres marcos son estables y maduros, y se puede crear excelentes aplicaciones web con cualquiera de ellos. Independientemente del marco que elija, obtiene todas las ventajas y características de ASP.NET en todas partes.

Cada marco tiene como destino un estilo de desarrollo diferente. La que elija dependerá de una combinación de los recursos de programación (conocimientos, aptitudes y experiencia de desarrollo), el tipo de aplicación que está creando y el enfoque de desarrollo con el que se sienta cómodo.

Con ASP.NET Web Forms, puede crear sitios web dinámicos mediante un modelo conocido controlado por eventos y arrastrar y colocar. Una superficie diseño y cientos de controles y componentes le permiten crear rápidamente sitios potentes y sofisticados sitios controlados por IU con datos.

Los tres marcos de ASP.NET se basan en .NET Framework y comparten la funcionalidad principal de .NET y de ASP.NET. Por ejemplo, los tres marcos ofrecen un modelo de seguridad de inicio de sesión basado en la pertenencia, y los tres comparten las mismas instalaciones para administrar solicitudes, controlar sesiones, etc., que forman parte de la funcionalidad principal ASP.NET.[5]

1.6. HTML: Lenguaje de etiquetas de hipertexto

HTML (Lenguaje de Marcas de Hipertexto, del inglés HyperText Markup Language) es el componente más básico de la Web. Define el significado y la estructura del contenido web. Además de HTML, generalmente se utilizan otras tecnologías para describir la apariencia/presentación de una página web (CSS) o la funcionalidad/comportamiento (JavaScript).

"Hipertexto" hace referencia a los enlaces que conectan páginas web entre sí, ya sea dentro de un único sitio web o entre sitios web. Los enlaces son un aspecto fundamental de la Web. Al subir contenido a Internet y vincularlo a las páginas creadas por otras personas, te conviertes en un participante activo en la «World Wide Web» (Red Informática Mundial).

HTML utiliza "marcas" para etiquetar texto, imágenes y otro contenido para mostrarlo en un navegador Web. Las marcas HTML incluyen "elementos" especiales como <head>, <title>,

<body>, <header>, <footer>, <article>, <section>, <p>, <div>, , , <progress>, <video>, , , y muchos otros.

Un elemento HTML se distingue de otro texto en un documento mediante "etiquetas", que consisten en el nombre del elemento rodeado por "<" y ">". El nombre de un elemento dentro de una etiqueta no distingue entre mayúsculas y minúsculas. Es decir, se puede escribir en mayúsculas, minúsculas o una mezcla. Por ejemplo, la etiqueta <title> se puede escribir como <Title>, <TITLE> o de cualquier otra forma.[6]

1.7. CSS: Cascading Style Sheets

Hojas de Estilo en Cascada (del inglés Cascading Style Sheets) o CSS es el lenguaje de estilos utilizado para describir la presentación de documentos HTML o XML (en-US) (incluyendo varios lenguajes basados en XML como SVG, MathML o XHTML). CSS describe como debe ser renderizado el elemento estructurado en la pantalla, en papel, en el habla o en otros medios.

CSS es utilizado para diseñar y dar estilo a las páginas web, por ejemplo, alterando la fuente, color, tamaño y espaciado del contenido, dividirlo en múltiples columnas o agregar animaciones y otras características decorativas.[7]

2. Metodología de estudio

El desarrollo del programa en C# se sustenta en una robusta estructura de clases que eficientemente refleja el funcionamiento de una agencia de alquiler de vehículos. En el núcleo de esta arquitectura, destaca la clase central denominada "BaseDeDatos", desempeñando un papel crucial. Esta clase no incorpora atributos directos; en su lugar, encapsula listas responsables de contener información vital, como alquileres, ventas, vehículos, clientes y usuarios de la agencia. En consecuencia, esta clase integra métodos esenciales, como la abstracción de las listas para que sean accesibles en cualquier parte del proyecto y la carga inicial de datos para el manejo de información existente.

La funcionalidad del programa se apoya significativamente en las clases de Ventas y Alquileres. Inicialmente concebidas para facilitar el almacenamiento de datos específicos de cada transacción, incluyendo detalles, información del cliente y la matrícula del vehículo vendido/alquilado, estas clases se utilizan en la construcción de las páginas de Ventas y Alquileres. A través de formularios web, se permite al usuario realizar ventas/alquileres basándose en los clientes existentes y los vehículos disponibles. Posterior a la concreción de una transacción, se despliega una lista con todos los objetos "Alquiler" y "Venta" para el manejo de dicha información, brindando la posibilidad de editar y/o eliminar cualquier registro si es necesario.

El concepto de herencia se ha implementado en relación con los vehículos, destacando las clases "Auto", "Moto" y "Camión" como descendientes directos de la clase base "Vehículo". Esta jerarquía proporciona una representación estructurada y eficiente de los distintos tipos de vehículos disponibles para alquilar y vender, abarcando una variedad de opciones

acorde a las necesidades específicas de cada cliente. Se ha creado una página para gestionar los vehículos, permitiendo agregar, modificar y eliminar de la lista según sea necesario.

Además, se han desarrollado páginas para facilitar la creación y gestión de Clientes y Usuarios. Estas clases están programadas para permitir el manejo de sus atributos desde listas, obteniendo los datos principales necesarios para realizar ventas y alquileres. También se permite la edición y eliminación de estos datos según corresponda.

Enlace al repositorio del programa en GitHub:

[Obligatorio Programación 2 \(github.com\)](#) - [Enlace a Proyecto en Drive.](#)

2.1. Diagrama UML

El diagrama UML a continuación brinda una representación visual de la estructura de clases que define la dinámica del programa desarrollado. Este diseño ofrece claridad en las relaciones entre las entidades fundamentales que impulsan el sistema de gestión de renta y venta de vehículos.

En el núcleo de la jerarquía se encuentra la clase base "Vehículo," de la cual se derivan las clases especializadas "Auto," "Moto" y "Camión." La relación de herencia asegura que estas clases derivadas hereden las propiedades esenciales de la clase base, al mismo tiempo que permiten la introducción de atributos específicos para cada tipo de vehículo.

Una entidad clave es la clase "Cliente," que posibilita el registro de datos fundamentales, como el documento único del cliente y otros detalles relevantes. Este cliente será parte integral de una venta o un alquiler, estableciendo relaciones con estas clases a través de su atributo único.

A continuación, se encuentra la clase "Venta," que registra información relevante para llevar a cabo esta acción. Incluye un identificador único, el registro del costo, la fecha de la venta, el documento del cliente que realiza la compra y la matrícula única del vehículo vendido. Asimismo, posee un atributo para registrar al usuario responsable de la venta.

De manera similar, la clase "Alquiler" contiene detalles cruciales para el registro de alquileres. Al igual que la clase "Venta," incluye el documento del cliente y la matrícula del vehículo a alquilar, junto con la cantidad de días y la fecha de inicio del alquiler. Estos atributos permiten calcular la duración del alquiler y se relacionan con el estado del mismo, dependiendo de si el cliente devuelve el vehículo en la fecha acordada.

Otra entidad fundamental es la clase "Usuario," que contiene atributos identificativos y permisos para el acceso a distintas funcionalidades de la aplicación web, adaptados a cada usuario.

Finalmente, la clase que engloba todo el programa es la "BaseDeDatos," que alberga listas para contener objetos de las clases mencionadas anteriormente. A través de esta clase, se accede a los datos de cada entidad en todas las partes de la aplicación, y los datos ingresados mediante la interfaz gráfica se almacenan en las listas contenidas en esta clase central.

