**AE-4. JPA**

**Requerimiento 2**

**Ivan Nuñez Rodriguez**

2º DAM

Acceso a Datos

**Índice:**

1. Código.
2. Ejecución y resultado.
3. Enlace a GitHub.

**Código:**

**Main**

import controller.CursoController;  
import database.HibernateUtil;  
import model.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 CursoController cursoController = new CursoController();  
  
 // Crear usuarios  
 Usuario usuario1 = new Usuario("Jose@example.com", "1234");  
 Usuario usuario2 = new Usuario("Alba@example.com", "4321");  
 Usuario usuario3 = new Usuario("Daniel@example.com", "5678");  
 Usuario usuario4 = new Usuario("Ivan@example.com", "8765");  
  
 cursoController.agregarUsuario(usuario1);  
 cursoController.agregarUsuario(usuario2);  
 cursoController.agregarUsuario(usuario3);  
 cursoController.agregarUsuario(usuario4);  
  
 // Crear perfiles  
 cursoController.agregarPerfil(new Perfil("Jose", "Perez", usuario1));  
 cursoController.agregarPerfil(new Perfil("Alba", "Nuñez", usuario2));  
 cursoController.agregarPerfil(new Perfil("Daniel", "Gomez", usuario3));  
 cursoController.agregarPerfil(new Perfil("Ivan", "Nuñez", usuario4));  
  
 // Crear estudiantes  
 cursoController.agregarEstudiante(new Estudiante("Jose", usuario1.getEmail(), usuario1));  
 cursoController.agregarEstudiante(new Estudiante("Alba", usuario2.getEmail(), usuario2));  
 cursoController.agregarEstudiante(new Estudiante("Daniel", usuario3.getEmail(), usuario3));  
 cursoController.agregarEstudiante(new Estudiante("Ivan", usuario4.getEmail(), usuario4));  
  
 // Crear profesores  
 Profesor profesor1 = new Profesor("Manuel", "SGE");  
 Profesor profesor2 = new Profesor("Ana", "FOL");  
 Profesor profesor3 = new Profesor("Daniel", "DEVops");  
  
 cursoController.agregarProfesor(profesor1);  
 cursoController.agregarProfesor(profesor2);  
 cursoController.agregarProfesor(profesor3);  
  
 // Crear cursos  
 cursoController.agregarCurso(new Curso("DAM", 40, profesor1));  
 cursoController.agregarCurso(new Curso("DAM2", 50, profesor2));  
  
 // Mostrar resultados  
 cursoController.listarProfesores();  
 cursoController.listarEstudiantes();  
 cursoController.listarCursos();  
 cursoController.listarUsuarios();  
 cursoController.listarPerfiles();  
  
  
 HibernateUtil.*close*();  
 }  
}

**Model.curso**

package model;  
  
import jakarta.persistence.\*;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
@Entity  
  
public class Curso {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 @Column(nullable = false)  
 private String nombre;  
  
 @Column(nullable = false)  
 private int duracionHoras;  
  
 @ManyToOne  
 @JoinColumn(name = "profesor\_id", nullable = false)  
 private Profesor profesor;  
  
  
 public Curso(String nombre, int duracionHoras, Profesor profesor) {  
 this.nombre = nombre;  
 this.duracionHoras = duracionHoras;  
 this.profesor = profesor;  
 }  
}

**Model.estudiante**

package model;  
  
import jakarta.persistence.\*;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
import java.util.Set;  
  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
@Entity  
  
public class Estudiante {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 @Column(nullable = false)  
 private String nombre;  
  
 @Column(nullable = false, unique = true)  
 private String email;  
  
 @ManyToOne  
 @JoinColumn(name = "usuario\_id", nullable = false)  
 private Usuario usuario;  
  
 @ManyToMany  
 @JoinTable(name = "inscripciones", joinColumns = @JoinColumn(name = "estudiante\_id"), inverseJoinColumns = @JoinColumn(name = "curso\_id"))  
 private Set<Curso> cursos;  
  
 public Estudiante(String nombre, String email, Usuario usuario) {  
 this.nombre = nombre;  
 this.email = email;  
 this.usuario = usuario;  
 }  
}

**Model.perfil**

package model;  
  
import jakarta.persistence.\*;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
@Entity  
  
public class Perfil {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 @Column(nullable = false)  
 private String nombre;  
  
 @Column(nullable = false)  
 private String apellido;  
  
 @OneToOne  
 @JoinColumn(name = "usuario\_id", nullable = false)  
 private Usuario usuario;  
  
  
 public Perfil(String nombre, String apellido, Usuario usuario) {  
 this.nombre = nombre;  
 this.apellido = apellido;  
 this.usuario = usuario;  
 }  
}

