Novo metodo API

Como criar um novo método na API

- Novo Get API com parâmetro

Novo Get API com parâmetro

Passo a passo para criar um novo método Get na API com parâmetro.

No projeto:

Na camada Data\Model:

Criar o nome do novo método:

```
public class Produto
{
    [Key]
    [CssWidth(40)]
    public int Id { get; set; }

    [DisplayName("Código de Produto")]
    [Required(ErrorMessage = CampoObrigatorio)]
    public string CodigoProduto { get; set; }

    [Required(ErrorMessage = CampoObrigatorio)]
    public string Nome { get; set; }

}

public static class ProdutoAPI
{
    public const string Deactivate = "Deactivate";
    public const string CustomProdutoUpdate = "CustomProdutoUpdate";
    public const string ObterProdutoPorId= "ObterProdutoPorId";
}
```

No arquivo .Razor, fazer a chamada para o Novo Método:

```
protected override async Task OnAfterRenderAsync(bool firstRender)
{
    if (firstRender)
    {
        await base.OnAfterRenderAsync(firstRender);
```

```
        produtoObj = await
_produtoService.ObterProdutoPorId(Convert.ToInt32(CurrentId), LoginInfo);

        if (produtoObj is not null && produtoObj.FornecedorProdutos is not null &&
produtoObj.FornecedorProdutos.Count > 0)
            fornobj = produtoObj.FornecedorProdutos[0].Fornecedor;

        await Task.Delay(1);
        await InvokeAsync(StateHasChanged);
    }
}
```

Na camada Services: criar a nova chamada Crud para o novo método

```
using SupersoftData.Context;
using SupersoftData.Model.Comum;
using SupersoftERP.Utils;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace SupersoftERP.Services.Comum
{
    public class ProdutoService : BaseService<Produto>
    {
        public ProdutoService(ApplicationDbContext context) : base(context) { }

        public async Task<Produto> ObterProdutoPorId(int Id, string LoginInfo)
        {
            var response = await CrudHttpRequest.GetRequest<Produto>(PathApi.ErpApi,
typeof(Produto).Name.ToLower(),
 LoginInfo, null, ProdutoAPI.ObterProdutoPorId, Id.ToString());
            var responseObj = await
CrudHttpRequest.GetResponseObjectAsync<Produto>(response);
            return responseObj;
        }

        public async Task<Produto> DeactivateProdutoAsync(Produto obj, string LoginInfo)
        {
            var response = await CrudHttpRequest.PutRequest(PathApi.ErpApi,
typeof(Produto).Name.ToLower(), obj, LoginInfo,  ProdutoAPI.Deactivate);
            var responseObj = await CrudHttpRequest.GetResponseObjectAsync<Produto>(response);
            return responseObj;
        }

        public async Task<Produto> CustomUpdateProdutoAsync(Produto obj, string LoginInfo,
```

```
List<int> produtoImagemDelete = null)
        {
          var tuplaObj = new Tuple<Produto, List<int>>(item1: obj, item2: produtoImagemDelete);

          var response = await CrudHttpRequest.PutRequest(PathApi.ErpApi,
typeof(Produto).Name.ToLower(), tuplaObj, LoginInfo,     ProdutoAPI.CustomProdutoUpdate);
           var responseObj = await
CrudHttpRequest.GetResponseObjectAsync<Produto>(response);
          return responseObj;
        }
    }
}
```

Na API:

Criar o novo controller:

```
[HttpGet(ProdutoAPI.ObterProdutoPorId+ "/{id}")]
public async Task<ActionResult<Produto>>
ObterProdutoPorIdAsync([FromHeader(Name = "RequestInfo")] string encryptedCon, int
id)
{
    try
    {
       using (var conn = new ConnectionService(encryptedCon))
       {
          using (var repos = new ProdutoRepository(conn.Context))
          {
             var obj = await repos.ObterProdutoPorIdAsync(id);
             return obj;
          }
       }
    }
    catch (UnauthorizedAccessException)
    {
       return Unauthorized("Unauthorized users cannot access this method.");
    }
    catch (ArgumentNullException)
    {
       return null;
```

```
    }
}
```

Criar o novo repository:

```
public async Task<Produto> ObterProdutoPorIdAsync(int id)
{
    var obj = await _db.Produtos
                            .Include(p => p.FornecedorProdutos)
                            .FirstOrDefaultAsync(p => p.Id == Convert.ToInt32(id));
    return obj;
}
```