



PLAN DE TRABAJO DEL ESTUDIANTE

RDA 
RECURSO DIDÁCTICO PARA EL APRENDIZAJE

DATOS DEL ESTUDIANTE

Apellidos y Nombres:	Yataco Saravia Luis Alonso	ID:	1552780
Dirección Zonal/CFP:	Ica – Ayacucho		
Carrera:	Ingeniería de Software con Inteligencia Artificial	Semestre:	V
Curso/ Mód. Formativo:	Fullstack Developer Software		
Tema de Trabajo Final:	Entregable 1		

1. INFORMACIÓN

▪ Identifica la problemática del caso práctico propuesto.

El caso práctico propuesto es:

La problemática principal es la necesidad de diseñar e implementar un sistema de gestión integral (CRUD) para un catálogo educativo, el cual debe manejar correctamente la jerarquía de contenido (Categoría > Subcategoría > Curso) y asegurar la integridad de los datos (asociación de Cursos con Docentes por FK).

▪ Identifica propuesta de solución y evidencias.

La solución reside en la construcción de un modelo de datos, utilizando MySQL para crear las tablas necesarias (Categorías, Subcategorías, Cursos). La implementación de Claves Foráneas (FK) en este modelo hará vincular cada curso con un Docente único. A nivel de aplicación, se ha establecido archivos HTML, CSS y JavaScript organizados en carpetas, creando el esqueleto de la interfaz administrativa necesaria para ejecutar las operaciones CRUD.

2. PLANIFICACIÓN DEL TRABAJO

- Cronograma de actividades:**

Nº	ACTIVIDADES	CRONOGRAMA					
		20	10	2025			
	Entregable 01	20	10	2025			

- Lista de recursos necesarios:**

1. MÁQUINAS Y EQUIPOS		
	Descripción	Cantidad
Computadora		1

2. HERRAMIENTAS E INSTRUMENTOS		
	Descripción	Cantidad
XAMPP		1
Visual Studio Code		1
Postman		1
MySQL		1

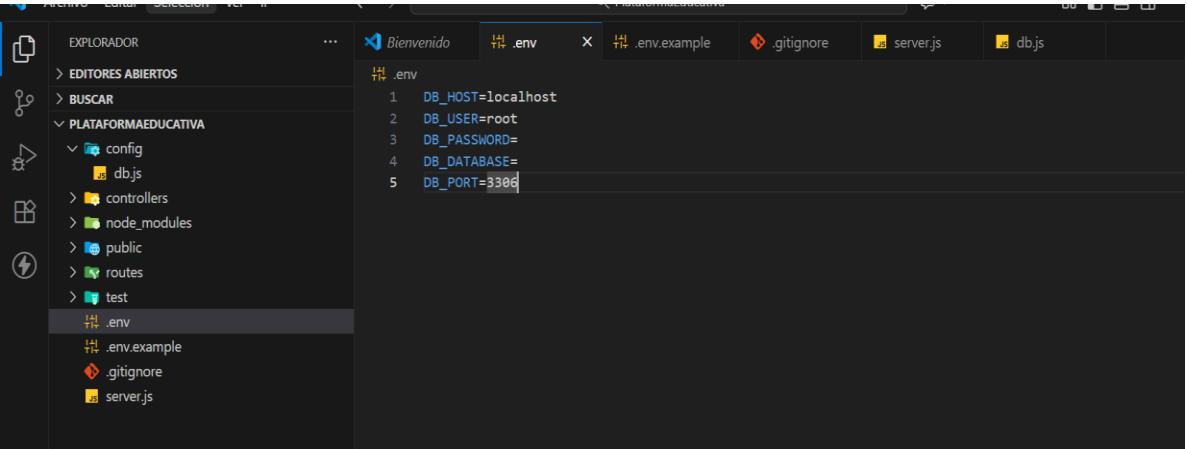
3. MATERIALES E INSUMOS		
	Descripción	Cantidad
USB		1
Escritorio		1

3. DECIDIR PROPUESTA

- Describe la propuesta determinada para la solución del caso práctico

PROPIUESTA DE SOLUCIÓN

Creación de la carpeta y archivos en Visual Studio Code



The screenshot shows the Visual Studio Code interface with the following details:

- Explorador (Left Panel):** Shows the project structure:
 - EDITORES ABIERTOS
 - BUSCAR
 - PLATAFORMAEDUCATIVA
 - config
 - db.js
 - controllers
 - node_modules
 - public
 - routes
 - test
 - .env
 - .env.example
 - .gitignore
 - server.js
- Editor (Right Panel):** Shows the contents of the .env file:

```
1 DB_HOST=localhost
2 DB_USER=root
3 DB_PASSWORD=
4 DB_DATABASE=
5 DB_PORT=3306
```

4. EJECUTAR

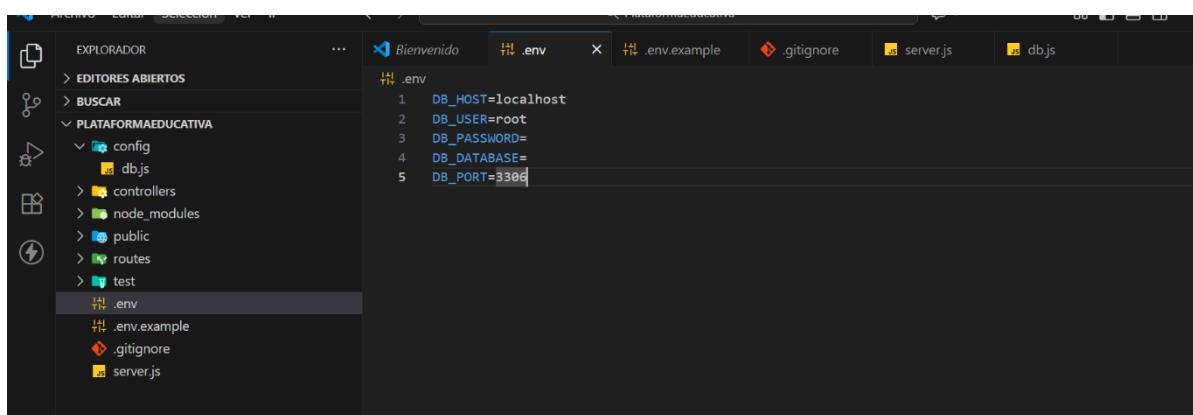
- **Resolver el caso práctico, utilizando como referencia el problema propuesto y las preguntas guía proporcionadas para orientar el desarrollo.**
 - **Fundamentar sus propuestas en los conocimientos adquiridos a lo largo del curso, aplicando lo aprendido en las tareas y operaciones descritas en los contenidos curriculares.**

INSTRUCCIONES: Ser lo más explícito posible. Los gráficos ayudan a transmitir mejor las ideas. Tomar en cuenta los aspectos de calidad, medio ambiente y SHI.