**Model.profesor**

package model;  
  
import jakarta.persistence.\*;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
import java.util.List;  
  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
@Entity  
  
public class Profesor {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 @Column(nullable = false)  
 private String nombre;  
  
 @Column(nullable = false)  
 private String especialidad;  
  
 @OneToMany(mappedBy = "profesor", cascade = CascadeType.*ALL*)  
 private List<Curso> cursos;  
  
  
 public Profesor(String nombre, String especialidad) {  
 this.nombre = nombre;  
 this.especialidad = especialidad;  
 }  
}

**Model.usuario**

package model;  
  
import jakarta.persistence.\*;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
@Entity  
  
public class Usuario {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 @Column(nullable = false, unique = true)  
 private String email;  
  
 @Column(nullable = false)  
 private String password;  
  
 @OneToOne(mappedBy = "usuario", cascade = CascadeType.*ALL*)  
 private Perfil perfil;  
  
  
 public Usuario(String email, String password) {  
 this.email = email;  
 this.password = password;  
 }  
}

**database.hibernateutil**

package database;  
  
import org.hibernate.Session;  
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().buildSessionFactory();  
 } catch (Throwable ex) {  
 throw new ExceptionInInitializerError("Error al crear la SessionFactory: " + ex);  
 }  
 }  
  
 public static Session getSession() {  
 return *sessionFactory*.openSession();  
 }  
  
 public static void close() {  
 *sessionFactory*.close();  
 }  
}

**DAO.cursoDAO**

package dao;  
  
import database.HibernateUtil;  
import model.Curso;  
import org.hibernate.Session;  
  
import java.util.List;  
  
public class CursoDAO {  
  
  
 public void agregarCurso(Curso curso) {  
 try (Session session = HibernateUtil.*getSession*()) {  
 session.beginTransaction();  
 session.persist(curso);  
 session.getTransaction().commit();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public List<Curso> obtenerTodosLosCursos() {  
 try (Session session = HibernateUtil.*getSession*()) {  
 session.beginTransaction();  
 List<Curso> cursos = session.createQuery("SELECT c FROM Curso c", Curso.class).getResultList();  
 session.getTransaction().commit();  
 return cursos;  
  
  
 }  
 }  
}

**DAO.estudianteDAO**

package dao;  
  
import database.HibernateUtil;  
import model.Estudiante;  
import org.hibernate.Session;  
  
import java.util.List;  
  
public class EstudianteDAO {  
  
  
 public void agregarEstudiante(Estudiante estudiante) {  
 try (Session session = HibernateUtil.*getSession*()) {  
 session.beginTransaction();  
 Long count = session.createQuery("SELECT COUNT(e) FROM Estudiante e WHERE e.email = :email", Long.class).setParameter("email", estudiante.getEmail()).getSingleResult();  
 if (count == 0) {  
 session.persist(estudiante);  
 System.*out*.println("Estudiante agregado: " + estudiante.getNombre());  
 } else {  
 System.*out*.println("El estudiante con email " + estudiante.getEmail() + " ya existe.");  
 }  
 session.getTransaction().commit();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
  
 public List<Estudiante> obtenerTodosLosEstudiantes() {  
 try (Session session = HibernateUtil.*getSession*()) {  
 session.beginTransaction();  
 List<Estudiante> estudiantes = session.createQuery("SELECT e FROM Estudiante e", Estudiante.class).getResultList();  
 session.getTransaction().commit();  
 return estudiantes;  
 }  
 }  
}

**DAO.perfilDAO**

package dao;  
  
import database.HibernateUtil;  
import model.Perfil;  
import org.hibernate.Session;  
  
import java.util.List;  
  
public class PerfilDAO {  
  
 public void agregarPerfil(Perfil perfil) {  
 try (Session session = HibernateUtil.*getSession*()) {  
 session.beginTransaction();  
 Long count = session.createQuery("SELECT COUNT(p) FROM Perfil p WHERE p.usuario.id = :usuarioId", Long.class).setParameter("usuarioId", perfil.getUsuario().getId()).getSingleResult();  
 if (count == 0) {  
 session.persist(perfil);  
 System.*out*.println("Perfil agregado: " + perfil.getNombre());  
 } else {  
 System.*out*.println("Ya existe un perfil asociado al usuario " + perfil.getUsuario().getEmail());  
 }  
 session.getTransaction().commit();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public List<Perfil> listarPerfil() {  
 try (Session session = HibernateUtil.*getSession*()) {  
 session.beginTransaction();  
 List<Perfil> perfil = session.createQuery("SELECT p from Perfil p", Perfil.class).getResultList();  
 session.getTransaction().commit();  
 return perfil;  
 }  
 }  
}

