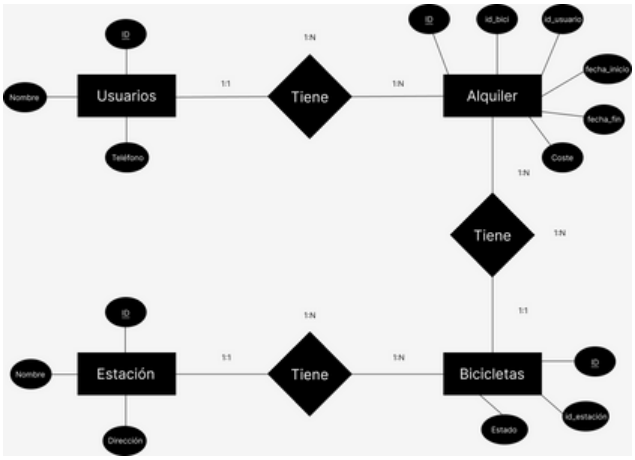
Desarrollo de Aplicación con Hibernate

Pedro Manuel Liébana Beltrán

Diagrama entidad - relación



Mi actividad está pensada para realizar un sistema de alquiler de bicicletas. He usado cuatro tablas.

Usuarios: Guarda la información de cada persona que usa el servicio.

Estación: Representa las estaciones donde están las bicicletas.

Bicicletas: Cada bicicleta tiene un estado y pertenece a una estación.

Alquiler: Aquí se guardan los alquileres que hacen los usuarios, con su fecha.

Relaciones

Relación **1:N** entre **Usuario y alquiler**. (Un usuarios puede tener varios alquileres, pero cada alquiler solo puede tener un usuario).

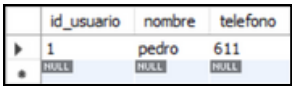
Relación **1:N** entre **Estación y Bicicletas**. (Una estación puede tener varias bicicletas, pero una bicicleta solo puede tener una estación).

Relación **1:N** entre **Bicicleta y Alquiler**. (Una bicicleta puede tener varios alquileres, pero cada alquiler es de una sola bici).

Operaciones CRUD

Agregar usuario →

```
if (eleccion == 1) {
    System.out.println("Escriba el nombre:");
    String nombre = sc.next();

    System.out.println("Escriba el telefono:");
    int telefono = sc.nextInt();

    SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
    Session session = sessionFactory.openSession();
    session.beginTransaction();

    Usuario usuario = new Usuario(nombre, telefono);
    session.persist(usuario);

    session.getTransaction().commit();
    session.close();
}
```



	id_usuario	nombre	telefono
▶	1	pedro	611
*	NULL	NULL	NULL

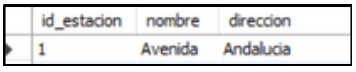
En primer lugar introducimos el “Nombre del usuario” y “Teléfono”.

- Después de introducir los datos, el programa crea una conexión con Hibernate para guardar los datos en la base de datos.
- Abre la sesión y crea un objeto Usuario con los datos que hemos pasado anteriormente.
- Los guarda en la base de datos usando persist() y confirma los cambios con commit().

Estos mismos pasos se realizan para agregar una estación, una bicicleta y un alquiler pero con datos diferentes.

Agregar estación →

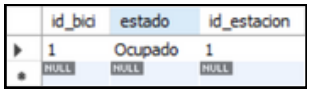
```
else if (eleccion == 2) {
    System.out.println("Escriba el nombre:");
    String nombre = sc.next();

    System.out.println("Escriba la direccion:");
    String direccion = sc.next();

    SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
    Session session = sessionFactory.openSession();
    session.beginTransaction();

    Estacion estacion = new Estacion(nombre, direccion);
    session.persist(estacion);

    session.getTransaction().commit();
    session.close();
}
```



