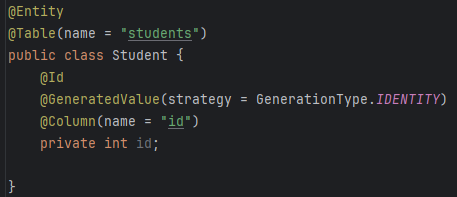
**Aula 3 – Spring JPA Notações**

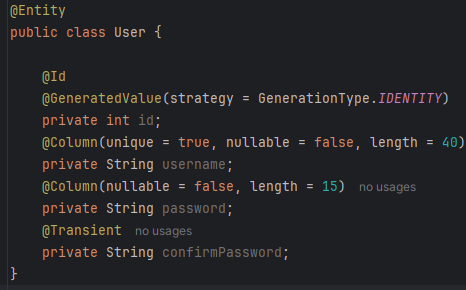
JPA + MySql Driver + Spring Web + Spring Dev tools + Lombok

1 - Configurar conexão com o banco de dados

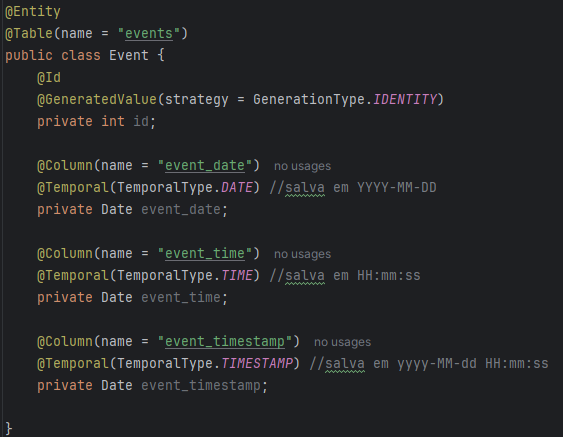
2 – Criando classe Student e fazendo o mapeamento



3 – Criando classe user e mapeando

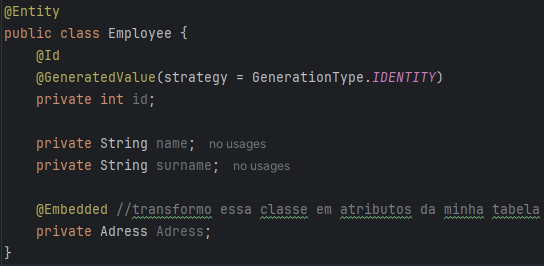


4 – Criando classe de evento com os tipos de data

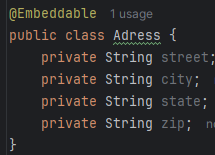


5 – Criando a classe Employee e unificando com a classe Adress

Employee

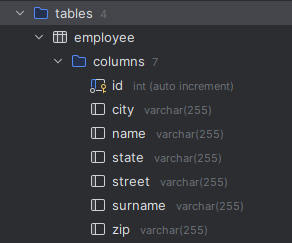


Adress



Nesse caso a classe Employee terá todos atributos da classe Adress como se já fossem dele

Resultado:



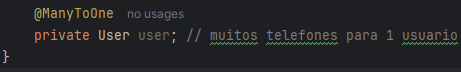
6 – Cardinalidade das relações

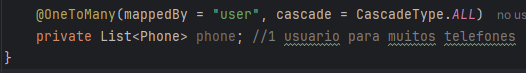
Criando relação **Um para Muitos** entre a classe User e a classe Phone

Na classe User



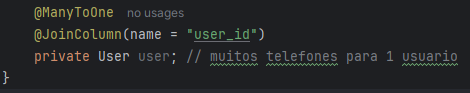
Na classe Phone





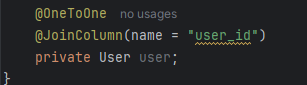
JoinColumn coloca uma chave estrangeira baseada no user\_id

Coloca a chave estrangeira onde está com o ManyToOne

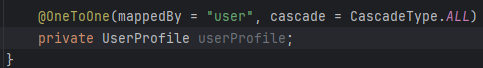


Agora mapeando uma relação **Um para Um** na Classe User e na Classe UserProfile

Classe UserProfile

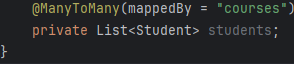


Classe User

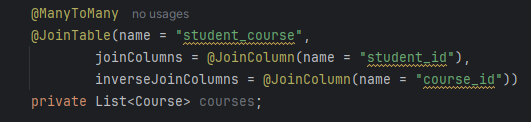


Agora mapeando uma relação **Muitos para Muitos**

Classe Course



Classe Student



No caso do ManyToMany a relação de N para N resulta em uma entidade nova, então usamos JoinTable, junto com JoinColumns e inverseJoinColumns pois essa nova entidade terá as 2 chaves estrangeiras

O mappedBy é sempre no lado onde começa com One, no caso onde possui o OneToMany, casos onde tem OneToOne, ManyToMany, tanto faz qual lado colocar.

O nome utilizado no mappedBy é baseado no nome da variável que está na outra classe conectada

Em geral em um dos lados precisa ter o mappedBy, e o JoinColumn precisa estar na classe que receberá a chave estrangeira e em casos de ManyToMany utilizar o TableColumn com JoinColumns e InverseJoinColumns