EJERCICIO 3-1. USAR HIBERNATE PARA LA TABLA CLIENTES

> Nota: en el mismo proyecto donde tienes la clase EmpresaHibernate, puedes crear otra clase “EmpresaClientes” para hacer las consultas sobre Clientes. No necesitas hacer la instalación de nuevo.

Crea en la clase, haz un constructor que tenga conexión con la bdd y haz los siguientes métodos:

1. Mostrar todos los clientes
2. Añadir un cliente con la información en los argumentos
3. Borrar un cliente con la id en el argumento. Si la id no existe, debe avisarme de que no lo ha borrado.
4. actualizarCliente(int id): te muestra los valores del cliente si existe y te pregunta si quieres un nuevo nombre, si dices que si te lo pide, luego dice que si quieres un nuevo país, si dices que sí te lo pide, y actualiza según lo que hayas introducido
5. borrarCliente(String nombre): borra los clientes con ese nombre
6. mostrarPorPais(String pais): muestra el número de clientes que tenemos en ese país y luego los datos de cada uno.
7. buscarPaisDe(String nombre): muestra el país del nombre pasado por argumento.

> Los métodos deben tener try/catch y avisar si no existe lo que estamos buscando, borrarndo, actualizando, etc. Y avisar también de que se ha hecho el borrado, inserción, etc.

Metodos:

```
package martinmd.martinmd;


import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.query.Query;


import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;
```

```
import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.Table;


import java.util.List;

import java.util.Scanner;

@Entity

@Table(name = "clientes")

public class Clientes {

@Id

   @GeneratedValue(strategy = GenerationType.IDENTITY) // Asegúrate de que el id se genere
correctamente


   private static final long serialVersionUID = 1L;


   private SessionFactory sessionFactory;


   private Integer id;

   private String nombre;

   private String pais;


   public Clientes(SessionFactory sessionFactory) {

      this.sessionFactory = sessionFactory;

   }

   // Constructores

   public Clientes() {

   }
```

```
public Clientes(String nombre, String pais) {
    this.nombre = nombre;
    this.pais = pais;
}

// Getters y Setters
public Integer getId() {
    return this.id;
}

public void setId(Integer id) {
    this.id = id;
}

public String getNombre() {
    return this.nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public String getPais() {
    return this.pais;
}
```

```
public void setPais(String pais) {
    this.pais = pais;
}



@Override
public String toString() {
    return "Clientes [id=" + id + ", nombre=" + nombre + ", pais=" + pais + "]";
}




public void mostrarTodosLosClientes() {
    try (Session session = sessionFactory.openSession()) {
        List<Object[]> clientes = session.createQuery("FROM Clientes", Object[].class).list();
        if (clientes.isEmpty()) {
            System.out.println("No hay clientes registrados.");
        } else {
            for (Object[] cliente : clientes) {
                System.out.println(cliente);
            }
        }
    } catch (Exception e) {
        System.err.println("Error al mostrar los clientes: " + e.getMessage());
    }
}
```

