Logotipo, nombre de la empresa

Descripción generada automáticamente

**Sistema de Gestión de Tareas**

**Ingeniería en Informática**

INTEGRANTES:

LUIS ANTONIO FERNANDEZ LOPEZ

[luis.fernandez08894@edu.ipchile.cl](mailto:%6c%75%69s%2efe%72%6e%61%6e%64%65%7a%3088%394@%65%64%75%2eipc%68%69l%65%2e%63%6c)

ANDRES REYNALDO TORRES CALQUIN

[andres.torres37367@edu.ipchile.cl](mailto:andres.torres37367@edu.ipchile.cl)

SOL NINOSKA BRICEÑO ALVAREZ

[sol.briceno79454@edu.ipchile.cl](mailto:sol.briceno79454@edu.ipchile.cl)

PABLO DAVID ORMAZABAL ROJAS

[pablo.ormazabal15483@edu.ipchile.cl](mailto:pablo.ormazabal15483@edu.ipchile.cl)

CAROLINE PATRICIA CAMUS RIVAS

[caroline.camus08844@edu.ipchile.cl](mailto:caroline.camus08844@edu.ipchile.cl)

# SCRIPT BASE DE DATOS

-- Crear la base de datos

CREATE DATABASE gestion\_estudiantes;

-- Usar la base de datos

USE gestion\_estudiantes;

-- Tabla para almacenar estudiantes

CREATE TABLE estudiante (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    nombre VARCHAR(100) NOT NULL,

    edad INT NOT NULL

);

-- Tabla para almacenar cursos

CREATE TABLE curso (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    nombre VARCHAR(100) NOT NULL

);

-- Tabla de relación muchos a muchos entre estudiantes y cursos

CREATE TABLE estudiante\_curso (

    estudiante\_id INT,

    curso\_id INT,

    PRIMARY KEY (estudiante\_id, curso\_id),

    FOREIGN KEY (estudiante\_id) REFERENCES estudiante(id) ON DELETE CASCADE,

    FOREIGN KEY (curso\_id) REFERENCES curso(id) ON DELETE CASCADE

);

curso

CODIGOS FUENTES DEL SISTEMA DE ACUERDO A LAS PAGINAS SOLICITADAS

package com.mycompany.model;

import jakarta.persistence.\*; // Importación corregida para Jakarta Persistence API

import java.io.Serializable;

import java.util.HashSet;

import java.util.Set;

@Entity

public class Student implements Serializable {

    private static final long serialVersionUID = 1L; // ID de versión para serialización

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String name;

    private int age;

    @ManyToMany(mappedBy = "students")

    private Set<Course> courses = new HashSet<>(); // Usar Set para evitar duplicados y asegurar integridad

    // Constructor

    public Student() {}

    // Getters y Setters

    public Long getId() {

        return id;

    }

    public void setId(Long id) {

        this.id = id;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public int getAge() {

        return age;

    }

    public void setAge(int age) {

        this.age = age;

    }

    public Set<Course> getCourses() {

        return courses;

    }

    public void setCourses(Set<Course> courses) {

        this.courses = courses;

    }

    // En caso de que quieras añadir funcionalidad para manejar el email, añádelo aquí

    // Si no es necesario, puedes eliminar este método

    public void setEmail(String email) {

        throw new UnsupportedOperationException("Email field is not supported.");

    }

}

package com.mycompany.model;

import jakarta.persistence.\*; // Cambiado de javax.persistence a jakarta.persistence

import java.io.Serializable;

import java.util.HashSet;

import java.util.Set;

@Entity

public class Course implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String name;

    @ManyToMany

    @JoinTable(name = "student\_course",

               joinColumns = @JoinColumn(name = "course\_id"),

               inverseJoinColumns = @JoinColumn(name = "student\_id"))

    private Set<Student> students = new HashSet<>(); // Usar Set para evitar duplicados

    // Constructor

    public Course() {}

    // Getters y Setters

    public Long getId() {

        return id;

    }

    public void setId(Long id) {

        this.id = id;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public Set<Student> getStudents() {

        return students;

