Universidad Tecnológica Del Cibao Oriental (UTECO)

Lenguaje de Programación IV

Nombre

Osnilda Furcal Domínguez

Matricula

2022-1411

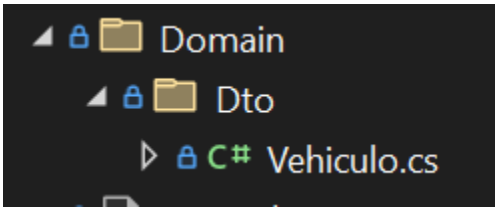
Tema

Informe sobre aplicación hecha con Blazor.

Docente

Felix Benzan

Aplicación sobre una lista de vehículos

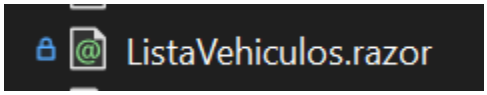


- Lo primero que hice después de haber creado el proyecto fue crear 2 carpetas una Domain y Otra llamada Dto donde cree una clase llamada Vehiculo.

```
7 referencias
public class Vehiculo
{
    3 referencias
    public Vehiculo()
    {
    }
    7 referencias
    public string Nombre {  get; set; }

    7 referencias
    public string Color { get; set; }

    2 referencias
    public DateTime FechaAgregado { get; set; }

}
```

Donde declare los atributos que mas me interesaban para la lista nombre, color, Fecha y hora (pero la fecha en la que el usuario agrego un elemento a la lista)

Eso si la mayoría de las capturas son de la versión final.



- Luego cree un componente llamado ListaVehiculos.razor.

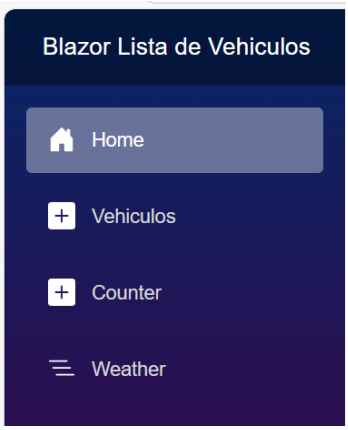
```
@page "/lista"
@rendermode InteractiveServer
```

- Donde especifique que el componente es una pagina y que se renderice del lado del servidor.

Fui al NavMenu y agregué la página al menú.

Home.razor | NavMenu.razor | ListaVehiculos.razor | Vehiculo.cs

```
6
7     <input type="checkbox" title="Navigation menu" class="navbar-toggler" />
8
9     <div class="nav-scrollable" onclick="document.querySelector('.navbar-toggler').click()">
10        <nav class="flex-column">
11            <div class="nav-item px-3">
12                <NavLink class="nav-link" href="" Match="NavLinkMatch.All">
13                    <span class="bi bi-house-door-fill-nav-menu" aria-hidden="true"></span> Home
14                </NavLink>
15            </div>
16
17            <div class="nav-item px-3">
18                <NavLink class="nav-link" href="lista">
19                    <span class="bi bi-plus-square-fill-nav-menu" aria-hidden="true"></span> Vehiculos
20                </NavLink>
21            </div>
```



Ahora volviendo al componente ListaVehiculos me surgió la siguiente idea: que en lugar de utilizar inputs para la lista de vehículos que sea un elemento select con options, para evitar que el usuario ingrese cosas no relacionadas con los vehículos. Y que estos select pasaran su valor a los atributos de la clase Vehiculo y así poder hacer la lista.

```
<div style="margin-right:10px;">
    <label for="" class=""><strong>Nombre del auto</strong></label>
    <select @bind="vehiculos.Nombre" class="form-select" style="width: 450px;">
        <option value="">Seleccione</option>
        <option value="Hyundai Sonata SE">Hyundai Sonata SE</option>
        <option value="Lamborghini Urus">Lamborghini Urus</option>
        <option value="Toyota Supra Mk4">Toyota Supra Mk4</option>
        <option value="Camaro Zl1 1LE">Camaro Zl1 1LE</option>
        <option value="Kia Picanto Zenith">Kia Picanto Zenith</option>
        <option value="Ford Bronco Raptor">Ford Bronco Raptor</option>
        <option value="Mitsubishi Lancer Evo XI">Mitsubishi Lancer Evo XI</option>
        <option value="Tesla Model X">Tesla Model X</option>
    </select>
```

```
<div>
        <label for="" class=""><strong>Color del vehiculo</strong></label>
        <select @bind="vehiculos.Color" class="form-select" style=" width: 450px;">
            <option value="">Seleccione</option>
            <option value="Rojo">Rojo</option>
            <option value="Amarillo">Amarillo</option>
            <option value="Blanco">Blanco</option>
            <option value="Negro">Negro</option>
            <option value="Naranja">Naranja</option>
            <option value="Azul">Azul</option>
            <option value="Verde">Verde</option>
            <option value="Plateado">Plateado</option>
    </select>
```

