Criação de models e Entity Framework

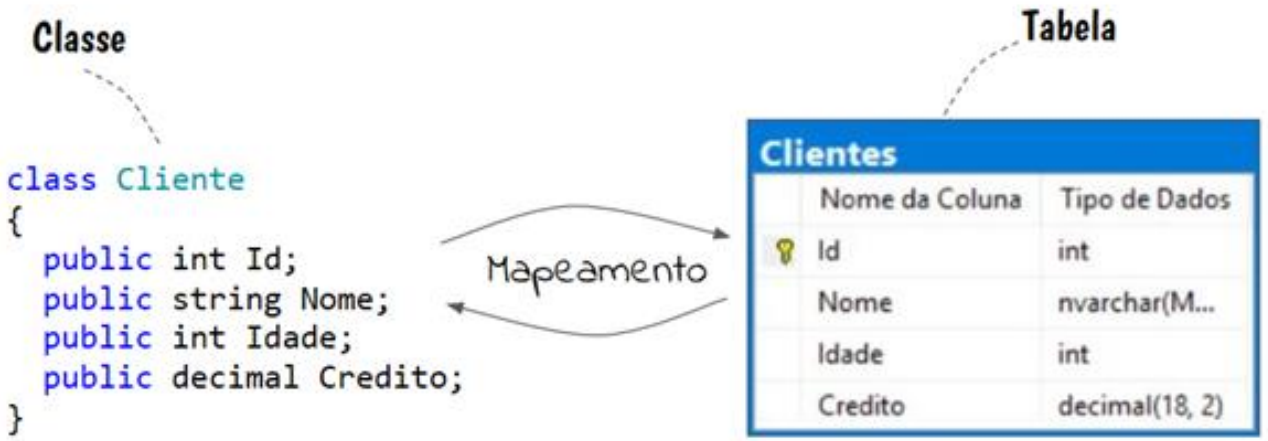
Geucimar Briatore
geucimar@up.edu.br

Atualizado em 07/2021

Objetivo

Utilizar framework de persistência para acesso a dados e mapeamento objeto-relacional.

O que é mapeamento objeto-relacional (ORM)?



Autor: Joel Rodrigues

Mapeamento objeto-relacional

- **Mapeamento objeto-relacional** (ORM) é uma técnica de conversão das classes da aplicação para tabelas do banco de dados e vice-versa;
- As relações, no mapeamento objeto-relacional, podem ser de associação **normal**, associação **agregação**, associação **composição** ou **herança**;
- Com relação à muliplicidade elas podem ser **Um-para-Um** (1..1), **Um-para-Muitos** (1..*), **Muitos-para-Muitos** (*..*) e suas variações **Zero-para-Um** (0..1) e **Zero-para-Muitos** (0..*);
- Com relação à direção elas podem ser **Unidirecional** ou **Bidirecional**.

Lista de frameworks ORM para .NET

- Biblioteca **Base One Foundation Component**, gratuita e comercial;
- **Dapper**, código aberto;
- **Entity Framework**, incluso no framework .NET;
- **iBATIS**, código aberto, mantido pela ASF, mas agora desativado;
- **NHibernate**, código aberto;
- **nHydrate**, código aberto;
- **Quick Objects**, gratuito e comercial;
- **XPO**, gratuito, suporte técnico comercial.

https://en.wikipedia.org/wiki/List_of_object-relational_mapping_software

Entity Framework

- **Entity Framework** (EF) é um mapeador objeto-relacional para trabalhar com dados relacionais em C#. O EF elimina a escrita de muitos "**códigos desnecessários**" para a gravação de dados no banco de dados;
- Para trabalhar com Entity Framework para persistência de dados é preciso incluir o EF no projeto que vai acessar as tabelas no banco de dados;
- O EF utiliza um string de conexão padrão para saber qual banco de dados utilizar. Se o banco indicado (Ex.: **Initial Catalog = ExemploDB;**) não existir o EF cria o banco na base indicada;
- No mercado, 90% dos projetos trabalham sobre um banco de dados já existente, neste caso é preciso fazer engenharia reversa das tabelas existentes no banco de dados.

