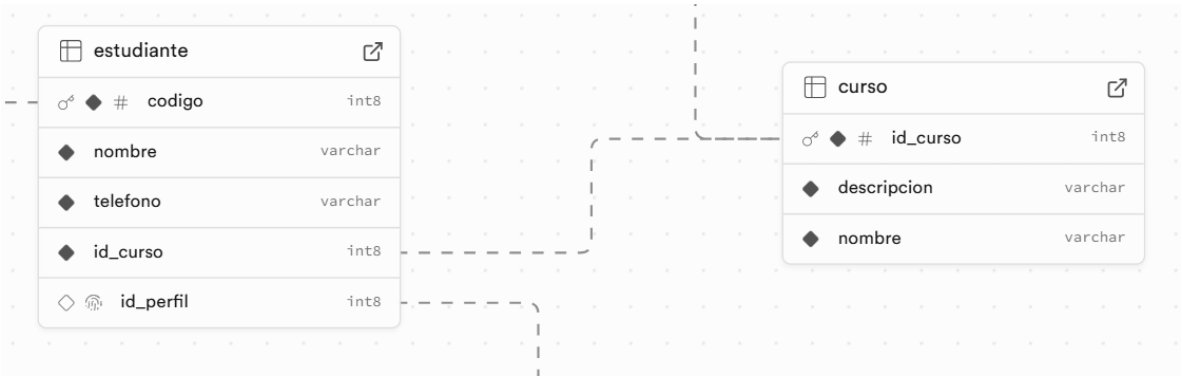
Jeimy Alejandra Herrera Rincón

Relaciones entre Entidades

Relación de uno a muchos: Un curso puede tener muchos estudiantes



Entidad Estudiante

```
@ManyToOne  3 usages
@JoinColumn(name = "id_curso", nullable = false)
private Curso curso;
```

@ManyToOne: indica que la entidad Estudiante tiene una relación de muchos a uno con la entidad Curso.

@JoinColumn (name = "id_curso", nullable = false) : se utiliza para especificar la columna en la base de datos que se usará para la relación. En este caso, se está indicando que la columna id_curso de la tabla Estudiante será la clave foránea para relacionar las dos tablas y que este campo no puede ser nulo.

Private Curso curso: declara un atributo de tipo Curso en la clase Estudiante. Este atributo representa la relación entre el estudiante y el curso al que está inscrito, permitiendo el acceso al curso asociado a un estudiante

Entidad Curso

```
@OneToMany(mappedBy = "curso", cascade = CascadeType.ALL, fetch = FetchType.LAZY)
private List<Estudiante> estudiantes;
```

@OneToMany: indica que la entidad Curso tiene una relación de uno a muchos con la entidad Estudiantes. En este caso, se está definiendo que un Curso puede tener múltiples Estudiantes.

mappedBy = "curso" : especifica el nombre del campo en la entidad Estudiante en donde hay un campo llamado curso que hace referencia a la entidad Curso. Con mappedBy, se indica

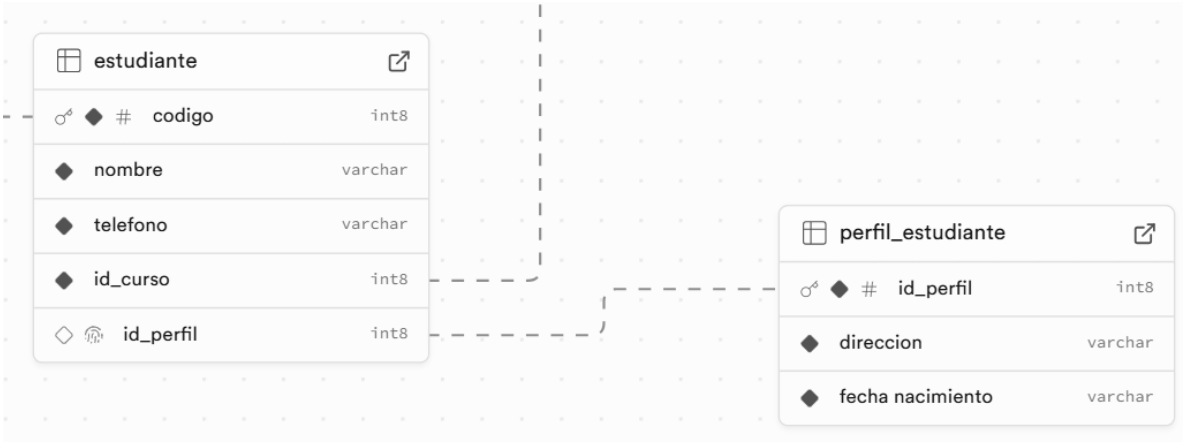
que la relación no es gestionada por la entidad Curso, sino por la entidad Estudiante. Esto es importante para evitar la duplicación de la relación y para que JPA sepa que la clave foránea se encuentra en la tabla de Estudiante.