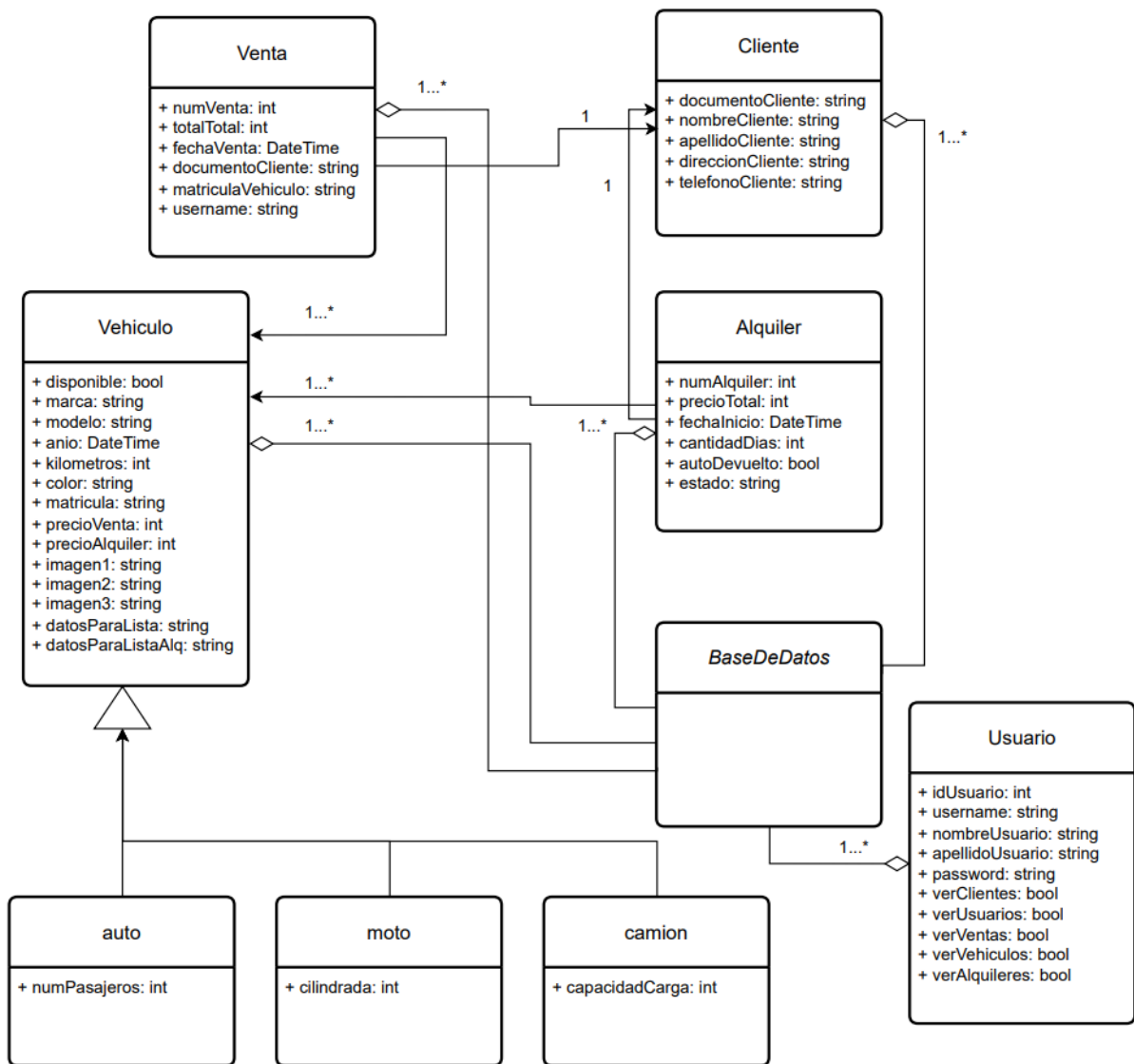


Figura 2.1.1: Diagrama UML.

2.2. Clases

Como se ve en el punto anterior, las clases son la estructura principal detrás de la aplicación web. Es por medio de estas entidades que se puede hacer uso de las funciones de la aplicación, tales como la venta y alquiler de vehículos. A continuación se encuentra cada clase a detalle y sus funciones principales.

2.2.1. Clase BaseDeDatos

La clase "BaseDeDatos" se erige como la piedra angular del programa, albergando las listas que contienen toda la información disponible en el sistema, tales como las listas de vehículos, alquileres, ventas, clientes y usuarios. Esta clase se presenta como abstracta y estática, lo que posibilita su accesibilidad desde cualquier parte del código.

Adicionalmente, se implementó la carga inicial de datos en esta clase, con el objetivo de dotar al programa de información preexistente para facilitar la gestión del sistema y la verificación de las funcionalidades.

```
public abstract class BaseDeDatos
{
    public static List<Vehiculo> listaVehiculos = new List<Vehiculo>();
    public static List<Usuario> listaUsuarios = new List<Usuario>();
    public static List<Cliente> listaClientes = new List<Cliente>();
    public static List<Venta> listaVentas = new List<Venta>();
    public static List<Alquiler> listaAlquileres = new List<Alquiler>();
}
```

Figura 2.2.1: Atributos de la clase “BaseDeDatos”.

2.2.2. Clase Vehículo

La clase “Vehículo” es padre de las subclases “Auto”, “Camion” y “Moto” y contiene la mayoría de los atributos comunes de dicha clase. También posee dos atributos que se componen de otros atributos ya existentes; estos se utilizan para mostrar al usuario información del vehículo en la interfaz gráfica.

```
public class Vehiculo
{
    15 referencias
    public bool disponible { get; set; } public string matricula { get; set; }
    7 referencias
    public string marca { get; set; } public string modelo { get; set; }
    3 referencias
    public int kilometros { get; set; } public DateTime anio { get; set; }
    5 referencias
    public string color { get; set; } public int precioVenta { get; set; }
    5 referencias
    public int precioAlquiler { get; set; } public string imagen1 { get; set; }
    3 referencias
    public string imagen2 { get; set; } public string imagen3 { get; set; }
    4 referencias
    public string datosParaLista
    {
        get { return $"{this.GetType().Name} - Matrícula: {matricula} - Marca:" +
            $" {marca} - Año: {anio.Year} - Precio: ${precioVenta}"; }
    }

    0 referencias
    public string datosParaListaAlq
    {
        get { return $"{this.GetType().Name} - Matrícula: {matricula} - Marca:" +
            $" {marca} - Año: {anio.Year} - Precio diario: ${precioAlquiler}"; }
    }
}
```

Figura 2.2.2: Atributos de la clase “Vehiculos”.

2.2.3. Clase Moto

La clase “Moto” es hija de “Vehículo” y hereda todos sus atributos adicionalmente al atributo Cilindrada que es particular de esta clase.

```

public class Moto : Vehiculo
{
    3 referencias
    public int cilindrada { get; set; }
    1 referencia
    public Moto() { }
    5 referencias
    public Moto(bool disponible, string matricula, string marca,
        string modelo, int kilometros, DateTime anio, string color,
        int precioVenta, int precioAlquiler, string imagen1,
        string imagen2, string imagen3, int cilindrada) :
        base(disponible, matricula, marca, modelo, kilometros,
        anio, color, precioVenta, precioAlquiler, imagen1, imagen2, imagen3)
    {
        this.cilindrada = cilindrada;
    }
    0 referencias
    public int getCilindrada() { return cilindrada; }
    1 referencia
    public void setCilindrada(int cilindrada) { this.cilindrada = cilindrada; }
}

```

Figura 2.2.3: Atributos y constructor de la clase “Moto”.

2.2.4. Clase Auto

La clase “Auto” es hija de “Vehículo” y hereda todos sus atributos adicionalmente al atributo numPasajeros que es particular de esta clase.

```

public class Auto : Vehiculo
{
    3 referencias
    public int numPasajeros { get; set; }
    1 referencia
    public Auto() { }
    5 referencias
    public Auto(bool disponible, string matricula,
        string marca, string modelo, int kilometros,
        DateTime anio, string color, int precioVenta,
        int precioAlquiler, string imagen1, string imagen2,
        string imagen3, int numPasajeros) :
        base(disponible, matricula, marca, modelo,
        kilometros, anio, color, precioVenta,
        precioAlquiler, imagen1, imagen2, imagen3)
    {
        this.numPasajeros = numPasajeros;
    }
    0 referencias
    public int getNumPasajeros() { return numPasajeros; }
    1 referencia
    public void setNumPasajeros(int numPasajeros) { this.numPasajeros = numPasajeros; }
}

```

Figura 2.2.4: Atributos y método constructor de la clase “Auto”.

2.2.5. Clase Camion

La clase “Camion” es hija de “Vehículo” y hereda todos sus atributos adicionalmente al atributo capacidadCarga que es particular de esta clase.

```

public class Camion : Vehiculo
{
    3 referencias
    public int capacidadCarga { get; set; }
    1 referencia
    public Camion() { }
    5 referencias
    public Camion(bool disponible, string matricula,
        string marca, string modelo, int kilometros,
        DateTime anio, string color, int precioVenta,
        int precioAlquiler, string imagen1, string imagen2,
        string imagen3, int capacidadCarga) :
        base(disponible, matricula, marca, modelo,
            kilometros, anio, color, precioVenta,
            precioAlquiler, imagen1, imagen2, imagen3)
    {
        this.capacidadCarga = capacidadCarga;
    }
    0 referencias
    public int getCapacidadCarga() { return capacidadCarga; }
    1 referencia
    public void setCapacidadCarga(int capacidadCarga) { this.capacidadCarga = capacidadCarga; }
}

```

Figura 2.2.5: Atributos y método constructor de la Clase "Camion".

2.2.6. Clase Cliente

La clase "Cliente" encapsula de manera integral la información correspondiente a cada cliente. Esta información se utiliza posteriormente durante la ejecución de operaciones como alquileres o ventas, quedando registrada de manera adecuada en los eventos asociados.

Además, la clase incorpora un atributo adicional denominado "datosClientes". Este atributo desempeña un papel crucial al presentar los datos del cliente en el formulario, facilitando la selección del cliente durante eventos de venta o alquiler.

```

public class Cliente
{
    11 referencias
    public string documentoCliente { get; set; }
    6 referencias
    public string nombreCliente { get; set; }
    6 referencias
    public string apellidoCliente { get; set; }
    3 referencias
    public string direccionCliente { get; set; }
    4 referencias
    public string telefonoCliente { get; set; }

    0 referencias
    public string datosCliente
    {
        get { return $"Documento: {documentoCliente} - " +
            $"Nombre y Apellido: {nombreCliente} {apellidoCliente}" +
            $" - Teléfono: {telefonoCliente}"; }
    }
}

```

Figura 2.2.6: Atributos de la clase "Cliente".

