Ejercicio No. 5. Categoria-Producto (Consola)

Barrera Pérez Carlos Tonatihu
Profesor: José Asunción Enríquez Zárate
Web Application Development
Grupo: 3CM9

2 de junio de 2019

Índice

1. Introducción

Este ejercicio tuvo como objetivo desarrollar un pequeño CRUD para la base de datos que se muestra en la figura 1.

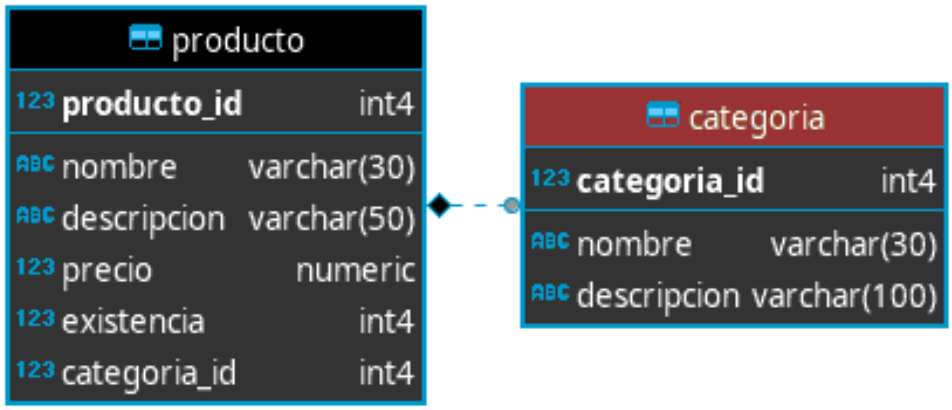

Figura 1: Base de datos que se trabajo en PostgreSQL

A diferencia del CRUD en consola que se realizo con anterioridad, en este se utilizo Hibernate para agilizar el acceso a datos. Y de nueva cuenta se utilizo el patrón DAO para realizar una implementación que no generara problemas en un futuro.

2. Desarrollo

2.1. Código

Las siguientes dos secciones de código se utilizan para configurar Hibernate y poder conectarse a la base de datos.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE hibernate-configuration PUBLIC
3         "-//Hibernate/Hibernate Configuration DTD//EN"
4         "http://www.hibernate.org/dtd/hibernate-
    configuration-3.0.dtd">
5 <hibernate-configuration>
6   <session-factory>
```

```
7       <property
8  name="hibernate.dialect">org.hibernate.dialect.
       PostgreSQLDialect</property>
9       <property
10 name="hibernate.connection.driver_class">org.postgresql.
       Driver</property>
11      <property
12 name="hibernate.connection.url">jdbc:postgresql://
       localhost:5432/tiendita</prope
13 rty>
14      <property name="hibernate.connection.username">postgres
       </property>
15      <property name="connection.password">respuesta42</
       property>
16      <property name="show_sql">true</property>
17          <property name="format_sql">true</property>
18      <property name="hibernate.current_session_context_class
       ">thread</property>
19
20      <mapping class="tiendita.entity.CategoriaEntity"/>
21      <mapping class="tiendita.entity.ProductoEntity"/>
22    </session-factory>
23 </hibernate-configuration>
```

Archivo 1: hibernate.cfg.xml

```
1  package tiendita.util;
2
3  import org.hibernate.HibernateException;
4  import org.hibernate.Session;
5  import org.hibernate.SessionFactory;
6  import org.hibernate.cfg.Configuration;
7
8  /**
9   * Hibernate Utility class with a convenient method to get
       Session Factory
10  * object.
11  *
12  * @author tonatihu
13  */
14 public class HibernateUtil {
15      private static final SessionFactory ourSessionFactory;
16
```

```
17      static {
18          try {
19              Configuration configuration = new Configuration
    ();
20              configuration.configure();
21
22              ourSessionFactory = configuration.
    buildSessionFactory();
23          } catch (HibernateException ex) {
24              throw new ExceptionInInitializerError(ex);
25          }
26      }
27
28      public static Session getSession() throws
    HibernateException {
29          return ourSessionFactory.openSession();
30      }
31  }
```

Archivo 2: HibernateUtil.java

A continuación se muestran las entidades que se configuraron en el archivo de configuración y que mapean las tablas que se encuentran en la base de datos.