Con un botón que se encargue de agregar las listas:

```
<div style="text-align:center;"><button @onclick="agregar" class="btn btn-primary" style="margin-top: 37px; width:2
```

Ahora para la funcionalidad cree un arreglo /lista (esta basado en el proyecto HolaMundo antiguo) que es donde vamos a guardar los objetos de la lista, y un objeto llamado vehiculos (una instancia) que es el objeto que se va a estar agregando a la lista.

```
@code {
    public Vehiculo vehiculos { get; set; } = new Vehiculo();
    public List<Vehiculo> VehiculosList { get; set; } = [];

    void agregar()
    {
        if (!string.IsNullOrEmpty(vehiculos.Nombre) && !string.IsNullOrEmpty(vehiculos.Color))
        {
            VehiculosList.Add(new Vehiculo
                {
                    Nombre = vehiculos.Nombre,
                    Color = vehiculos.Color,
                    FechaAgregado = DateTime.Now
                });

            vehiculos = new Vehiculo();

        }
```

La función agregar tiene una condición la cual se asegura de que el usuario no pueda agregar opciones vacías, la instancia de Vehiculo se añade a VehiculoList y al final el objeto vehiculos se reinicia quedando vacío y volviendo a repetir el proceso.

```
private void Eliminar(Vehiculo item)
{
    if (item != null)
    {
        VehiculosList.Remove(item);
    }
}
```

Esta función elimina un elemento especifico de VehiculosList.

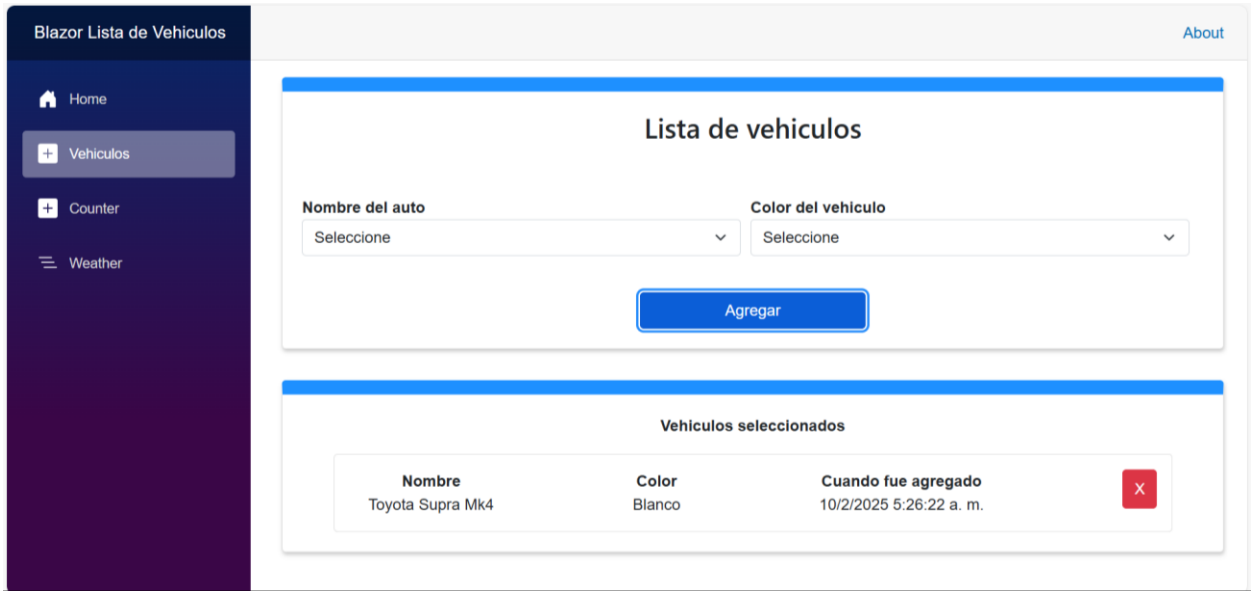
```
@if (VehiculosList.Any()){
    <div class="card" style="padding:20px; border-top: 15px solid DodgerBlue; box-shadow: 1px 3px 4px rgba(0,
    <label style="margin-bottom: 13px"><strong>Vehiculos seleccionados</strong></label>

            @foreach (var item in VehiculosList)
            {
            <div class="card" style="padding:15px; width:860px; height: 80px; margin-top: 4px; display:flex; flex-
            <div style="margin-left:20px;">
                <label for="" class=""><strong>Nombre</strong></label>
                <p>@item.Nombre</p>
            </div>
            <div>
                <label for="" class=""><strong>Color</strong></label>
                <p>@item.Color</p>
            </div>
            <div>
                <label for="" class=""><strong>Cuando fue agregado</strong></label>
                <p>@item.FechaAgregado</p>
            </div>
```