Microsoft | .NET About Learn Architecture Docs Downloads Community Get Started Search

Entity Framework Documentation

Entity Framework | EF Core | EF 6

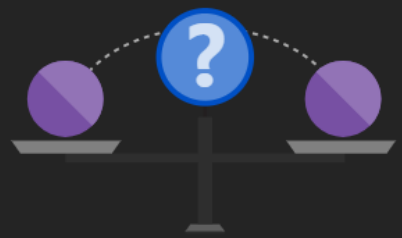
Entity Framework is an object-relational mapper (O/RM) that enables .NET developers to work with a database using .NET objects. It eliminates the need for most of the data-access code that developers usually need to write.

Entity Framework Core

EF Core is a lightweight, extensible, and cross-platform version of Entity Framework.

Entity Framework 6

EF 6 is a tried and tested data access technology with many years of features and stabilization.

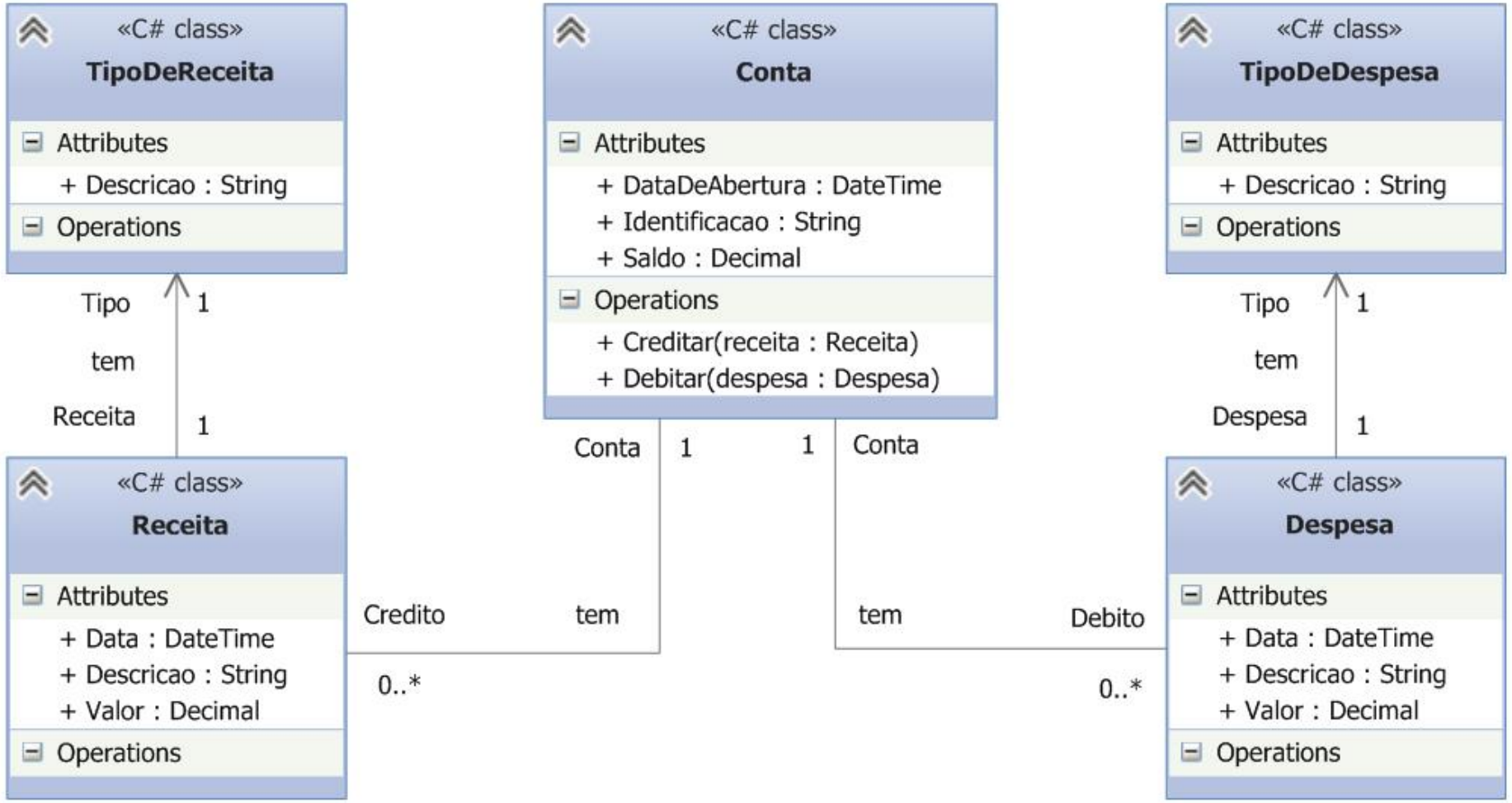


Choosing

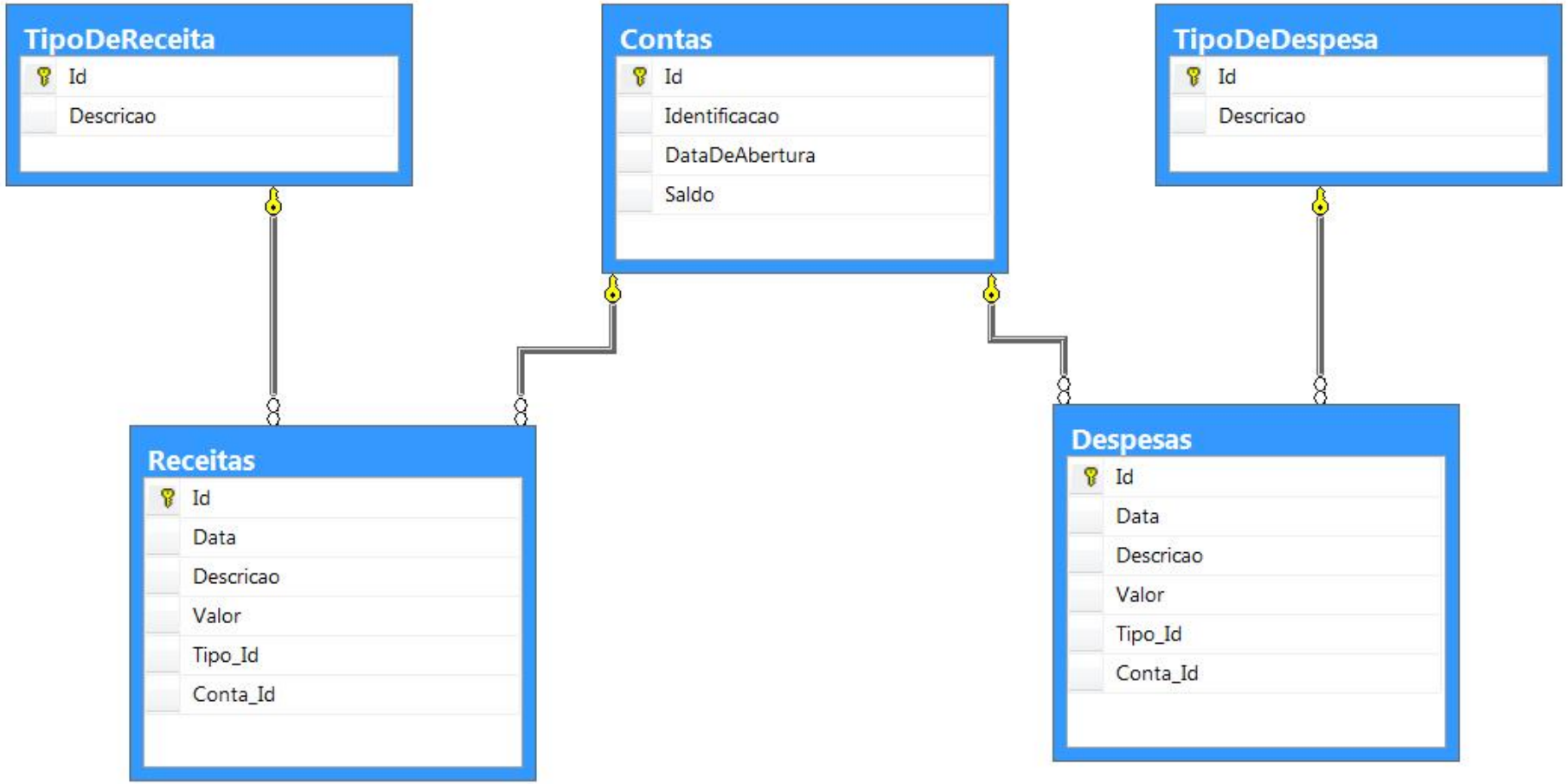
Find out which version of EF is right for you.

https://docs.microsoft.com/en-us/ef/#pivot=entityfmwk

Exemplo de diagrama de classes (UML)



