En una primera instancia hace que el controlador obtenga información directamente desde la BBDD. Luego hace que pase por una capa de servicios, invocando la idea de controladores estúpidos.

Para ello, crea esta clase:

public class ServiceResponse<T>

{

public T? Data { get; set; }

public bool Success { get; set; } = true;

public string Message { get; set; } = string.Empty;

}

Así estaba el controlador con el método GET.

[HttpGet]

public async Task<ActionResult<List<Product>>> GetProducts()

{

var products = await \_context.Products.ToListAsync();

return Ok(products);

}

Luego queda asÍ (en realidad acá implementa el objetoServiceResponse nada mas. El controlado sigue conectando con la BBDD):

[HttpGet]

public async Task<ActionResult<ServiceResponse<List<Product>>>> GetProducts()

{

var products = await \_context.Products.ToListAsync();

var response = new ServiceResponse<List<Product>>()

{

Data = products

};

return Ok(response);

}

Asimismo. El Razor recibía products, tendrá que hacerse cargo del response.

LUUEEEEEEGOOOOO

Agrega carpeta Servicios y ProductServices en Server. Con la interfaz e implementación. Asimismo lo inyecta en program.cs.

Así queda la implementación del método GetProductAsync.

public async Task<ServiceResponse<List<Product>>> GetProductsAsync()

{

var response = new ServiceResponse<List<Product>>

{

Data = await \_context.Products.ToListAsync()

};

return response;

}

Entonces retoca el controlador, al que ya no le llega el contexto, sino el servicio este:

[HttpGet]

public async Task<ActionResult<ServiceResponse<List<Product>>>> GetProducts()

{

var result = await \_productService.GetProductsAsync();

return Ok(result);

}

Queda bien peladito.

Luego agrega ProductService al Cliente. Se registra en el program.cs de Client y se modifica el componente razor correspondiente.

Detalle

Tenés un listado de productos. Hacés click en uno y accedes a su página, con el detalle. Así tiene el esqueleto inicial, sin HTML:

@page "/product/{id:int}"

@inject IProductService ProductService

@code {

private Product? product=null;

[Parameter]

public int Id { get; set; }

protected override async Task OnParametersSetAsync()

{

product = ProductService.Products.Find(p => p.Id == Id);

}

}

Crea la funcionalidad de traer de la BBDD un solo producto, pasándole su Id.

public async Task<ServiceResponse<Product>> GetSingleProductAsync(int productId)

{

var response = new ServiceResponse<Product>();

var product=await \_context.Products.FindAsync(productId);

if (product == null) {

response.Success = false;

response.Message = "Sorry, this product doesn´t exist";

}

else

{

response.Data= product;

}

return response;

}

Esto es en el productService del Server. Ahora toca el controlador:

[HttpGet("{productId}")]

public async Task<ActionResult<ServiceResponse<Product>>> GetSingleProductsWithId(int productId)

{

var result = await \_productService.GetSingleProductAsync(productId);

return Ok(result);

}

Así queda el productService del Cliente:

public async Task<ServiceResponse<Product>> GetSingleProductWithId(int productId)

{

var result = await \_http.GetFromJsonAsync<ServiceResponse<Product>>($"api/product/{productId}");

return result;

}