2.2.7. Clase Usuario

La clase "Usuario" almacena la información relativa a los usuarios que emplean el sistema. En este contexto, se destaca la presencia de un usuario precargado: "Usuario Admin" con la contraseña "Admin". Este usuario ostenta todos los permisos y posee la capacidad exclusiva de crear nuevos usuarios. Cabe señalar que los usuarios creados posteriormente no tienen la facultad de asignar permisos para la creación de usuarios.

En el proceso de cada venta y alquiler, se registra la información del usuario responsable de llevar a cabo dicho evento.

```
public class Usuario
{
    4 referencias
    public int idUsuario { get; set; } public string username { get; set; }
    3 referencias
    public string nombreUsuario { get; set; } public string apellidoUsuario { get; set; }
    3 referencias
    public string password { get; set; } public bool verClientes { get; set; }
    3 referencias
    public bool verUsuarios { get; set; } public bool verVentas { get; set; }
    3 referencias
    public bool verVehiculos { get; set; } public bool verAlquileres { get; set; }
```

Figura 2.2.7: Atributos de la clase "Usuario"

2.2.8. Clase Venta

La clase "Venta" desempeña un papel crucial al registrar una de las funciones principales de la aplicación. Sus atributos están cuidadosamente seleccionados para capturar la información esencial relacionada con una transacción de venta. Para construir una instancia de esta clase, se requiere la identificación del usuario responsable de la venta, la especificación del cliente que desea realizar la compra, identificado mediante su documento único, la matrícula del vehículo que se está vendiendo y el costo asociado, el cual se obtiene de los atributos del vehículo en cuestión. Además, la clase "Venta" registra la fecha exacta en que se lleva a cabo la transacción y el monto total involucrado en la venta, proporcionando así un seguimiento detallado de cada operación de venta realizada a través de la aplicación.

```
public class Venta
{
    4 referencias
    public int numVenta { get; set; }
    3 referencias
    public int totalVenta { get; set; }
    3 referencias
    public DateTime fechaVenta { get; set; }
    3 referencias
    public string documentoCliente { get; set; }
    3 referencias
    public string matriculaVehiculo { get; set; }
    3 referencias
    public string username { get; set; }
```

Figura 2.2.8: Atributos de la clase "Venta".

2.2.9. Clase Alquiler

La clase "Alquiler", al igual que "Ventas", desempeña un rol fundamental al gestionar una de las funciones principales de la aplicación. Sus atributos están cuidadosamente seleccionados para capturar información esencial relacionada con una transacción de alquiler. Para instanciar esta clase, se solicita la identificación del usuario responsable del alquiler, la especificación del cliente interesado, identificado mediante su documento único, la matrícula del vehículo objeto del alquiler y el costo asociado. Además, la clase "Alquiler" registra la cantidad de días solicitados para el alquiler y la fecha de inicio de la transacción, proporcionando un registro detallado de cada operación de alquiler realizada a través de la aplicación.

Adicionalmente, la clase "Alquiler" incluye dos atributos adicionales que son fundamentales para determinar la disponibilidad del vehículo. En el caso de que el vehículo sea alquilado y se devuelva antes de tiempo o atrasado, el atributo "estado" refleja esta condición en función de las fechas de alquiler, la fecha actual y la cantidad de días alquilados. Estos elementos proporcionan un mecanismo efectivo para gestionar y registrar el estado y la disponibilidad de los vehículos en el contexto de las operaciones de alquiler.

```
public class Alquiler
{
    4 referencias
    public int numAlquiler { get; set; } public int precioTotal { get; set; }
    4 referencias
    public DateTime fechaInicio { get; set; } public int cantidadDias { get; set; }
    0 referencias
    public string estado {
        get {
            if (!autoDevuelto && DateTime.Now > fechaInicio.AddDays(cantidadDias))
            { return "Atrasado"; }
            else if (!autoDevuelto)
            { return "Al día"; }
            else
            { return "Vehículo devuelto"; }
        }
    }
    5 referencias
    public bool autoDevuelto { get; set; }
    3 referencias
    public string documentoCliente { get; set; }
    3 referencias
    public string matriculaVehiculo { get; set; }
    3 referencias
    public string username { get; set; }
```

Figura 2.2.9: Atributos de la clase "Alquiler".

2.3. HTML: ingreso de datos

El sistema, desarrollado en ASP.NET, presenta como interfaz gráfica un sitio web que utiliza HTML, CSS y Bootstrap. Las páginas están diseñadas de manera intuitiva para facilitar la navegación del usuario, empleado de la automotora, a través de las distintas pestañas disponibles en la Barra de Navegación.

A continuación se detallan secciones claves del código HTML que logran la funcionalidad deseada:

2.3.1. Ingreso de datos tipo string

Para facilitar el ingreso de datos por parte del usuario a través de la interfaz, se emplean casillas de texto, donde se introducen los datos requeridos. En el desarrollo del programa, se ha utilizado principalmente el control "TextBox", que posibilita la entrada de datos tanto de tipo string como numéricos.

En un diseño de fila y columnas, la estructura "div class='row'" indica la creación de una fila en la interfaz gráfica. Dentro de esta fila, se encuentra otra estructura "div class='col-lg-5'", que representa una columna de tamaño grande (large) con un ancho equivalente a 5 columnas según el sistema de grillas de Bootstrap.

Dentro de esta columna, se encuentra un control de ASP.NET TextBox con el ID "txtMatricula". Este TextBox está diseñado para capturar la matrícula de un vehículo. Se le ha aplicado una clase de estilo "form-control" de Bootstrap para mejorar su presentación visual y se ha establecido un ancho específico de 250 píxeles.

Además, se ha incluido un validador de campo requerido ("RequiredFieldValidator") con el ID "rfvMatricula". Este validador está vinculado al TextBox "txtMatricula" y tiene como objetivo asegurar que se ingrese una matrícula. En caso de que el usuario intente enviar el formulario sin completar este campo, se mostrará un mensaje de error en color rojo.

```
<div class="row">
  <div class="col-lg-5">
    <asp:TextBox ID="txtMatricula" runat="server" CssClass="form-control"
      placeholder="Matrícula del Vehículo" Style="width: 250px;">
    </asp:TextBox>
    <asp:RequiredFieldValidator ID="rfvMatricula" runat="server"
      ControlToValidate="txtMatricula" ErrorMessage="Ingrese una matrícula"
      ForeColor="Red" Display="Dynamic" ValidationGroup="Guardar">
    </asp:RequiredFieldValidator>
  </div>
</div>
```

Figura 2.3.1: Formulario de matrícula con validación.

2.3.2. Ingreso de datos tipo numérico

Este fragmento de código HTML está diseñado para capturar y validar la entrada de datos relacionados con la cantidad de kilómetros de un vehículo. Utiliza un cuadro de texto (TextBox) con la capacidad de aceptar solo valores numéricos (TextMode="Number").

Además, se ha establecido un límite mínimo de 0 (min="0") para garantizar que se ingresen valores no negativos. Se proporciona un espacio de presentación claro y estilizado mediante las clases de estilo CSS "form-control" y se define un ancho específico de 250 píxeles.

Para validar la entrada del usuario, se ha incorporado un validador de campo requerido (RequiredFieldValidator) llamado "rfvKilometros", que mostrará un mensaje de error en color rojo si no se proporciona la información necesaria. Este componente se encuentra dentro de una estructura de filas y columnas (row y col-lg-5) para gestionar el diseño y la presentación.

```
<div class="row">
  <div class="col-lg-5">
    <asp:TextBox TextMode="Number" min="0" ID="txtKilometros"
      runat="server" CssClass="form-control" Style="width: 250px;"
      placeholder="Cantidad de Kilometros" ></asp:TextBox>
    <asp:RequiredFieldValidator ID="rfvKilometros" runat="server"
      ControlToValidate="txtKilometros"
      ErrorMessage="Ingrese los kilometros del vehículo"
      ForeColor="Red" Display="Dynamic" ValidationGroup="Guardar">
    </asp:RequiredFieldValidator>
  </div>
</div>
```

Figura 2.3.2: Formulario de kilómetros con validación.

2.3.3. Ingreso de datos tipo DateTime

Este bloque de código HTML se enfoca en la entrada y validación de la fecha de fabricación de un vehículo. Se presenta al usuario un encabezado de nivel 6 (h6) que indica la acción requerida: "Elija la fecha de fabricación del vehículo". A continuación, se proporciona un cuadro de texto (TextBox) con la capacidad de ingresar fechas (TextMode="Date"). Este componente, identificado como "txtAnio", utiliza una clase de estilo CSS ("form-control") para presentar una interfaz gráfica amigable y se define con un ancho específico de 250 píxeles para un diseño consistente.

