ETEC PROF. HORÁCIO AUGUSTO DA SILVEIRA

DESENVOLVIMENTO DE SISTEMAS – 2023/1

Luiz Fernando Souza / Quitéria Danno

Aula 09 – Relacionamentos: One to Many, One to One e Many to Many

Com a criação da classe **Usuario** criaremos uma tabela no banco de dados chamada ***Usuarios*** através do mapeamento da classe e das migrações. Esta classe e tabela serão importantes, pois, criará um relacionamento com os dados do personagem do tipo um para muitos, em que um usuário poderá ter diversos personagens atrelados a ele.

No banco de dados usaremos um tipo de dado chamado de hash para não expor a senha do usuário e um salt que nada mais é do que caracteres que são concatenados combinados antes, durante ou depois do hash a fim de evitar que a senha seja descoberta com técnicas de quebras de segurança. Mais detalhes poderão ser entendidos com as referências abaixo:

- Hash e Salt de senhas: https://www.brunobrito.net.br/seguranca-salt-hash-senha/
- Exemplo de criação de hash em C#: https://www.youtube.com/watch?v=ggPgk4znUEY

1. Abra o projeto **RpgApi** e crie a classe **Usuario.cs** dentro da pasta Models, codificando conforme a seguir.

```
1 reference
public int Id { get; set; } //Atalho para propridade (PROP + TAB)
1 reference
public string Username { get; set; }
1 reference
public byte[] PasswordHash { get; set; }
1 reference
public byte[] PasswordSalt { get; set; }
0 references
public byte[] Foto { get; set; }
0 references
public double? Latitude { get; set; }
0 references
public double? Longitude { get; set; }
0 references
public DateTime? DataAcesso { get; set; } //using System;

[NotMapped] // using System.ComponentModel.DataAnnotations.Schema
1 reference
public string PasswordString { get; set; }
0 references
public List<Personagem> Personagens { get; set; }//using System.Collections.Generic;
2 references
public string Perfil { get; set; }
0 references
public string Email { get; set; }
```

- Note que além das propriedades normais estamos criando uma lista de personagens. Isso definirá que um Usuário pode possuir vários personagens.

---

2. Abra a classe Personagem e adicione a codificação sinalizada. Para saber para qual Usuário um objeto do tipo Personagem estará atrelado, faremos a declaração do objeto na classe Personagem conforme abaixo. Vamos aproveitar e criar uma propriedade que futuramente armazenará a foto do Personagem.

```
public ClasseEnum Classe { get; set; }
0 references
public byte[] FotoPersonagem { get; set; }
0 references
public Usuario Usuario { get; set; }
```

3. Crie uma pasta chamada **Utils** e dentro dela crie a classe **Criptografia** e adicione o método abaixo. Esse método é estático, ou seja, não precisará da classe estanciada para chama-lo futuramente.

```
public static void CriarPasswordHash(string password, out byte[] hash, out byte[] salt)
{
    using (var hmac = new System.Security.Cryptography.HMACSHA512())
    {
        salt = hmac.Key;
        hash = hmac.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password));
    }
}
```

4. Abra a classe DataContext e adicione a referência à classe Usuario recém-criada para o contexto do banco de dados, o que chamamos de mapeamento. Procure a região onde temos outros mapeamentos feitos.

```
public DbSet<Usuario> Usuarios { get; set; }
```

5. Ainda na classe DataContext, posicione o cursor antes do fechamento do método OnModelCreating para preparar um usuário padrão para quando a tabela for alimentada. Exigirá o using para RpgApi.Utils para reconhecer a classe Criptografia.

```
        new Arma() { Id = 6, Nome = "Foice", Dano = 33 },
        new Arma() { Id = 7, Nome = "Cajado", Dano = 32 }
    );
    //Início da criação do usuário padrão.
    Usuario user = new Usuario();
    Criptografia.CriarPasswordHash("123456", out byte[] hash, out byte[]salt);
    user.Id = 1;
    user.Username = "UsuarioAdmin";
    user.PasswordString = string.Empty;
    user.PasswordHash = hash;
    user.PasswordSalt = salt;
    user.Perfil = "Admin";
    user.Email = "seuEmail@gmail.com";
    user.Latitude = -23.5200241;
    user.Longitude = -46.596498;

    modelBuilder.Entity<Usuario>().HasData(user);
    //Fim da criação do usuário padrão.

    //Define que se o Perfil não for informado, o valor padrão será jogador
    modelBuilder.Entity<Usuario>().Property(u => u.Perfil).HasDefaultValue("Jogador");
}//Fim do método OnModelCreating
```