cascade = CascadeType.ALL : define el comportamiento de la cascada para las operaciones de persistencia. CascadeType.ALL significa que todas las operaciones de estado (crear, actualizar, eliminar, etc.) realizadas en la entidad Curso se aplicarán automáticamente a las entidades Estudiantes.

fetch = FetchType.LAZY : define la estrategia de carga para la relación. FetchType.LAZY indica que los Estudiantes asociados a un Curso no se cargarán automáticamente cuando se cargue el Curso. En su lugar, se cargarán solo cuando se acceda explícitamente a la colección de Estudiantes.

private List<Estudiante> estudiantes : declara un atributo de tipo List<Estudiante> en la clase Curso. Este atributo representa la colección de estudiantes que están inscritos en el curso y permite el acceso a la lista de estudiantes asociados a un curso

Relación de uno a uno: un perfil para un estudiante



Entidad Estudiante

```
@OneToOne(cascade = CascadeType.ALL)  3 usages
@JoinColumn(name = "id_perfil", referencedColumnName = "id")
private PerfilEstudiante perfilEstudiante;
```

@OneToOne(cascade = CascadeType.ALL) : OneToOne indica que existe una relación uno a uno entre la entidad Estudiante y la entidad PerfilEstudiante. Cascade = CascadeType.ALL especifica que todas las operaciones de persistencia (como persist, merge, remove, etc.) realizadas en la entidad Estudiante se aplicarán automáticamente a la entidad PerfilEstudiante

@JoinColumn (name = "id_perfil", referencedColumnName = "id") : especifica la columna en la base de datos que se usará para la relación. En este caso, se está indicando que la columna que almacenará la referencia al PerfilEstudiante en la tabla de Estudiante se llamará id_perfil. name = "id_perfil" se encarga de definir el nombre de la columna en la tabla Estudiante que actuará como clave foránea, apuntando a la clave primaria de la tabla PerfilEstudiante. referencedColumnName = "id" especifica que la columna id_perfil en la tabla Estudiante se refiere a la columna id en la tabla PerfilEstudiante. Esto establece la relación entre las dos tablas en la base de datos.

private PerfilEstudiante perfilEstudiante : declara un atributo de tipo PerfilEstudiante en la clase Estudiante. Este atributo representa la relación entre el estudiante y su perfil asociado.