Además, se ha integrado un validador de campo requerido (RequiredFieldValidator) denominado "rfvAnio" para asegurar que el usuario proporcione la información esencial. En caso de que no se ingrese el año del vehículo, este validador mostrará un mensaje de error en color rojo. El diseño del formulario se organiza mediante una estructura de filas y columnas (row y col-lg-5), contribuyendo a una presentación ordenada y clara.

```
<h6>Elija la fecha de fabricación del vehículo</h6>
<div class="row">
  <div class="col-lg-5">
    <asp:TextBox TextMode="Date" ID="txtAnio" runat="server"
      CssClass="form-control" placeholder="Año del Vehículo"
      Style="width: 250px;"></asp:TextBox>
    <asp:RequiredFieldValidator ID="rfvAnio" runat="server"
      ControlToValidate="txtAnio" ErrorMessage="Ingrese el año del vehiculo"
      ForeColor="Red" Display="Dynamic" ValidationGroup="Guardar">
    </asp:RequiredFieldValidator>
  </div>
</div>
```

Figura 2.3.3: Formulario de fecha con validación.

2.3.4. Ingreso de datos tipo Url

Este bloque de código HTML se centra en la entrada y validación de la URL de la primera imagen asociada a un vehículo. Se presenta al usuario un cuadro de texto (TextBox) con la capacidad de ingresar URLs de imágenes (TextMode="Url"). Este componente, identificado como "txtimagen1", utiliza una clase de estilo CSS ("form-control") para presentar una interfaz gráfica amigable y se define con un ancho específico de 250 píxeles para un diseño consistente.

Además, se ha integrado un validador de campo requerido (RequiredFieldValidator) que se llama "rfvImagen1". Este validador asegura que el usuario proporcione la URL de la imagen. En caso de que no se ingrese la URL de la primera imagen del vehículo, el validador mostrará un mensaje de error en color rojo. El diseño del formulario se organiza mediante una estructura de filas y columnas (row y col-lg-5), contribuyendo a una presentación ordenada y clara.

```
<div class="row">
  <div class="col-lg-5">
    <asp:TextBox ID="txtimagen1" TextMode="Url" runat="server"
      CssClass="form-control" placeholder="Imagen 1 del Vehículo"
      Style="width: 250px;"></asp:TextBox>
    <asp:RequiredFieldValidator ControlToValidate="txtimagen1"
      runat="server" ErrorMessage="Debe ingresar 3 imagenes"
      ForeColor="Red" Display="Dynamic" ValidationGroup="Guardar">
    </asp:RequiredFieldValidator>
  </div>
</div>
```

Figura 2.3.4: Ingreso y validación de la URL de imágenes.

2.3.5. Selección del tipo de vehículo

Este fragmento de código implementa un RadioButtonList en la interfaz web, proporcionando al usuario la capacidad de elegir el tipo de vehículo entre Moto, Auto y Camión. Se utiliza la propiedad AutoPostBack para habilitar la ejecución de un evento de servidor, en este caso, el método rblTipoVehiculo_SelectedIndexChanged. Este evento se activa automáticamente cuando el usuario cambia la selección. La interfaz utiliza estilos personalizados (customRadioButtonList) para mejorar la presentación visual. Esta funcionalidad es crucial para el ingreso y clasificación de vehículos en el sistema.

```

<div class="row" id="tipoVehiculo">
    <div class="col-lg-5">
        <asp:RadioButtonList CssClass="customRadioButtonList"
            ID="rblTipoVehiculo" runat="server"
            OnSelectedIndexChanged="rblTipoVehiculo_SelectedIndexChanged"
            AutoPostBack="true">
            <asp:ListItem Value="Moto" Selected="True">Moto</asp:ListItem>
            <asp:ListItem Value="Auto">Auto</asp:ListItem>
            <asp:ListItem Value="Camión">Camion</asp:ListItem>
        </asp:RadioButtonList>
    </div>
</div>

```

Figura 2.3.5: Selector de tipo de vehículo para ingreso en el sistema.

2.3.6. Evento Guardar

Este código implementa un botón de guardado en una interfaz de usuario ASP.NET. El botón, identificado como "btnGuardar", está diseñado para activar la lógica de código asociada cuando se hace clic en él. Al hacer clic en este botón, se ejecutará el evento "btnGuardar_Click", el cual contiene la lógica para procesar y almacenar la información ingresada en la interfaz, según el contexto de la aplicación web.

```

<div class="row">
    <div class="col-lg-5">
        <asp:Button ID="btnGuardar" runat="server"
            CssClass="btn btn-primary" Text="Guardar"
            OnClick="btnGuardar_Click" ValidationGroup="Guardar" />
    </div>
</div>

```

Figura 2.11.6: Implementación del botón de guardado en la interfaz de usuario ASP.NET.

Evento de Guardado para la Interfaz de Usuario ASP.NET: Este código representa la lógica asociada al botón de 'Guardar' en la interfaz de usuario. Se encarga de validar la matrícula del vehículo, crear instancias de las clases de vehículos (Moto, Auto, Camión) según la selección del usuario, asignar valores a sus atributos correspondientes y agregarlos a la lista de vehículos. Además, actualiza la interfaz de usuario (GridView) para reflejar los cambios y limpia los campos de entrada después de completar el proceso de guardado.

```

protected void btnGuardar_Click(object sender, EventArgs e)
{
    bool matriculaCorrecta = true;
    foreach (Vehiculo vehiculos in BaseDeDatos.listaVehiculos)
    {
        if (vehiculos.matricula == txtMatricula.Text)
        {
            lblError.Visible = true;
            txtMatricula.Text = string.Empty;
            matriculaCorrecta = false;
        }
    }
}

```

Figura 2.11.6.1: Logica del botón de guardado en backend.

```

if (rblTipoVehiculo.SelectedItem.Value == "Moto" && matriculaCorrecta == true)
{
    Moto vehiculo = new Moto();
    vehiculo.setDisponible(true);
    vehiculo.setMatricula(txtMatricula.Text);
    vehiculo.setMarca(txtMarca.Text);
    vehiculo.setModelo(txtModelo.Text);
    int kilometros = Convert.ToInt32(txtKilometros.Text);
    vehiculo.setKilometros(kilometros);
    vehiculo.setColor(txtColor.Text);
    int precioventa = Convert.ToInt32(txtPrecioVenta.Text);
    vehiculo.setPrecioVenta(precioventa);
    int precioAlquiler = Convert.ToInt32(txtPrecioAlquiler.Text);
    vehiculo.setPrecioAlquiler(precioAlquiler);
    vehiculo.setImagen1(txtimagen1.Text);
    vehiculo.setImagen2(txtimagen2.Text);
    vehiculo.setImagen3(txtimagen3.Text);
    int cilindrada = Convert.ToInt32(txtCilindradas.Text);
    vehiculo.setCilindrada(cilindrada);

    BaseDeDatos.listaVehiculos.Add(vehiculo);
    this.gvVehiculos.DataSource = BaseDeDatos.listaVehiculos;
    this.gvVehiculos.DataBind();
}

```

Figura 2.11.6.2: Ejemplo de creación de objeto "Moto" en botón Guardar.

2.4. HTML: listado de datos

Además de la entrada de datos, la información se muestra en cada pantalla en forma de listas GridView, permitiendo la modificación y eliminación de registros según sea necesario.

2.4.1. Configuración de GridView

Este código implementa un GridView en ASP.NET (denominado "gvVehiculos") que presenta una interfaz tabular para visualizar y gestionar datos de vehículos. El GridView tiene propiedades y eventos configurados para permitir operaciones como la edición, eliminación y actualización de registros.

Los datos se vinculan a través de la clave primaria "Matricula", y el diseño visual incluye bordes sólidos para mejorar la presentación y comprensión de la información. Además, se ha implementado un manejo de eventos específicos para cancelar ediciones, eliminar registros y realizar actualizaciones en la interfaz.

```

<div class="row">
  <div class="col-lg-8">
    <asp:GridView ID="gvVehiculos" runat="server" Width="100%"
      BorderWidth="2px" CellSpacing="5" ItemStyle-BorderStyle="Solid"
      ItemStyle-HorizontalAlign="Center" HeaderStyle-BorderWidth="2px"
      HeaderStyle-BorderStyle="Solid" HeaderStyle-HorizontalAlign="Center"
      OnRowCancelingEdit="gvVehiculos_RowCancelingEdit"
      OnRowDeleting="gvVehiculos_RowDeleting"
      OnRowEditing="gvVehiculos_RowEditing"
      OnRowUpdating="gvVehiculos_RowUpdating"
      AutoGenerateColumns="false"
      OnRowDataBound="gvVehiculos_RowDataBound"
      DataKeyNames="Matricula">

```

Figura 2.4.1: .Implementación y Gestión de Datos con GridView en ASP.NET.

