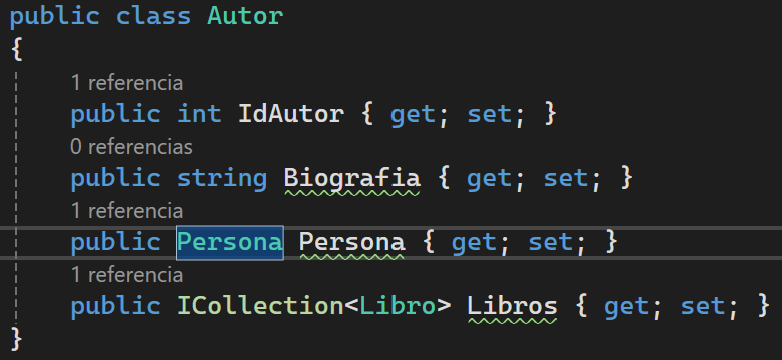
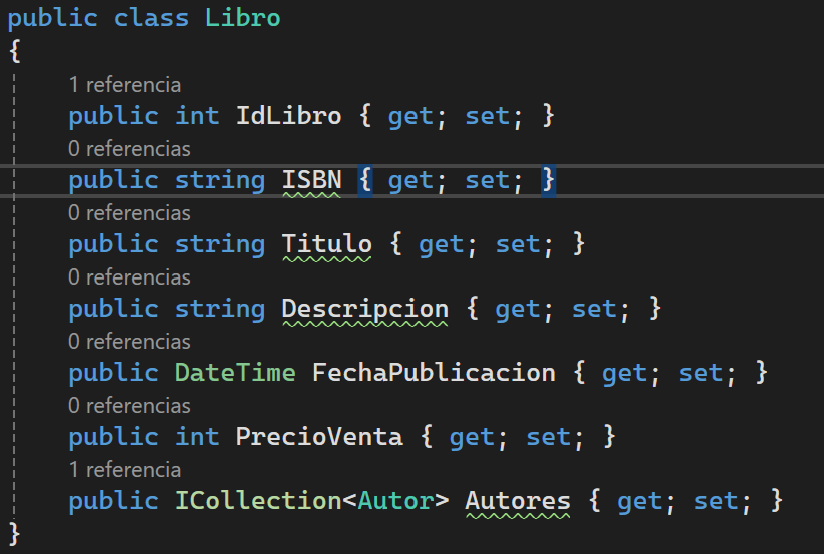
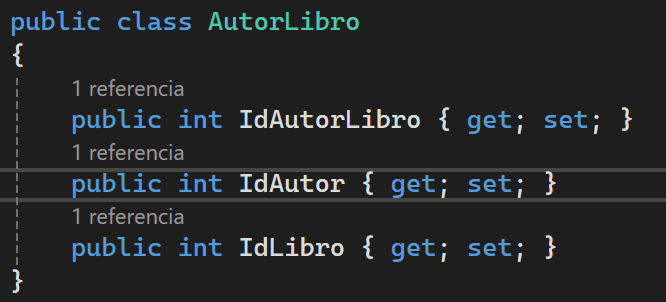
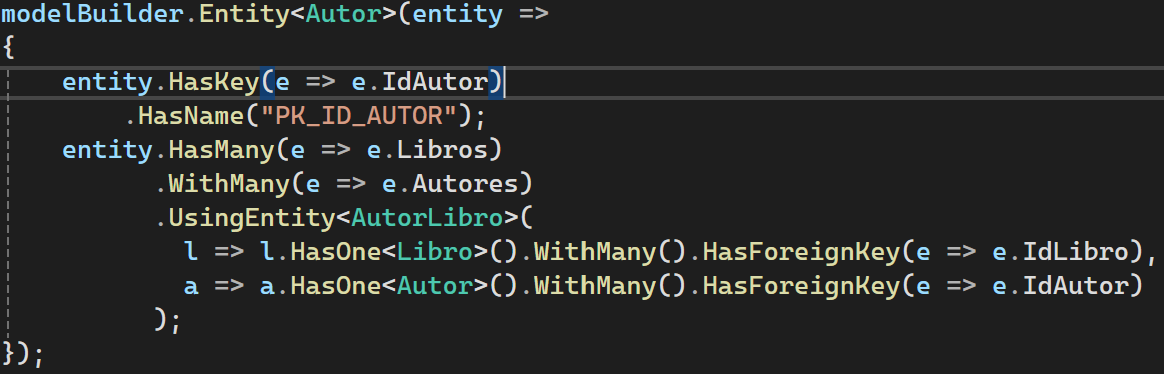
Para relaciones (n;n) se hace una tabla intermedia, y en cada tabla relacionada se hace un listado de objetos del otro tipo de tabla

ejemplo:





En llibreria context



que teóricamente sería (muchos a muchos)::

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Post>()

.HasMany(e => e.Tags)

.WithMany(e => e.Posts)

.UsingEntity(

"PostTag",

l => l.HasOne(typeof(Tag)).WithMany().HasForeignKey("TagsId").HasPrincipalKey(nameof(Tag.Id)),

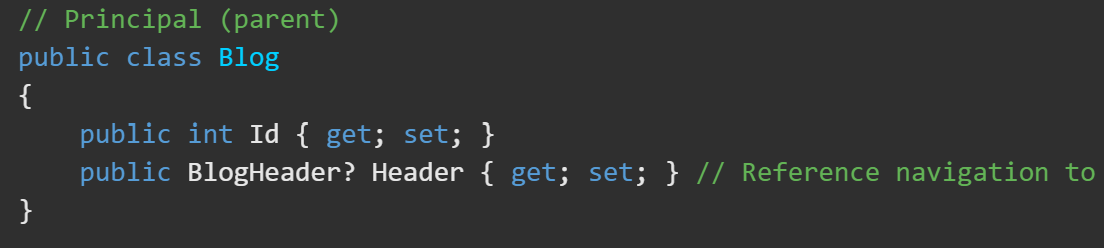
r => r.HasOne(typeof(Post)).WithMany().HasForeignKey("PostsId").HasPrincipalKey(nameof(Post.Id)),

j => j.HasKey("PostsId", "TagsId"));

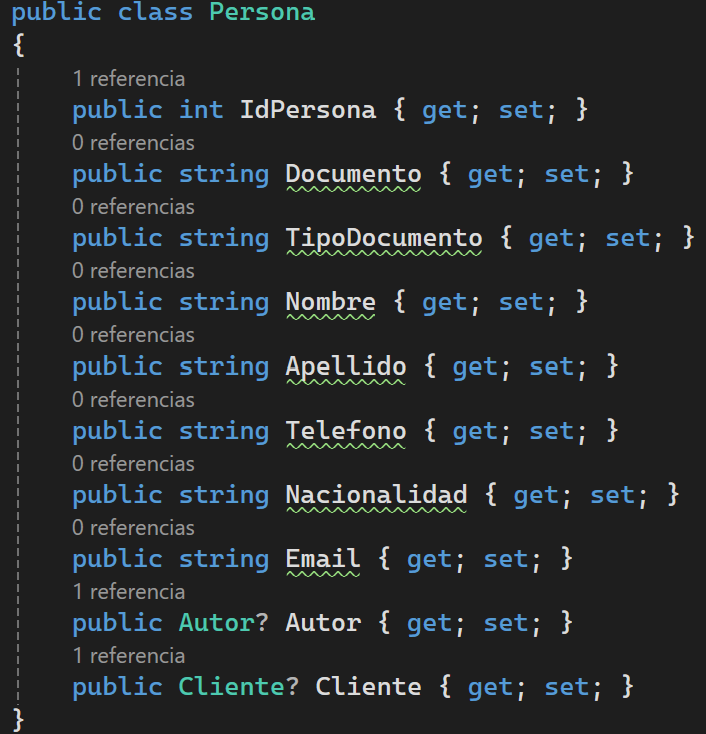
}

Para una relación 1:1 (ejemplo persona - autor o persona - cliente)

En el código, teóricamente sería:

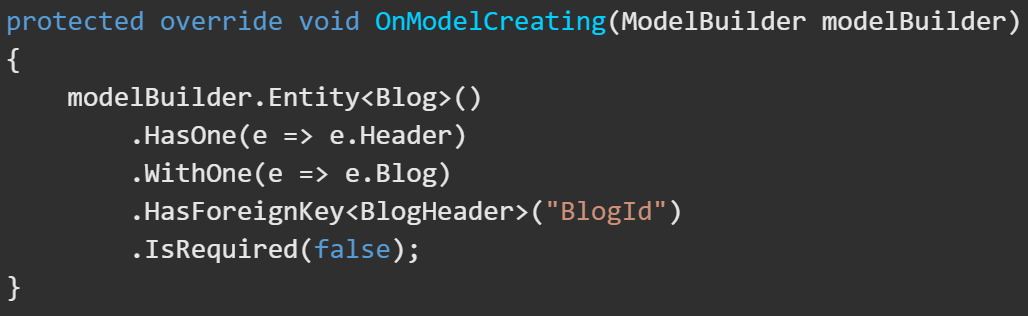


En el ejemplo (aplicado para autor y cliente (hijos), en persona (padre) se veria asi:

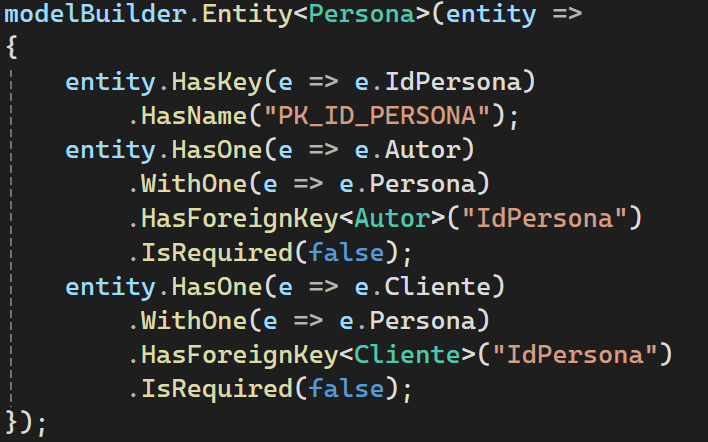


Siendo la última línea de la teoría, las dos últimas líneas del ejemplo (Y la primera el id del padre). En la clase cliente y autor hay que declarar persona

En libreria context teoricamente seria:

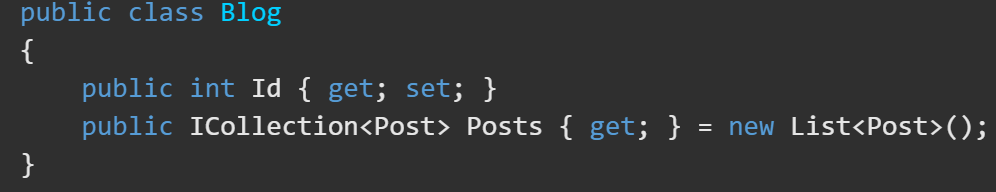


Como las FK están en Autor y cliente, conviene ponerlas en persona

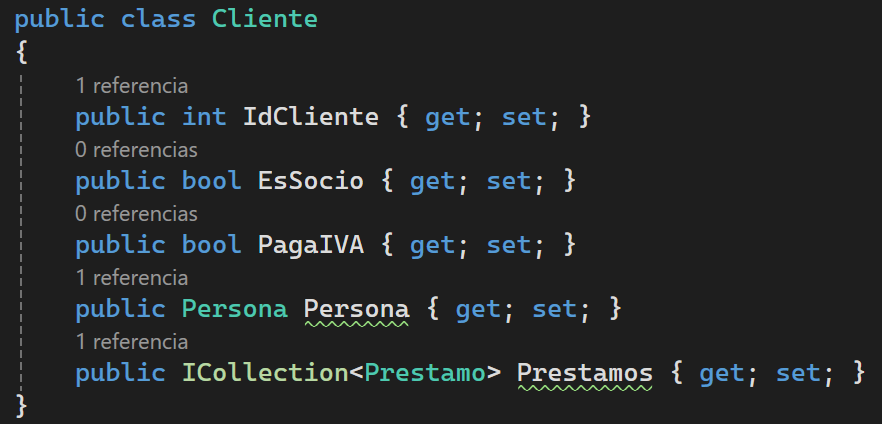


Para relacion (1;n) ejemplo cliente - préstamo

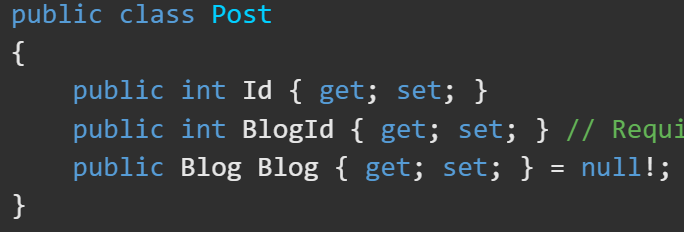
En la entidad padre (la que se relaciona a muchas otras entidades) teóricamente es:



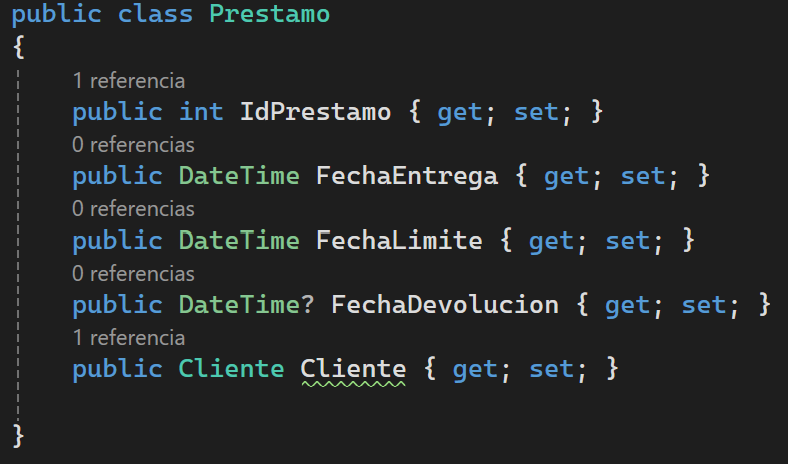
En el ejemplo es Cliente, y quedaría:



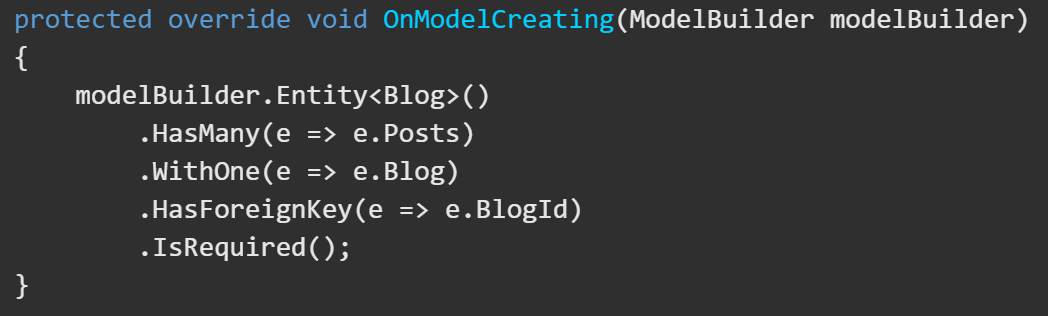
En la clase hijo teóricamente es:



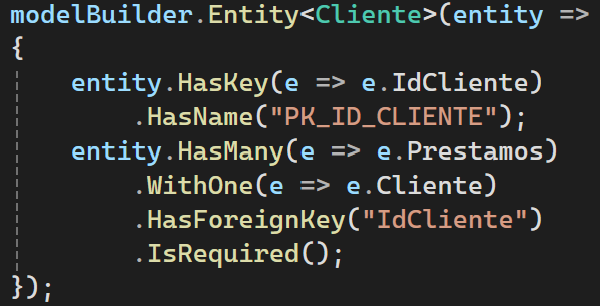
En el ejemplo es pedido, y quedaría:



En el Libreria Context en el padre teóricamente va:



Y en el ejemplo, en la clase cliente (padre) va:



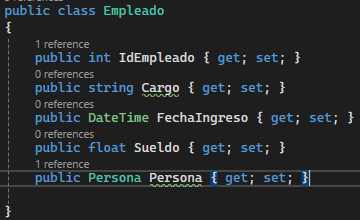
Se usa shadow id para no poner el idCliente en préstamo

13:31 migracion

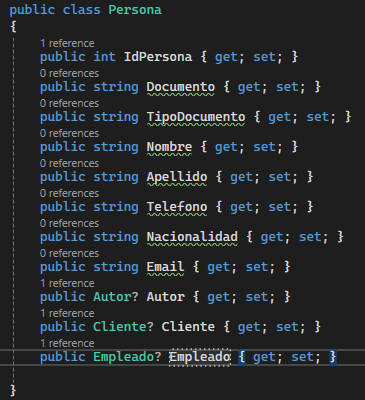
Ejercitación:

Relación 1:1 (persona - empleado)

Delaclaramos Persona en Empleado



y Empleado en Persona



En libreria context buscamos a persona y agregamos:

entity.HasOne(e => e.Empleado)

.WithOne(e => e.Persona)

.HasForeignKey<Cliente>("IdPersona")

.IsRequired(false);

Relación 1:n (empleado - préstamo)(padre-hijo)

En código:

Entidad padre (empleado) se le agrega la colección de préstamos

Entidad hijo (préstamo) se le agrega un campo Empleado (tipo empleado)

En Librería:

En la entidad padre (Empleado) va:

.WithOne(e => e.Empleado)

.HasForeignKey("IdEmpleado")

.IsRequired();

Relación 1:n (FormaPago - préstamo)

En código:

Entidad padre (FP) se le agrega la colección de préstamos

Entidad hijo (préstamo) se le agrega un campo FP (tipo FP)

En la libreria:

En la entidad padre (FormaPago) va:

entity.HasMany(e => e.Prestamos)

.WithOne(e => e.FormaPago)

.HasForeignKey("IdFormaPago")

.IsRequired();

VENTA

Relación 1:n (Cliente-Venta)

En código:

Entidad padre (Cliente) se le agrega la colección de Ventas

Entidad hijo (Ventas) se le agrega un campo Cliente (tipo Cliente)

En la libreria:

En la entidad padre (Cliente) va:

entity.HasMany(e => e.Venta) hijo

.WithOne(e => e.Cliente) padre

.HasForeignKey("IdCliente") padre

.IsRequired();

Relación 1:n (Empleado-Venta)

En código:

Entidad padre (Empleado) se le agrega la colección de Ventas

Entidad hijo (Ventas) se le agrega un campo Empleado (tipo Empleado)

En la libreria:

En la entidad padre (Empleado) va:

entity.HasMany(e => e.Venta)

.WithOne(e => e.Empleado)

.HasForeignKey("IdEmpleado")

.IsRequired();

Relación 1:n (FormaPago-Venta)

En código:

Entidad padre (FormaPago) se le agrega la colección de Ventas

Entidad hijo (Venta) se le agrega un campo FormaPago (tipo FormaPago)

En la libreria:

En la entidad padre (FormaPago) va:

entity.HasMany(e => e.Venta)

.WithOne(e => e.FormaPago)

.HasForeignKey("idFormaPago")

.IsRequired();

LIBRO

Relación 1:n (Editorial-Libro)

En código:

Entidad padre (Editorial) se le agrega la colección de Libro

Entidad hijo (Libro) se le agrega un campo Editorial (tipo Editorial)

En la libreria:

En la entidad padre (Editorial) va:

entity.HasMany(e => e.Libro)

.WithOne(e => e.Editorial)

.HasForeignKey("IdEditorial")

.IsRequired();

Relación 1:n (Libro-CopiaLibro)

En código:

Entidad padre (Libro) se le agrega la colección de CopiaLibro

Entidad hijo (CopiaLibro) se le agrega un campo Libro (tipo Libro)

En la libreria:

En la entidad padre (Libro) va:

entity.HasMany(e => e.CopiaLibro)

.WithOne(e => e.Libro)

.HasForeignKey("IdLibro")

.IsRequired();

Relacion n:m Genero-Libro

Se crea una clase intermedia GeneroLibro

Se agregan los id de Genero y Libro

En la clase Genero agregamos un listado de libros

En la clase Libro agregamos un listado de Genero

En la libreria Context (Genero):

entity.HasMany(e => e.Libros)

.WithMany(e => e.Generos)

.UsingEntity<GeneroLibro>(

l => l.HasOne<Libros>().WithMany().HasForeignKey(e => e.IdLibro),

r => r.HasOne<Generos>().WithMany().HasForeignKey(e => e.IdGenero));

});

Relacion n:m Prestamo-CopiaLibro

Se crea una clase intermedia PrestamoCopiaLibro

Se agregan los id de Prestamo y CopiaLibro

En la clase Prestamo agregamos un listado de CopiasLibro

En la clase CopiaLibro agregamos un listado de Prestamos

En la libreria Context (Prestamo):

entity.HasMany(e => e.Prestamos)

.WithMany(e => e.CopiaLibros)

.UsingEntity<PrestamoCopiaLibro>(

l => l.HasOne<CopiaLibro>().WithMany().HasForeignKey(e => e.IdCopiaLibro),

r => r.HasOne<Prestamo>().WithMany().HasForeignKey(e => e.IdPrestamo));

});

Relacion n:m Libro-Venta

Se crea una clase intermedia LibroVenta

Se agregan los id de Libro y Venta

En la clase Libro agregamos un listado de Ventas

En la clase Venta agregamos un listado de Libros

En la libreria Context (Libro):

entity.HasMany(e => e.Ventas)

.WithMany(e => e.Libros)

.UsingEntity<LibroVenta>(

l => l.HasOne<Venta>().WithMany().HasForeignKey(e => e.IdVenta),

r => r.HasOne<Libro>().WithMany().HasForeignKey(e => e.IdLibro));

});