CREACIÓN DE OBJETOS

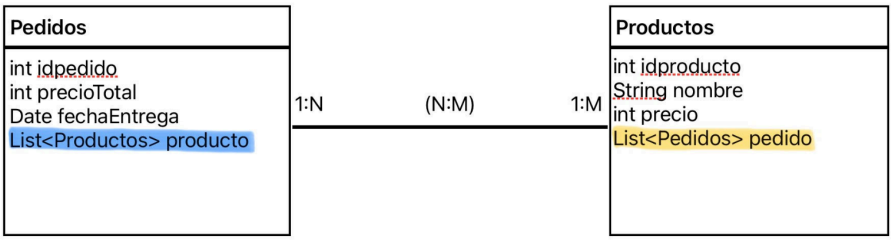
OneToOne

@JoinColumn(name="PRODUCTO", referencedColumnName="PRODUCTO")
@OneToOne
private Productos producto;

@OneToOne(mappedBy="producto", orphanRemoval=true)
private Pedidos pedido;

ManyToMany

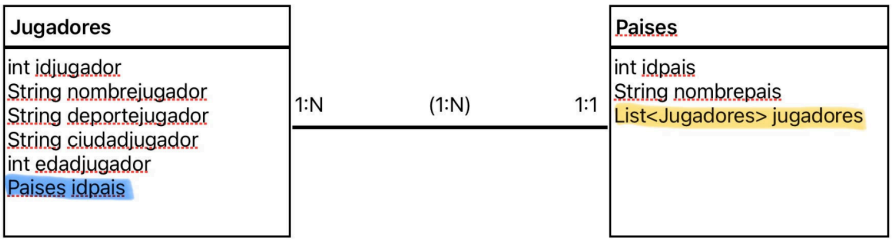
@ManyToMany(mappedBy="pedido")
private List<Productos> producto;



@JoinTable(
name="rel_productos_pedidos",
joinColumns=@JoinColumn(name="FK_PRODUCTO",nullable=false),
inverseJoinColumns=@JoinColumn(name="FK_PEDIDO",nullable=false))
@ManyToMany(cascade=CascadeType.ALL)
private List<Pedidos> pedido;

OneToMany // ManyToOne

@JoinColumn(name="IDPAIS", referencedColumnName="IDPAIS")
@ManyToOne
private Paises idpais;



@OneToMany(mappedBy="idpais", orphanRemoval=true)
private List<Jugadores> jugadores;

INICIO PROYECTO

```
private static EntityManagerFactory emf;
private static EntityManager em;
public static void main (String[] argos) {
        crearConexion();        → Conexión al final del doc
        em.close();
        emf.close();
}
```

em.refresh() → DON'T FORGET IT

CONSULTAS, UPDATES Y DELETES

Consultas sin JPQL

em.getTransaction().begin();
Usuarios usuario = em.find(Usuarios.class, idUsuario, LockModeType.PESSIMISTIC_READ);

- **CONSULTAS**
  usuario.get…();
- **UPDATE**
  usuario.set…();
- **DELETE**
  em.remove(usuario);

em.getTransaction().commit();

Consultas con JPQL

TypedQuery

TypedQuery<nombre_objeto> query = em.createQuery("Consulta",nombre_objeto.class);
query. setParameter("parametro_en_consulta", parametro_a_introducir);
- SI DEVUELVE UN UNICO OBJETO

```
nombre_objeto … = query.getSingleResult();
```

- SI DEVUELVE UNA LISTA DE OBJETOS

1.
```
Collection<nombre_objeto> coleccion = query.getResultList();
Iterator<nombre_objeto> it = coleccion.iterator();
while(it.hasNext()){
        nombre_objeto … = it.next();
        …
}
```

2.
```
List<nombre_objeto> lista = query.getResultList();
for(nombre_objeto o: lista){
        …
}
```

Query

Query query = em.createQuery("Consulta");
query. setParameter("parametro_en_consulta", parametro_a_introducir);
- PARA UPDATE Y DELETE

```
em.getTransaction().begin();
int … = query.executeUpdate;
em.getTransaction().commit();
```

- PARA SELECT
  - SI DEVUELVE UN UNICO OBJETO

```
nombre_objeto … = query.getSingleResult();
```

  - SI DEVUELVE UNA LISTA DE OBJETOS

1.
```
Collection<nombre_objeto> coleccion = query.getResultList();
Iterator<nombre_objeto> it = coleccion.iterator();
while(it.hasNext()){
        nombre_objeto … = it.next();
        …
}
```

2.
```
List<nombre_objeto> lista = query.getResultList();
for(nombre_objeto o: lista){
        …
}
```

MappedQuery

Para usar las MappedQuery deben estar creadas las namedQueries en la clase

```
@NamedQueries({
        @NamedQuery(name="Jugadores.findById", query="SELECT from Jugadores WHERE
                idjugador=:IDJUGADORP"),
        @NamedQuery(name="Jugadores.findAll", query="SELECT from Jugadores")
})
```

```
TypedQuery<nombre_objeto> query = em.createNamedQuery("Consulta",nombre_objeto.class);
query. setParameter("parametro_en_consulta", parametro_a_introducir);
```

- SI DEVUELVE UN UNICO OBJETO

```
        nombre_objeto … = query.getSingleResult();
```

- SI DEVUELVE UNA LISTA DE OBJETOS

1.
```
Collection<nombre_objeto> coleccion = query.getResultList();
Iterator<nombre_objeto> it = coleccion.iterator();
while(it.hasNext()){
        … = it.next();
        …
}
```
2.
```
List<nombre_objeto> lista = query.getResultList();
for(nombre_objeto o: lista){
        …
}
```

Consultas CriteriaQuery

nombre_objeto = clase_tabla → Si se extrae una tabla completa, sino suele ser Object[]

```
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<nombre_objeto> query = cb.createQuery(nombre_objeto.class);
Root<clase_tabla> u = query.from(clase_tabla.class);
        *campos_a_seleccionar
                u → Si lo que queremos seleccionar es una tabla completa
                u.get(nombre_columna) → Si lo que queremos es una columna,
                                        se pueden pedir tantas como quieras.
        *condiciones
                Predicate condiciones = cb.equal( u.get(nombre_columna), parametro_a_comparar)
query.select(campos_a_seleccionar).where(condiciones)
List<clase_tabla> lista = em.createQuery(query).getResultList();
        List<clase_tabla> lista = query.getResultList();
        for(clase_tabla o: lista){
                …
        }
```

CREAR CONEXIÓN

```
private static void crearConexion(){
        //CONEXIÓN DOCKER (Version objectdb 2.8.7)
        emf = Persistence.createEntityManagerFactory("objectdb://localhost/proyecto.odb;
                                                      user=admin;
                                                      password=admin");
        //CONEXIÓN EN LOCAL
        emf = Persistence.createEntityManagerFactory("./db/proyecto.odb");
        em = emf.createEntityManager();
}
```