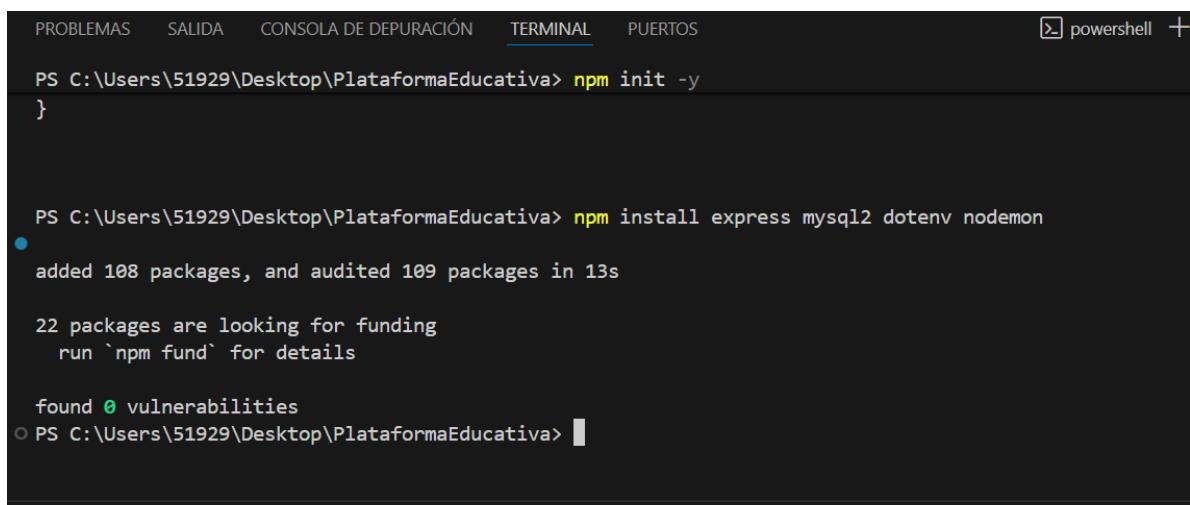
OPERACIONES / PASOS / SUBPASOS	NORMAS TÉCNICAS -
ESTÁNDARES / SEGURIDAD	/ MEDIO AMBIENTE
Crear la base de datos	
Activar XAMPP	
Abrir VisualStudio	
Desarrollo Frontend	
desarrollo Backend (CRUD)	

DIBUJO / ESQUEMA / DIAGRAMA DE PROPUESTA

(Adicionar las páginas que sean necesarias)



Instalación del json



```

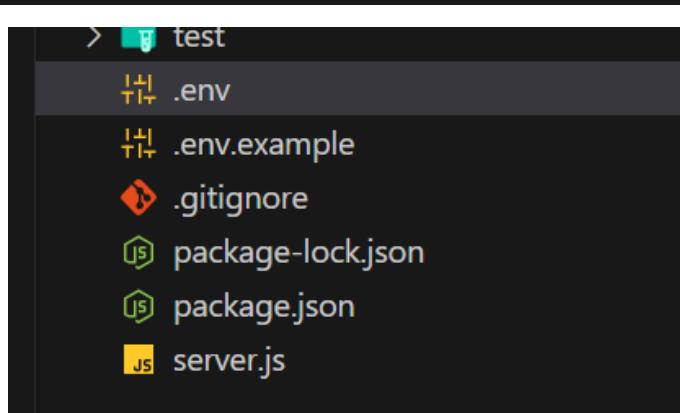
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
powershell + 

PS C:\Users\51929\Desktop\PlataformaEducativa> npm init -y
}

PS C:\Users\51929\Desktop\PlataformaEducativa> npm install express mysql2 dotenv nodemon
●
  added 108 packages, and audited 109 packages in 13s
  22 packages are looking for funding
    run `npm fund` for details
  found 0 vulnerabilities
○ PS C:\Users\51929\Desktop\PlataformaEducativa>

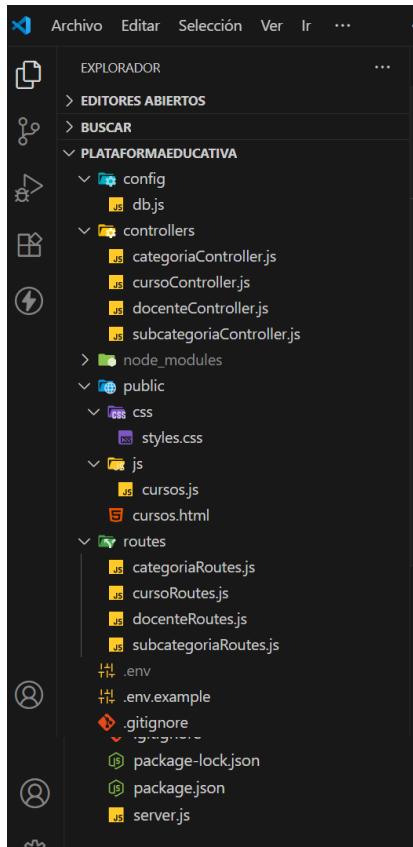
```

Lín. 5, col. 13 Espacios: 4 UTF-8 CRLF {} Propiedades





Creación de los archivos



Código Config/db.js

```
const mysql = require('mysql2');

const pool = mysql.createPool({
  connectionLimit: 10,
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'proyecto_cursos'
});

module.exports = pool.promise();
```

PS C:\Users\51929\Desktop\PlataformaEducativa> nodemon server
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Servidor iniciado en http://localhost:3000