Exemplo de diagrama entidade-relacionamento (DER)



Código um-para-um (totalmente definidos)

```
public class Pessoa {
    public int Id { get; set; }
    public string Nome { get; set; }
    public Endereco EnderecoResidencial { get; set;}
}

public class Endereco {
    public int Id { get; set; }
    public string Rua { get; set; }
    public string Numero { get; set; }
    public int PessoaId { get; set; }
    public Pessoa Pessoa { get; set; }
}
```

Código um-para-muitos (sem chave estrangeira)

```
public class Pessoa {
    public Pessoa () { Enderecos = new List<Endereco>(); }
    public int Id { get; set; }
    public string Nome { get; set; }
    public List<Endereco> Enderecos { get; set; }
}

public class Endereco {
    public int Id { get; set; }
    public string Rua { get; set; }
    public string Numero { get; set; }
    public Pessoa Pessoa { get; set; }
}
```

Código muitos-para-muitos

```
public class Pessoa {
    public Pessoa () { Enderecos = new List<Endereco>(); }
    public int Id { get; set; }
    public string Nome { get; set; }
    public List<Endereco> Enderecos { get; set; }
}

public class Endereco {
    public Endereco(){ Pessoas = new List<Pessoa>(); }
    public int Id { get; set; }
    public string Rua { get; set; }
    public string Numero { get; set; }
    public List<Pessoa> Pessoas { get; set; }
}
```

Configuração de relação na API fluente (DbContext)

```
public class ExemploContext : DbContext {

    protected override void OnModelCreating(ModelBuilder mb) {

        mb.Entity<Pessoa>()
            .HasOne(p => p.Endereco)
            .WithOne(e => e.Pessoa);

        ou

        mb.Entity<Pessoa>()
            .HasMany(p => p.Endereco)
            .WithOne();
    }
}
```

https://docs.microsoft.com/pt-br/ef/core/modeling/relationships?tabs=fluent-api

Mapeamento de herança com EF

- EF Core dá suporte ao padrão de tabela por hierarquia (TPH). O TPH usa uma única tabela para armazenar os dados de todos os tipos na hierarquia, e uma coluna discriminadora é usada para identificar qual tipo cada linha representa;
- Por convenção, o EF irá configurar a herança somente se dois ou mais tipos herdados forem explicitamente incluídos no DbContext;
- Também é possível configurar nome e tipo da coluna discriminadora e os valores que são usados para identificar cada tipo na hierarquia na API fluente.

https://docs.microsoft.com/pt-br/ef/core/modeling/inheritance

Passos para utilização do Entity Framework

- Passo 1: Criar a classe de domínio;
- Passo 2: Instalar o Entity Framework;
- Passo 3: Criar o DbContext;
- Passo 4: Criar o banco de dados.
- Passo 5: Gravar dados no banco.

https://www.devmedia.com.br/entity-framework-como-fazer-seu-primeiro-mapeamento-objeto-relacional/38756

Passo 1: Criar a classe de domínio

```
public class Pessoa {
    public int Id { get; set; }
    public string Nome { get; set; }
    public int Idade { get; set;}
    public int CPF { get; set;}
}
```

Passo 2: Instalar o Entity Framework (Core)

```
dotnet add package Microsoft.EntityFrameworkCore
dotnet add package Microsoft.EntityFrameworkCore.SqlServer
dotnet add package Microsoft.EntityFrameworkCore.Design
```

Passo 3: Criar o DbContext com string de conexão

O objeto **DbContext** é responsável por gerenciar todas as transações com o banco de dados.

```
public class ExemploContext : DbContext {
    //public ExemploContext() : base("ConexaoExemplo"){}
    public DbSet<Pessoa> Pessoas { get; set; }
    public DbSet<Endereco> Enderecos { get; set; }
    protected override void OnConfiguring(DbContextOptionsBuilder dbcob) {
        dbcob.UseSqlServer(@"Server=(localdb)\mssqllocaldb;
            Database=Blogging;Integrated Security=True");
    }
}
```