```
1  package tiendita.entity;
2
3  import java.io.Serializable;
4  import java.util.ArrayList;
5  import java.util.List;
6  import java.util.Objects;
7  import javax.persistence.Basic;
8  import javax.persistence.Column;
9  import javax.persistence.Entity;
10 import javax.persistence.GeneratedValue;
11 import javax.persistence.GenerationType;
12 import javax.persistence.Id;
13 import javax.persistence.OneToMany;
14 import javax.persistence.Table;
15
16 /**
17  *
18  * @author tonatihu
19  * Created on 02-Jun-2019
```

```
20  */
21 @Entity
22 @Table(name = "categoria", schema = "public", catalog = "
       tiendita")
23 public class CategoriaEntity implements Serializable {
24     @Id
25     @GeneratedValue(strategy = GenerationType.IDENTITY)
26     @Column(name = "categoria_id", nullable = false)
27     private int categoriaId;
28     @Basic
29     @Column(name = "nombre", nullable = false, length = 30)
30     private String nombre;
31     @Basic
32     @Column(name = "descripcion", nullable = false, length
       = 100)
33     private String descripcion;
34
35     @OneToMany(mappedBy = "categoria")
36     private List<ProductoEntity> productos = new ArrayList
       <>();
37
38     public int getCategoriaId() {
39         return categoriaId;
40     }
41
42     public void setCategoriaId(int categoriaId) {
43         this.categoriaId = categoriaId;
44     }
45
46     public String getNombre() {
47         return nombre;
48     }
49
50     public void setNombre(String nombre) {
51         this.nombre = nombre;
52     }
53
54     public String getDescripcion() {
55         return descripcion;
56     }
57
58     public void setDescripcion(String descripcion) {
59         this.descripcion = descripcion;
```

```
60      }
61
62
63      public List<ProductoEntity> getProductos() {
64          return productos;
65      }
66
67      public void setProductos(List<ProductoEntity> productos
        ) {
68          this.productos = productos;
69      }
70
71      public void addProducto(ProductoEntity producto) {
72          this.productos.add(producto);
73      }
74
75      @Override
76      public boolean equals(Object o) {
77          if (this == o) return true;
78          if (o == null || getClass() != o.getClass()) return
        false;
79          CategoriaEntity that = (CategoriaEntity) o;
80          return categoriaId == that.categoriaId && Objects.
       equals(nombre,
81  that.nombre) &&
82                  Objects.equals(descripcion, that.
       descripcion);
83      }
84
85      @Override
86      public int hashCode() {
87          return Objects.hash(categoriaId, nombre,
       descripcion);
88      }
89
90      @Override
91      public String toString() {
92          return "CategoriaEntity{" + "categoriaId=" +
       categoriaId + ", nombre='"
93  + nombre + '\'' +
94                  ", descripcion='" + descripcion + '\'' + '}
       ';
95      }
```

```
96 }
```

Archivo 3: CategoriaEntity.java

```
1  package tiendita.entity;
2
3  import java.io.Serializable;
4  import java.math.BigDecimal;
5  import java.util.Objects;
6  import javax.persistence.Basic;
7  import javax.persistence.Column;
8  import javax.persistence.Entity;
9  import javax.persistence.GeneratedValue;
10 import javax.persistence.GenerationType;
11 import javax.persistence.Id;
12 import javax.persistence.JoinColumn;
13 import javax.persistence.ManyToOne;
14 import javax.persistence.Table;
15
16 /**
17  *
18  * @author tonatihu
19  * Created on 02-Jun-2019
20  */
21 @Entity
22 @Table(name = "producto", schema = "public", catalog = "
      tiendita")
23 public class ProductoEntity implements Serializable {
24     @Id
25     @GeneratedValue(strategy = GenerationType.IDENTITY)
26     @Column(name = "producto_id", nullable = false)
27     private int productoId;
28     @Basic
29     @Column(name = "nombre", nullable = false, length = 30)
30     private String nombre;
31     @Basic
32     @Column(name = "descripcion", nullable = false, length
      = 50)
33     private String descripcion;
34     @Basic
35     @Column(name = "precio", nullable = false)
36     private BigDecimal precio;
37     @Basic
```

