AD

Práctica 3

HERRAMIENTAS DE MAPEO OBJETO-RELACIONAL. HIBERNATE

Álvaro Manuel Navarro Cruz
2º DAM
10/12/2024

Índice

Introducción (↑)

Para el desarrollo de esta práctica, se nos pide realizar el código de persistencia necesario para un programa de gestión de hotel, concretamente el apartado del almacenamiento de clientes.

Debemos utilizar un proyecto de tipo Maven e implementar los métodos necesarios para las operaciones CRUD de la tabla que se creará a partir de nuestra clase POJO "ClienteANC".

Para ello, debemos completar la clase POJO con anotaciones Hibernate directamente sobre la misma, sin uso de ficheros ".hbm.xml".

También debemos configurar el archivo hibernate.cfg.xml con los parámetros de conexión a la Base de Datos "hotelanc".

Para la ejecución del programa, debemos implementar una clase TestHotel con método Main, donde realizaremos las operaciones de alta, baja, modificación y consulta.

Repositorio en GitHub

https://github.com/AlvaroMfco/AD_Practica3.git

1.- Fichero pom.xml (↑)

```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="h
  <modelVersion>4.0.0</modelVersion>
  <groupId>es.studium.Practica3</groupId>
  <artifactId>AD_Practica3</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>AD_Practica3</name>

    <dependencies>
    <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
    <!-- https://mvnrepository.com/artifact/com.mysql/mysql-connector-j -->
        <dependency>
            <groupId>com.mysql</groupId>
            <artifactId>mysql-connector-j</artifactId>
            <version>9.1.0</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.hibernate.orm/hibernate-core -->
        <dependency>
            <groupId>org.hibernate.orm</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>6.6.2.Final</version>
        </dependency>

    </dependencies>

    <properties>
        <maven.compiler.source>22</maven.compiler.source>
        <maven.compiler.target>22</maven.compiler.target>
    </properties>
</project>
```

2.- Fichero de configuración de Hibernate (↑)

```
-//Hibernate/Hibernate Configuration DTD//EN (doctype)
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD//EN" "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.connection.driver_class">
            com.mysql.cj.jdbc.Driver
        </property>
        <property name="hibernate.connection.url">
            jdbc:mysql://localhost:3306/hotelanc?createDatabaseIfNotExist=true
        </property>
        <property name="hibernate.connection.username">root</property>
        <property
            name="hibernate.connection.password">Studium2023;</property>

        <!-- <property name="show_sql">true</property>  -->
        <property name="hibernate.format_sql">true</property>
        <property name="hibernate.use_sql_comments">true</property>

        <property name="hibernate.hbm2ddl.auto">update</property>

        <!-- Mapping files -->
        <mapping class="es.studium.Practica3.ClienteANC" />

    </session-factory>
</hibernate-configuration>
```

En este archivo se establece el conector JDBC, la URL de conexión a la base de datos, el usuario y la contraseña.

Se ha comentado la línea 15 para que no aparezcan las sentencias SQL por consola.

3.- Clase ClienteANC

```java
package es.studium.Practica3;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

//Entity para crear la tabla
@Entity
public class ClienteANC {
    //Indicar que va a ser el ID y Auto Increment
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="idCliente")
    private int id;

    //Con @Column especificamos el nombre que llevará esta columna en la tabla
    @Column(name="nombreCliente")
    private String nombre;

    @Column(name="apellidosCliente")
    private String apellidos;

    @Column(name="emailCliente")
    private String email;

    @Column(name="dniCliente")
    private String dni;

    @Column(name="claveCliente")
    private String clave;

    //Constructor vacío
    public ClienteANC() {
        nombre = "";
        apellidos = "";
        email = "";
        dni = "";
        clave = "";
    }

    public ClienteANC(int idCliente) {
        this.id = idCliente;
    }

    //Constructor por parámetros
    public ClienteANC(String nombre, String apellidos, String email, String dni, String clave) {
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.email = email;
        this.dni = dni;
        this.clave = clave;
    }

    //Constructor con todos los parámetros
    public ClienteANC(int idCliente, String nombre, String apellidos, String email, String dni, String clave) {
        this.id = idCliente;
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.email = email;
        this.dni = dni;
        this.clave = clave;
    }

    public int getId() {
        return id;
    }

    public void setId(int idCliente){
        this.id = idCliente;
    }
```

```java

    public String getNombre() {
        return nombre;
    }

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

    public String getApellidos() {
        return apellidos;
    }

    public void setApellidos(String apellidos) {
        this.apellidos = apellidos;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getDni() {
        return dni;
    }

    public void setDni(String dni) {
        this.dni = dni;
    }

    public String getClave() {
        return clave;
    }

    public void setClave(String clave) {
        this.clave = clave;
    }
}
```