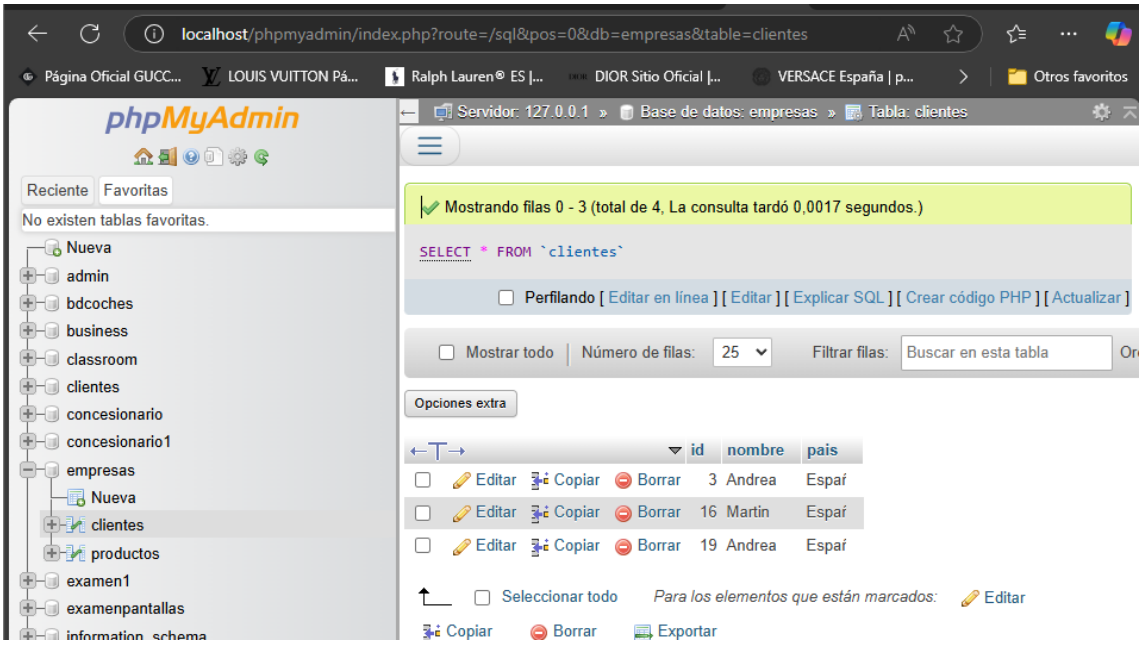
```
public void anadirCliente(String nombre, String pais) {
    Transaction tx = null;
    try (Session session = sessionFactory.openSession()) {
        tx = session.beginTransaction();
        Clientes cliente = new Clientes();
        cliente.setNombre(nombre);
        cliente.setPais(pais);
        session.save(cliente);
        tx.commit();
        System.out.println("Cliente añadido correctamente.");
    } catch (Exception e) {
        if (tx != null) tx.rollback();
        System.err.println("Error al añadir el cliente: " + e.getMessage());
    }
}

public void borrarCliente(int id) {
    Transaction tx = null;
    try (Session session = sessionFactory.openSession()) {
        tx = session.beginTransaction();
        Clientes cliente = session.get(Clientes.class, id);
        if (cliente != null) {
            session.delete(cliente);
            tx.commit();
            System.out.println("Cliente eliminado correctamente.");
        } else {
            System.out.println("Cliente con ID " + id + " no encontrado.");
```

```
            }
        } catch (Exception e) {
            if (tx != null) tx.rollback();
            System.err.println("Error al borrar el cliente: " + e.getMessage());
        }
    }

    public void actualizarCliente(int id) {
        Transaction tx = null;
        try (Session session = sessionFactory.openSession(); Scanner scanner = new
Scanner(System.in)) {
            Clientes cliente = session.get(Clientes.class, id);
            if (cliente == null) {
                System.out.println("Cliente con ID " + id + " no encontrado.");
                return;
            }

            System.out.println("Cliente actual: " + cliente);
            tx = session.beginTransaction();

            System.out.print("¿Desea cambiar el nombre? (si/no): ");
            if (scanner.nextLine().equalsIgnoreCase("si")) {
                System.out.print("Nuevo nombre: ");
                cliente.setNombre(scanner.nextLine());
            }

            System.out.print("¿Desea cambiar el país? (si/no): ");
            if (scanner.nextLine().equalsIgnoreCase("si")) {
```

```
            System.out.print("Nuevo país: ");
            cliente.setPais(scanner.nextLine());
        }

        session.update(cliente);
        tx.commit();
        System.out.println("Cliente actualizado correctamente.");
    } catch (Exception e) {
        if (tx != null) tx.rollback();
        System.err.println("Error al actualizar el cliente: " + e.getMessage());
    }
}

public void borrarCliente(String nombre) {
    Transaction tx = null;
    try (Session session = sessionFactory.openSession()) {
        tx = session.beginTransaction();
        Query<Clientes> query = session.createQuery("FROM Clientes WHERE nombre =
:nombre", Clientes.class);
        query.setParameter("nombre", nombre);
        List<Clientes> clientes = query.list();

        if (clientes.isEmpty()) {
            System.out.println("No se encontraron clientes con el nombre: " + nombre);
        } else {
            for (Clientes cliente : clientes) {
                session.delete(cliente);
            }
```

```
            tx.commit();
            System.out.println("Clientes con el nombre " + nombre + " eliminados correctamente.");
        }
    } catch (Exception e) {
        if (tx != null) tx.rollback();
        System.err.println("Error al borrar los clientes: " + e.getMessage());
    }
  }

  public void mostrarPorPais(String pais) {
    try (Session session = sessionFactory.openSession()) {
        Query<Clientes> query = session.createQuery("FROM Clientes WHERE pais = :pais",Clientes.class);
        query.setParameter("pais", pais);
        List<Clientes> clientes = query.list();

        if (clientes.isEmpty()) {
            System.out.println("No se encontraron clientes en el país: " + pais);
        } else {
            System.out.println("Número de clientes en " + pais + ": " + clientes.size());
            for (Clientes cliente : clientes) {
                System.out.println(cliente);
            }
        }
    } catch (Exception e) {
        System.err.println("Error al mostrar clientes por país: " + e.getMessage());
    }
```

```
    }


    public void buscarPaisDe(String nombre) {
        try (Session session = sessionFactory.openSession()) {
            Query<Clientes> query = session.createQuery("FROM Clientes WHERE nombre =
:nombre", Clientes.class);
            query.setParameter("nombre", nombre);
            Clientes cliente = query.uniqueResult();


            if (cliente == null) {
                System.out.println("No se encontró un cliente con el nombre: " + nombre);
            } else {
                System.out.println("El cliente " + nombre + " es de: " + cliente.getPais());
            }
        } catch (Exception e) {
            System.err.println("Error al buscar el país del cliente: " + e.getMessage());
        }
    }
}
```



Main:

```
package martinmd.martinmd;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class App {
    public static void main(String[] args) {
        // Configuración de Hibernate
        SessionFactory sessionFactory = new
Configuration().configure().buildSessionFactory();

        // Instancia de la clase ClienteDAO
        Clientes clienteManager = new Clientes(sessionFactory);

        try {
            // 1. Mostrar todos los clientes
            System.out.println("Mostrando todos los clientes:");
            clienteManager.mostrarTodosLosClientes();

            // 2. Añadir un cliente
            System.out.println("\nAñadiendo un cliente:");
            clienteManager.anadirCliente("Andrea", "España");

            // 3. Borrar un cliente por ID
            System.out.println("\nBorrando cliente:");
            clienteManager.borrarCliente(17);
```

```
        // 4. Actualizar un cliente por ID
        System.out.println("\nActualizando cliente:");
        clienteManager.actualizarCliente(3);

        // 5. Mostrar clientes por país
        System.out.println("\nMostrando clientes de España:");
        clienteManager.mostrarPorPais("España");

        // 6. Buscar el país de un cliente por nombre
        clienteManager.buscarPaisDe("Nerea");
    } finally {
        // Cerrar la sesión de Hibernate
        sessionFactory.close();
    }
  }
}
```