SENAI

Trabajo Final

CategoriaController.js

```
const db = require('../config/db');

exports.obtenerCategorias = async (req, res) => {
  try {
    const [rows] = await db.query('SELECT id_categoria, nombre_categoria FROM categoria ORDER BY nombre_categoria');
    res.json(rows);
  } catch (error) {
    console.error("Error al obtener categorías:", error);
    res.status(500).json({ error: 'Error interno del servidor al obtener categorías' });
  }
};
```

cursoController.js

```
const db = require('../config/db');

exports.obtenerCursos = async (req, res) => {
  try {
    const query = `SELECT
      c.id_curso,
      c.titulo,
      c.fecha_inicio,
      c.fecha_fin,
      c.duracion_horas,
      c.precio,
      cat.nombre_categoria,
      scat.nombre_subcategoria,
      d.nombre_completo AS nombre_docente
    FROM
      cursos c
    JOIN
      subcategoria scat ON c.id_subcategoria = scat.id_subcategoria
    JOIN
      docente d ON scat.id_docente = d.id_docente`;
    res.json(query);
  } catch (error) {
    console.error("Error al obtener cursos:", error);
    res.status(500).json({ error: 'Error interno del servidor al obtener cursos' });
  }
};
```



Trabajo Final

Explorador:

- EDTORES ABIERTOS
- BUSCAR
- PLATAFORMAEDUCATIVA
 - config
 - db.js
 - controllers
 - categoriaController.js
 - cursoController.js
 - docenteController.js
 - subcategoriaController.js
 - node_modules
 - public
 - css
 - styles.css
 - js
 - cursos.js
 - cursos.html
 - routes
 - categoriaRoutes.js
 - cursoRoutes.js
 - docenteRoutes.js
 - subcategoriaRoutes.js
- .env
- .env.example
- .gitignore
- package-lock.json
- LÍNEA DE TIEMPO

Tabs: Release Notes, localhost:3000/api/cursos, categoriaController.js, cursoController.js, db.js

cursoController.js (Lineas 3-47):

```
3 exports.obtenerCursos = async (req, res) => {
5   const query = `
```

categoria cat ON scat.id_categoria = cat.id_categoria
JOIN
docente d ON c.id_docente = d.id_docente
ORDER BY c.id_curso DESC;

```
19
20   const [rows] = await db.query(query);
21   res.json(rows);
22 } catch (error) {
23   console.error('Error al obtener cursos:', error);
24   res.status(500).json({ error: 'Error interno del servidor' });
25 }
```

exports.obtenerCursoPorId = async (req, res) => {
35 const { id_curso } = req.params;
36 try {
37 const query = `

SELECT
c.*,
scat.id_categoria
FROM
cursos c
JOIN
subcategoria scat ON c.id_subcategoria = scat.id_subcategoria
WHERE c.id_curso = ?;

```
38
39   const [rows] = await db.query(query, [id_curso]);
40   if (rows.length === 0) return res.status(404).json({ message: 'Curso no encontrado' });
41   res.json(rows[0]);
42 } catch (error) {
43   res.status(500).json({ error: 'Error interno del servidor' });
44 }
```

// REGISTRAR (CREATE)

exports.registrarCurso = async (req, res) => {
58 const { titulo, descripcion, fecha_inicio, fecha_fin, duracion_horas, precio, id_subcategoria, id_categoria } = req.body;
59 try {
60 const [result] = await db.query(
61 'INSERT INTO cursos (titulo, descripcion, fecha_inicio, fecha_fin, duracion_horas, precio, id_subcategoria, id_categoria)'
62 , [titulo, descripcion, fecha_inicio, fecha_fin, duracion_horas, precio, id_subcategoria, id_categoria]
63);
64 res.status(201).json({ message: 'Curso registrado con éxito', id: result.insertId });
65 } catch (error) {
66 console.error('Error al registrar curso:', error);
67 res.status(500).json({ error: 'Error interno del servidor', details: error.message });
68 }
69 }

// ACTUALIZAR (UPDATE)

exports.actualizarCurso = async (req, res) => {
73 const { id_curso } = req.params;
74 const { titulo, descripcion, fecha_inicio, fecha_fin, duracion_horas, precio, id_subcategoria, id_categoria } = req.body;
75 try {

const [result] = await db.query(
76 'UPDATE cursos SET titulo = ?, descripcion = ?, fecha_inicio = ?, fecha_fin = ?, duracion_horas = ?, precio = ?, id_subcategoria = ?, id_categoria = ?',
77 [titulo, descripcion, fecha_inicio, fecha_fin, duracion_horas, precio, id_subcategoria, id_categoria]
78);
79 if (result.affectedRows === 0) return res.status(404).json({ message: 'Curso no encontrado' });
80 res.json({ message: 'Curso actualizado con éxito' });
81 } catch (error) {
82 console.error('Error al actualizar curso:', error);
83 res.status(500).json({ error: 'Error interno del servidor', details: error.message });
84 }

// ELIMINAR (DELETE)

exports.eliminarCurso = async (req, res) => {
89 const { id_curso } = req.params;
90 try {
91 const [result] = await db.query('DELETE FROM cursos WHERE id_curso = ?', [id_curso]);
92 if (result.affectedRows === 0) return res.status(404).json({ message: 'Curso no encontrado' });
93 res.json({ message: 'Curso eliminado con éxito' });
94 } catch (error) {
95 console.error('Error al eliminar curso:', error);
96 res.status(500).json({ error: 'Error interno del servidor', details: error.message });
97 }
98 }

};



docenteController.js

```
controllers > docenteController.js > ...
1 const db = require('../config/db');
2
3 exports.obtenerDocentes = async (req, res) => {
4   try {
5     const [rows] = await db.query('SELECT id_docente, nombre_completo FROM docente ORDER BY nombre_completo');
6     res.json(rows);
7   } catch (error) {
8     console.error("Error al obtener docentes:", error);
9     res.status(500).json({ error: 'Error interno del servidor al obtener docentes' });
10  }
11};
```

subcategoryController.js

```
controllers > subcategoriaController.js > ...
1 const db = require('../config/db');
2
3 exports.obtenerSubcategorias = async (req, res) => {
4   try {
5     const [rows] = await db.query('SELECT id_subcategoria, nombre_subcategoria, id_categoria FROM subcategoria');
6     res.json(rows);
7   } catch (error) {
8     console.error("Error al obtener subcategorias:", error);
9     res.status(500).json({ error: 'Error interno del servidor' });
10  }
11};
12
13 exports.obtenerSubcategoriasPorCategoria = async (req, res) => {
14   const { id_categoria } = req.params;
15   try {
16     const [rows] = await db.query('SELECT id_subcategoria, nombre_subcategoria FROM subcategoria WHERE id_categoria = ?;', [id_categoria]);
17     res.json(rows);
18   } catch (error) {
19     console.error("Error al obtener subcategorias por categoria:", error);
20     res.status(500).json({ error: 'Error interno del servidor' });
21   }
22};
```