2.4.2. Campos en GridView

Este código corresponde a un campo de plantilla utilizado en un GridView de ASP.NET para mostrar y editar la matrícula de vehículos. En el modo de visualización, se presenta como un Label que muestra la matrícula actual. En el modo de edición, se utiliza también un Label, permitiendo la modificación de la matrícula directamente en la celda correspondiente. Este elemento es parte de una interfaz de usuario que facilita la gestión de datos relacionados con vehículos en el contexto de una aplicación web desarrollada con ASP.NET.

```

<asp:TemplateField HeaderText="Matrícula" ItemStyle-BorderWidth="2px"
  ItemStyle-BorderStyle="Solid" ItemStyle-HorizontalAlign="Center"
  HeaderStyle-BorderWidth="2px" HeaderStyle-BorderStyle="Solid"
  HeaderStyle-HorizontalAlign="Center">
  <ItemTemplate>
    <asp:Label ID="lbl1" runat="server" Text='<%# Bind("matricula") %>'></asp:Label>
  </ItemTemplate>
  <EditItemTemplate>
    <asp:Label ID="lbl2" runat="server" Text='<%# Bind("matricula") %>'></asp:Label>
  </EditItemTemplate>
</asp:TemplateField>

```

Figura 2.4.2: .Campo de Plantilla para la Matrícula en un GridView de ASP.NET.

2.4.3. Configuración de botones Editar/Eliminar

Este fragmento de código representa dos campos de plantilla en un control GridView de ASP.NET. El primer campo, 'Editar', proporciona botones para iniciar la edición de un registro y para actualizar o cancelar los cambios realizados. El segundo campo, 'Eliminar', presenta un botón para eliminar el registro correspondiente en la GridView. Estos campos facilitan las operaciones de edición y eliminación dentro de la interfaz de usuario asociada al control GridView.

```

<asp:TemplateField HeaderText="Editar" ItemStyle-BorderWidth="2px"
ItemStyle-BorderStyle="Solid" ItemStyle-HorizontalAlign="Center"
HeaderStyle-BorderWidth="2px" HeaderStyle-BorderStyle="Solid"
HeaderStyle-HorizontalAlign="Center">
<ItemTemplate>
<asp:LinkButton CssClass="btn btn-primary" ID="lnkEdit" runat="server"
CausesValidation="False" CommandName="Edit" Text="Editar"></asp:LinkButton>
</ItemTemplate>
<EditItemTemplate>
<asp:LinkButton CssClass="btn btn-primary" ID="lnkUpdate" runat="server"
CausesValidation="True" CommandName="Update" Text="Actualizar"></asp:LinkButton>
<asp:LinkButton CssClass="btn btn-secondary" ID="lnkCancel" runat="server"
CausesValidation="False" CommandName="Cancel" Text="Cancelar"></asp:LinkButton>
</EditItemTemplate>
</asp:TemplateField>

<asp:TemplateField HeaderText="Eliminar" ItemStyle-BorderWidth="2px"
ItemStyle-BorderStyle="Solid" ItemStyle-HorizontalAlign="Center"
HeaderStyle-BorderWidth="2px" HeaderStyle-BorderStyle="Solid"
HeaderStyle-HorizontalAlign="Center">
<ItemTemplate>
<asp:LinkButton CssClass="btn btn-dark" ID="lnkDelete" runat="server"
CausesValidation="False" CommandName="Delete" Text="Eliminar"></asp:LinkButton>
</ItemTemplate>
</asp:TemplateField>

```

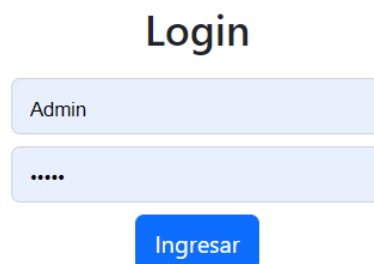
Figura 2.4.3: Campos de Plantilla para Edición y Eliminación en GridView.

2.5. Interfaz de usuario

A continuación se detallarán todas las pantallas de la interfaz de usuario con sus funcionalidades.

2.5.1. Login

El sistema cuenta con una interfaz de usuario diseñada para garantizar un acceso seguro y controlado. Para ingresar al sistema, se utiliza una pantalla de inicio de sesión que solicita al usuario ingresar su nombre de usuario y contraseña. Se ha establecido un usuario predeterminado con privilegios administrativos, con las credenciales "Default Admin - Admin", para facilitar el acceso inicial al sistema. Este usuario proporciona la capacidad de ingresar y explorar las funcionalidades del sistema de manera inmediata.



The image shows a login interface. At the top, the word "Login" is centered. Below it, there are two light blue input fields. The first field contains the text "Admin". The second field contains six dots ".....". Below these fields is a blue button with the text "Ingresar" in white.

Figura 2.5.1: Pantalla de ingreso al sistema.

2.5.2. Inicio

La pantalla de inicio presenta una barra de navegación que facilita el acceso a todas las funciones desarrolladas, siempre y cuando el usuario cuente con los permisos necesarios. Además, se destacan las cuatro principales funcionalidades del programa: Ventas, Alquileres, Clientes y Vehículos. Cada una de ellas cuenta con un botón correspondiente que redirige al usuario al área respectiva. La visibilidad de estos botones está condicionada por los permisos asignados a cada usuario.

Por último sobre el marco superior derecho está la opción de Cerrar Sesión para poder ingresar con otro usuario.

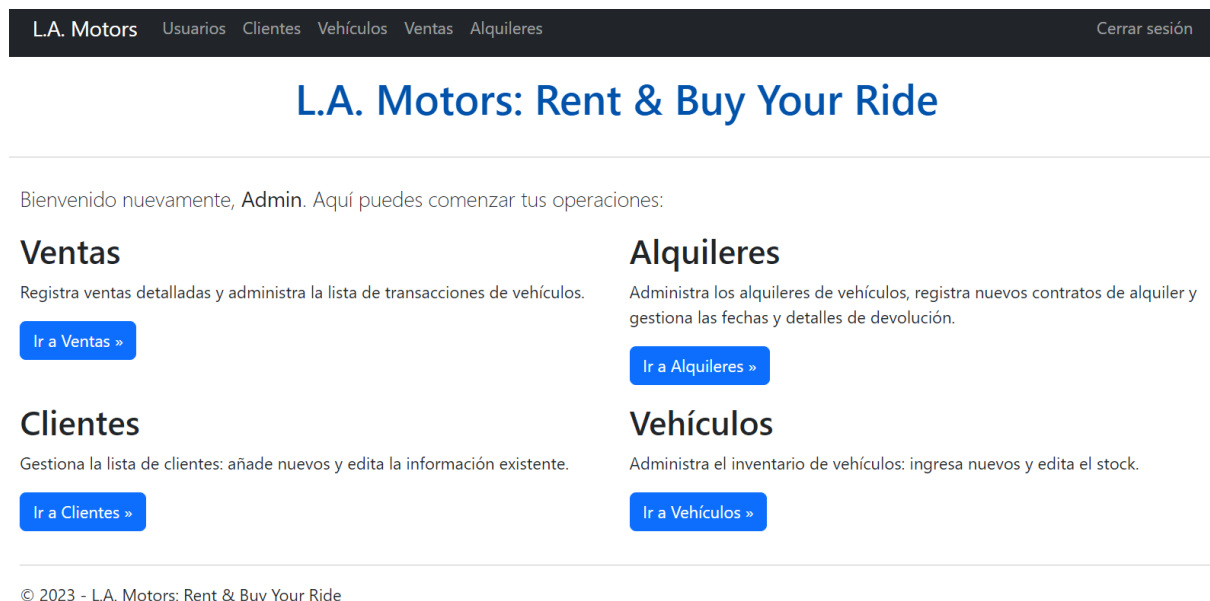


Figura 2.5.2: Pantalla de Inicio del sistema.

2.5.3. Usuarios

La página de Usuario reviste una importancia particular, ya que solo es accesible por el administrador, el usuario "Admin". Desde esta interfaz, el administrador puede registrar otros usuarios, asignándoles atributos específicos y otorgándoles permisos para diversas funciones dentro de la aplicación web. Una vez ingresados los datos del nuevo usuario y configurados sus permisos, se puede guardar la información en la lista correspondiente.

Esta pantalla también proporciona una vista de la lista de usuarios ya creados. Permite la edición de la información de cada usuario, con la excepción de su atributo único, el "username". Además, ofrece la posibilidad de ajustar los permisos asignados a cada usuario previamente registrado.

Agregar un nuevo Usuario

Username

Nombre del Usuario

Apellido del Usuario

Contraseña

Confirmar Contraseña

Permisos:

- ☐ Ver Clientes
- ☐ Ver Ventas
- ☐ Ver Vehículos
- ☐ Ver Alquileres

Guardar

Figura 2.5.3: Registro de un Usuario.

Lista de Usuarios

Número ID Usuario	Username	Nombre	Apellido	Ver Clientes	Ver Ventas	Ver Vehículos	Ver Alquileres	Editar	Eliminar
1	Admin	Admin	Admin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Editar	Eliminar

Figura 2.5.4: Listado de usuarios.