```
38      @Column(name = "existencia", nullable = false)
39      private int existencia;
40
41      @ManyToOne
42      @JoinColumn(name = "categoria_id", referencedColumnName
        = "categoria_id",
43  nullable = false)
44      private CategoriaEntity categoria;
45
46
47      public int getProductoId() {
48          return productoId;
49      }
50
51      public void setProductoId(int productoId) {
52          this.productoId = productoId;
53      }
54
55      public String getNombre() {
56          return nombre;
57      }
58
59      public void setNombre(String nombre) {
60          this.nombre = nombre;
61      }
62
63      public String getDescripcion() {
64          return descripcion;
65      }
66
67      public void setDescripcion(String descripcion) {
68          this.descripcion = descripcion;
69      }
70
71      public BigDecimal getPrecio() {
72          return precio;
73      }
74
75      public void setPrecio(BigDecimal precio) {
76          this.precio = precio;
77      }
78
79      public int getExistencia() {
```

```
80          return existencia;
81      }
82
83      public void setExistencia(int existencia) {
84          this.existencia = existencia;
85      }
86
87      @Override
88      public boolean equals(Object o) {
89          if (this == o) return true;
90          if (o == null || getClass() != o.getClass()) return
        false;
91          ProductoEntity that = (ProductoEntity) o;
92          return productoId == that.productoId && existencia
      == that.existencia &&
93  Objects.equals(
94                  nombre, that.nombre) && Objects.equals(
      descripcion,
95  that.descripcion) &&
96                  Objects.equals(precio, that.precio);
97      }
98
99      @Override
100     public int hashCode() {
101         return Objects.hash(productoId, nombre, descripcion
      , precio,
102 existencia);
103     }
104
105     public CategoriaEntity getCategoria() {
106         return categoria;
107     }
108
109     public void setCategoria(CategoriaEntity categoria) {
110         this.categoria = categoria;
111     }
112
113     @Override
114     public String toString() {
115         return "ProductoEntity{" + "productoId=" +
      productoId + ", nombre='" +
116 nombre + '\'' +
117                 ", descripcion='" + descripcion + '\'' + ",
```

```
        precio=" + precio +
118 ", existencia=" +
119                 existencia + ", categoria=" + categoria + '
    }';
120     }
121 }
```

Archivo 4: ProductoEntity.java

La siguientes tres secciones de código es la implementación del patrón DAO, utilizando primero una clase abstracta que modela el comportamiento del resto de DAOs.

```
 1
 2 package dao;
 3
 4 import java.io.Serializable;
 5 import java.util.List;
 6 import org.hibernate.Session;
 7 import org.hibernate.Transaction;
 8 import tiendita.util.HibernateUtil;
 9
10 /**
11  *
12  * @author tonatihu
13  * Created on 02-Jun-2019
14  */
15 public abstract class GenericDAO <T, Id extends
    Serializable>{
16     private Session currentSession;
17     private Transaction currentTransaction;
18
19     public void openCurrentSession() {
20         currentSession = HibernateUtil.getSession();
21     }
22
23     public void openCurrentSessionWithTransaction() {
24         currentSession = HibernateUtil.getSession();
25         currentTransaction = currentSession.
    beginTransaction();
26     }
27
28     public void closeCurrentSession() {
29         currentSession.close();
```

```
30      }
31
32      public void closeCurrentSessionwithTransaction() {
33          currentTransaction.commit();
34          currentSession.close();
35      }
36
37      public void rollback() {
38          if (currentTransaction != null)
39              currentTransaction.rollback();
40      }
41
42      protected Session getCurrentSession() {
43          return currentSession;
44      }
45
46      public abstract void create(T entity);
47      public abstract void update(T entity);
48      public abstract T findById(Id id);
49      public abstract void delete(T entity);
50      public abstract List<T> findAll();
51  }
```

Archivo 5: GenericDAO.java