css/styles.css

```
public > css > styles.css > btn-danger
1 .navbar-header-custom {
2     background-color: #28a745;
3     padding-top: 5px;
4     padding-bottom: 5px;
5 }
6 .navbar-brand {
7     font-weight: bold;
8     color: #343a40;
9 }
10 #tabla-cursos thead tr {
11     background-color: #343a40 !important;
12     color: white;
13 }
14 #tabla-cursos th.text-center,
15 #tabla-cursos td.text-center {
16     text-align: center !important;
17 }
18 #tabla-cursos th.text-end,
19 #tabla-cursos td.text-end {
20     text-align: right !important;
21 }
22 .btn-info {
23     --bs-btn-bg: #ede321;
24     --bs-btn-border-color: #ede321;
25 }
26 .btn-danger {
27     --bs-btn-bg: #dc3545;
28     --bs-btn-border-color: #dc3545;
29 }
30 
```

cursos.js

```
public > js > cursos.js > ...
1 const API_URL_CURSOS = "http://localhost:3000/api/cursos";
2 const API_URL_CATEGORIAS = "http://localhost:3000/api/categorias";
3 const API_URL_SUBCATEGORIAS = "http://localhost:3000/api/subcategorias";
4 const API_URL_DOCENTES = "http://localhost:3000/api/docentes";
5
6 const formularioCurso = document.getElementById('form-curso');
7 const idCursoInput = document.getElementById('id_curso');
8 const tablaCursosBody = document.querySelector('#tabla-cursos tbody');
9 const selectCategorias = document.getElementById('id_categoria');
10 const selectSubcategorias = document.getElementById('id_subcategoria');
11 const selectDocentes = document.getElementById('id_docente');
12 const btnGuardar = document.getElementById('btn-guardar');
13
14 function formatDateForInput(dateString) {
15     if (!dateString) return '';
16     const date = new Date(Date.parse(dateString));
17     const year = date.getFullYear();
18     const month = String(date.getUTCMonth() + 1).padStart(2, '0');
19     const day = String(date.getUTCDate()).padStart(2, '0');
20     return `${year}-${month}-${day}`;
21 }
22
23 function formatDisplayDate(dateString) {
24     if (!dateString) return '';
25     const date = new Date(Date.parse(dateString));
26     const day = String(date.getUTCDate()).padStart(2, '0');
27     const month = String(date.getUTCMonth() + 1).padStart(2, '0');
28     const year = date.getUTCFullYear();
29     return `${day}/${month}/${year}`;
30 }
```



Trabajo Final

The image displays three vertically stacked code editors, each showing a different file from a web application's source code. The application appears to be a platform for managing courses, categories, and subcategories.

- Editor 1:** Shows the `cursos.js` file. The code defines functions to format dates and to fetch and display categories and subcategories based on selected IDs. It uses `fetch` to get JSON data and `innerHTML` to update dropdown menus.
- Editor 2:** Shows the `cursos.js` file again, but this version includes code to fetch and display teachers (docentes). It follows a similar pattern of using `fetch` and updating `innerHTML` for dropdown menus.
- Editor 3:** Shows the `cursos.js` file once more, this time focusing on the logic to retrieve and display courses (cursos). It includes handling for course creation, reading course details, and updating course information. It uses `fetch` to get course data and `innerHTML` to build tables.

The left sidebar of each editor shows the project structure, including `config`, `controllers`, `routes`, `public`, and `node_modules` directories, along with `.env`, `.env.example`, `.gitignore`, and `package-lock.json`.



Trabajo Final

public > js > cursos.js > ...

```
95     async function obtenerCursos() {
183         cursos.forEach(curso => {
122             // Botón Editar
123             const editButton = document.createElement('button');
124             editButton.textContent = 'Editar';
125             editButton.classList.add('btn', 'btn-info', 'btn-sm', 'me-2');
126             editButton.title = 'Editar';
127             editButton.onclick = () => cargarParaEdicion(curso.id_curso);
128             actionsCell.appendChild(editButton);
129
130             // Botón Eliminar
131             const deleteButton = document.createElement('button');
132             deleteButton.textContent = 'Eliminar';
133             deleteButton.classList.add('btn', 'btn-danger', 'btn-sm');
134             deleteButton.title = 'Eliminar';
135             deleteButton.onclick = () => eliminarCurso(curso.id_curso, curso.titulo);
136             actionsCell.appendChild(deleteButton);
137
138         } catch (error) {
139             console.error("Error al obtener cursos:", error);
140             tablaCursosBody.innerHTML = `<tr><td colspan="10" class="text-center text-danger">Error al ca
141         }
142
143     // 2. CARGAR PARA EDICIÓN (READ ONE)
144     async function cargarParaEdicion(id_curso) {
145         window.scrollTo({ top: 0, behavior: 'smooth' });
146     }
```

public > js > cursos.js > ...

```
145     async function cargarParaEdicion(id_curso) {
147
148         try {
149             const response = await fetch(`${API_URL_CURSOS}/${id_curso}`);
150             if (!response.ok) throw new Error('Curso no encontrado.');
151             const curso = await response.json();
152
153             // Llenar campos del formulario
154             idCursoInput.value = curso.id_curso;
155             document.getElementById('titulo').value = curso.titulo;
156             document.getElementById('descripcion').value = curso.descripcion;
157             document.getElementById('fecha_inicio').value = formatDateForInput(curso.fecha_inicio);
158             document.getElementById('fecha_fin').value = formatDateForInput(curso.fecha_fin);
159             document.getElementById('duracion_horas').value = curso.duracion_horas;
160             document.getElementById('precio').value = parseFloat(curso.precio).toFixed(2);
161
162             // Cargar y seleccionar selects
163             await obtenerCategorias(curso.id_categoria);
164             await obtenerSubcategoriasPorCategoria(curso.id_categoria, curso.id_subcategoria);
165             await obtenerDocentes(curso.id_docente);
166
167             // Actualizar botón de Guardar/Actualizar
168             btnGuardar.innerHTML = 'Actualizar';
169             btnGuardar.classList.add('btn-warning');
170
171         } catch (error) {
172             console.error("Error al cargar curso para edición:", error);
173             alert('No se pudo cargar el curso para edición.');
174         }
175     }
```