Lista de Usuarios

Número ID Usuario	Username	Nombre	Apellido	Ver Clientes	Ver Ventas	Ver Vehículos	Ver Alquileres	Editar	Eliminar
1	Admin	<input type="text"/>	<input type="text"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Actualizar Cancelar	Eliminar

Figura 2.5.5: Edición de un Usuario.

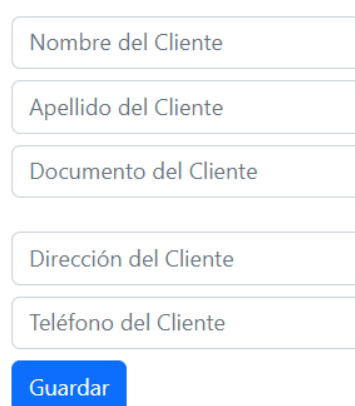
Se establece una restricción para que únicamente el administrador de la aplicación web tenga acceso a esta página. Esta medida se implementa para prevenir posibles inconvenientes derivados del acceso de usuarios no autorizados a funciones críticas. De esta manera, una vez que se crea un usuario, este puede acceder a la página con su "username" y la contraseña establecida en el momento de su registro.

2.5.4. Clientes

La sección de clientes brinda la posibilidad de añadir nuevos registros ingresando información clave como nombre, apellido, documento, dirección y teléfono. Se implementa un control para evitar la inserción de clientes duplicados, basado en el número de documento, y se verifica la validez de los documentos uruguayos ingresados.

Adicionalmente, se incorporan controles que impiden la inserción de clientes si faltan datos o si se presentan errores en los tipos de datos. Al final de la página, se muestra una lista completa de los clientes activos con sus respectivos detalles. Esta sección permite editar los registros existentes y eliminarlos según sea necesario.

Agregar un nuevo Cliente



Formulario para agregar un nuevo cliente. El formulario contiene cinco campos de texto con el siguiente orden de arriba hacia abajo: 'Nombre del Cliente', 'Apellido del Cliente', 'Documento del Cliente', 'Dirección del Cliente' y 'Teléfono del Cliente'. Debajo de estos campos hay un botón azul con el texto 'Guardar'.

Figura 2.5.6: Registro de un Cliente

Al final de la página, se muestra una lista completa de los clientes activos con sus respectivos detalles. Esta sección permite editar los registros existentes y eliminarlos según sea necesario.

Lista de Clientes

Documento	Nombre	Apellido	Dirección	Teléfono	Editar	Eliminar
45866580	Juan	Perez	dr Edye 3456	098765531	Editar	Eliminar
45899989	Javier	Lopez	dr Edye 5585	098509731	Editar	Eliminar
37854682	Luis	Gomez	Aigua 3588	098712313	Editar	Eliminar

Figura 2.5.7: Listado de Clientes.

Lista de Clientes

Documento	Nombre	Apellido	Dirección	Teléfono	Editar	Eliminar
45866580	<input type="text" value="Juan"/>	<input type="text" value="Perez"/>	<input type="text" value="dr Edye 3456"/>	<input type="text" value="098765531"/>	<input type="button" value="Actualizar"/> <input type="button" value="Cancelar"/>	<input type="button" value="Eliminar"/>
45899989	Javier	Lopez	dr Edye 5585	098509731	<input type="button" value="Editar"/>	<input type="button" value="Eliminar"/>
37854682	Luis	Gomez	Aigua 3588	098712313	<input type="button" value="Editar"/>	<input type="button" value="Eliminar"/>

Figura 2.5.8: Edición de Clientes.

2.5.5. Vehículos

La página de Vehículos desempeña un papel crucial en la funcionalidad de la aplicación, ya que habilita al usuario para registrar nuevos vehículos con detalles específicos. En esta pantalla, el usuario tiene la capacidad de agregar imágenes del vehículo y seleccionar su tipo entre las tres categorías disponibles: Auto, Moto o Camión. Posteriormente, puede agregar características únicas según el tipo de vehículo y guardar la información en la lista correspondiente.

Agregar un nuevo Vehículo


<input type="text" value="Matrícula del Vehículo"/>	<input type="text" value="Imagen 3 del Vehículo"/>
<input type="text" value="Marca del Vehículo"/>	<input type="text" value="Precio de venta"/>
<input type="text" value="Modelo del Vehículo"/>	<input type="text" value="Precio de alquiler"/>
<input type="text" value="Cantidad de Kilometros"/>	<input checked="" type="radio"/> Moto
Elija la fecha de fabricación del vehículo	<input type="radio"/> Auto
<input type="text" value="dd/mm/aaaa"/> 	<input type="radio"/> Camion
<input type="text" value="Color del Vehículo"/>	<input type="text" value="Cilindradas del Vehículo"/>
<input type="text" value="Imagen 1 del Vehículo"/>	<input type="button" value="Guardar"/>
<input type="text" value="Imagen 2 del Vehículo"/>	

Figura 2.5.9: Registro de un Vehículo.

La Base de Datos ya incluye vehículos registrados, los cuales se presentan en una grilla visible en la pantalla. Dentro de esta grilla, se ofrece la posibilidad de editar los datos e imágenes de cada vehículo, así como eliminarlos de la lista si es necesario.

Matrícula	Marca	Modelo	Kilometraje	Año	Color	Precio Venta	Precio Alquiler
XYZ789	Mercedes-Benz	Actros	60000	2019	Azul	55000	750
DEF456	Scania	R-Series	80000	2021	Blanco	65000	900

Figura 2.5.10: Listado de vehículos.

Imagen 1	Imagen 2	Imagen 3	Editar	Eliminar
			Editar	Eliminar
			Editar	Eliminar

Figura 2.5.11: Listado de vehículos y botones.

Esta interfaz permite la gestión completa del stock de vehículos almacenados en la Base de Datos. Estos vehículos pueden estar disponibles para alquiler o venta en las páginas correspondientes. En el caso de las ventas, al concretarse, los vehículos se eliminan automáticamente del stock y, por ende, de la lista visualizada en esta página.

2.5.6. Ventas

En la página de Ventas, se llevan a cabo transacciones de vehículos al elegir previamente al cliente y seleccionar el vehículo a ser vendido. La lista de vehículos se presenta de manera ordenada según el tipo de objeto, y se incluyen la mayoría de los atributos identificatorios para proporcionar información detallada sobre el vehículo seleccionado.

Realizar la venta de un vehículo

Clientes:

Documento: 45866580 - Nombre y Apellido: Juan Perez - Teléfono: 098765531
Documento: 45899989 - Nombre y Apellido: Javier Lopez - Teléfono: 098509731
Documento: 37854682 - Nombre y Apellido: Luis Gomez - Teléfono: 098712313

Vehiculos:

Auto - Matrícula: ASD7892 - Marca: Chevrolet - Año: 2022 - Precio: \$26000

Vender

Figura 2.5.12: Registro de Ventas.

Al final, se muestra el listado de vehículos vendidos con información detallada, incluyendo datos como el modelo del vehículo, el documento del cliente y el nombre del cliente. Además, se ofrece la posibilidad de eliminar registros según sea necesario. No se permite editar en el entendido de que los datos existentes de vehículos no son modificables.

Lista de vehículos vendidos

Número de Venta	Matrícula	Marca	Modelo	Color	Año	Precio Venta	Documento Cliente	Nombre y Apellido Cliente	Eliminar
1	M123	Honda	CBR600RR	Rojo	2022	\$12000	45866580	Juan Perez	Eliminar
2	FFG1237	Nissan	Altima	Verde	2018	\$20000	45899989	Javier Lopez	Eliminar

Figura 2.5.13: Listado de Clientes.

2.5.7. Alquileres

La página de alquileres posibilita registrar nuevos alquileres mediante la selección de un cliente existente y la elección de un vehículo de la lista disponible. Se solicita también la especificación de la duración del alquiler y la fecha de inicio.

Realizar el alquiler de un vehículo

Cientes:

Documento: 45866580 - Nombre y Apellido: Juan Perez - Teléfono: 098765531

Documento: 45899989 - Nombre y Apellido: Javier Lopez - Teléfono: 098509731

Documento: 37854682 - Nombre y Apellido: Luis Gomez - Teléfono: 098712313

Vehiculos:

Auto - Matrícula: ASD7892 - Marca: Chevrolet - Año: 2022 - Precio diario: \$2600

Cantidad de días de alquiler

Indique la fecha de inicio del Alquiler

dd/mm/aaaa

Alquilar

Figura 2.5.14: Registro de Alquileres.

En el listado de vehículos disponibles para alquiler, se proporciona información detallada sobre cada alquiler, incluyendo datos específicos. Además, se ofrece la opción de dejar disponible un registro de alquiler si el vehículo es devuelto antes de la fecha prevista, señalando si el alquiler está al día o atrasado. No se permite la edición de datos del alquiler ya que estos son inherentes a cada vehículo.

Lista de vehículos alquilados

Número de Alquiler	Matrícula	Marca	Modelo	Color	Año	Cantidad de días	Fecha de Inicio	Precio Alquiler
1	ABC123	Volvo	VNL	Rojo	2020	3	10/12/2023	\$2400
2	GHI789	Kenworth	T680	Negro	2022	10	09/12/2023	\$8500

Figura 2.5.15: Listado de Alquileres.