    }

    public void setStudents(Set<Student> students) {

        this.students = students;

    }

}

package com.mycompany.controller;

import com.mycompany.model.Student;

import com.mycompany.model.Course;

import jakarta.persistence.EntityManager;

import jakarta.persistence.EntityManagerFactory;

import jakarta.persistence.Persistence;

import jakarta.persistence.Query;

import java.util.List;

public class StudentController {

    private final EntityManagerFactory emf;

    public StudentController() {

        this.emf = Persistence.createEntityManagerFactory("StudentManagementPU");

    }

    public void createStudent(Student student) {

        EntityManager em = emf.createEntityManager();

        try {

            em.getTransaction().begin();

            em.persist(student);

            em.getTransaction().commit();

        } catch (Exception e) {

            e.printStackTrace(); // Imprime la traza del error para depuración

        } finally {

            em.close(); // Asegúrate de cerrar el EntityManager

        }

    }

    public void updateStudent(Student student) {

        EntityManager em = emf.createEntityManager();

        try {

            em.getTransaction().begin();

            em.merge(student); // Utiliza merge para actualizar el estudiante

            em.getTransaction().commit();

        } catch (Exception e) {

            e.printStackTrace(); // Imprime la traza del error para depuración

        } finally {

            em.close(); // Asegúrate de cerrar el EntityManager

        }

    }

    public List<Student> getAllStudents() {

        EntityManager em = emf.createEntityManager();

        try {

            Query query = em.createQuery("SELECT s FROM Student s");

            return query.getResultList();

        } finally {

            em.close(); // Asegúrate de cerrar el EntityManager

        }

    }

    public Student getStudentById(Long studentId) {

        EntityManager em = emf.createEntityManager();

        try {

            return em.find(Student.class, studentId);

        } finally {

            em.close(); // Asegúrate de cerrar el EntityManager

        }

    }

    public void enrollStudentInCourse(Long studentId, Long courseId) {

        EntityManager em = emf.createEntityManager();

        try {

            em.getTransaction().begin();

            Student student = em.find(Student.class, studentId);

            Course course = em.find(Course.class, courseId);

            if (student != null && course != null) {

                student.getCourses().add(course);

                em.getTransaction().commit();

            } else {

                throw new IllegalArgumentException("Student or Course not found");

            }

        } catch (IllegalArgumentException e) {

            e.printStackTrace(); // Imprime la traza del error para depuración

        } finally {

            em.close(); // Asegúrate de cerrar el EntityManager

        }

    }

}

/\*

 \* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

 \* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

 \*/

package com.mycompany.controller;

/\*\*

 \*

 \* @author Luis

 \*/

import com.mycompany.model.Course;

import com.mycompany.model.Student;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import jakarta.persistence.EntityManager;

import jakarta.persistence.EntityManagerFactory;

import jakarta.persistence.Persistence;

import jakarta.persistence.Query;

import java.io.IOException;

import java.util.List;

@WebServlet("/course")  // Define el servlet para manejar las solicitudes en esta ruta

public class CourseController extends HttpServlet {

    private final EntityManagerFactory emf;

    public CourseController() {

        this.emf = Persistence.createEntityManagerFactory("StudentManagementPU");

    }

    public void createCourse(Course course) {

        EntityManager em = emf.createEntityManager();

        try {

            em.getTransaction().begin();

            em.persist(course);

            em.getTransaction().commit();

        } catch (Exception e) {

            if (em.getTransaction().isActive()) {

                em.getTransaction().rollback();

            }

            e.printStackTrace();

        } finally {

            em.close();

        }

    }

    public List<Course> getAllCourses() {

        try (EntityManager em = emf.createEntityManager()) {

            Query query = em.createQuery("SELECT c FROM Course c");

            return query.getResultList();

        }

    }

    public List<Student> getStudentsByCourse(Long courseId) {

        try (EntityManager em = emf.createEntityManager()) {

            Query query = em.createQuery("SELECT s FROM Student s JOIN s.courses c WHERE c.id = :courseId");

            query.setParameter("courseId", courseId);

            return query.getResultList();