public > js > cursos.js > ...

```
178     // 3. REGISTRAR/ACTUALIZAR (CREATE/UPDATE)
179     formularioCurso.addEventListener('submit', async (event) => {
180         event.preventDefault();
181
182         const id_curso = idCursoInput.value;
183         const method = id_curso ? 'PUT' : 'POST';
184         const url = id_curso ? `${API_URL_CURSOS}/${id_curso}` : API_URL_CURSOS;
185
186         const data = {
187             titulo: document.getElementById('titulo').value,
188             descripcion: document.getElementById('descripcion').value,
189             fecha_inicio: document.getElementById('fecha_inicio').value,
190             fecha_fin: document.getElementById('fecha_fin').value,
191             duracion_horas: parseInt(document.getElementById('duracion_horas').value),
192             precio: parseFloat(document.getElementById('precio').value),
193             id_subcategoria: parseInt(selectSubcategorias.value),
194             id_docente: parseInt(selectDocentes.value)
195         };
196
197         try {
198             let response = await fetch(url, {
199                 method: method,
200                 headers: { 'Content-Type': 'application/json' },
201                 body: JSON.stringify(data),
202             });
203
204             if (!response.ok) {
205                 const responseData = await response.json();
206                 throw new Error(`Error en la operación: ${response.status} - ${responseData.details || error
207             }
208         }
```



> EDITORES ABIERTOS
> BUSCAR
✓ PLATAFORMAEDUCATIVA
 > config
 db.js
 > controllers
 categoriaController.js
 cursoController.js
 docenteController.js
 subcategoriaController.js
 > node_modules
 > public
 > css
 styles.css
 > js
 cursos.js
 cursos.html
 > routes
 categoriaRoutes.js
 cursoRoutes.js
 docenteRoutes.js
 subcategoriaRoutes.js
.env
.env.example
.gitignore
package-lock.json
LÍNEA DE TIEMPO

```
public > js > cursos.js > ...
178  formularioCurso.addEventListener('submit', async (event) => {
179
180    const result = await response.json();
181    alert(result.message);
182    resetFormulario();
183    obtenerCursos();
184
185  } catch (error) {
186    console.error("Error al guardar curso:", error);
187    alert(`Error al guardar/actualizar el curso: ${error.message}`);
188  }
189});
190
191 // 4. ELIMINAR CURSO (DELETE)
192 async function eliminarCurso(id, titulo_curso) {
193
194   // Configuración para los botones de SweetAlert usando clases de Bootstrap
195   const swalWithBootstrapButtons = Swal.mixin({
196     customClass: {
197       confirmButton: "btn btn-danger mx-2", // Rojo para eliminar
198       cancelButton: "btn btn-secondary" // Gris para cancelar
199     },
200     buttonsStyling: false
201   });
202
203   // Muestra el modal de confirmación
204   swalWithBootstrapButtons.fire({
205     title: "¿Estás seguro?",
206     text: "Estás a punto de eliminar el curso: "${titulo_curso}". ¡Esta acción no se puede revertir!",
207     icon: "warning",
208     showCancelButton: true,
209   }).then(async (result) => {
210
211     if (result.isConfirmed) {
212       try {
213         const response = await fetch(`${API_URL_CURSOS}/${id}`, {
214           method: 'DELETE',
215         });
216
217         if (!response.ok) {
218           throw new Error('Error al eliminar el curso');
219         }
220         await response.json();
221         swalWithBootstrapButtons.fire({
222           title: "¡Eliminado!",
223           text: "El curso "${titulo_curso}" ha sido eliminado.",
224           icon: "success"
225         });
226
227         obtenerCursos();
228         resetFormulario();
229
230       } catch (error) {
231         console.error("Error al eliminar curso:", error);
232       }
233     }
234   });
235}
```

> EDITORES ABIERTOS
> BUSCAR
✓ PLATAFORMAEDUCATIVA
 > config
 db.js
 > controllers
 categoriaController.js
 cursoController.js
 docenteController.js
 subcategoriaController.js
 > node_modules
 > public
 > css
 styles.css
 > js
 cursos.js
 cursos.html
 > routes
 categoriaRoutes.js
 cursoRoutes.js
 docenteRoutes.js
 subcategoriaRoutes.js
.env
.env.example
.gitignore

```
public > js > cursos.js > ...
219  async function eliminarCurso(id, titulo_curso) {
220
221   confirmButtonText: "Sí, ¡Eliminar!",  
   cancelButtonText: "No, cancelar",  
   reverseButtons: true
222 }) .then(async (result) => {
223
224   if (result.isConfirmed) {
225     try {
226       const response = await fetch(`${API_URL_CURSOS}/${id}`, {  
         method: 'DELETE',  
       });
227
228       if (!response.ok) {
229         throw new Error('Error al eliminar el curso');
230       }
231       await response.json();
232       swalWithBootstrapButtons.fire({  
         title: "¡Eliminado!",  
         text: "El curso "${titulo_curso}" ha sido eliminado.",  
         icon: "success"
233       });
234
235       obtenerCursos();
236       resetFormulario();
237
238     } catch (error) {
239       console.error("Error al eliminar curso:", error);
240     }
241   }
242 }
```

> EDITORES ABIERTOS
> BUSCAR
✓ PLATAFORMAEDUCATIVA
 > config
 db.js
 > controllers
 categoriaController.js
 cursoController.js
 docenteController.js
 subcategoriaController.js
 > node_modules
 > public
 > css
 styles.css
 > js
 cursos.js
 cursos.html
 > routes
 categoriaRoutes.js
 cursoRoutes.js
 docenteRoutes.js
 subcategoriaRoutes.js
.env
.env.example
.gitignore

```
public > js > cursos.js > resetFormulario
219  async function eliminarCurso(id, titulo_curso) {
220
221   swalWithBootstrapButtons.fire({
222     title: "Error",
223     text: `No se pudo eliminar el curso. Verifique si hay dependencias.`,
224     icon: "error"
225   });
226
227   } else if (result.dismiss === Swal.DismissReason.cancel) {
228     swalWithBootstrapButtons.fire({
229       title: "Cancelación",
230       text: "El curso está a salvo :)",
231       icon: "info"
232     });
233   }
234 });
235
236 function resetFormulario() {
237   formularioCurso.reset();
238   idCursoInput.value = '';
239
240   btnGuardar.innerHTML = 'Guardar';
241   btnGuardar.classList.remove('btn-warning');
242   btnGuardar.classList.add('btn-primary');
243 }
```



```
public > js > cursos.js > resetFormulario
280 function resetFormulario() {
281     btnGuardar.classList.add('btn-primary');
282
283     obtenerCategorias();
284     obtenerSubcategoriasPorCategoria(null);
285     obtenerDocentes();
286
287     document.getElementById('btn-cancelar').addEventListener('click', (e) => {
288         e.preventDefault();
289         resetFormulario();
290     });
291
292     // Inicializar la aplicación
293     document.addEventListener('DOMContentLoaded', () => {
294         resetFormulario();
295         obtenerCursos();
296     });
297
298     // Inicializar la aplicación
299     document.addEventListener('DOMContentLoaded', () => {
300         resetFormulario();
301         obtenerCursos();
302     });
303
304 }
```

cursos.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Proyecto - Cursos</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.2/css/all.min.css" rel="stylesheet">
    <link rel="stylesheet" href="/css/styles.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
</head>
<body>
    <header class="navbar-header-custom"></header>

