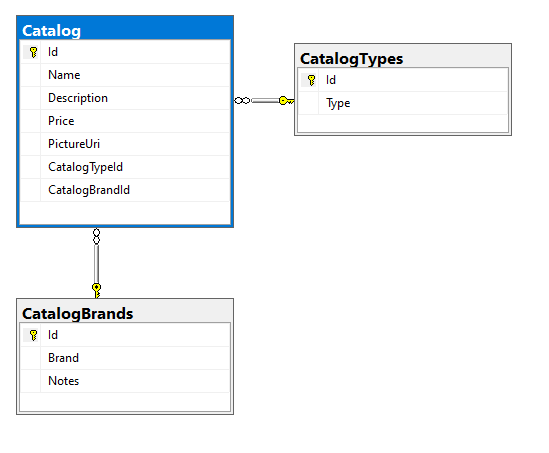
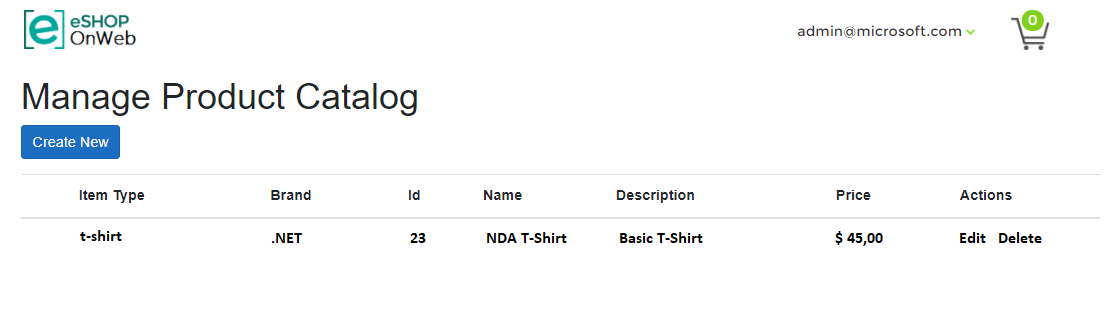
# Aula2. Iniciando a criação de um crud completo

Vamos fazer o CRUD de Item de Catalogo. Nessa aula vamos fazer a consulta e edição.

Banco de dados:



E a tela inicial do CRUD deverá ser assim:



## Criando a query

Abra a pasta Specifications do ApplicationCore. Veja que tem 3 Specs de Catalog:

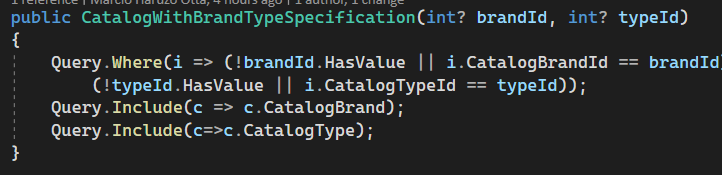
* CatalogFilterSpecification, que filtra por brandId e tipeId,
* CatalogFilterPaginatedSpecification, que filtra a tabela e traz apenas os registros de 1 pagina;
* CatalogItemsSpecification, que traz alguns por ID.

Em nenhum desses métodos são trazidos os dados das tabelas auxiliares, CatalogBrands e CatalogTypes, então vamos fazer uma spec que faça isso.

Crie a CatalogWithBrandTypeSpecification (copie a classe CatalogFilterSpecification e renomeie).

Altere o construtor fazendo include de CatalogBrands e CatalogTypes.

Note que a query para o specification é um pouco diferente:



Não iremos criar testes unitários porque esse comportamento de Eager Loading exige que seja feito teste de integração.

## Criando a tela de consulta

Na pasta Pages/Admin, crie a Index.cshtml

Copie esse html para dentro dela:

@page

@model Microsoft.eShopWeb.Web.Pages.Admin.IndexModel

@{

ViewData["Title"] = "Manage Product Catalog";

}

<**form** class="container" method="post">

<h1>@ViewData["Title"]</h1>

<**a** class="btn btn-primary" **asp-page**="./Create">

Create New

</**a**>

<table class="table table-striped table-hover">

<thead>

<tr>

<th>Item Type</th>

<th>Brand</th>

<th>Id</th>

<th>Name</th>

<th>Price</th>

<th>Actions</th>

</tr>

</thead>

<tbody class="cursor-pointer">

@foreach (var item in @Model.Lista)

{

<tr>

<td>@item.CatalogType</td>

<td>@item.CatalogBrand</td>

<td>@item.Id</td>

<td>@item.Name</td>

<td>@item.Price</td>

<td>

<**a** class="btn btn-primary" **asp-page**="./EditCatalogItem" **asp-route-catalogId**="@item.Id">

Edit

</**a**>

<**button** type="submit" class="btn btn-primary" **asp-page-handler**="delete"