4.- Clase HotelManager

```java
package es.studium.Practica3;

import java.util.List;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

public class HotelManager {

    private static Session getSession() {
        SessionFactory sf = new Configuration().addAnnotatedClass(ClienteANC.class).configure().buildSessionFactory();
        //Session permite la conexión con la BBDD.
        Session session = sf.openSession();
        return session;
    }

    //Método para crear clientes
    public static void crearCliente(ClienteANC c) {
        Session session = getSession();
        Transaction tx = session.beginTransaction();
        session.persist(c);
        //Con el commit se guardarán los datos en la BBDD
        tx.commit();
        // Cerrar sesión
        session.close();
        System.out.println(
                "\nCliente con ID: " + c.getId() + " insertado correctamente en la base de datos hotelanc."
                        + "\n  -Nombre: " + c.getNombre() + "\n  -Apellidos: " + c.getApellidos()
                        + "\n  -Email: " + c.getEmail() + "\n  -DNI: " + c.getDni() + "\n  -Clave: " + c.getClave());
    }

    //Método para leer todos los usuarios de la tabla de la BBDD
    public static List<ClienteANC> leerClientes() {
        String consulta = "FROM " + ClienteANC.class.getName();
        Session sessionObj = getSession();
        List<ClienteANC> listaResultado = sessionObj.createQuery(consulta, ClienteANC.class).list();
        System.out.println("\n** Clientes encontrados en la tabla 'clienteanc' **");
        for (ClienteANC c : listaResultado) {
            System.out.println(c.getId() + " - " + c.getNombre() + " - " + c.getApellidos() + " - " + c.getEmail()
            + " - " + c.getDni() + " - " + c.getClave());
        }
        sessionObj.close();
        return listaResultado;
    }

    //Método para actualizar clientes
    public static void actualizarCliente(ClienteANC c) {
        Session sessionObj = getSession();
        Transaction transObj = sessionObj.beginTransaction();

        //Actualizamos el objeto
        sessionObj.merge(c);
        transObj.commit();
        sessionObj.close();
        System.out.println("\n** Actualización Completada correctamente **");
        System.out.println("   -Actualizado el Cliente con id: " + c.getId());
    }

    //Método para eliminar clientes
    public static void eliminarCliente(ClienteANC c) {
        Session sessionObj = getSession();
        Transaction transObj = sessionObj.beginTransaction();
        sessionObj.remove(c);
        transObj.commit();
        sessionObj.close();
        System.err.println("\n** Cliente eliminado con éxito **");
        System.out.println("  -Eliminado el Cliente con id: "  + c.getId());
    }
}
```

5.- Clase TestHotel (↑)

```java
package es.studium.Practica3;

public class TestHotel {

    public static void main(String[] args) {
        //Utilizamos el constructor por parámetros sin ID para crear los clientes, ya que este es Auto Increment
        ClienteANC cl1 = new ClienteANC("Álvaro", "Navarro Cruz", "alvaromfco@gmail.com", "49545195K",
                "Studium2023;");
        ClienteANC cl2 = new ClienteANC("Pepe", "García Ruíz", "pepe@gmail.com", "82945876L",
                "clav3Pepe");
        ClienteANC cl3 = new ClienteANC("Raúl", "Escudero Ortíz", "raul@gmail.com", "25487632A",
                "clav3R4ul");
        //Actualizamos mediante el constructor por parámetros con todos los datos.
        ClienteANC cl4 = new ClienteANC(2, "Patricia", "Velasco Reverendo", "patricia@gmail.com", "05689458J",
                "clavePatricia;");
        //Eliminamos mediante el constructor con un solo atributo, ID.
        ClienteANC cl5 = new ClienteANC(3);

        //Creamos 3 clientes y realizamos una lectura
        HotelManager.crearCliente(cl1);
        HotelManager.crearCliente(cl2);
        HotelManager.crearCliente(cl3);
        HotelManager.leerClientes();

        //Actualizamos el cliente con id = 2 y leemos para confirmar el cambio
        HotelManager.actualizarCliente(cl4);
        HotelManager.leerClientes();

        //Eliminamos el cliente con id = 3 y leemos para confirmar el cambio
        HotelManager.eliminarCliente(cl5);
        HotelManager.leerClientes();
    }
}
```