    <nav class="navbar bar-light bg-light border-bottom">
        <div class="container-fluid">
            <span class="navbar-brand mb-0 h1 ps-3">Gestión de Cursos</span>
        </div>
    </nav>

    <main class="container-fluid mt-3">
        <div class="card mb-4 shadow-sm">
            <div class="card-header bg-white border-0">
            </div>
            <div class="card-body">
                <form id="form-curso" autocomplete="off" method="POST">
                    <input type="hidden" name="id_curso" id="id_curso" value="" />
                </form>
            </div>
        </div>
    </main>
</body>

```



EDTORES ABIERTOS

BUSCAR

PLATAFORMAEDUCATIVA

- config
 - db.js
- controllers
 - categoriaController.js
 - cursoController.js
 - docenteController.js
 - subcategoriaController.js
- node_modules
- public
 - css
 - styles.css
 - js
 - cursos.js
 - cursos.html
- routes
 - categoriaRoutes.js
 - cursoRoutes.js
 - docenteRoutes.js
 - subcategoriaRoutes.js
- .env
- .env.example
- .gitignore
- packagelock.json

LÍNEA DE TIEMPO

```
public > cursos.html > ...
2  <html lang="es">
12 <body>
21   <main class="container-fluid mt-3">
22     <div class="card mb-4 shadow-sm">
25       <div class="card-body">
29         <div class="row g-3 mb-3">
30           <div class="col-md-6">
31             <label for="id_categoria" class="form-label">Categorías</label>
32             <select id="id_categoria" name="id_categoria" class="form-select" required>
33               <option value="">Seleccione</option>
34             </select>
35           </div>
36           <div class="col-md-6">
37             <label for="id_subcategoria" class="form-label">SubCategorías</label>
38             <select id="id_subcategoria" name="id_subcategoria" class="form-select" required>
39               <option value="">Seleccione</option>
40             </select>
41           </div>
42         </div>
43       <div class="row g-3 mb-3">
44         <div class="col-md-12">
45           <label for="titulo" class="form-label">Título del Curso</label>
46           <input type="text" class="form-control" id="titulo" name="titulo" required>
47         </div>
48       </div>
49     <div class="row g-3 mb-3">
50       <div class="col-md-12">
```

EDTORES ABIERTOS

BUSCAR

PLATAFORMAEDUCATIVA

- config
 - db.js
- controllers
 - categoriaController.js
 - cursoController.js
 - docenteController.js
 - subcategoriaController.js
- node_modules
- public
 - css
 - styles.css
 - js
 - cursos.js
 - cursos.html
- routes
 - categoriaRoutes.js
 - cursoRoutes.js
 - docenteRoutes.js
 - subcategoriaRoutes.js
- .env
- .env.example
- .gitignore
- packagelock.json

LÍNEA DE TIEMPO

```
public > cursos.html > ...
2  <html lang="es">
12 <body>
21   <main class="container-fluid mt-3">
22     <div class="card mb-4 shadow-sm">
25       <div class="card-body">
53         <label for="descripcion" class="form-label">Descripción Básica del Curso</label>
54         <textarea class="form-control" id="descripcion" name="descripcion" rows="3" required></textarea>
55       </div>
56     <div class="row g-3 mb-3">
57       <div class="col-md-3">
58         <label for="fecha_inicio" class="form-label">Fecha de Inicio del Curso</label>
59         <input type="date" class="form-control" id="fecha_inicio" name="fecha_inicio" required>
60       </div>
61       <div class="col-md-3">
62         <label for="fecha_fin" class="form-label">Fecha del Fin del Curso</label>
63         <input type="date" class="form-control" id="fecha_fin" name="fecha_fin" required>
64       </div>
65       <div class="col-md-6">
66         <label for="id_docente" class="form-label">Docentes</label>
67         <select id="id_docente" name="id_docente" class="form-select" required>
68           <option value="">Seleccione</option>
69         </select>
70       </div>
71     </div>
72     <div class="row g-3 mb-3">
73       <div class="col-md-6">
74         <label for="precio" class="form-label">Precio del Curso</label>
```

EDTORES ABIERTOS

BUSCAR

PLATAFORMAEDUCATIVA

- config
 - db.js
- controllers
 - categoriaController.js
 - cursoController.js
 - docenteController.js
 - subcategoriaController.js
- node_modules
- public
 - css
 - styles.css
 - js
 - cursos.js
 - cursos.html
- routes
 - categoriaRoutes.js
 - cursoRoutes.js
 - docenteRoutes.js
 - subcategoriaRoutes.js
- .env
- .env.example
- .gitignore
- packagelock.json

LÍNEA DE TIEMPO

```
public > cursos.html > ...
2  <html lang="es">
12 <body>
21   <main class="container-fluid mt-3">
22     <div class="card mb-4 shadow-sm">
25       <div class="card-body">
28         <input type="number" class="form-control" id="precio" name="precio" step="0.01" required>
29       </div>
30       <div class="col-md-6">
31         <label for="duracion_horas" class="form-label">Duración del Curso en Horas</label>
32         <input type="number" class="form-control" id="duracion_horas" name="duracion_horas" required>
33       </div>
34     </div>
35   <div class="card-footer text-end mt-4 bg-white border-0">
36     <button type="reset" class="btn btn-secondary me-2" id="btn-cancelar">Cancelar</button>
37     <button type="submit" class="btn btn-primary" id="btn-guardar">Guardar</button>
38   </div>
39 </div>
40 </div>
41 </div>
42 </div>
43 </div>
44 <div class="card shadow-sm">
45   <div class="card-body">
46     <div class="table-responsive">
47       <table class="table table-bordered table-striped align-middle" id="tabla-cursos">
48         <thead>
49           <tr>
50             <th class="text-center">ID</th>
51             <th>Título del Curso</th>
```



