docker run --name dk_pPrueba -e POSTGRES_USER=admin -e POSTGRES_PASSWORD=admin -e POSTGRES_DB=pPruebaDB -p 5432:5432 -d postgres

Paso 1: Crear un Proyecto Maven

Esto generará la estructura de un proyecto web básico.

Paso 2: Agregar Dependencias en `pom.xml`

Añade las siguientes dependencias en el archivo `pom.xml`:

```
<dependencies>
    <!-- Hibernate Core -->
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-core</artifactId>
        <version>6.3.1.Final</version>
    </dependency>

    <!-- Hibernate Entity Manager -->
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-entitymanager</artifactId>
        <version>6.3.1.Final</version>
    </dependency>

    <!-- Driver JDBC de la base de datos (Ejemplo con MySQL) -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-j</artifactId>
        <version>8.0.33</version>
    </dependency>

    <!-- Dependencias de Java EE (para Servlets y JSP si es necesario) -->
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>4.0.1</version>
        <scope>provided</scope>
    </dependency>

    <!-- Dependencias opcionales -->
    <dependency>
        <groupId>jakarta.persistence</groupId>
        <artifactId>jakarta.persistence-api</artifactId>
        <version>3.1.0</version>
    </dependency>

    <dependency>
        <groupId>org.glassfish.jaxb</groupId>
        <artifactId>jaxb-runtime</artifactId>
        <version>3.0.2</version>
    </dependency>
</dependencies>
```

para descargar las dependencias.

Paso 3: Configurar `hibernate.cfg.xml`

En la carpeta `src/main/resources`, crea el archivo `hibernate.cfg.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration
DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
        <!-- Configuración de la Base de Datos -->
        <property
name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/mi_base_datos?
useSSL=false</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password">12345</property>

        <!-- Dialecto de Hibernate -->
        <property
name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>

        <!-- Mostrar consultas SQL en consola -->
        <property name="hibernate.show_sql">true</property>
        <property name="hibernate.format_sql">true</property>

        <!-- Actualizar automáticamente la BD -->
        <property name="hibernate.hbm2ddl.auto">update</property>

        <!-- Especificar las clases mapeadas -->
        <mapping class="com.ejemplo.modelo.Usuario"/>
    </session-factory>
</hibernate-configuration>
```

Paso 4: Crear una Clase Modelo (Entidad)

Ejemplo de entidad `Usuario.java`:

```
package com.ejemplo.modelo;

import jakarta.persistence.*;

@Entity
@Table(name = "usuarios")
public class Usuario {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "nombre", nullable = false)
    private String nombre;

    @Column(name = "email", unique = true, nullable = false)
    private String email;

    // Constructores
    public Usuario() {}
```

```
    public Usuario(String nombre, String email) {
        this.nombre = nombre;
        this.email = email;
    }

    // Getters y Setters
    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }

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

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

Paso 5: Crear una Clase de Utilidad para la Sesión

Crea la clase `HibernateUtil.java` en `com.ejemplo.util`:

```
package com.ejemplo.util;

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

public class HibernateUtil {
    private static final SessionFactory sessionFactory = buildSessionFactory();

    private static SessionFactory buildSessionFactory() {
        try {
            return new
Configuration().configure("hibernate.cfg.xml").buildSessionFactory();
        } catch (Throwable ex) {
            System.err.println("Error al inicializar Hibernate: " + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }

    public static void shutdown() {
        getSessionFactory().close();
    }
}
```

Paso 6: Crear una Clase de Prueba

Crea `TestHibernate.java` para probar la conexión con la base de datos:

```
package com.ejemplo.test;

import com.ejemplo.modelo.Usuario;
import com.ejemplo.util.HibernateUtil;
import org.hibernate.Session;
import org.hibernate.Transaction;
```

```
public class TestHibernate {
    public static void main(String[] args) {
        Session session = HibernateUtil.getSessionFactory().openSession();
        Transaction transaction = session.beginTransaction();

        try {
            Usuario usuario = new Usuario("Shinji", "shinji@example.com");
            session.save(usuario);

            transaction.commit();
            System.out.println("Usuario guardado con éxito: " +
usuario.getId());
        } catch (Exception e) {
            transaction.rollback();
            e.printStackTrace();
        } finally {
            session.close();
            HibernateUtil.shutdown();
        }
    }
}
```

Paso 7: Ejecutar la Prueba

Compila y ejecuta:

Si todo está bien, deberías ver la consulta SQL generada en la consola y el usuario guardado en la base de datos.