**Módulo – Componentes e Injeção de Dependências**

As versões estáveis do Spring Boot **NÃO** possuem o SNAPSHOT, nem m1, m2, m3, rc1, rc2.

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

**Revisão OO e SQL**

No comando de create table de uma está ela pivô N:N há 2 Primary Keys

A JPA e o Hibernate fazem o meio de campo entre nossos objetos e o banco de dados

1 entity manager por requisição (thread)

Sempre de preferência aos impostos do **Javax.persistence**

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

**Módulo – Modelo e Domínio e ORM**

Para acessarmos nosso banco de dados H2 em memória vá para <http://localhost:8080/h2-console> e na url coloque: jdbc:h2:mem:testdb

**Relacionamentos N:1 – Pedidos e Usuários**

A **Foreing Key** ficara no lado N da relação, ou seja, pedidos terá o atributo client.

**No lado N: Tem o @ManyToOne junto com @JoinColumn com o nome da coluna que ficará salva no banco**

**@ManyToOne**

**@JoinColumn**(name = "client\_id")

private **User** **client**;

**No outro lado: Tem o @OneToMany com mappedBy**

**@OneToMany**(mappedBy = "**client**")

private **List**<**Order**> orders = new ArrayList<>();

**IMPORTANTE: Não se cria o set para coleções, deve-se criar apenas o getter, pois nunca trocamos uma lista, nós a manipulamos.**

Para **salvar um instante no formato UTC**, sem o fuso-horário, faça:

**@Column(columnDefinition = "TIMESTAMP WITHOUT TIME ZONE")**

private **Instant** moment;

**Relacionamentos 1:1 – Pedidos e Pagamentos**

Ambos terão a annotation **@OneToOne**

De acordo com o diagrama, o pagamento precisa ter no mínimo 1 pedido, portanto ele é dependente do pedido. Já o pedido pode existir sem ter um pagamento, ou seja, o pagamento pode estar nulo.

**Lado dependente – Pagamento**

**@OneToOne**

**@MapsId**

private **Order** **order**;

**Lado mais independente – Pedido 🡪 Tem o mappedBy e cascade**

**@OneToOne(mappedBy = "order", cascade = CascadeType.ALL)**

private **Payment** payment;