

FIA/P GRADUAÇÃO

# ENTERPRISE APPLICATION DEVELOPMENT

Prof. Me. Thiago T. I. Yamamoto

#07 - RELACIONAMENTOS

# TRAJETÓRIA

---



- ✓ Plataforma .NET
- ✓ Linguagem C# e Orientação a Objetos
- ✓ ASP.NET Core – Rotas e Controller
- ✓ ASP.NET Core – Razor e Tag Helpers
- ✓ ASP.NET Core – Layout e Partial Views
- ✓ Entity Framework Core
- ✓ Entity Framework Core - Relacionamentos

# #07 - AGENDA

---

- Relacionamentos
- Um-para-um
- Um-para-muitos
- Muitos-para-muitos



- Existem 3 tipos de relacionamentos:
  - **Uma-para-um** (One to one): é um relacionamento onde uma entidade só possui uma ligação com outra entidade e esta só possui a ligação de volta;
  - **Um-para-muitos** (One to Many): uma entidade possui várias ligações com outra entidade e esta só possui a ligação de volta;
  - **Muitos-para-muitos** (Many to many): uma entidade possui várias ligações com outra entidade e esta possui também várias ligações de volta;



# UM-PARA-UM

- Para mapear um relacionamento **um-para-um** é preciso adicionar uma **propriedade de navegação** para a outra entidade:

```
public class Cliente
{
    public int ClientId { get; set; }
    public string Nome { get; set; }
    public CarteiraMotorista Cnh { get; set; }
}
```

Propriedade de navegação para a classe CarteiraMotorista.

```
public class CarteiraMotorista
{
    public int CarteiraMotoristaId { get; set; }
    public DateTime DataValidade { get; set; }
}
```

- Podemos adicionar também uma propriedade de **Foreign Key** na classe para facilitar a manipulação da relação:

```
public class Cliente
{
    public int ClientId { get; set; }
    public string Nome { get; set; }
    public CarteiraMotorista Cnh { get; set; }
    public int CarteiraMotoristaId { get; set; }
}
```

Mapeia a **Chave estrangeira**, deve ter o mesmo nome da propriedade da chave primária da classe **CarteiraMotorista**.





# UM-PARA-MUITOS

- Adicione uma propriedade de navegação para a outra entidade, porém o tipo da propriedade será **ICollection**;

```
public class Cliente
{
    public int ClientId { get; set; }
    public string Nome { get; set; }
    public virtual ICollection<Qualificacao> Qualificacao { get; set; }
}
```

Propriedade do tipo **ICollection** de **Qualificacao**.

```
public class Qualificacao
{
    public int QualificacaoId { get; set; }
    public int Estrelas { get; set; }
}
```

- Podemos ter um relacionamento **bi-direcional**:

```
public class Cliente
{
    public int ClientId { get; set; }
    public string Nome { get; set; }
    public virtual ICollection<Qualificacao> Qualificacao { get; set; }
}
```

Relacionamento **bi-direcional**

```
public class Qualificacao
{
    public int QualificacaoId { get; set; }
    public int Estrelas { get; set; }
    public Cliente Cliente { get; set; }
    public int ClientId { get; set; }
}
```



# MUITOS-PARA-MUITOS

- O primeiro passo é criar uma **nova classe** para mapear a **tabela associativa**:

```
public class ClienteVeiculo
{
    public int ClientId { get; set; }
    public Cliente Cliente { get; set; }
    public int VeiculoId { get; set; }
    public Veiculo Veiculo { get; set; }
}
```

- Implemente o **relacionamento** com a tabela associativa em ambas as classes:

```
public class Veiculo
{
    public int Veiculoid { get; set; }
    public string Modelo { get; set; }
    public ICollection<ClienteVeiculo> ClienteVeiculo { get; set; }
}
```

```
public class Cliente
{
    public int Clienteid { get; set; }
    public string Nome { get; set; }
    public ICollection<ClienteVeiculo> ClienteVeiculo { get; set; }
}
```

- Para finalizar, na **classe** de **contexto**, sobrescreva o método **OnModelCreating()** para configurar a tabela associativa através de **Fluent API**:

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<ClienteVeiculo>()
        .HasKey(c => new { c.ClientId, c.VeiculoId });
    modelBuilder.Entity<ClienteVeiculo>()
        .HasOne(c => c.Veiculo)
        .WithMany(c => c.ClienteVeiculo)
        .HasForeignKey(c => c.VeiculoId);
    modelBuilder.Entity<ClienteVeiculo>()
        .HasOne(c => c.Cliente)
        .WithMany(c => c.ClienteVeiculo)
        .HasForeignKey(c => c.ClientId);
    base.OnModelCreating(modelBuilder);
}
```

# VOCÊ APRENDEU..

---

- Como mapear relacionamentos com Entity Framework Core;
- Mapear relacionamentos Uma-para-um, Um-para-Muitos e Muitos-para-Muitos;





# Copyright © 2013 – 2023

## Prof. Me. Thiago T. I. Yamamoto

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).

*A vida vai ficando cada vez mais dura perto do topo.  
Friedrich Nietzsche*