6.- Ejecución (↑)

6.1.- Crear y Leer clientes

En primer lugar, vamos a comentar las líneas de Update y de Delete para crear los clientes y mostrarlos por consola.

```
19          //Creamos 3 clientes y realizamos una lectura
20          HotelManager.crearCliente(cl1);
21          HotelManager.crearCliente(cl2);
22          HotelManager.crearCliente(cl3);
23          HotelManager.leerClientes();
24
25          //Actualizamos el cliente con id = 2 y leemos para confirmar el cambio
26 //       HotelManager.actualizarCliente(cl4);
27 //       HotelManager.leerClientes();
28
29          //Eliminamos el cliente con id = 3 y leemos para confirmar el cambio
30 //       HotelManager.eliminarCliente(cl5);
31 //       HotelManager.leerClientes();
32      }
33 }
```

Al ejecutar el programa, por consola aparecerán los clientes al ser creados y, finalmente, todos los clientes encontrados en la BBDD.

Cliente1:

```
dic 03, 2024 12:05:04 A. M. org.hibernate.Version logVersion
INFO: HHH000412: Hibernate ORM core version 6.6.2.Final
dic 03, 2024 12:05:04 A. M. org.hibernate.cache.internal.RegionFactoryIn
INFO: HHH000026: Second-level cache disabled
dic 03, 2024 12:05:04 A. M. org.hibernate.engine.jdbc.connections.interr
WARN: HHH10001002: Using built-in connection pool (not intended for proc
dic 03, 2024 12:05:05 A. M. org.hibernate.engine.jdbc.env.internal.JdbcE
INFO: HHH10001005: Database info:
        Database JDBC URL [jdbc:mysql://localhost:3306/hotelanc?createDa
        Database driver: com.mysql.cj.jdbc.Driver
        Database version: 8.0.35
        Autocommit mode: false
        Isolation level: undefined/unknown
        Minimum pool size: 1
        Maximum pool size: 20
dic 03, 2024 12:05:05 A. M. org.hibernate.engine.transaction.jta.platfor
INFO: HHH000489: No JTA platform available (set 'hibernate.transaction.j
dic 03, 2024 12:05:05 A. M. org.hibernate.resource.transaction.backend.j
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hi

Cliente con ID: 1 insertado correctamente en la base de datos hotelanc.
  -Nombre: Álvaro
  -Apellidos: Navarro Cruz
  -Email: alvaromfco@gmail.com
  -DNI: 49545195K
  -Clave: Studium2023;
```

Cliente2:

```
dic 03, 2024 12:05:05 A. M. org.hibernate.cache.internal.RegionFactoryIn
INFO: HHH000026: Second-level cache disabled
dic 03, 2024 12:05:05 A. M. org.hibernate.engine.jdbc.connections.intern
WARN: HHH10001002: Using built-in connection pool (not intended for prod
dic 03, 2024 12:05:05 A. M. org.hibernate.engine.jdbc.env.internal.JdbcE
INFO: HHH10001005: Database info:
        Database JDBC URL [jdbc:mysql://localhost:3306/hotelanc?createDa
        Database driver: com.mysql.cj.jdbc.Driver
        Database version: 8.0.35
        Autocommit mode: false
        Isolation level: undefined/unknown
        Minimum pool size: 1
        Maximum pool size: 20
dic 03, 2024 12:05:05 A. M. org.hibernate.engine.transaction.jta.platfor
INFO: HHH000489: No JTA platform available (set 'hibernate.transaction.j
dic 03, 2024 12:05:05 A. M. org.hibernate.resource.transaction.backend.j
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hi

Cliente con ID: 2 insertado correctamente en la base de datos hotelanc.
  -Nombre: Pepe
  -Apellidos: García Ruíz
  -Email: pepe@gmail.com
  -DNI: 82945876L
  -Clave: clav3Pepe
```