Passo 4: Gerar o banco de dados

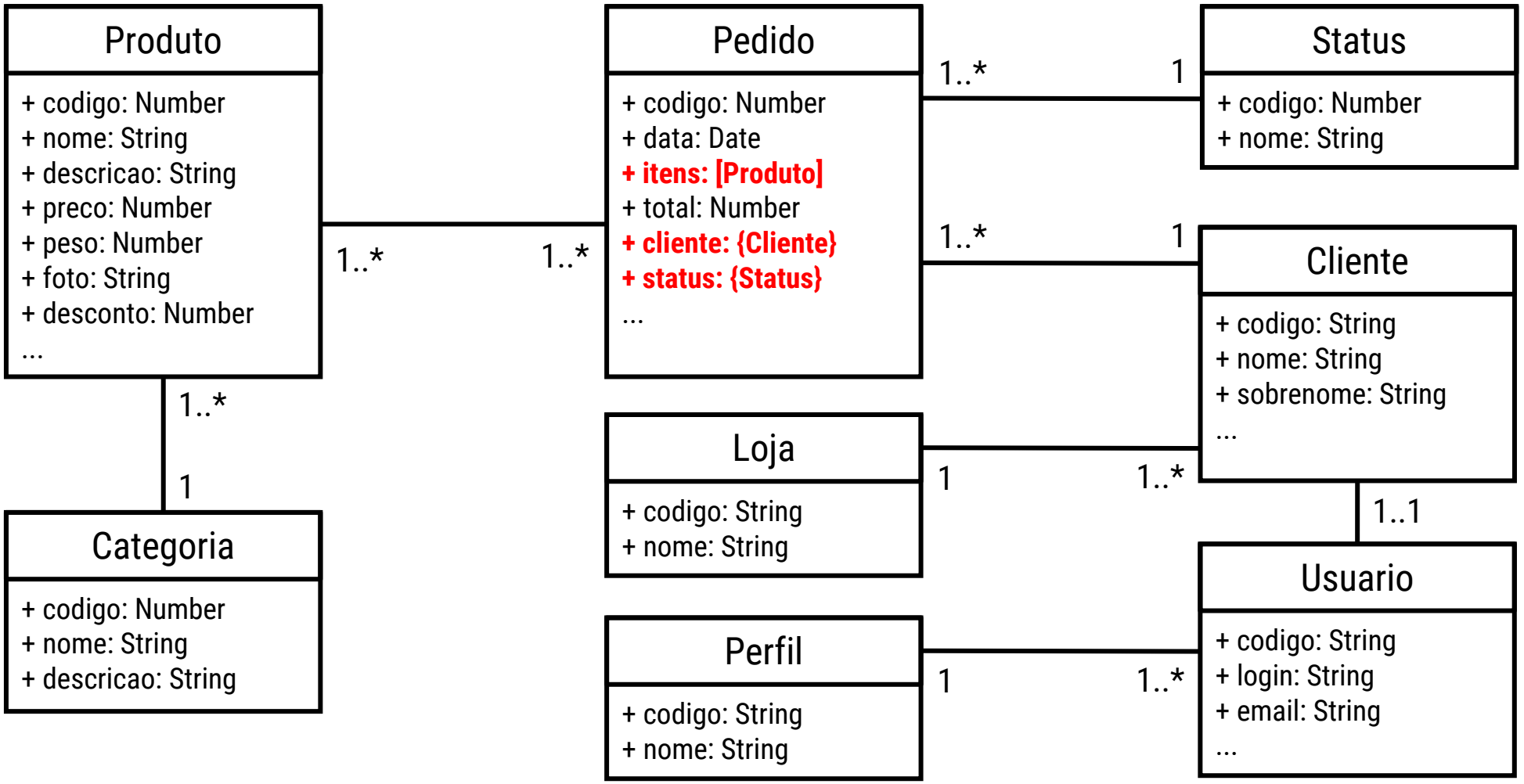
```
dotnet tool install dotnet-ef
dotnet ef migrations add Initial
dotnet ef database update

//dotnet ef migrations remove
```

Passo 5: Gravar dados no banco

Para salvar os objetos mapeados no objeto **DbContext** bastar criar as classes e adicioná-las ao contexto e invocar **SaveChanges()**;

```
public class Exemplo
{
    static void Main(string[] args)
    {
        ExemploContext ctx = new ExemploContext();
        ctx.Pessoas.Add(new Pessoa("João"));
        ctx.SaveChanges();
    }
}
```

Modelo de dados MyFood



Atividade 3

- Fazer o mapeamento objeto-relacional do projeto disciplinar;
- Versionar as classes e apresentar na próxima aula.