**asp-route-catalogId**="@item.Id">Delete

</**button**>

</td>

</tr>

}

</tbody>

</table>

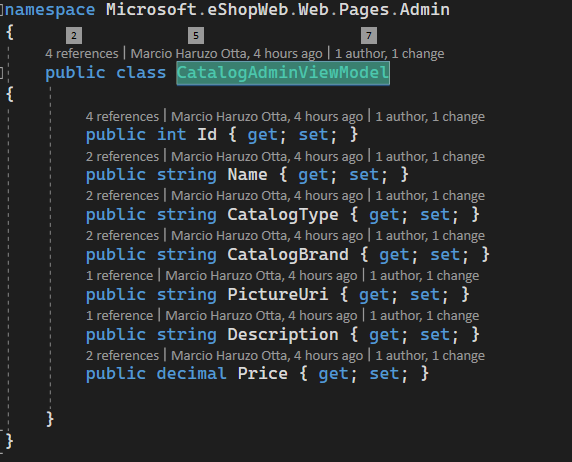
</**form**>

**Observações**:

* Note que o modelo quer uma variável Lista contendo um ViewModel com as propriedades Id, CatalogType, etc.
* O Link Create e Delete apontam para paginas que ainda não criamos
* O link Delete aponta para uma ação nessa pagina que ainda não criamos.

Entao vamos criar o viewModel dessa página, infelizmente os viewModels já existentes não servem pois não trazem os dados de Brand e Type.

Nessa mesma página crie o CatalogAdminViewModel:

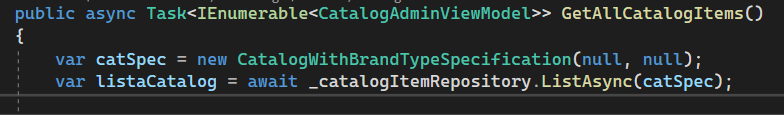


Na PageModel (Index.cshtml.cs), crie a variável ou property Lista que conterá a lista desse viewModel:

public IEnumerable<CatalogAdminViewModel> Lista = new List<CatalogAdminViewModel>();

## Criando o serviço para popular a lista

Como já fizemos na aula1, vamos criar um método de serviço no projeto WEB para popular a viewModel. Na pasta Services, já existe a CatalogItemViewModelService.cs, crie um método para retornar uma IEnumerable<CatalogAdminViewModel> usando a spec que criamos:



Agora basta fazer o mapping, use esse código:

var listaCatalogAdminViewModel =

listaCatalog.Select(c => new CatalogAdminViewModel

{

Id = c.Id,

Name = c.Name,

Description = c.Description,

Price = c.Price,

CatalogBrand = c.CatalogBrand.Brand,

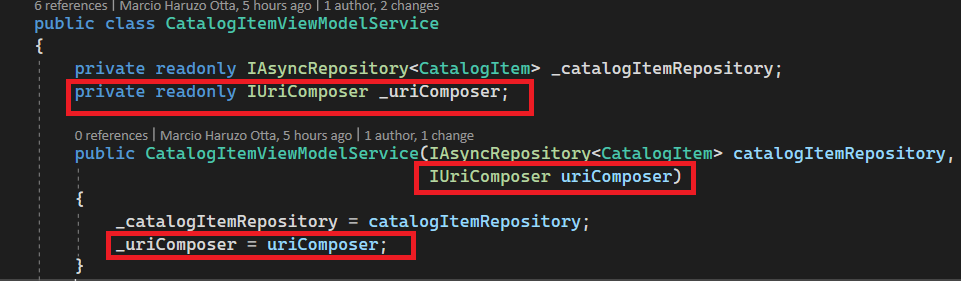
CatalogType = c.CatalogType.Type,

PictureUri = \_uriComposer.ComposePicUri(c.PictureUri)

});

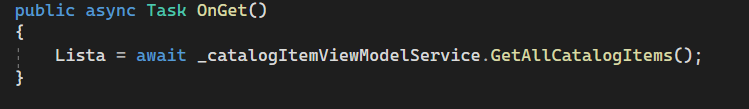
return listaCatalogAdminViewModel;

Note que precisamos usar o serviço IUriComposer, que apenas faz um replace da URL que está gravada no banco de dados, receba essa Classe no Construtor:



## Concluindo a tela de consulta

Na PageModel (Index.cshtml.cs), receba no construtor o catalogItemViewModelService, e no onGet preencha a variável Lista:



## Fazendo a tela de Edição

A página EditCatalogItem já existe, só precisamos fazer uma alteração pois ela não recebe como parâmetro o ID (era utilizada por outro projeto e recebia como parâmetro o viewModel inteiro).

Troque a linha:

@page

Por:

@page "{catalogId}"

No PageModel (.cs), altere o método OnGet para isso:

public async Task OnGet(int catalogId)

{

CatalogModel = await \_catalogItemViewModelService

.GetCatalogItemViewModelById(catalogId);

}

Esse GetCatalogItemViewModelById não existe, faça sua implementação na classe de serviço CatalogItemViewModelService. Ela deve retornar um CatalogItemViewModel dado um ID.