Cliente3:

```
dic 03, 2024 12:05:05 A. M. org.hibernate.cache.internal.RegionFactoryIn
INFO: HHH000026: Second-level cache disabled
dic 03, 2024 12:05:05 A. M. org.hibernate.engine.jdbc.connections.intern
WARN: HHH10001002: Using built-in connection pool (not intended for prod
dic 03, 2024 12:05:05 A. M. org.hibernate.engine.jdbc.env.internal.JdbcE
INFO: HHH10001005: Database info:
        Database JDBC URL [jdbc:mysql://localhost:3306/hotelanc?createDa
        Database driver: com.mysql.cj.jdbc.Driver
        Database version: 8.0.35
        Autocommit mode: false
        Isolation level: undefined/unknown
        Minimum pool size: 1
        Maximum pool size: 20
dic 03, 2024 12:05:05 A. M. org.hibernate.engine.transaction.jta.platfor
INFO: HHH000489: No JTA platform available (set 'hibernate.transaction.j
dic 03, 2024 12:05:05 A. M. org.hibernate.resource.transaction.backend.j
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hi

Cliente con ID: 3 insertado correctamente en la base de datos hotelanc.
  -Nombre: Raúl
  -Apellidos: Escudero Ortíz
  -Email: raul@gmail.com
  -DNI: 25487632A
  -Clave: clav3R4ul
```

Resultado de la consulta:

```
dic 03, 2024 12:05:05 A. M. org.hibernate.cache.internal.RegionFactoryIniti
INFO: HHH000026: Second-level cache disabled
dic 03, 2024 12:05:05 A. M. org.hibernate.engine.jdbc.connections.internal.
WARN: HHH10001002: Using built-in connection pool (not intended for product
dic 03, 2024 12:05:06 A. M. org.hibernate.engine.jdbc.env.internal.JdbcEnvi
INFO: HHH10001005: Database info:
        Database JDBC URL [jdbc:mysql://localhost:3306/hotelanc?createDatab
        Database driver: com.mysql.cj.jdbc.Driver
        Database version: 8.0.35
        Autocommit mode: false
        Isolation level: undefined/unknown
        Minimum pool size: 1
        Maximum pool size: 20
dic 03, 2024 12:05:06 A. M. org.hibernate.engine.transaction.jta.platform.i
INFO: HHH000489: No JTA platform available (set 'hibernate.transaction.jta.
dic 03, 2024 12:05:06 A. M. org.hibernate.resource.transaction.backend.jdbc
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hiber

** Clientes encontrados en la tabla 'clienteanc' **
1 - Álvaro - Navarro Cruz - alvaromfco@gmail.com - 49545195K - Studium2023;
2 - Pepe - García Ruíz - pepe@gmail.com - 82945876L - clav3Pepe
3 - Raúl - Escudero Ortíz - raul@gmail.com - 25487632A - clav3R4ul
```

Si realizamos un SELECT en la BBDD, aparecerá la siguiente tabla con los datos introducidos anteriormente:

```
1 •  SELECT * FROM hotelanc.clienteanc;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

idCliente	nombreCliente	apellidosCliente	emailCliente	dniCliente	claveCliente
1	Álvaro	Navarro Cruz	alvaromfco@gmail.com	49545195K	Studium2023;
2	Pepe	García Ruíz	pepe@gmail.com	82945876L	clav3Pepe
3	Raúl	Escudero Ortíz	raul@gmail.com	25487632A	clav3R4ul
NULL	NULL	NULL	NULL	NULL	NULL

6.2.- Modificar y Leer cliente

Para este paso, vamos a comentar las líneas de creación de usuarios y habilitamos las de modificación:

```
19          //Creamos 3 clientes y realizamos una lectura
20  //      HotelManager.crearCliente(cl1);
21  //      HotelManager.crearCliente(cl2);
22  //      HotelManager.crearCliente(cl3);
23  //      HotelManager.leerClientes();
24
25          //Actualizamos el cliente con id = 2 y leemos para confirmar el cambio
26          HotelManager.actualizarCliente(cl4);
27          HotelManager.leerClientes();
28
29          //Eliminamos el cliente con id = 3 y leemos para confirmar el cambio
30  //      HotelManager.eliminarCliente(cl5);
31  //      HotelManager.leerClientes();
32      }
33  }
```

Esto debería actualizar los datos del cliente con id = 2 de la siguiente manera:

```
13      //Actualizamos mediante el constructor por parámetros con todos los datos.
14      ClienteANC cl4 = new ClienteANC(2, "Patricia", "Velasco Reverendo", "patricia@gmail.com", "05689458J",
15              "clavePatricia;");
```

Al ejecutar el programa, se nos mostrará por consola el cliente con id 2 modificado con los valores pasados:

```
** Actualización Completada correctamente **
   -Actualizado el Cliente con id: 2
dic 03, 2024 12:13:46 A. M. org.hibernate.cache.internal.RegionFactoryInitiator in
INFO: HHH000026: Second-level cache disabled
dic 03, 2024 12:13:46 A. M. org.hibernate.engine.jdbc.connections.internal.DriverM
WARN: HHH10001002: Using built-in connection pool (not intended for production use
dic 03, 2024 12:13:46 A. M. org.hibernate.engine.jdbc.env.internal.JdbcEnvironment
INFO: HHH10001005: Database info:
        Database JDBC URL [jdbc:mysql://localhost:3306/hotelanc?createDatabaseIfNo
        Database driver: com.mysql.cj.jdbc.Driver
        Database version: 8.0.35
        Autocommit mode: false
        Isolation level: undefined/unknown
        Minimum pool size: 1
        Maximum pool size: 20
dic 03, 2024 12:13:47 A. M. org.hibernate.engine.transaction.jta.platform.internal
INFO: HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platfor
dic 03, 2024 12:13:47 A. M. org.hibernate.resource.transaction.backend.jdbc.intern
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.en

** Clientes encontrados en la tabla 'clienteanc' **
1 - Álvaro - Navarro Cruz - alvaromfco@gmail.com - 49545195K - Studium2023;
2 - Patricia - Velasco Reverendo - patricia@gmail.com - 05689458J - clavePatricia;
3 - Raúl - Escudero Ortíz - raul@gmail.com - 25487632A - clav3R4ul
```

Si volvemos a realizar un SELECT en la BBDD, deberíamos confirmar que el cliente con id = 2 ahora tiene los datos modificados:

```
1 •  SELECT * FROM hotelanc.clienteanc;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

idCliente	nombreCliente	apellidosCliente	emailCliente	dniCliente	claveCliente
1	Álvaro	Navarro Cruz	alvaromfco@gmail.com	49545195K	Studium2023;
2	Patricia	Velasco Reverendo	patricia@gmail.com	05689458J	clavePatricia;
3	Raúl	Escudero Ortíz	raul@gmail.com	25487632A	clav3R4ul
NULL	NULL	NULL	NULL	NULL	NULL

6.3.- Eliminar y Leer cliente

Para finalizar, vamos a comentar las líneas de modificación y habilitaremos las de eliminación:

```
19          //Creamos 3 clientes y realizamos una lectura
20 //       HotelManager.crearCliente(cl1);
21 //       HotelManager.crearCliente(cl2);
22 //       HotelManager.crearCliente(cl3);
23 //       HotelManager.leerClientes();
24
25          //Actualizamos el cliente con id = 2 y leemos para confirmar el cambio
26 //       HotelManager.actualizarCliente(cl4);
27 //       HotelManager.leerClientes();
28
29          //Eliminamos el cliente con id = 3 y leemos para confirmar el cambio
30          HotelManager.eliminarCliente(cl5);
31          HotelManager.leerClientes();
32      }
33 }
```

Esto debería eliminar el cliente con id = 3, según esta declaración:

```
16          //Eliminamos mediante el constructor con un solo atributo, ID.
17          ClienteANC cl5 = new ClienteANC(3);
```

