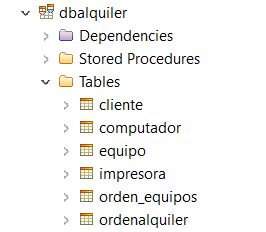
Refrescar la conexion a la base de datos para que se puede reflejar la nueva tabla orden\_equipos



**En la clase Equipo**

Agregar nuevo atributo

//bi-directional many-to-many association to Ordenalquiler

@ManyToMany(mappedBy="equipos")

private List<Ordenalquiler> ordenalquilers;

Agregar setter y getter

public List<Ordenalquiler> getOrdenalquilers() {

return this.ordenalquilers;

}

public void setOrdenalquilers(List<Ordenalquiler> ordenalquilers) {

this.ordenalquilers = ordenalquilers;

}

Con este atributo se podrán recuperar las ordenes en las que ese equipo ha sido incluido para alquiler

En el constructor agregar

ordenalquilers = new ArrayList<Ordenalquiler>();

**En la clase Ordenalquiler**

Agregar nuevo atributo equipos

//bi-directional many-to-many association to Equipo

@ManyToMany

@JoinTable(

name="orden\_equipos"

, joinColumns={

@JoinColumn(name="id\_orden")

}

, inverseJoinColumns={

@JoinColumn(name="id\_equipo")

}

)

private List<Equipo> equipos;

En este atributo se indica cual es la tabla para la relación

//se agrega tambien setter y getter

public List<Equipo> getEquipos() {

return this.equipos;

}

public void setEquipos(List<Equipo> equipos) {

this.equipos = equipos;

}

Y métodos para agregar o eliminar equipos

public void addEquipo(Equipo equipo) {

equipos.add(equipo);

equipo.getOrdenalquilers().add(this);

}

public void removeEquipo(Equipo equipo) {

equipos.remove(equipo);

equipo.getOrdenalquilers().remove(this);

}

En el constructor agregar

equipos = new ArrayList<Equipo>();

**En la clase Test**

Para consulta agregaremos tres nuevos métodos :

public static void consultaEOrdenesEquipo() {

System.out.println("consultar ordenes de equipo....");

EntityManagerFactory emf = Persistence.createEntityManagerFactory("JPATest3");

EntityManager em = emf.createEntityManager();

try {

//buscar equipo

Equipo equipo = em.find(Equipo.class, "1001");

//recuperar sus orddenesy mostrar

for (Ordenalquiler orden:equipo.getOrdenalquilers()) {

System.out.println(orden);

}

} catch (Exception e) {

System.out.println("Error:" + e.getMessage());

} finally {

em.close();

}

}

public static void consultaEquiposOrden() {

System.out.println("consultar equipos de orden....");

EntityManagerFactory emf = Persistence.createEntityManagerFactory("JPATest3");

EntityManager em = emf.createEntityManager();

try {

//buscar orden

Ordenalquiler orden = em.find(Ordenalquiler.class, 1);

//recuperar sus equipos y mostrar

for (Equipo equipo:orden.getEquipos()) {

System.out.println(equipo);

}

} catch (Exception e) {

System.out.println("Error:" + e.getMessage());

} finally {

em.close();

}

}

public static void insertarOrdenYEquipos() {

System.out.println("Insertar orden Y EQUIPOS..");

EntityManagerFactory emf = Persistence.createEntityManagerFactory("JPATest3");

EntityManager em = emf.createEntityManager();

EntityTransaction et = em.getTransaction();

try {

//buscar clliente

Cliente cliente = em.find(Cliente.class, "777777");

SimpleDateFormat sdf = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");

String fecha =sdf.format(new Date());

short dias=4;

Ordenalquiler orden = new Ordenalquiler(dias,fecha,cliente);

//buscar equipo que se van a agregar a la orden

Equipo equipo = em.find(Equipo.class, "1001");

Equipo equipo2 = em.find(Equipo.class, "1002");

orden.addEquipo(equipo);

orden.addEquipo(equipo2);

et.begin();

em.persist(orden);

et.commit();

} catch (Exception e) {

System.out.println("Error:" + e.getMessage());

} finally {

em.close();

}

}

En el método main llamaremos a los métodos para verificar la salida

consultaEquiposOrden();

consultaEOrdenesEquipo();

insertarOrdenYEquipos();