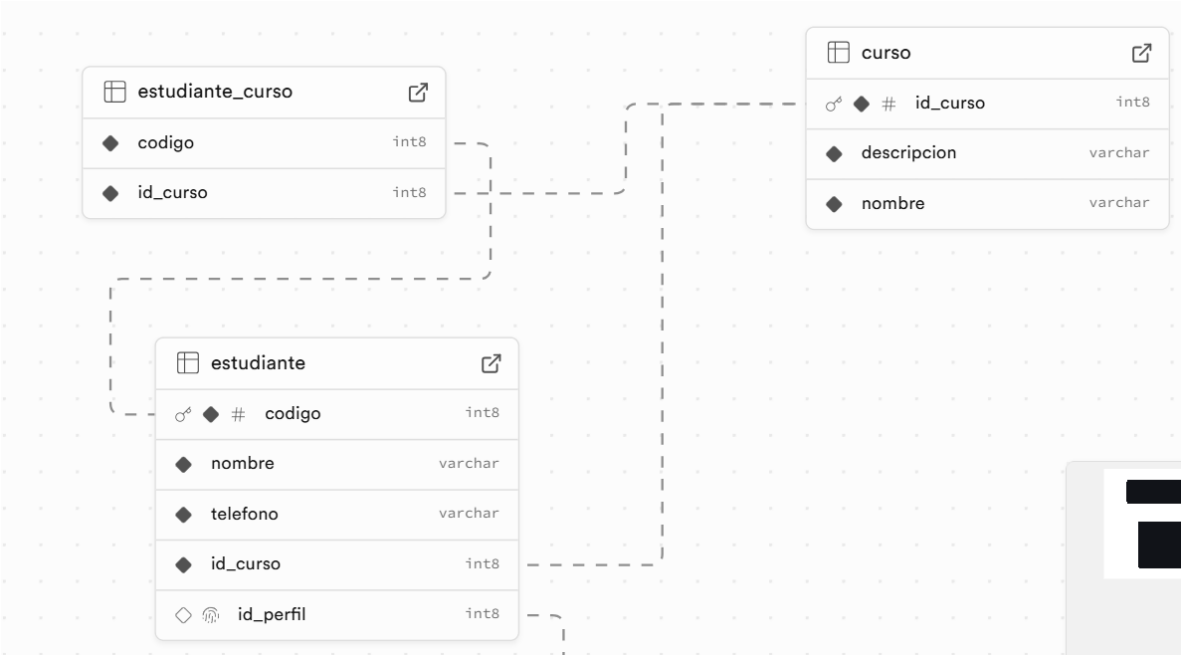
Entidad PerfilEstudiante

```
@OneToOne(mappedBy = "perfilEstudiante", cascade = CascadeType.ALL, fetch =  FetchType.LAZY)
private Estudiante estudiante;
```

@OneToOne(mappedBy = "perfilEstudiante", cascade = CascadeType.ALL, fetch = FetchType.LAZY) : OneToOne indica que existe una relación uno a uno entre la entidad PerfilEstudiante y la entidad Estudiante. mappedBy = "perfilEstudiante" especifica que la relación es gestionada por el campo perfilEstudiante en la clase Estudiante, de modo que la clave foránea que establece la relación se encuentra en la tabla de Estudiante.

private Estudiante estudiante : declara un atributo de tipo Estudiante en la clase PerfilEstudiante. Este atributo representa la relación entre el perfil y el estudiante asociado.

Relación de muchos a muchos: Muchos estudiantes pueden tener muchos cursos



Entidad estudiante

```
@ManyToMany(cascade = CascadeType.ALL)
@JoinTable(
        name = "Estudiante_Curso",
        joinColumns = @JoinColumn(
                name = "codigo",
                referencedColumnName = "codigo"
        ),
        inverseJoinColumns = @JoinColumn(
                name = "id_curso",
                referencedColumnName = "id_curso"
        )
)
private List<Curso> cursosList;
```

@ManyToMany(cascade = CascadeType.ALL) : @ManyToMany indica que existe una relación de muchos a muchos entre la entidad Estudiante y la entidad Curso. Esto significa que un estudiante puede estar inscrito en múltiples cursos y, a su vez, un curso puede tener múltiples estudiantes inscritos.

@JoinTable : definir la tabla intermedia que se utilizará para gestionar la relación muchos a muchos. En este caso, la tabla intermedia se llamará Estudiante_Curso declarado con name = "Estudiante_Curso".

joinColumns = @JoinColumn (name = "codigo", referencedColumnName = "codigo") : joinColumns = @JoinColumn(...) define las columnas en la tabla intermedia que se utilizarán para referenciar la entidad Estudiante. name = "codigo" especifica el nombre de la columna en la tabla intermedia que actuará como clave foránea para la entidad Estudiante y referencedColumnName = "codigo" indica que la columna codigo en la tabla intermedia se refiere a la columna codigo en la tabla Estudiante. Esto establece la relación entre las dos tablas

inverseJoinColumns = @JoinColumn(...) : define las columnas en la tabla intermedia que se utilizarán para referenciar la entidad Curso.

private List<Curso> cursosList : declara un atributo de tipo List<Curso> en la clase Estudiante. Este atributo representa la lista de cursos en los que está inscrito el estudiante.