```
1  package dao.impl;
2
3  import dao.GenericDAO;
4  import java.util.List;
5  import java.util.logging.Level;
6  import java.util.logging.Logger;
7  import org.hibernate.HibernateException;
8  import tiendita.entity.CategoriaEntity;
9
10 /**
11  *
12  * @author tonatihu
13  * Created on 02-Jun-2019
14  */
15 public class CategoriaDAOImpl extends GenericDAO<
       CategoriaEntity, Integer> {
16      private static final Logger LOGGER = Logger
17              .getLogger(CategoriaDAOImpl.class.getName());
```

```
18      @Override
19      public void create(CategoriaEntity entity) {
20          openCurrentSessionWithTransaction();
21          try {
22              getCurrentSession().save(entity);
23          } catch (HibernateException hehe) {
24              LOGGER.log(Level.SEVERE, "Error en create",
    hehe);
25              rollback();
26          } finally {
27              closeCurrentSessionwithTransaction();
28          }
29      }
30
31      @Override
32      public void update(CategoriaEntity entity) {
33          openCurrentSessionWithTransaction();
34          try {
35              getCurrentSession().update(entity);
36          } catch (HibernateException hehe) {
37              LOGGER.log(Level.SEVERE, "Error en update",
    hehe);
38              rollback();
39          } finally {
40              closeCurrentSessionwithTransaction();
41          }
42      }
43
44      @Override
45      public CategoriaEntity findById(Integer integer) {
46          openCurrentSession();
47          CategoriaEntity categoriaEntity = (CategoriaEntity)
    getCurrentSession()
48                  .get(CategoriaEntity.class, integer);
49          closeCurrentSession();
50          return categoriaEntity;
51      }
52
53      @Override
54      public void delete(CategoriaEntity entity) {
55          openCurrentSessionWithTransaction();
56          try {
57              getCurrentSession().delete(entity);
```

```
58          } catch (HibernateException hehe) {
59              LOGGER.log(Level.SEVERE, "Error en delete",
    hehe);
60              rollback();
61          } finally {
62              closeCurrentSessionwithTransaction();
63          }
64      }
65
66      @Override
67      @SuppressWarnings("unchecked")
68      public List<CategoriaEntity> findAll() {
69          openCurrentSession();
70          List<CategoriaEntity> lista = (List<CategoriaEntity
    >)
71  getCurrentSession()
72                  .createQuery("from CategoriaEntity").list()
    ;
73          closeCurrentSession();
74          return lista;
75      }
76  }
```

Archivo 6: CategoriaDAOImpl.java

```
1  package dao.impl;
2
3  import dao.GenericDAO;
4  import java.util.List;
5  import java.util.logging.Level;
6  import java.util.logging.Logger;
7  import org.hibernate.HibernateException;
8  import tiendita.entity.ProductoEntity;
9
10 /**
11  *
12  * @author tonatihu
13  * Created on 02-Jun-2019
14  */
15 public class ProductoDAOImpl extends GenericDAO<
    ProductoEntity, Integer> {
16      private static final Logger LOGGER = Logger
17              .getLogger(ProductoDAOImpl.class.getName());
```

```
18      @Override
19      public void create(ProductoEntity entity) {
20          openCurrentSessionWithTransaction();
21          try {
22              getCurrentSession().save(entity);
23          } catch (HibernateException hehe) {
24              LOGGER.log(Level.SEVERE, "Error en create",
    hehe);
25              rollback();
26          } finally {
27              closeCurrentSessionwithTransaction();
28          }
29      }
30
31      @Override
32      public void update(ProductoEntity entity) {
33          openCurrentSessionWithTransaction();
34          try {
35              getCurrentSession().update(entity);
36          } catch (HibernateException hehe) {
37              LOGGER.log(Level.SEVERE, "Error en update",
    hehe);
38              rollback();
39          } finally {
40              closeCurrentSessionwithTransaction();
41          }
42      }
43
44      @Override
45      public ProductoEntity findById(Integer integer) {
46          openCurrentSession();
47          ProductoEntity productoEntity = (ProductoEntity)
    getCurrentSession()
48                  .get(ProductoEntity.class, integer);
49          closeCurrentSession();
50          return productoEntity;
51      }
52
53      @Override
54      public void delete(ProductoEntity entity) {
55          openCurrentSessionWithTransaction();
56          try {
57              getCurrentSession().delete(entity);
```