	id_estacion	nombre	direccion
▶	1	Avenida	Andalucia

Agregar bicicleta →

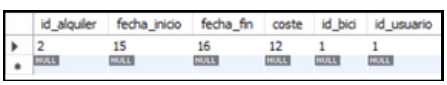
```
else if (eleccion == 3) {
    System.out.println("Inserte id estacion:");
    int id_estacion = sc.nextInt();

    System.out.println("Estado:");
    String estado = sc.next();

    SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
    Session session = sessionFactory.openSession();
    session.beginTransaction();

    Estacion e = session.get(Estacion.class, id_estacion);
    Bicicleta bicicleta = new Bicicleta(e, estado);
    session.persist(bicicleta);

    session.getTransaction().commit();
    session.close();
}
```



	id_bici	estado	id_estacion
▶	1	Ocupado	1
*	NULL	NULL	NULL

Realizar alquiler →

```
else if (eleccion == 4) {
    System.out.println("Inserte id del usuario:");
    int id_usuario = sc.nextInt();

    System.out.println("Inserte el id de la bicicleta:");
    int id_bicicleta = sc.nextInt();

    System.out.println("Fecha inicio:");
    String fechainicio = sc.next();

    System.out.println("Fecha final:");
    String fechafinal = sc.next();

    System.out.println("Precio:");
    int precio = sc.nextInt();

    SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
    Session session = sessionFactory.openSession();
    session.beginTransaction();

    Bicicleta b = session.get(Bicicleta.class, id_bicicleta);
    Usuario u = session.get(Usuario.class, id_usuario);

    Alquiler alquiler = new Alquiler(fi: "12", ff: "13", u, b, precio);

    session.persist(alquiler);
    session.getTransaction().commit();
    session.close();
```



	id_alquiler	fecha_inicio	fecha_fin	coste	id_bici	id_usuario
▶	2	15	16	12	1	1
*	NULL	NULL	NULL	NULL	NULL	NULL

Ejemplos de consultas HQL

Mostrar alquiler por persona →

```
else if (eleccion == 5) {

    SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
    Session session = sessionFactory.openSession();

    System.out.println("Introduce el id del usuario: ");
    int idusuario = sc.nextInt();
    sc.nextLine();

    List<Alquiler> pedidos = session.createQuery(
            "FROM Alquiler a WHERE a.id_usuario.id = :id ORDER BY a.fecha_inicio DESC",
            Alquiler.class)
            .setParameter("id", idusuario)
            .getResultList();

    for (Object obj : pedidos) {
        Alquiler al = (Alquiler) obj;
        System.out.println(
                "Alquiler " + al.getId() +
                        " - Fecha inicio: " + al.getFecha_inicio() +
                        " - Fecha fin: " + al.getFecha_fin());
    }
}
```

```
Alquiler 2 - Fecha inicio: 15 - Fecha fin: 16
```

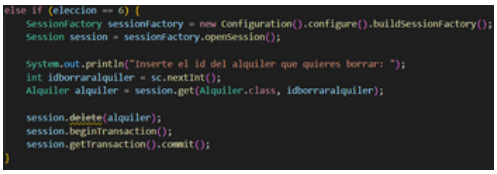
Cuando elegimos la opción 5 del menú, el programa nos pide que pasemos el ID del usuario que queremos ver sus alquileres.

Una vez que hemos pasado los datos, se conecta con la base de datos con Hibernate.

Ejecuta una consulta HQL que busca todos los alquileres donde el campo id_usuario.id sea igual al que hemos metido.

Además lo ordena por fecha de inicio.

Borrar un alquiler →



```
else if (eleccion == 6) {
    SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
    Session session = sessionFactory.openSession();

    System.out.println("Inserte el id del alquiler que quieres borrar: ");
    int idborraralquiler = sc.nextInt();
    Alquiler alquiler = session.get(Alquiler.class, idborraralquiler);

    session.delete(alquiler);
    session.beginTransaction();
    session.getTransaction().commit();
}
```

Cuando elegimos la opción 6 del menú, el programa nos pide que pasemos el ID del alquiler que queremos borrar.

