# Selects Elaborados - ASP.net

ASP.NET

Nícolas Maisonnette Duarte
Disciplina: Desenvolvimento para Internet III

# Introdução rápida

Neste material, é proposta uma maneira de realizar selects mais elaborados em ASP. Mas, o que seriam esses selects? Basicamente são aqueles com where e que podem ou não se relacionar a outra tabela.

Para que seja possível acompanhar o passo a passo aqui descrito, é recomendado que já se tenha lido a apostila “ASP.NET Core 3.1 com Entity Framework SQL Server” da professora Patrícia.

# O que interessa: Como fazer?

Basicamente, começaremos criando um novo controlador na pasta Controllers. Esse controlador pode ter o nome que desejar com o sufixo Controller, mas neste exemplo utilizaremos ImagensDoPinController, que possui um contexto de “mundo”. Mas, para que criaremos uma classe nova? Para conseguirmos passar um mesmo parâmetro, no caso PinID, duas vezes sem haver conflito, porque as rotas serão diferentes.

```
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using backend.Data;
using Microsoft.AspNetCore.Http;

namespace ProjetoMundo_API.Controllers
{
    [Route("[controller]")]
    [ApiController]
    public class ImagensDoPinController : Controller
    {
        public IRepository Repo { get; }
    }
}
```

Podemos observar que os usings já foram colocados para facilitar a implementação da classe. Agora, faremos o construtor dela, que utiliza o IRepository.

```
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using backend.Data;
using Microsoft.AspNetCore.Http;

namespace ProjetoMundo_API.Controllers
{
    [Route("[controller]")]
    [ApiController]
    public class ImagensDoPinController : Controller
    {
        public IRepository Repo { get; }

        public ImagensDoPinController(IRepository repo)
        {
            this.Repo = repo;
        }
    }
}
```

Após essa implementação, devemos finalmente escrever o nosso método. Para fins de exemplo, faremos um que coleta todas as imagens de um determinado pin (pin de mapa, uma localização).

```
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using backend.Data;
using Microsoft.AspNetCore.Http;

namespace ProjetoMundo_API.Controllers
{
    [Route("[controller]")]
    [ApiController]
    public class ImagensDoPinController : Controller
    {
        public IRepository Repo { get; }

        public ImagensDoPinController(IRepository repo)
        {
            this.Repo = repo;
        }

        [HttpGet("{PinID}")]
        public async Task<IActionResult> Get(int PinID)
        {
            try
            {
                var result = await this.Repo.GetAllImagesAsyncByIdPin(PinI
D);
                return Ok(result);
            }
            catch
            {
                return this.StatusCode(StatusCodes.Status500InternalServer
Error, "Falha no acesso ao banco de dados.");
            }
        }
    }
}
```

Aqui, utilizamos um método do Repository chamado GetAllImagesAsyncByIdPin, onde o parâmetro recebido é o id de um pin. Ele serve para coletarmos todas as imagens que possuem um determinado ID de um pin, que podem ser uma ou mais. Vamos, então, implementar esse método na classe e colocá-lo na IRepository também!

```
using System.Threading.Tasks;
using backend.Models;
using System.Linq;
using Microsoft.EntityFrameworkCore;

namespace ProjetoMundo_API.Data
{
    public class Repository : IRepository
    {
        public MundoContext Context { get; }

        public Repository(MundoContext context)
        {
            this.Context = context;
        }


      (...) //resto da classe


        public async Task<ImagemPin[]> GetAllImagesAsyncByIdPin(int ID)
        {
            IQueryable<ImagemPin> consultaImagens=this.Context.ImagemPin;
            consultaImagens = consultaImagens.OrderBy(a => a.IdImagem)
            .Where(ImagemPin => ImagemPin.IdPin == ID);

            return await consultaImagens.ToArrayAsync();
        }
    }
}
```

Agora que já implementamos o método no Repository, é hora do IRepository, mas antes, uma explicação: utilizamos o ".Where" para coletarmos somente aquelas imagens que possuem o ID do pin que desejamos. Também ordenamos as imagens coletadas por meio de seus próprios IDs e retornamos tudo em formato de lista.

```
using System.Threading.Tasks;
using backend.Models;

namespace ProjetoMundo_API.Data
{
    public interface IRepository
    {
        (...) //resto da classe

        Task<ImagemPin[]> GetAllImagesAsyncByIdPin(int ID);
    }
}
```

Pronto! Terminamos a nossa última implementação, que serve para o IRepository saber que o Repository possui o método de coletar imagens. Agora, é só testar por meio do aplicativo Postman e colocar tudo para funcionar.

# Fontes

- Imagem da capa: https://www.google.com/search?q=asp.net+logo&tbm=isch&client=firefox-b-d&hl=pt-BR&ved=2ahUKEwis6pWxoPDoAhWfALkGHURTCZ4QrNwCKAF6BAgBEDY&biw=1349&bih=632#imgrc=Y1xsj02L_Me_nM

- Apostila ASP.NET Core 3.1 com Entity Framework SQL Server da professora Patrícia.