```
58          } catch (HibernateException hehe) {
59              LOGGER.log(Level.SEVERE, "Error en delete",
    hehe);
60              rollback();
61          } finally {
62              closeCurrentSessionwithTransaction();
63          }
64      }
65
66      @Override
67      @SuppressWarnings("unchecked")
68      public List<ProductoEntity> findAll() {
69          openCurrentSession();
70          List<ProductoEntity> lista = (List<ProductoEntity>)
    getCurrentSession()
71                  .createQuery("from ProductoEntity").list();
72          closeCurrentSession();
73          return lista;
74      }
75  }
```

Archivo 7: ProductoDAOImpl.java

Finalmente se tiene la clase principal en donde se trabajó el CRUD y se utilizo todo lo anterior.

```
1  package tiendita;
2
3  import dao.impl.CategoriaDAOImpl;
4  import java.io.BufferedReader;
5  import java.io.IOException;
6  import java.io.InputStreamReader;
7  import java.util.List;
8  import tiendita.entity.CategoriaEntity;
9
10 /**
11  *
12  * @author tonatihu
13  * Created on 02-Jun-2019
14  */
15 public class Tiendita {
16
17     /**
18      * @param args the command line arguments
```

```
19      * @throws java.io.IOException
20      */
21     public static void main(String[] args) throws
       IOException {
22         boolean continuar = true;
23         int opcion = 0;
24         BufferedReader br = new BufferedReader(new
25 InputStreamReader(System.in));
26         CategoriaDAOImpl dao = new CategoriaDAOImpl();
27         CategoriaEntity categoria;
28 
29         while (continuar) {
30             System.out.println("1) Alta categoria");
31             System.out.println("2) Baja categoria");
32             System.out.println("3) Cambio categoria");
33             System.out.println("4) Consultar categoria");
34             System.out.println("5) Consultar todas las
       categorias");
35             System.out.println("6) Salir");
36             System.out.println("Selecciona una opcion:");
37             opcion = Integer.valueOf(br.readLine());
38             switch(opcion) {
39                 case 1: // alta
40                     System.out.println("ALTA");
41                     categoria = new CategoriaEntity();
42                     System.out.println("Ingrese el nombre:"
       );
43                     categoria.setNombre(br.readLine());
44                     System.out.println("Ingrese la
       descripcion:");
45                     categoria.setDescripcion(br.readLine())
       ;
46                     dao.create(categoria);
47                     System.out.println("Listo...");
48                     break;
49                 case 2: // baja
50                     System.out.println("BAJA");
51                     categoria = new CategoriaEntity();
52                     System.out.println("Ingrese el id:");
53                     categoria.setCategoriaId(Integer.
       valueOf(br.readLine()));
54                     categoria = dao.findById(categoria.
       getCategoriaId());
```

```
55                      dao.delete(categoria);
56                      System.out.println("Listo...");
57                      break;
58                  case 3: // cambio
59                      System.out.println("CAMBIO");
60                      categoria = new CategoriaEntity();
61                      System.out.println("Ingrese el id:");
62                      categoria.setCategoriaId(Integer.
    valueOf(br.readLine()));
63                      System.out.println("Ingrese la
    descripcion:");
64                      categoria.setDescripcion(br.readLine())
    ;
65                      System.out.println("Ingrese el nombre:"
    );
66                      categoria.setNombre(br.readLine());
67                      System.out.println("Listo...");
68                      dao.update(categoria);
69                      break;
70                  case 4: // consulta
71                      System.out.println("CONSULTA");
72                      categoria = new CategoriaEntity();
73                      System.out.println("Ingrese el id:");
74                      categoria.setCategoriaId(Integer.
    valueOf(br.readLine()));
75                      categoria = dao.findById(categoria.
    getCategoriaId());
76                      if (categoria != null)
77                          System.out.println(categoria.
    toString());
78                      System.out.println("Listo...");
79                      break;
80                  case 5: // consulta todos
81                      System.out.println("CONSULTAR TODOS");
82                      List<CategoriaEntity> categorias = dao.
    findAll();
83                      for (CategoriaEntity c : categorias) {
84                          System.out.println(c.toString());
85                      }
86                      System.out.println("Listo");
87                      break;
88                  default:
89                      continuar = false;
```

