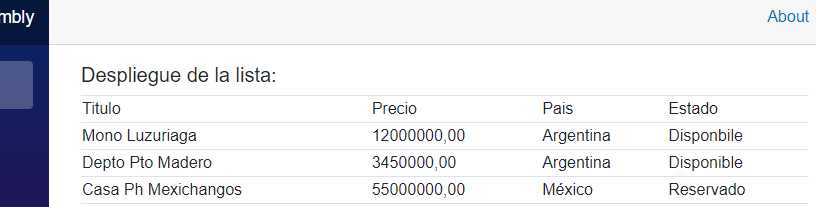
Tabla



@page "/"

@using Inmobiliaria.DTO;

@using Inmobiliaria.WebAssembly.Servicios.Contrato;

@inject IInmobiliariaServicios inmobiliariaServicios;

<div>

<h5>Despliegue de la lista:</h5>

<table class="table table-bordered">

<tr>

<td>Titulo</td>

<td>Precio</td>

<td>Pais</td>

<td>Estado</td>

</tr>

@foreach (var item in listaInmuebles)

{

<tr>

<td>@item.Titulo</td>

<td>@item.Precio</td>

<td>@item.Pais</td>

<td>@item.Estado</td>

</tr>

}

</table>

</div>

@code{

List<InmuebleDTO> listaInmuebles = new List<InmuebleDTO>();

string buscar = "";

private async Task ObtenerCatalogo()

{

var response = await inmobiliariaServicios.Catalogo(buscar);

if (response.EsCorrecto == true)

{

listaInmuebles = (List<InmuebleDTO>)response.Resultado!;

}

else

{

listaInmuebles = new List<InmuebleDTO>();

}

}

protected override async Task OnInitializedAsync()

{

await ObtenerCatalogo();

}

}

Servicios de la API:

public async Task<List<InmuebleDTO>> Catalogo(string buscar)

{

try

{

var consulta = \_modeloRepositorio.Consultar(i => i.Titulo.ToLower().Contains(buscar.ToLower()));

List<InmuebleDTO> lista = \_mapper.Map<List<InmuebleDTO>>(await consulta.ToListAsync());

return lista;

}

catch (Exception ex)

{

throw ex;

}

}

Buscar puede definirse por medio de la interfaz de la página o bien puede quedar vacío, como al inicializarse la página. Si está vacío, mostrará todos los elementos de la tabla.

Barra de búsqueda +

Tarjetas con info de tabla e imágenes





@page "/"

@using Inmobiliaria.DTO;

@using Inmobiliaria.WebAssembly.Servicios.Contrato;

@inject IInmobiliariaServicios inmobiliariaServicios;

<div>

<div class="row justify-content-center mt-3 mb-4">

<div class="col-sm-6 col-md-8 col-xl-9">

<div class="input-group mb-3">

<input type="text" class="form-control" placeholder="Buscar producto..." @bind-value="buscar">

<button class="btn btn-outline-secondary" type="button" @onclick="(async () => await ObtenerCatalogo())">

<i class="oi oi-magnifying-glass"></i>

</button>

</div>

</div>

</div>

Cuando se toca el botón se ejecuta el método ObtenerCatalogo y se le manda el bind-value como parámetro.

Creo, que es mas correcto decir que se le da el valor del input al string buscar. El cual ya era el parámetro del método. Si se pusiera otro bind-value se le daría valor a esa variable y buscar quedaría inmutable.

<div class="col-sm-10">

<div class="row gx-4 row-cols-1 row-cols-md-3 row-cols-xl-4 justify-content-start">

@foreach (var item in listaInmuebles)

{

<div class="col mb-5">

<div class="card h-100">

<div class="card-body">

<div class="text-center">

<img class="card-img-top border-dark border-1" src="/bombones.jpg" style="width:100%;height:250px">

<h3 class="fw-bolder">@item.Titulo</h3>

@item.Precio

@item.Pais

@item.Estado

</div>

</div>

</div>

</div>

}

</div>

</div>

</div>

@code{

List<InmuebleDTO> listaInmuebles = new List<InmuebleDTO>();

string buscar = "";

string locura = "mexi";

private async Task ObtenerCatalogo()

{

var response = await inmobiliariaServicios.Catalogo(buscar);

if (response.EsCorrecto == true)