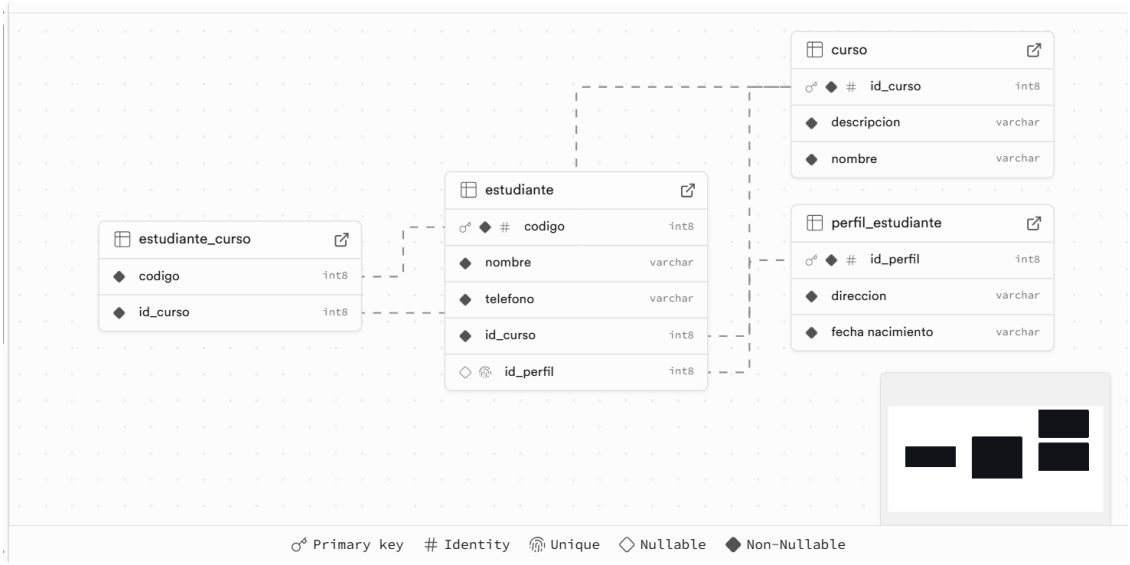
Entidad Curso

```
@ManyToMany(mappedBy = "cursosList", cascade = CascadeType.ALL)
private List<Estudiante> estudiantesList;
```

@ManyToMany(mappedBy = "cursosList", cascade = CascadeType.ALL) : @ManyToMany indica que existe una relación de muchos a muchos entre la entidad Curso y la entidad Estudiante. Esto significa que un curso puede tener múltiples estudiantes inscritos y, a su vez, un estudiante puede estar inscrito en múltiples cursos. mappedBy = "cursosList": especifica que la relación es gestionada por el campo cursosList en la clase Estudiante de modo que la clave foránea que establece la relación se encuentra en la tabla de Estudiante.

private List<Estudiante> estudiantesList : declara un atributo de tipo List<Estudiante> en la clase Curso. Este atributo representa la lista de estudiantes que están inscritos en el curso.

Esquema completo



Etiquetas Lombock
@NoArgsConstructor : Crea un constructor vacío de forma automática.
@AllArgsConstructor : Crea un constructor que incluye todos los parámetros de forma automática.
@Setter : Crea los setter para cada parámetro
@Getter : Crea los getter para cada parámetro

Etiquetas definición de entidades

@Table(name = "Estudiante") : @Table se utiliza para definir las propiedades de la tabla y name para establecer el nombre que tendra

@Column(name = "Nombre", nullable = false, length = 20) : @Column se utiliza para definir las propiedades de las columnas de la tabla. Name me indica el nombre la columna, nullable=false que no puede ser falso y lenght la longitud máxima de los caracteres.

Link Repositorio: https://github.com/JeimyH/ProyectoPostgres?tab=readme-ov-file#proyectopostgres

Referencias

JPA entity relationships. (2015, enero 10). Tutorialspoint.com; TutorialsPoint. https://www.tutorialspoint.com/es/jpa/jpa_entity_relationships.htm

Moisset, D. (s/f). *Spring Data JPA - Mapeo de relaciones entre entidades - uno a muchos y muchos a uno*. Tutorialesprogramacionya.com. Recuperado el 1 de abril de 2025, de https://www.tutorialesprogramacionya.com/springbootya/detalleconcepto.php?punto=14&codigo=15&inicio=0