Se conecta a la base de datos con Hibernate.

Busca el alquiler con el ID que anteriormente le hemos pasado. Si ese alquiler existe, se borra con .delete(). `session.delete(alquiler);`

Ejemplos de consultas HQL

Actualizar fecha de alquiler →

```
else if (eleccion == 7) {

    SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();
    Session session = sessionFactory.openSession();

    System.out.println("Introduce el id del alquiler: ");
    int idalquiler = sc.nextInt();

    System.out.println("Introduce fecha inicio actualizada: ");
    String fechainicio = sc.next();

    System.out.println("Introduce fecha fin actualizada: ");
    String fechafin = sc.next();

    Query q = session.createQuery("update Alquiler set fecha_inicio = :fi, fecha_fin = :ff where id = :id");
    q.setParameter("fi", fechainicio);
    q.setParameter("ff", fechafin);
    q.setParameter("id", idalquiler);
    session.beginTransaction();
    q.executeUpdate();
    session.getTransaction().commit();
}
```

Cuando elegimos la opción 7 del menú, el programa nos pide que pasemos el ID del alquiler, la nueva fecha inicio y la nueva fecha fin.

Una vez que hemos pasado los datos, se conecta con la base de datos con Hibernate.

Ejecuta una consulta HQL que actualiza solo las fechas de alquiler que queremos cambiar.

Rellena los valores fi, ff y id con los datos facilitados.

Antes

id_alquiler	fecha_inicio	fecha_fin	coste	id_bici	id_usuario
2	15	16	12	1	1
NULL	NULL	NULL	NULL	NULL	NULL

Después

id_alquiler	fecha_inicio	fecha_fin	coste	id_bici	id_usuario
2	20	22	12	1	1
NULL	NULL	NULL	NULL	NULL	NULL

Desafío personal

A continuación explicaré el procedimiento de crear un pequeño servidor donde podremos mostrar un formulario y poder enseñar los alquileres de un usuario.

- En primer lugar, creamos un servidor que escuchará por el puerto 8080 y que podremos navegar por el.

```
HttpServer server = HttpServer.create(new InetSocketAddress(8080), 0);
```

- Seguidamente definimos lo que se mostrará cuando un usuario lo visite.

```
server.createContext("/formulario", (HttpExchange exchange) -> {
    String html = "<html><body>" +
                  "<h1>Buscar alquileres por usuario</h1>" +
                  "<form action='/resultados'>" +
                  "ID Usuario: <input name='id'><br>" +
                  "<button type='submit'>Buscar</button>" +
                  "</form></body></html>";

    exchange.sendResponseHeaders(200, html.length());
    try (OutputStream os = exchange.getResponseBody()) {
        os.write(html.getBytes());
    }
});
```

- Esta parte responde cuando buscamos algún ID.

```
server.createContext("/resultados", (HttpExchange exchange) -> {
    String query = exchange.getRequestURI().getQuery();
    int id = Integer.parseInt(query.split("=")[1]);

    SessionFactory sf = new Configuration().configure().buildSessionFactory();
    Session session = sf.openSession();

    List<Alquiler> lista = session.createQuery(
            "FROM Alquiler a WHERE a.id_usuario.id = :id", Alquiler.class)
            .setParameter("id", id)
            .list();

    session.close();

    StringBuilder html = new StringBuilder("<html><body><h1>Resultados</h1>");

    for (Alquiler a : lista) {
        html.append("Alquiler ")
                .append(a.getId())
                .append(" - Inicio: ").append(a.getFecha_inicio())
                .append(" - Fin: ").append(a.getFecha_fin())
                .append("<br>");
    }

    html.append("</body></html>");

    exchange.sendResponseHeaders(200, html.length());
    try (OutputStream os = exchange.getResponseBody()) {
        os.write(html.toString().getBytes());
    }
});
```