**DAO.profesorDAO**

package dao;  
  
import database.HibernateUtil;  
import model.Profesor;  
import org.hibernate.Session;  
  
import java.util.List;  
  
public class ProfesorDAO {  
  
 public void agregarProfesor(Profesor profesor) {  
 try (Session session = HibernateUtil.*getSession*()) {  
 session.beginTransaction();  
 session.persist(profesor);  
 session.getTransaction().commit();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public List<Profesor> obtenerTodosLosProfesores() {  
 try (Session session = HibernateUtil.*getSession*()) {  
 session.beginTransaction();  
 List<Profesor> profesores = session.createQuery("SELECT p FROM Profesor p", Profesor.class).getResultList();  
 session.getTransaction().commit();  
 return profesores;  
  
  
 }  
 }  
}

**DAO.usuarioDAO**

package dao;  
  
import database.HibernateUtil;  
import model.Usuario;  
import org.hibernate.Session;  
  
import java.util.List;  
  
public class UsuarioDAO {  
  
 public void agregarUsuario(Usuario usuario) {  
 try (Session session = HibernateUtil.*getSession*()) {  
 session.beginTransaction();  
 Long count = session.createQuery("SELECT COUNT(u) FROM Usuario u WHERE u.email = :email", Long.class).setParameter("email", usuario.getEmail()).getSingleResult();  
 if (count == 0) {  
 session.persist(usuario);  
 System.*out*.println("Usuario agregado: " + usuario.getEmail());  
 } else {  
 System.*out*.println("El usuario con email " + usuario.getEmail() + " ya existe. No se insertará nuevamente.");  
 }  
 session.getTransaction().commit();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
  
 public List<Usuario> listarUsuarios() {  
 try (Session session = HibernateUtil.*getSession*()) {  
 session.beginTransaction();  
 List<Usuario> usuarios = session.createQuery("FROM Usuario", Usuario.class).getResultList();  
 session.getTransaction().commit();  
 return usuarios;  
 }  
 }  
}

**controller.cursocontroller**

package controller;  
  
import dao.\*;  
import model.\*;  
  
public class CursoController {  
 private final CursoDAO cursoDAO = new CursoDAO();  
 private final EstudianteDAO estudianteDAO = new EstudianteDAO();  
 private final ProfesorDAO profesorDAO = new ProfesorDAO();  
 private final PerfilDAO perfilDAO = new PerfilDAO();  
 private final UsuarioDAO usuarioDAO = new UsuarioDAO();  
  
 public void agregarEstudiante(Estudiante estudiante) {  
 estudianteDAO.agregarEstudiante(estudiante);  
 }  
  
 public void agregarProfesor(Profesor profesor) {  
 profesorDAO.agregarProfesor(profesor);  
 }  
  
 public void agregarCurso(Curso curso) {  
 cursoDAO.agregarCurso(curso);  
 }  
  
 public void agregarPerfil(Perfil perfil) {  
 perfilDAO.agregarPerfil(perfil);  
 }  
  
 public void agregarUsuario(Usuario usuario) {  
 usuarioDAO.agregarUsuario(usuario);  
 }  
  
 public void listarCursos() {  
 cursoDAO.obtenerTodosLosCursos().forEach(c -> System.*out*.println("Curso: " + c.getNombre() + " - " + c.getDuracionHoras() + " horas"));  
 }  
  
 public void listarEstudiantes() {  
 estudianteDAO.obtenerTodosLosEstudiantes().forEach(e -> System.*out*.println("Estudiante: " + e.getNombre() + " - " + e.getEmail()));  
 }  
  
 public void listarProfesores() {  
 profesorDAO.obtenerTodosLosProfesores().forEach(p -> System.*out*.println("Profesor: " + p.getNombre() + " - " + p.getEspecialidad()));  
 }  
  
 public void listarUsuarios() {  
 usuarioDAO.listarUsuarios().forEach(u -> System.*out*.println("Usuario: " + u.getId() + " - " + u.getEmail()));  
 }  
  
 public void listarPerfiles() {  
 perfilDAO.listarPerfil().forEach(p -> System.*out*.println("Perfil: " + p.getId() + " - " + p.getNombre() + " " + p.getApellido()));  
 }  
}

**Ejecución y resultado.**

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

**Captura de pantalla de un videojuego

El contenido generado por IA puede ser incorrecto.**

**Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.**

**Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.**

**Captura de pantalla de un videojuego

El contenido generado por IA puede ser incorrecto.**

**Texto

El contenido generado por IA puede ser incorrecto.**

**Enlace a GitHub:**

<https://github.com/Ivannunezrodriguez/Acceso_a_Datos_24-25/tree/7af1c26c9ac5e6a2a204f755064fec96e5894a42/Acitvidades/AE-4.%20JPA_Requerimiento%202>