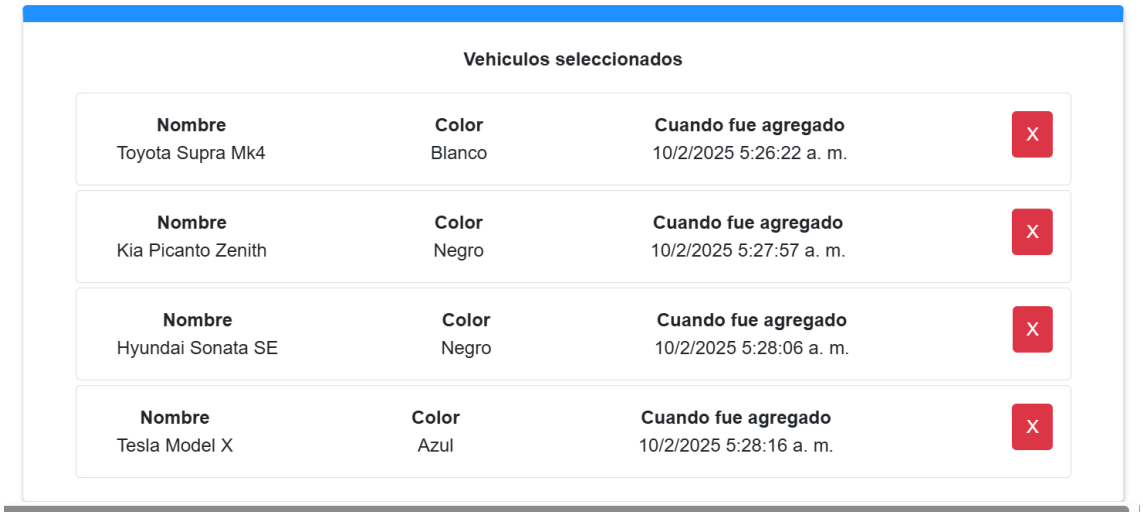
Para evitar que el espacio donde aparecen las listas se vea, utilice una condicional if que chequea si la lista tiene algún elemento, sino entonces se oculta. El ciclo foreach es el responsable de crear los múltiples elementos por cada item en la lista.

Y el botón de eliminar de cada elemento:

```
<button class="btn btn-danger" style="height:40px; width:35px; padding:5px; float:right;" @onclick="()=> Eliminar(item)"
```

Y este es el resultado:





Durante la creación de este proyecto aprendí:

- A hacer commits y push desde visual studio.

- Que similar a angular Blazor te permite visualizar el contenido de una variable usando @ y su nombre.

- Las sentencias @if y el ciclo @foreach con html.

- A la hora de hacer listas es bueno utilizar un Id para evitar duplicados (cosa que no hice, el proyecto agrega duplicados).

- Los atributos tomados directamente de la clase y asignados a un elemento html se pueden actualizar en tiempo real y eso es problemático en esta clase de proyectos.

- Utilizar la propiedad style en línea puede resultar caótico (el próximo proyecto será puro boostrap o archivo css aparte porque lo que hice fue una aberración, es que solo se css puro)