INTERFACE -> SERVIÇO

* Eu tenho um “IEntidadeService“ (uma interface) que herda de : **IDisposable** (liberar recursos não utilizados).
  + Nessa classe, teremos os métodos que serão implementados pela classe de “EntidadeService” (uma classe de serviço que não é uma interface).
* Na minha classe “EntidadeService”, ela vai herdar de nossa interface IEntidadeService e fazer as injeções de dependência referentes aos repositórios.
  + Trabalhar nas implementações dos métodos da interface.

REPOSITORIO GENERICO -> REPOSITORIOS PERSONALIZADOS

* Primeiro passo é criar uma interface public IRepository<TEntity> (TEntity é uma entidade genérica, podendo ser acionada por qualquer uma). Herdando de IDisposable Where TEntity : Entity. E nela, dentro, adicionar as pesquisas padrões que faremos nos repositórios. Ex:

public interface IRepository<TEntity> : IDisposable where TEntity : Entity

{

Task Adicionar(TEntity entity);

Task<TEntity> ObterPorId(Guid id);

Task<List<TEntity>> ObterTodos();

Task Atualizar(TEntity entity);

Task Remover(Guid id);

Task<IEnumerable<TEntity>> Buscar(Expression<Func<TEntity, bool>> predicate);

Task<int> SaveChanges();

}

* A nossa model, que será o tipo da entidade que será passada como parâmetro, ela precisa herdar de Entity, pois seguira com um padrão. Essa classe, Entity, é uma abstrata, não podendo ser estanciada. Ex:

public abstract class Entity

{

protected Entity()

{

Id = Guid.NewGuid();

}

public Guid Id { get; set; }

}

* Por fim, nosso repositório personalizado ele herda de : IRepository<Model> passando entre o tipo a sua model. Dentro da classe, implementamos os tipos da interface.