        }

    }

    public List<Course> getCoursesByStudent(Long studentId) {

        try (EntityManager em = emf.createEntityManager()) {

            Query query = em.createQuery("SELECT c FROM Course c JOIN c.students s WHERE s.id = :studentId");

            query.setParameter("studentId", studentId);

            return query.getResultList();

        }

    }

    @Override

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        try {

            String name = request.getParameter("name");

            Course course = new Course();

            course.setName(name);

            createCourse(course);  // Reutilizamos el método createCourse

            response.sendRedirect("lista\_cursos.jsp"); // Redirige a la lista de cursos

        } catch (IOException e) {

            request.setAttribute("errorMessage", "Error al crear el curso.");

            request.getRequestDispatcher("error.jsp").forward(request, response);

        }

    }

}

package com.mycompany.servlet;

import com.mycompany.controller.StudentController;

import com.mycompany.model.Student;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.util.List;

@WebServlet("/StudentServlet")

public class StudentServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    private StudentController studentController;

    @Override

    public void init() throws ServletException {

        studentController = new StudentController();

    }

    @Override

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        String id = request.getParameter("id"); // Puede ser nulo si es un nuevo estudiante

        String name = request.getParameter("name");

        int age;

        try {

            age = Integer.parseInt(request.getParameter("age"));

        } catch (NumberFormatException e) {

            request.setAttribute("errorMessage", "Invalid age format.");

            request.getRequestDispatcher("error.jsp").forward(request, response);

            return;

        }

        if (id == null || id.isEmpty()) {

            // Crear nuevo estudiante

            Student student = new Student();

            student.setName(name);

            student.setAge(age);

            studentController.createStudent(student);

        } else {

            // Editar estudiante existente

            Long studentId;

            try {

                studentId = Long.parseLong(id);

            } catch (NumberFormatException e) {

                request.setAttribute("errorMessage", "Invalid student ID format.");

                request.getRequestDispatcher("error.jsp").forward(request, response);

                return;

            }

            Student student = studentController.getStudentById(studentId);

            if (student != null) {

                student.setName(name);

                student.setAge(age);

                studentController.updateStudent(student);

            } else {

                request.setAttribute("errorMessage", "Student not found.");

                request.getRequestDispatcher("error.jsp").forward(request, response);

                return;

            }

        }

        response.sendRedirect("lista\_estudiantes.jsp");

    }

    @Override

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        String action = request.getParameter("action");

        if ("edit".equals(action)) {

            // Obtener los datos del estudiante para editar

            Long id;

            try {

                id = Long.parseLong(request.getParameter("id"));

            } catch (NumberFormatException e) {

                request.setAttribute("errorMessage", "Invalid student ID format.");

                request.getRequestDispatcher("error.jsp").forward(request, response);

                return;

            }

            Student student = studentController.getStudentById(id);

            request.setAttribute("student", student);

            request.getRequestDispatcher("formulario\_estudiante.jsp").forward(request, response);

        } else if ("list".equals(action)) {

            // Listar todos los estudiantes

            List<Student> students = studentController.getAllStudents();

            request.setAttribute("students", students);

            request.getRequestDispatcher("lista\_estudiantes.jsp").forward(request, response);

        }

    }

}

import com.mycompany.controller.CourseController;

import com.mycompany.model.Course;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.util.List;

@WebServlet("/CourseServlet")

public class CourseServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    private CourseController courseController;

    // Inicializa el controlador de cursos

    @Override

    public void init() throws ServletException {

        courseController = new CourseController();

    }

    // Maneja las solicitudes POST para crear un nuevo curso

    @Override

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        String name = request.getParameter("name");  // Obtiene el nombre del curso desde el formulario

        Course course = new Course();

        course.setName(name);

        // Crea un nuevo curso utilizando el controlador

        courseController.createCourse(course);

        // Redirige a la lista de cursos después de crear el curso

        response.sendRedirect("lista\_cursos.jsp");

    }

    // Maneja las solicitudes GET, como listar cursos

    @Override

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        String action = request.getParameter("action");

        if ("list".equals(action)) {

            // Obtiene todos los cursos utilizando el controlador

            List<Course> courses = courseController.getAllCourses();

            // Envía la lista de cursos al JSP para mostrarlos

            request.setAttribute("courses", courses);

            request.getRequestDispatcher("lista\_cursos.jsp").forward(request, response);

        }

        // Puedes agregar más acciones aquí, como editar o eliminar cursos

    }

}

<%--

    Document   : index

    Created on : 11-09-2024, 5:47:20 p. m.

