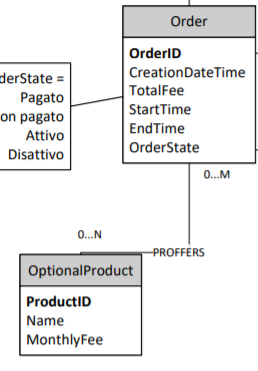
Relationship Service Package “offers” Service



ServicePackage -> Service @ManyToMany is necessary to know what services are offered by a service package.

Service -> ServicePackage @ManyToMany is not requested by the specification, but it is mapped for simplicity and for potential future purpose.

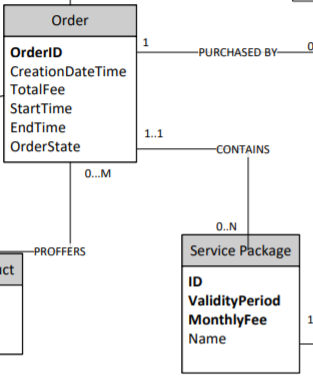
Relationship Order “proffers” OptionalProduct



Order -> OptionalProduct @ManyToMany is necessary to know what optional product are part of the order.

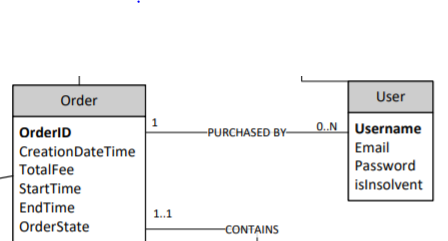
OptionalProduct -> Order @ManyToMany is requested by the specification.

Relationship Order “contains” ServicePackage

Order -> Service Package @OneToMany is required to know which is the service package offered by the order.

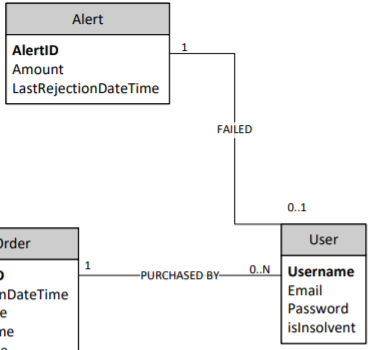
Service Package -> Order @ManyToOne is requested by the specification.

Relationship Order “purchased by” User

 User -> Order @ManyToOne is required to know which orders have been purchased by the user

Order -> User @OneToMany is not requested by the specification, but it is mapped for simplicity and for potential future purpose.

Relationship User “Failed” Alert

User -> Alert @OneToOne is requested by the specification.  
Alert -> User @OneToOne is not requested by the specification, but it is mapped for simplicity and for potential future purpose.

**Order Entity**

@ManyToOne

@JoinColumn(name = “Username”)

UserEntity user;

@ManyToMany(fetch = FetchType.EAGER )

List<OptionalProductEntity> optionalProducts ;

**OptionalProduct Entity**

@ManyToMany(mappedBy = ”optionalProducts”, fetch = FetchType.LAZY, cascade = {CascadeType.REMOVE, CascadeType.PERSIST, CascadeType.MERGE, CascadeType.REFRESH})

List<OrderEntity> orderEntities ;