```
90                          break;
91                  }
92              }
93              System.out.println("See you space cowboy...");
94          }
95  }
```

Archivo 8: Tiendita.java

2.2. Consultar todos

```
1) Alta categoria
2) Baja categoria
3) Cambio categoria
4) Consultar categoria
5) Consultar todas las categorias
6) Salir
Selecciona una opcion:
5
CONSULTAR TODOS
Hibernate:
    select
        categoriae0_.categoria_id as categori1_0_,
        categoriae0_.descripcion as descripc2_0_,
        categoriae0_.nombre as nombre3_0_
    from
        tiendita.public.categoria categoriae0_
CategoriaEntity{categoriaId=1, nombre='Electrónica', descripcion='Articulos para el hogar'}
CategoriaEntity{categoriaId=2, nombre='Computación', descripcion='Articulos para el hogar'}
CategoriaEntity{categoriaId=3, nombre='Línea Blanca', descripcion='Articulos para el hogar'}
CategoriaEntity{categoriaId=4, nombre='Alimentos', descripcion='Los alimentos son chidos'}
Listo
```

Figura 2: Se consultan todos

2.3. Cambio

```
1) Alta categoria
2) Baja categoria
3) Cambio categoria
4) Consultar categoria
5) Consultar todas las categorias
6) Salir
Selecciona una opcion:
3
CAMBIO
Ingrese el id:
2
Ingrese la descripcion:
La computacion es una subcategoria de la electronica
Ingrese el nombre:
Computación
Listo...
Hibernate:
    update
        tiendita.public.categoria
    set
        descripcion=?,
        nombre=?
    where
        categoria_id=?
```

Figura 3: Se realizan cambios en una categoría

2.4. Consulta

```
1) Alta categoria
2) Baja categoria
3) Cambio categoria
4) Consultar categoria
5) Consultar todas las categorias
6) Salir
Selecciona una opcion:
4
CONSULTA
Ingrese el id:
2
Hibernate:
    select
        categoriae0_.categoria_id as categori1_0_0_,
        categoriae0_.descripcion as descripc2_0_0_,
        categoriae0_.nombre as nombre3_0_0_
    from
        tiendita.public.categoria categoriae0_
    where
        categoriae0_.categoria_id=?
CategoriaEntity{categoriaId=2, nombre='Computación', descripcion='La computacion es una subcategoria de la electronica'}
Listo...
```

Figura 4: Se realiza una consulta en el que se realizo el cambio

2.5. Alta

```
1) Alta categoria
2) Baja categoria
3) Cambio categoria
4) Consultar categoria
5) Consultar todas las categorias
6) Salir
Selecciona una opcion:
1
ALTA
Ingrese el nombre:
No sé
Ingrese la descripcion:
Una descripción bien chida
Hibernate:
    insert
    into
        tiendita.public.categoria
        (descripcion, nombre)
    values
        (?, ?)
Listo...
```

Figura 5: Alta de una categoría

2.6. Baja

```
1) Alta categoria
2) Baja categoria
3) Cambio categoria
4) Consultar categoria
5) Consultar todas las categorias
6) Salir
Selecciona una opcion:
2
BAJA
Ingrese el id:
6
Hibernate:
    select
        categoriae0_.categoria_id as categori1_0_0_,
        categoriae0_.descripcion as descripc2_0_0_,
        categoriae0_.nombre as nombre3_0_0_
    from
        tiendita.public.categoria categoriae0_
    where
        categoriae0_.categoria_id=?
Hibernate:
    delete
    from
        tiendita.public.categoria
    where
        categoria_id=?
Listo...
```

Figura 6: Baja de una categoría

3. Conclusiones

El uso de Hibernate facilita bastante el trabajar con bases de datos ya que se pueden evitar bastantes pasos que se tenían que realizar usando solamente JDBC como es el caso de la conexión a la base de datos.

Es importante tener en cuenta que se debe de usar correctamente ya que de lo contrario podría generar más problemas de los que pretende solucionar. En este caso, debido a el como se implemento puede ser usado en futuros

trabajos de esta materia ya que el uso de Hibernate no se restringe solo a proyectos web o hechos solo con Java.