    Author     : Luis

--%>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Administración de Estudiantes</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <header>

        <h1>Bienvenidos al Sistema de Ingreso de Estudiantes</h1>

    </header>

    <main>

        <nav>

            <ul>

                <li><a href="lista\_estudiantes.jsp">Lista de Estudiantes</a></li>

                <li><a href="agregar\_estudiante.jsp">Agregar Estudiante</a></li>

            </ul>

        </nav>

    </main>

    <footer>

        <p>&copy; 2024 Administración de Estudiantes. Todos los derechos reservados.</p>

    </footer>

</body>

</html>

<%--

    Document   : lista\_estudiantes

    Created on : 05-09-2024, 11:55:18 p. m.

    Author     : Luis

--%>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Lista de Estudiantes</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <header>

        <h1>Lista de Estudiantes</h1>

    </header>

    <main>

        <table border="1">

            <thead>

                <tr>

                    <th>ID</th>

                    <th>Nombre</th>

                    <th>Edad</th>

                    <th>Acciones</th>

                </tr>

            </thead>

            <tbody>

                <c:forEach var="student" items="${students}">

                    <tr>

                        <td>${student.id}</td>

                        <td>${student.name}</td>

                        <td>${student.age}</td>

                            <a href="StudentServlet?action=edit&id=${student.id}">Editar</a>

                            <a href="delete\_student.jsp?id=${student.id}" onclick="return confirm('¿Estás seguro de que deseas eliminar este estudiante?');">Eliminar</a>

                    </tr>

                </c:forEach>

            </tbody>

        </table>

        <a href="formulario\_estudiante.jsp">Agregar Estudiante</a>

    </main>

    <footer>

        <p>&copy; 2024 Administración de Estudiantes. Todos los derechos reservados.</p>

    </footer>

</body>

</html>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<html>

<head>

    <title>Editar Estudiante</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <h2>Editar Estudiante</h2>

    <form action="StudentServlet" method="POST">

        <input type="hidden" name="id" value="${student.id}" /> <!-- Campo oculto para el ID del estudiante -->

        <label for="name">Nombre:</label>

        <input type="text" id="name" name="name" value="${student.name}" required /><br>

        <label for="age">Edad:</label>

        <!-- Verifica si age tiene un valor, si no, usa un valor por defecto como 0 -->

        <input type="number" id="age" name="age" value="${student.age != null ? student.age : 0}" required /><br>

        <input type="submit" value="Actualizar" />

    </form>

</body>

</html>

<%--

    Document   : agregar\_estudiante

    Created on : 05-09-2024, 11:57:53 p. m.

    Author     : Luis

--%>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Agregar Estudiante</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <header>

