



ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN



CONTENIDO

REALIZAR UNA APLICACIÓN CON PERSISTENCIA A BASES DE DATOS	3
INTRODUCCION.....	3
OBJETIVOS.....	3
Objetivo general:	3
Objetivos específicos:	3
DESARROLLO.....	4
Creacion del proyecto	4
Crear la tabla productos (id, nombre, fecha, precio)	4



	Realizar una aplicación con persistencia A BASES DE DATOS		Página 3 de 11	
	Programación Web Avanzada		Unidad:	3
	Bonilla Hidalgo Jairo Smith		NRC:	23275

REALIZAR UNA APLICACIÓN CON PERSISTENCIA A BASES DE DATOS

INTRODUCCION

En el presente trabajo se desarrolla una aplicación Java con persistencia de datos utilizando Hibernate como framework ORM (Object Relational Mapping) y MySQL como sistema de gestión de base de datos relacional. El proyecto fue implementado en IntelliJ IDEA y tiene como propósito



- **Documentar** el desarrollo del proyecto con capturas de pantalla y código fuente, generando un informe final en PDF que evidencie la funcionalidad implementada.

	Realizar una aplicación con persistencia A BASES DE DATOS		Página 4 de 11	
	Programación Web Avanzada		Unidad:	3
	Bonilla Hidalgo Jairo Smith		NRC:	23275

DESARROLLO

Creacion del proyecto

Para dar inicio al desarrollo, se procedió a crear un nuevo proyecto en IntelliJ IDEA utilizando el lenguaje Java. Durante la configuración inicial, se optó por Maven como gestor de dependencias, lo que permitió simplificar la incorporación de Hibernate y el conector de MySQL a través del archivo *pom.xml*.



Crear la tabla productos (id, nombre, fecha, precio)

En **mysql** se generó la tabla **productos** con los siguientes campos:

	Realizar una aplicación con persistencia A BASES DE DATOS		Página 5 de 11	
	Programación Web Avanzada		Unidad:	3
	Bonilla Hidalgo Jairo Smith		NRC:	23275

- id → entero, clave primaria, autoincremental.
- nombre → texto (VARCHAR 200).
- fecha → tipo DATE.
- precio → tipo DECIMAL(10,0)

```
mysql> desc productos;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id     | int(11)       | NO   | PRI | NULL    | auto_increment |
```



```
22     }
23     }
24 }
25
```

En esta fase se empleó Hibernate para recuperar la totalidad de los registros presentes en la tabla productos utilizando una consulta JPQL.

La aplicación estableció la conexión con la base de datos a través de un EntityManager, ejecutó la sentencia `SELECT c FROM Product c` y posteriormente iteró sobre la colección de objetos Product para mostrar su información en la consola.

The screenshot shows the IntelliJ IDEA IDE. On the left, the 'Project' view displays the file structure of the 'ormdwal' project. The 'main' directory contains a 'java' package, which in turn contains an 'org.espe' package. Inside 'org.espe', there is a 'native' package containing 'ExampleJdbc' and 'productsJdbc'. The 'orm' package contains 'entity', 'mains', and 'product'. The 'product' package contains 'HbCreateProduct', 'HbDeleteProduct', 'HbListProduct' (highlighted with a red box), 'HbProductFind', and 'HbProductWhere'. On the right, the 'HbListProduct.java' file is open, showing the following code:

```
1 package org.espe.orm.product;
2
3 import java.util.*;
4
5
6
7
8
9 public class HbListProduct {
10     public static void main(String[] args) {
11         EntityManager em = JpaUtil.getEntityManager();
12         List<Product> productList = em.createQuery("select c from Product c", Product.class).getResultList();
13
14         productList.forEach(System.out::println);
15         em.close();
16     }
17 }
18
19
20
```



```
9 public class HbProductFind {
10     public static void main(String[] args) {
11         EntityManager em = JpaUtil.getEntityManager();
12         System.out.print("Enter product id: ");
13         Scanner sn = new Scanner(System.in);
14         Long idKeyboard = sn.nextLong();
15         Product product = em.find(Product.class, idKeyboard);
16         System.out.println(product);
17         em.close();
18     }
19 }
```


	Realizar una aplicación con persistencia A BASES DE DATOS		Página 7 de 11	
	Programación Web Avanzada		Unidad:	3
	Bonilla Hidalgo Jairo Smith		NRC:	23275

```
HbProductFind x
ago 15, 2025 12:05:19 A.M. org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
Enter product id: 1
Hibernate: select product0_.id as id1_1_0_, product0_.fecha as fecha2_1_0_, product0_.nombre as nombre3_1_0_, product0_.pre
Product{id=1, nombre='Laptop', fecha=2025-07-18 00:00:00.0, precio=800.0}

Process finished with exit code 0
```

En esta etapa se utilizó Hibernate para ejecutar una consulta JPQL con cláusula WHERE,



Process finished with exit code 0

En esta fase se implementó la funcionalidad de creación de registros en la tabla productos utilizando Hibernate.