Documento Cliente	Nombre y Apellido Cliente	Devuelto	Estado	Editar	Eliminar
37854682	Luis Gomez	<input type="checkbox"/>	Al día	Editar	Eliminar
45899989	Javier Lopez	<input type="checkbox"/>	Al día	Editar	Eliminar

Figura 2.5.16: Listado de Alquileres y botones.

3. Resultados y discusión

A continuación se detalla el resultado de este minucioso proyecto que ha dado lugar a un programa robusto y funcional que ha superado las expectativas iniciales. Todos los requisitos previamente establecidos han sido implementados exitosamente, destacando especialmente las operaciones de ventas y alquileres, las cuales han sido desarrolladas de manera pragmática y eficiente. Además, se ha logrado implementar de manera efectiva las funcionalidades de las listas, proporcionando la información esencial sobre ventas, alquileres, vehículos, clientes y usuarios.

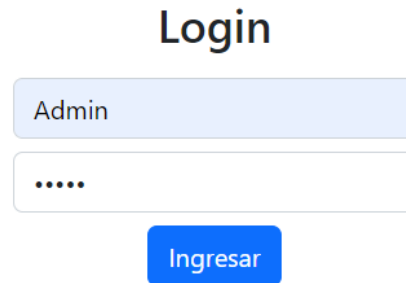
A pesar de estos logros, se identificó una limitación en la funcionalidad asociada a los usuarios. En particular, se observó que el Usuario Admin, quien posee permisos para acceder a todas las páginas, tiene la capacidad de editarse a sí mismo y eliminar sus propios permisos. Esta situación resulta incoherente con el propósito original de un usuario administrador y podría plantear riesgos de seguridad. Este aspecto podría ser abordado en futuras iteraciones del proyecto para garantizar una gestión más coherente y segura de los permisos de usuario.

Además, durante la evaluación del proyecto, se ha identificado una oportunidad de mejora que consiste en la implementación de la funcionalidad para descargar las listas en formatos como Excel, archivos de texto u otros. Esta característica adicional podría proporcionar a los usuarios una mayor versatilidad al gestionar y analizar la información generada por el sistema. Es importante destacar que, debido a limitaciones de tiempo, esta funcionalidad no fue implementada en la versión actual del proyecto, pero se considera una valiosa adición para futuras iteraciones.

3.1. Login

La página de login proporciona una experiencia inicial intuitiva y minimalista para los usuarios. Su diseño elegante cumple de manera efectiva con su función primaria al permitir el ingreso seguro al programa. La interfaz es amigable, asegurando que solo se permita el

acceso con credenciales válidas, mejorando así la seguridad del sistema. Este enfoque minimalista contribuye a una experiencia de usuario positiva al simplificar el proceso de inicio de sesión. La página de login establece una sólida primera impresión y sienta las bases para una navegación fluida en la aplicación.



The image shows a login interface with the title "Login" centered at the top. Below the title, there are two input fields: the first is a light blue box containing the text "Admin", and the second is a white box with five dots representing a password. Below these fields is a blue button with the text "Ingresar" in white.

Figura 3.1: Login con datos de administrador.

3.2. Inicio

La página de inicio proporciona una visión clara de la identidad del programa, dando la bienvenida al usuario con información relevante sobre la empresa. La disposición de botones de fácil acceso facilita la navegación, permitiendo a los usuarios dirigirse directamente a las áreas específicas del programa que desean utilizar. La inclusión de descripciones breves junto a los botones proporciona una guía adicional, ayudando a los usuarios a comprender las funciones de cada sección.

La barra de navegación superior, discreta y elegante, ofrece una alternativa eficiente para la exploración de las diferentes partes del programa. La presencia de un botón de cierre de sesión accesible en todo momento agrega comodidad y seguridad al permitir que los usuarios finalicen su sesión fácilmente.

Es importante destacar que la página de inicio adapta dinámicamente el contenido y los enlaces disponibles según los permisos del usuario, asegurando un acceso coherente y seguro a las funcionalidades específicas permitidas para cada usuario. Esta capacidad de personalización contribuye significativamente a la usabilidad y seguridad general del programa.

L.A. Motors: Rent & Buy Your Ride

Bienvenido nuevamente, **Admin**. Aquí puedes comenzar tus operaciones:

Ventas

Registra ventas detalladas y administra la lista de transacciones de vehículos.

[Ir a Ventas »](#)

Alquileres

Administra los alquileres de vehículos, registra nuevos contratos de alquiler y gestiona las fechas y detalles de devolución.

[Ir a Alquileres »](#)

Clientes

Gestiona la lista de clientes: añade nuevos y edita la información existente.

[Ir a Clientes »](#)

Vehículos

Administra el inventario de vehículos: ingresa nuevos y edita el stock.

[Ir a Vehículos »](#)

© 2023 - L.A. Motors: Rent & Buy Your Ride

Figura 3.2: Página de Inicio.

3.3. Usuarios

La página de Usuarios desempeña un papel crucial en la administración del sistema, siendo exclusivamente accesible para el administrador, usuario con el rol "Admin". Desde esta página, el administrador tiene la capacidad de registrar nuevos usuarios, asignarles atributos específicos y definir los permisos necesarios para acceder a diversas funciones de la aplicación web.

La interfaz de la página de Usuarios es intuitiva y permite al administrador ingresar los datos necesarios para la creación de un nuevo usuario. Además, se brinda la posibilidad de visualizar una lista completa de usuarios registrados, con opciones para editar la información, excluyendo el atributo único, el "username". La capacidad de editar los permisos de cada usuario ya registrado proporciona flexibilidad y control sobre el acceso a las funciones del sistema.

La limitación del acceso a esta página exclusivamente al administrador se establece para prevenir posibles inconvenientes derivados de la manipulación de permisos por parte de usuarios no autorizados. Una vez creado un usuario, este puede acceder a la aplicación

web utilizando su "username" y la contraseña establecida durante el registro, contribuyendo así a la seguridad integral del sistema.

L.A. Motors

Usuarios

Cientes

Vehículos

Ventas

Alquileres

Cerrar sesión

Agregar un nuevo Usuario

Username

Nombre del Usuario

Apellido del Usuario

Contraseña

Confirmar Contraseña

Permisos:

☐ Ver Cientes

☐ Ver Ventas

☐ Ver Vehículos

☐ Ver Alquileres

Guardar

Lista de Usuarios

Número ID Usuario	Username	Nombre	Apellido	Ver Cientes	Ver Ventas	Ver Vehículos	Ver Alquileres	Editar	Eliminar
1	Admin	Admin	Admin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Editar	Eliminar

Figura 3.3: Página de Usuarios.

3.4. Clientes

La página de Clientes en el sistema de gestión de renta y venta de vehículos desempeña un papel fundamental en el registro y administración de información sobre los clientes. Desde esta interfaz, los usuarios autorizados pueden ingresar datos esenciales de los clientes, como nombre, apellido, dirección y número de teléfono, creando perfiles detallados que serán vinculados a transacciones de ventas o alquileres.

La página proporciona una experiencia de usuario intuitiva, permitiendo el registro sencillo de nuevos clientes y la visualización clara de una lista completa de clientes existentes. La capacidad de editar la información de los clientes, excepto su atributo único, el documento, agrega flexibilidad al sistema. Además, la opción de eliminar clientes contribuye a mantener actualizada la base de datos de clientes.

Es importante destacar que la página de Clientes se integra de manera coherente con otras partes del sistema, especialmente con las páginas de Ventas y Alquileres. La relación entre estas entidades es esencial para garantizar un seguimiento preciso de las transacciones y para brindar un servicio personalizado a cada cliente. La funcionalidad de la página de Clientes se alinea con la visión integral del sistema, donde la gestión eficiente de la información del cliente es esencial para el éxito de las operaciones de venta y alquiler de vehículos.

L.A. Motors

Usuarios

Clientes

Vehículos

Ventas

Alquileres

Cerrar sesión

Agregar un nuevo Cliente

Nombre del Cliente

Apellido del Cliente

Documento del Cliente

Dirección del Cliente

Teléfono del Cliente

Guardar

Lista de Clientes

Documento	Nombre	Apellido	Dirección	Teléfono	Editar	Eliminar
45866580	Juan	Perez	dr Edye 3456	098765531	Editar	Eliminar
45899989	Javier	Lopez	dr Edye 5585	098509731	Editar	Eliminar

Figura 3.4: Página de Clientes.

3.5. Vehículos

La página de Vehículos se erige como una herramienta central para la administración del inventario de la empresa. Esta interfaz proporciona a los usuarios la capacidad de registrar nuevos vehículos, detallando sus propiedades específicas y asignándoles una categoría entre "Auto", "Moto" y "Camión".

La inclusión de imágenes para cada vehículo permite una visualización más completa, facilitando la identificación y el reconocimiento por parte de los usuarios. La clasificación por categorías y la posibilidad de agregar características únicas según el tipo de vehículo mejoran la organización y la búsqueda eficiente dentro del inventario.