        <h1>Agregar Estudiante</h1>

    </header>

    <main>

        <form action="StudentServlet" method="POST">

            <label for="name">Nombre:</label>

            <input type="text" id="name" name="name" required><br>

            <label for="age">Edad:</label>

            <input type="number" id="age" name="age" required min="0"><br>

            <input type="submit" value="Agregar Estudiante">

        </form>

    </main>

    <footer>

        <p>&copy; 2024 Administración de Estudiantes. Todos los derechos reservados.</p>

    </footer>

</body>

</html>

<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.mycompany</groupId>

    <artifactId>SisGesTareas</artifactId>

    <version>1.0-SNAPSHOT</version>

    <packaging>war</packaging>

    <name>SisGesTareas-1.0-SNAPSHOT</name>

    <properties>

        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

        <jakartaee>10.0.0</jakartaee>

        <maven.compiler.source>11</maven.compiler.source>

        <maven.compiler.target>11</maven.compiler.target>

    </properties>

    <dependencies>

        <dependency>

            <groupId>jakarta.platform</groupId>

            <artifactId>jakarta.jakartaee-api</artifactId>

            <version>${jakartaee}</version>

            <scope>provided</scope>

        </dependency>

        <dependency>

            <groupId>com.mysql</groupId>

            <artifactId>mysql-connector-j</artifactId>

            <version>8.0.33</version>

        </dependency>

        <dependency>

            <groupId>jakarta.persistence</groupId>

            <artifactId>jakarta.persistence-api</artifactId>

            <version>3.2.0</version>

        </dependency>

        <dependency>

            <groupId>jakarta.servlet</groupId>

            <artifactId>jakarta.servlet-api</artifactId>

            <version>5.0.0</version>

            <scope>provided</scope>

        </dependency>

        <dependency>

            <groupId>jakarta.servlet.jsp.jstl</groupId>

            <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>

            <version>3.0.0</version>

        </dependency>

        <dependency>

            <groupId>org.apache.taglibs</groupId>

            <artifactId>taglibs-standard-impl</artifactId>

            <version>1.2.5</version>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

                <groupId>org.apache.maven.plugins</groupId>

                <artifactId>maven-compiler-plugin</artifactId>

                <version>3.13.0</version>

                <configuration>

                    <source>${maven.compiler.source}</source>

                    <target>${maven.compiler.target}</target>

                </configuration>

            </plugin>

            <plugin>

                <groupId>org.apache.maven.plugins</groupId>

                <artifactId>maven-war-plugin</artifactId>

                <version>3.4.0</version>

            </plugin>

        </plugins>

    </build>

    <repositories>

        <repository>

            <id>central</id>

            <url>https://repo.maven.apache.org/maven2</url>

        </repository>

    </repositories>

</project>

/\*

Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

Click nbfs://nbhost/SystemFileSystem/Templates/JSP\_Servlet/CascadeStyleSheet.css to edit this template

\*/

/\*

    Created on : 06-09-2024, 12:30:43 a. m.

    Author     : Luis

\*/

/\* Estilo general del cuerpo \*/

body {

    font-family: Arial, sans-serif;

    background-color: #f4f4f4;

    color: #333;

    margin: 0;

    padding: 0;

}

/\* Estilo del encabezado \*/

header {

    background-color: #007bff;

    color: white;

    padding: 20px;

    text-align: center;

}

header h1 {

    margin: 0;

    font-size: 2em;

}

/\* Estilo del contenido principal \*/

main {

    padding: 20px;

    text-align: center;

}

nav ul {

    list-style: none;

    padding: 0;

}

nav ul li {

    display: inline;

    margin: 0 15px;

}

nav ul li a {

    text-decoration: none;

    color: #007bff;

    font-size: 1.2em;

}

nav ul li a:hover {

    text-decoration: underline;

}

/\* Estilo del pie de página \*/

footer {

    background-color: #007bff;

    color: white;

    text-align: center;

    padding: 10px;

    position: absolute;

    bottom: 0;

    width: 100%;

}

footer p {

    margin: 0;

}

<?xml version="1.0" encoding="UTF-8"?>

<persistence version="2.1"

             xmlns="http://xmlns.jcp.org/xml/ns/persistence"

             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

             xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence

                                 http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_1.xsd">

    <!-- Define Persistence Unit -->

    <persistence-unit name="StudentManagementPU" transaction-type="RESOURCE\_LOCAL">

        <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>

        <class>com.mycompany.model.Student</class>

        <class>com.mycompany.model.Course</class>

        <properties>

            <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/universidad\_bd"/>

            <property name="javax.persistence.jdbc.user" value="root"/>

            <property name="javax.persistence.jdbc.password" value="LAfl2011"/>

            <property name="javax.persistence.jdbc.driver" value="com.mysql.cj.jdbc.Driver"/>

            <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect"/>

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

            <property name="hibernate.show\_sql" value="true"/>

            <property name="hibernate.format\_sql" value="true"/>

        </properties>

    </persistence-unit>

</persistence>