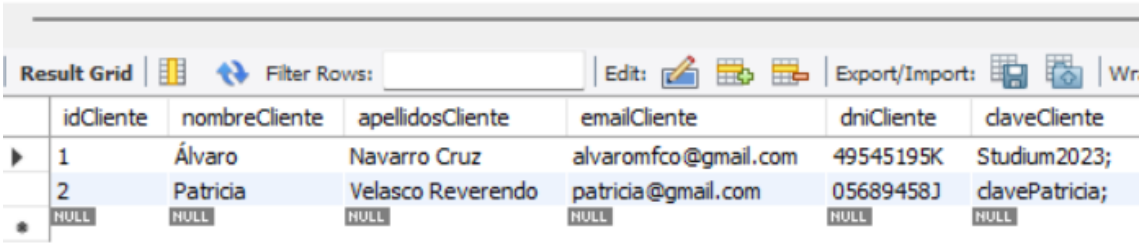
Si ejecutamos el programa, nos aparecerá por consola un aviso de éxito y se nos mostrará la lista completa de los clientes en la tabla actualizada.

```
** Cliente eliminado con éxito **
  -Eliminado el Cliente con id: 3
dic 03, 2024 12:17:55 A. M. org.hibernate.cache.internal.RegionFactoryInitiator in
INFO: HHH000026: Second-level cache disabled
dic 03, 2024 12:17:55 A. M. org.hibernate.engine.jdbc.connections.internal.DriverM
WARN: HHH10001002: Using built-in connection pool (not intended for production use
dic 03, 2024 12:17:55 A. M. org.hibernate.engine.jdbc.env.internal.JdbcEnvironment
INFO: HHH10001005: Database info:
        Database JDBC URL [jdbc:mysql://localhost:3306/hotelanc?createDatabaseIfNo
        Database driver: com.mysql.cj.jdbc.Driver
        Database version: 8.0.35
        Autocommit mode: false
        Isolation level: undefined/unknown
        Minimum pool size: 1
        Maximum pool size: 20
dic 03, 2024 12:17:55 A. M. org.hibernate.engine.transaction.jta.platform.internal
INFO: HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platfor
dic 03, 2024 12:17:55 A. M. org.hibernate.resource.transaction.backend.jdbc.intern
INFO: HHH10001501: Connection obtained from JdbcConnectionAccess [org.hibernate.en

** Clientes encontrados en la tabla 'clienteanc' **
1 - Álvaro - Navarro Cruz - alvaromfco@gmail.com - 49545195K - Studium2023;
2 - Patricia - Velasco Reverendo - patricia@gmail.com - 05689458J - clavePatricia;
```

Si realizamos un SELECT en la BBDD, deberíamos comprobar que el cliente con id = 3 ha sido eliminado:

```
1 •   SELECT * FROM hotelanc.clienteanc;
```



Result Grid | Filter Rows: | Edit: | Export/Import:

idCliente	nombreCliente	apellidosCliente	emailCliente	dniCliente	claveCliente
1	Álvaro	Navarro Cruz	alvaromfco@gmail.com	49545195K	Studium2023;
2	Patricia	Velasco Reverendo	patricia@gmail.com	05689458J	clavePatricia;
NULL	NULL	NULL	NULL	NULL	NULL

7.- Valoración Personal (↑)

Gracias al desarrollo de esta práctica, he conseguido afianzar los conocimientos vistos en la asignatura de Acceso a Datos, en particular, aquellos estudiados en el tema 3 de la asignatura.

Considero que el uso de Hibernate es más que satisfactorio, aportando rapidez y facilidad a la hora de realizar operaciones CRUD en Bases de Datos relacionales.

Tenía muchas ganas de probar este método por mi cuenta y, sinceramente, he salido más que satisfecho.

Todo lo estudiado en esta asignatura nos ha permitido ahorrar tiempo y recursos para realizar prácticas que ocuparían cientos de líneas de código en apenas 70 líneas como mucho, lo que me despierta curiosidad por explorar nuevas herramientas que faciliten el desarrollo de prácticas futuras.

8.-Bibliografía

Grupo Studium (↑)

1. *María José Martínez Navas, Acceso a Datos (Práctica Tema 3)*. Publicado en **Grupo Studium.**
Recuperado:
https://campustudium.com/pluginfile.php/15128/mod_resource/content/4/Tema%203%20-%20Herramientas%20de%20Mapeo%20Objeto-Relacional.%20Hibernate%20-%20Pr%C3%A1ctica_v2.pdf
Último acceso (03/12/2024).