The screenshot shows the 'Agregar un nuevo Vehículo' (Add a new Vehicle) form. At the top is a dark navigation bar with the text 'L.A. Motors' and links for 'Usuarios', 'Clientes', 'Vehículos', 'Ventas', 'Alquileres', and a 'Cerrar sesión' (Log out) button. The form itself is titled 'Agregar un nuevo Vehículo' and contains two columns of input fields. The left column includes: 'Matrícula del Vehículo', 'Marca del Vehículo', 'Modelo del Vehículo', 'Cantidad de Kilometros', a date picker labeled 'Elija la fecha de fabricación del vehículo' with the placeholder 'dd/mm/aaaa', 'Color del Vehículo', 'Imagen 1 del Vehículo', and 'Imagen 2 del Vehículo'. The right column includes: 'Imagen 3 del Vehículo', 'Precio de venta', 'Precio de alquiler', radio buttons for vehicle type (with 'Moto' selected), 'Cilindradas del Vehículo', and a blue 'Guardar' (Save) button.

Figura 3.5: Página de Vehículos, ingreso de datos.

La grilla que muestra los vehículos registrados ofrece opciones de edición y eliminación, lo que facilita la actualización y el mantenimiento del inventario. La conexión fluida entre la página de Vehículos y las páginas de Ventas y Alquileres asegura una integración coherente, fundamental para el proceso de transacciones. Además, la lógica de eliminación automática de vehículos vendidos contribuye a mantener la información actualizada y reflejar con precisión la disponibilidad de vehículos en tiempo real.

Lista de Vehículos

Matrícula	Marca	Modelo	Kilometraje	Año	Color	Precio Venta	Precio Alquiler
XYZ789	Mercedes-Benz	Actros	60000	2019	Azul	55000	750
DEF456	Scania	R-Series	80000	2021	Blanco	65000	900
JKL012	Peterbilt	579	50000	2018	Verde	50000	700

Figura 3.6: Página de Vehículos, datos en la grilla.










Imagen 1	Imagen 2	Imagen 3	Editar	Eliminar
			Editar	Eliminar
			Editar	Eliminar
			Editar	Eliminar

Figura 3.7: Página de Vehículos, imágenes en la grilla y botones.

3.6. Ventas

La página de Ventas constituye el punto focal para llevar a cabo transacciones comerciales de manera eficiente. Esta interfaz permite a los usuarios registrar ventas de vehículos, capturando detalles esenciales como el cliente, la matrícula del vehículo vendido, el costo y la fecha de la transacción.

La conexión directa con la página de Clientes asegura una integración sin problemas, permitiendo la selección fácil y rápida de clientes existentes al realizar una venta. La inclusión del documento como identificador único contribuye a una gestión más precisa de las transacciones y facilita la asociación de ventas a clientes específicos.

La grilla que muestra las ventas registradas ofrece la funcionalidad de eliminar registros innecesarios. Además, la lógica implementada para la actualización automática del inventario de vehículos garantiza la coherencia entre las páginas de Ventas y Vehículos.

L.A. Motors

Usuarios

Clientes

Vehículos

Ventas

Alquileres

Cerrar sesión

Realizar la venta de un vehículo

Cientes:

Documento: 45866580 - Nombre y Apellido: Juan Perez - Teléfono: 098765531
Documento: 45899989 - Nombre y Apellido: Javier Lopez - Teléfono: 098509731
Documento: 37854682 - Nombre y Apellido: Luis Gomez - Teléfono: 098712313

Vehiculos:

Auto - Matrícula: ASD7892 - Marca: Chevrolet - Año: 2022 - Precio: \$26000

Vender

Lista de vehículos vendidos

Número de Venta	Matrícula	Marca	Modelo	Color	Año	Precio Venta	Documento Cliente	Nombre y Apellido Cliente	Eliminar
1	M123	Honda	CBR600RR	Rojo	2022	\$12000	45866580	Juan Perez	Eliminar
2	FFG1237	Nissan	Altima	Verde	2018	\$20000	45899989	Javier Lopez	Eliminar

Figura 3.8: Página de Ventas.

3.7. Alquileres

La página de Alquileres representa la columna vertebral para administrar y supervisar los acuerdos de alquiler. Diseñada con precisión, ofrece una interfaz que permite a los usuarios registrar nuevos alquileres y gestionar los existentes de manera efectiva.

33

La vinculación directa con la página de Clientes facilita la asignación rápida de un cliente específico al proceso de alquiler, utilizando el documento del cliente como identificador único. Esta conexión asegura una gestión coherente y precisa de la información del cliente a lo largo de todo el sistema.

L.A. Motors [Usuarios](#) [Clientes](#) [Vehículos](#) [Ventas](#) [Alquileres](#) [Cerrar sesión](#)

Realizar el alquiler de un vehículo

Clientes:


Documento: 45866580 - Nombre y Apellido: Juan Perez - Teléfono: 098765531
Documento: 45899989 - Nombre y Apellido: Javier Lopez - Teléfono: 098509731
Documento: 37854682 - Nombre y Apellido: Luis Gomez - Teléfono: 098712313

Vehiculos:

Auto - Matrícula: ASD7892 - Marca: Chevrolet - Año: 2022 - Precio diario: \$2600

Cantidad de días de alquiler

Indique la fecha de inicio del Alquiler

dd/mm/aaaa 

Alquilar

Figura 3.9: Página de Alquileres, ingreso de datos.

La página de Alquileres incorpora funcionalidades de edición y eliminación en su grilla, proporcionando flexibilidad para ajustar detalles o eliminar registros según sea necesario. La lógica implementada garantiza que la disponibilidad de vehículos se actualice automáticamente, manteniendo la coherencia entre las páginas de Alquileres y Vehículos.

Lista de vehículos alquilados

Número de Alquiler	Matrícula	Marca	Modelo	Color	Año	Cantidad de días	Fecha de Inicio
1	ABC123	Volvo	VNL	Rojo	2020	3	10/12/2023
2	GHI789	Kenworth	T680	Negro	2022	10	09/12/2023

Precio Alquiler	Documento Cliente	Nombre y Apellido Cliente	Devuelto	Estado	Editar	Eliminar
\$2400	37854682	Luis Gomez	<input type="checkbox"/>	Al día	Editar	Eliminar
\$8500	45899989	Javier Lopez	<input type="checkbox"/>	Al día	Editar	Eliminar

Figura 3.10: Página de Alquileres, grilla y botones.

4. Conclusiones

La culminación de este proyecto ha dado lugar a un programa robusto y funcional que ha superado ampliamente las expectativas iniciales. La implementación exitosa de todos los requisitos, especialmente las operaciones de ventas y alquileres, destaca la solidez y eficiencia del sistema. La interfaz de usuario, diseñada con HTML, CSS y Bootstrap en ASP.NET, proporciona una experiencia intuitiva y fácil de usar para los empleados de la automotora, facilitando la navegación a través de las diversas funcionalidades mediante una barra de navegación bien estructurada.

Las funcionalidades de listas, que detallan información esencial sobre ventas, alquileres, vehículos, clientes y usuarios, son fundamentales para una gestión efectiva del sistema. Permiten acceder y analizar datos clave de manera eficiente, mejorando la toma de decisiones y la eficacia operativa.

Aunque los resultados son altamente positivos, se ha identificado una limitación en la gestión de usuarios, particularmente en la capacidad del Usuario Admin para editar y eliminar sus propios permisos. Esta observación resalta una oportunidad de mejora para futuras iteraciones, con el objetivo de garantizar una administración más coherente y segura de los permisos de usuario, minimizando posibles riesgos de seguridad.

Además, durante la evaluación del proyecto, se identificó una oportunidad de mejora que consiste en la implementación de la funcionalidad para descargar las listas en formatos como Excel, archivos de texto u otros. A pesar de que no se pudo incluir en la versión actual debido a limitaciones de tiempo, se considera una valiosa adición para futuras iteraciones, mejorando aún más la usabilidad y flexibilidad del programa. En conjunto, el proyecto ha alcanzado sus metas principales y presenta oportunidades claras para seguir evolucionando y mejorando en el futuro

5. Referencias

- [1]. [Clases y Objetos - Programación Orientada a Objetos \(gitbook.io\)](https://gitbook.io)
- [2]. [Herencia en C# | Microsoft Learn](https://learn.microsoft.com/es-es/csharp/fundamentals/inheritance)
- [3]. [Listas en C# - \(bugados.com\)](https://bugados.com/)
- [4]. [Manejo de excepciones - C# | Microsoft Learn](https://learn.microsoft.com/es-es/csharp/fundamentals/exceptions)
- [5]. [Información general de ASP.NET | Microsoft Learn](https://learn.microsoft.com/es-es/aspnet/)
- [6]. [HTML: Lenguaje de etiquetas de hipertexto | MDN \(mozilla.org\)](https://developer.mozilla.org/es/docs/Web/HTML)
- [7]. [CSS | MDN \(mozilla.org\)](https://developer.mozilla.org/es/docs/Web/CSS)