La aplicación solicita al usuario la información del nuevo producto nombre, fecha y precio, realiza la conversión de la fecha y el precio a los tipos de datos adecuados y, posteriormente, guarda el objeto **Product** en la base de datos dentro de una transacción controlada.



```
productsJdbc.java  HbListProduct.java  HbProductWhere.java  HbCreateProduct.java
1  package org.espe.orm.product;
2
3  > import ...
4
5  public class HbCreateProduct {
6      public static void main(String[] args) {
7          EntityManager em = JpaUtil.getEntityManager();
8          try {
9              String name = JOptionPane.showInputDialog("name: ");
10             String dateStr = JOptionPane.showInputDialog("Date (dd/MM/yyyy):");
11             // Convertir String a Date
12             SimpleDateFormat sdf = new SimpleDateFormat( pattern: "dd/MM/yyyy");
13             Date date = null;
14             try {
15                 date = sdf.parse(dateStr);
16             } catch (ParseException e) {
17                 e.printStackTrace();
18             }
19             // ...
20         }
21     }
22 }
23
```



utilizando Hibernate.

La aplicación solicita al usuario el ID del producto que desea modificar, obtiene la información actual del registro y ofrece la posibilidad de cambiar sus valores a través de cuadros de diálogo. En caso de que algún campo no sea modificado, se mantiene el dato original. Finalmente, los cambios se aplican en la base de datos empleando el método merge() dentro de una transacción.



```
> import ...  
  
public class HbUpdateProduct {  
    public static void main(String[] args) {  
        EntityManager em = JpaUtil.getEntityManager();  
        try {  
            Long id = Long.valueOf(JOptionPane.showInputDialog("Product to find: "));  
            Product productNew = em.find(Product.class, id);  
  
            String name = JOptionPane.showInputDialog( message: "name: ", productNew.getNombre());  
            SimpleDateFormat sdf = new SimpleDateFormat( pattern: "dd/MM/yyyy");  
            wInputDialog(  
                y):",  
                tFecha())  
        }  
    }  
}
```



En esta fase se desarrolló la operación de eliminación de registros en la tabla productos utilizando Hibernate.

La aplicación solicita al usuario el ID del producto que desea suprimir, localiza el registro correspondiente en la base de datos y, en caso de existir, procede a eliminarlo empleando el método `remove()` dentro de una transacción.

	Realizar una aplicación con persistencia A BASES DE DATOS	Página 10 de 11	
	Programación Web Avanzada	Unidad:	3
	Bonilla Hidalgo Jairo Smith	NRC:	23275

Input

?

Product to Delete (id):

1

OK

Cancel

Verificamos en la base de datos, y se elimino el producto con ID 1



CONCLUSIONES

1. Se comprobó que Hibernate facilita la gestión de la persistencia de datos en Java, permitiendo realizar operaciones CRUD de manera eficiente y reduciendo la complejidad de las consultas SQL nativas.

	Realizar una aplicación con persistencia A BASES DE DATOS		Página 11 de 11	
	Programación Web Avanzada		Unidad:	3
	Bonilla Hidalgo Jairo Smith		NRC:	23275

2. El uso de JPQL y consultas parametrizadas demostró la capacidad de Hibernate para realizar búsquedas dinámicas y filtradas, mejorando la interacción con la base de datos frente a consultas nativas.
3. La implementación de la clase ServiciosMain permitió centralizar las operaciones sobre la base de datos, promoviendo un diseño más limpio y facilitando futuras ampliaciones del proyecto.
4. La práctica permitió entender de forma práctica cómo crear, leer, actualizar y eliminar registros en la base de datos usando Hibernate, así como manejar transacciones y parámetros de manera segura.



Elaborado por:	Revisor por:
<p>-----</p> <p>Nombre: Jairo Smith Bonilla Hidalgo</p> <p>Fecha: 15/08/2025</p>	<p>-----</p> <p>Nombre:</p> <p>Fecha:</p>