{

listaInmuebles = (List<InmuebleDTO>)response.Resultado!;

}

else

{

listaInmuebles = new List<InmuebleDTO>();

}

}

protected override async Task OnInitializedAsync()

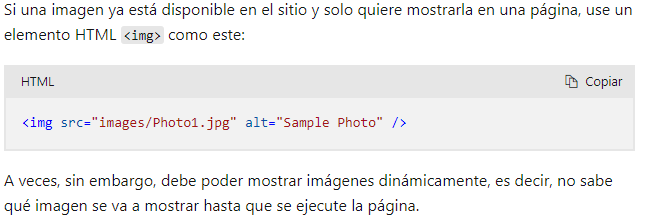
{

await ObtenerCatalogo();

}

}

Imágenes

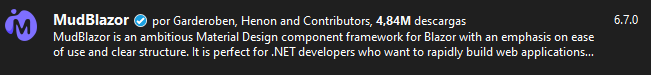


https://learn.microsoft.com/es-es/aspnet/web-pages/overview/ui-layouts-and-themes/9-working-with-images

Overflow:Hidden. Es para ocultar los elementos que están fuera del padre.

Nuget

MudBlazor



@using MudBlazor

Add font and style references

Add the following to your HTML head section, it's either index.html or \_Layout.cshtml/\_Host.cshtml depending on whether you're running WebAssembly or Server.

<link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap" rel="stylesheet" />

<link href="\_content/MudBlazor/MudBlazor.min.css" rel="stylesheet" />

Add script reference

In the same file but located in the end of it add the MudBlazor js file, it should be in the same location as the default blazor script.

<script src="\_content/MudBlazor/MudBlazor.min.js"></script>

Program.Cs

using MudBlazor.Services;

builder.Services.AddMudServices();

Add Components

Add the following components to your MainLayout.razor note that the ThemeProvider is essential for MudBlazor but the rest is optional.

<MudThemeProvider/>

<MudDialogProvider/>

<MudSnackbarProvider/>

<**MudCarousel** **Class**="mud-width-full" @ref="\_carousel" **ItemsSource**="@listaImagenes" @bind-SelectedIndex="selectedIndex" **Style**="height:200px;" **ShowArrows**="@\_arrows" **ShowBullets**="@\_bullets" **EnableSwipeGesture**="@\_enableSwipeGesture" **AutoCycle**="@\_autocycle">

<**ItemTemplate**>

<div class="d-flex flex-column flex-column justify-center" style="height:100%">

<**MudImage** **Src**="@context">@context</**MudImage**>

</div>

</**ItemTemplate**>

</**MudCarousel**>

<**MudSwitch** @bind-Checked="@\_arrows" **Color**="Color.Primary">Show Arrows</**MudSwitch**>

<**MudSwitch** @bind-Checked="@\_bullets" **Color**="Color.Primary">Show Bullets</**MudSwitch**>

<**MudSwitch** @bind-Checked="@\_enableSwipeGesture" **Color**="Color.Primary">Enable Swap Gesture</**MudSwitch**>

<**MudSwitch** @bind-Checked="@\_autocycle" **Color**="Color.Primary">Auto Cycle (Default: 5 secs)</**MudSwitch**>

<br />

<**MudSelect** @bind-Value="selectedIndex" **Label**="@($"Selected Item (index: {selectedIndex})")">

@{

int index = 0;

foreach (var item in listaImagenes)

{

<**MudSelectItem** **Value**="@index">@item</**MudSelectItem**>

index++;

}

}

</**MudSelect**>

@code {

private MudCarousel<string> \_carousel;

private bool \_arrows = true;

private bool \_bullets = true;

private bool \_enableSwipeGesture = true;

private bool \_autocycle = true;

private IList<string> \_source = new List<string>() { "Item 1", "Item 2", "Item 3", "Item 4", "Item 5" };

private int selectedIndex = 0;

private async Task AddAsync()

{

\_source.Add($"Item {\_source.Count + 1}");

await Task.Delay(1);

\_carousel.MoveTo(\_source.Count - 1);

}

private async Task DeleteAsync(int index)

{

if (\_source.Any())

{

\_source.RemoveAt(index);

await Task.Delay(1);

\_carousel.MoveTo(System.Math.Max(System.Math.Min(index, \_source.Count - 1), 0));

}

}

}