```
public > cursos.html > ...
  2   <html lang="es">
  3     <body>
  4       <main class="container-fluid mt-3">
  5         <div class="card shadow-sm">
  6           <div class="card-body">
  7             <table border="1">
  8               <thead>
  9                 <tr>
 10                   <th>Categoría</th>
 11                   <th>Subcategoria</th>
 12                   <th>Docente</th>
 13                   <th class="text-center">Inicio</th>
 14                   <th class="text-center">Fin</th>
 15                   <th class="text-center">Dur. (h)</th>
 16                   <th class="text-end">Precio</th>
 17                   <th class="text-center">Acciones</th>
 18               </tr>
 19             </thead>
 20             <tbody>
 21               <tr>
 22                 <td>Categoría A</td>
 23                 <td>Subcategoria A1</td>
 24                 <td>Docente A</td>
 25                 <td>2023-01-01</td>
 26                 <td>2023-01-02</td>
 27                 <td>10:00</td>
 28                 <td>100.00</td>
 29                 <td><a href="#">Ver</a> <a href="#">Actualizar</a> <a href="#">Eliminar</a></td>
 30               </tr>
 31             </tbody>
 32           </table>
 33         </div>
 34       </div>
 35     </main>
 36   <footer class="bg-dark text-white text-center py-3 mt-5">
 37     &copy; Gestión de Cursos.
 38   </footer>
 39
 40   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>
 41   <script src="./js/cursos.js"></script>
 42
 43 </body>
```

Routes

categoriaRoutes.js

```
routes > categoriaRoutes.js > ...
  1
  2   const express = require('express');
  3   const router = express.Router();
  4   const categoriaController = require('../controllers/categoriaController');
  5
  6   router.get('/', categoriaController.obtenerCategorias);
  7
  8   module.exports = router;
```

cursoRoutes.js

```
routes > cursoRoutes.js > ...
  1
  2   const express = require('express');
  3   const router = express.Router();
  4   const cursoController = require('../controllers/cursoController');
  5
  6   router.post('/', cursoController.registrarCurso);
  7   router.get('/', cursoController.obtenerCursos);
  8   router.get('/:id_curso', cursoController.obtenerCursoPorId);
  9   router.put('/:id_curso', cursoController.actualizarCurso);
 10   router.delete('/:id_curso', cursoController.eliminarCurso);
 11
 12   module.exports = router;
```



SENATI

Trabajo Final

docenteRoutes.js

```
routes > docenteRoutes.js > ...
1
2 const express = require('express');
3 const router = express.Router();
4 const docenteController = require('../controllers/docenteController');
5
6 router.get('/', docenteController.obtenerDocentes);
7
8 module.exports = router;
```

subcategoryRoutes.js

```
routes > subcategoriaRoutes.js > ...
1
2 const express = require('express');
3 const router = express.Router();
4 const subcategoriaController = require('../controllers/subcategoriaController');
5
6 router.get('/', subcategoriaController.obtenerSubcategorias);
7 router.get('/porCategoria/:id_categoria', subcategoriaController.obtenerSubcategoriasPorCategoria);
8
9 module.exports = router;
```

Conexión a la base de datos

```
.env
1 DB_HOST=localhost
2 DB_USER=root
3 DB_PASSWORD=
4 DB_DATABASE=projeto_cursos
5 DB_PORT=3306
```



.gitignore

```
gitignore
1 .env
2 node_modules
```

server.js

```
server.js > ...
1 // server.js
2 const express = require('express');
3 const cors = require('cors');
4 const path = require('path');
5
6 // Importar todas las rutas
7 const categoriaRoutes = require('../routes/categoriaRoutes');
8 const subcategoriaRoutes = require('../routes/subcategoriaRoutes');
9 const docenteRoutes = require('../routes/docenteRoutes');
10 const cursoRoutes = require('../routes/cursoRoutes');
11
12 const app = express();
13 const PORT = process.env.PORT || 3000;
14
15 // Configuración de CORS
16 app.use(cors({
17   origin: '*',
18   methods: ['GET', 'HEAD', 'PUT', 'PATCH', 'POST', 'DELETE'],
19   credentials: true
20 }));
21
22
23 app.use(express.json());
24 app.use(express.urlencoded({ extended: true }));
25
26 // Servir archivos estáticos (HTML/CSS/JS)
27 app.use(express.static(path.join(__dirname, 'public')));
28
29
30 // Rutas de la API (endpoint /api...)
31 app.use('/api/categorias', categoriaRoutes);
32 app.use('/api/subcategorias', subcategoriaRoutes);
33 app.use('/api/docentes', docenteRoutes);
34 app.use('/api/cursos', cursoRoutes);
35
36 // Ruta principal para servir cursos.html
37 app.get('/', (req, res) => {
38   res.sendFile(path.join(__dirname, 'public', 'cursos.html'));
39 }
40
41 app.listen(PORT, () => {
42   console.log(`Servidor iniciado en http://localhost:${PORT}`);
43 })
```



Thunder Client Método Get

The screenshot shows the Thunder Client interface. On the left, the activity log lists several recent requests, with the top one selected: `GET localhost:3000/api/cursos` (1 day ago). The main panel displays a GET request to `http://localhost:3000/api/cursos`. The response status is `200 OK`, size `771 Bytes`, and time `255 ms`. The response body is a JSON array containing two course objects:

```
[{"id_curso": 3, "titulo": "Matemáticas", "fecha_inicio": "2025-10-03T05:00:00.000Z", "fecha_fin": "2025-10-27T05:00:00.000Z", "duracion_horas": 5, "precio": "200.00", "nombre_categoria": "Informática", "nombre_subcategoria": "Lenguajes P.", "nombre_docente": "María López"}, {"id_curso": 2, "titulo": "SQL Avanzado", "fecha_inicio": "2023-10-15T05:00:00.000Z", "fecha_fin": "2023-12-01T05:00:00.000Z", "duracion_horas": 50, "precio": "149.99", "nombre_categoria": "Informática", "nombre_subcategoria": "BD", "nombre_docente": "Juan Pérez"}]
```

Buscar por id

The screenshot shows the Thunder Client interface. On the left, the activity log lists several recent requests, with the top one selected: `GET localhost:3000/api/cursos` (just now). The main panel displays a GET request to `http://localhost:3000/api/cursos/1`. The response status is `200 OK`, size `271 Bytes`, and time `400 ms`. The response body is a JSON object representing a single course:

```
{ "id_curso": 1, "titulo": "Python Básico", "descripcion": "Fundamentos de Python y estructuras de datos", "fecha_inicio": "2023-11-01T05:00:00.000Z", "fecha_fin": "2023-11-30T05:00:00.000Z", "duracion_horas": 40, "precio": "59.99", "id_subcategoria": 1, "id_docente": 1, "id_categoria": 1 }
```



Método PUT

The screenshot shows the Thunder Client interface. On the left, the activity log lists several GET requests. In the center, a new request is being made to `localhost:3000/api/cursos/1` using the PUT method. The JSON body of the request is identical to the one shown in the first screenshot. On the right, the response status is `200 OK`, size is `271 Bytes`, and time is `7 ms`. The response body contains the updated course data.

```
1 {
2   "id_curso": 1,
3   "titulo": "Python Básico",
4   "descripcion": "Fundamentos de Python y estructuras de datos",
5   "fecha_inicio": "2023-11-01T05:00:00.000Z",
6   "fecha_fin": "2023-11-30T05:00:00.000Z",
7   "duracion_horas": 40,
8   "precio": "59.99",
9   "id_subcategoria": 1,
10  "id_docente": 1,
11  "id_categoria": 1
12 }
```

Actualizado correctamente

The screenshot shows the Thunder Client interface. The activity log includes a PUT request to `localhost:3000/api/cursos`. In the center, a new request is being made to `localhost:3000/api/cursos/1` using the PUT method. The JSON body is identical to the previous screenshots. On the right, the response status is `200 OK`, size is `42 Bytes`, and time is `94 ms`. The response body contains a success message.

```
1 {
2   "message": "Curso actualizado con éxito"
3 }
```



Método POST, Creado correctamente

The screenshot shows the Thunder Client interface. In the left sidebar, there is a list of activity logs. In the main area, a POST request is being made to `http://localhost:3000/api/cursos`. The Body tab contains a JSON object representing a course. The Response tab shows a successful `201 Created` status with a message "Curso registrado con éxito" and an ID of `5`.

```
1 {
2   "titulo": "Desarrollo Web con Node.js",
3   "descripcion": "Curso intensivo sobre creación de aplicaciones web con Express y MySQL",
4   "fecha_inicio": "2023-12-05T05:00:00.000Z",
5   "fecha_fin": "2024-01-10T05:00:00.000Z",
6   "duracion_horas": 45,
7   "precio": "99.50",
8   "id_subcategoria": 2,
9   "id_docente": 3,
10  "id_categoria": 4
11 }
12 }
```

Método DELETE

The screenshot shows the Thunder Client interface. In the left sidebar, there is a list of activity logs. In the main area, a DELETE request is being made to `http://localhost:3000/api/cursos/5`. The Body tab contains the same JSON object as the previous POST request. The Response tab shows a successful `200 OK` status with a message "Curso eliminado con éxito".

```
1 {
2   "id_curso": 5,
3   "titulo": "Desarrollo Web con Node.js",
4   "descripcion": "Curso intensivo sobre creación de aplicaciones web con Express y MySQL",
5   "fecha_inicio": "2023-12-05T05:00:00.000Z",
6   "fecha_fin": "2024-01-10T05:00:00.000Z",
7   "duracion_horas": 45,
8   "precio": "99.50",
9   "id_subcategoria": 2,
10  "id_docente": 3,
11  "id_categoria": 1
12 }
```

Curso eliminado correctamente

The screenshot shows the Thunder Client interface. In the left sidebar, there is a list of activity logs. In the main area, a DELETE request is being made to `http://localhost:3000/api/cursos/5`. The Body tab contains the same JSON object as the previous requests. The Response tab shows a successful `200 OK` status with a message "Curso eliminado con éxito".

```
1 {
2   "message": "Curso eliminado con éxito"
3 }
```

```
1 •  CREATE DATABASE proyecto_cursos;
2 •  USE proyecto_cursos;
3
4 •  CREATE TABLE categoria (
5      id_categoria INT PRIMARY KEY AUTO_INCREMENT,
6      nombre_categoria VARCHAR(255) NOT NULL UNIQUE
7  );
8
9 •  CREATE TABLE subcategoria (
10     id_subcategoria INT PRIMARY KEY AUTO_INCREMENT,
11     nombre_subcategoria VARCHAR(255) NOT NULL UNIQUE,
12     id_categoria INT NOT NULL,
13     FOREIGN KEY (id_categoria) REFERENCES categoria(id_categoria)
14         ON DELETE RESTRICT
15         ON UPDATE CASCADE
16  );
17
18 •  CREATE TABLE docente (
19     id_docente INT PRIMARY KEY AUTO_INCREMENT,
20     nombre_completo VARCHAR(255) NOT NULL,
21     email VARCHAR(100) UNIQUE NOT NULL
22  );
23
24 •  CREATE TABLE cursos (
25     id_curso INT PRIMARY KEY AUTO_INCREMENT,
26     titulo VARCHAR(255) NOT NULL,
27     descripcion TEXT,
28     fecha_inicio DATE NOT NULL,
29     fecha_fin DATE NOT NULL,
30     duracion_horas INT NOT NULL,
31     precio DECIMAL(10, 2) NOT NULL,
32     id_subcategoria INT NOT NULL,
33     id_docente INT NOT NULL,
34     FOREIGN KEY (id_subcategoria) REFERENCES subcategoria(id_subcategoria)
35         ON DELETE RESTRICT
36         ON UPDATE CASCADE,
37     FOREIGN KEY (id_docente) REFERENCES docente(id_docente)
38         ON DELETE RESTRICT
39         ON UPDATE CASCADE
40  );
```



```
43 •     INSERT INTO categoria (nombre_categoria) VALUES ('Matemáticas');
44 •     INSERT INTO categoria (nombre_categoria) VALUES ('Idiomas');
45
46 •     INSERT INTO subcategoria (nombre_subcategoria, id_categoria) VALUES ('Lenguajes P.', 1);
47 •     INSERT INTO subcategoria (nombre_subcategoria, id_categoria) VALUES ('BD', 1);
48 •     INSERT INTO subcategoria (nombre_subcategoria, id_categoria) VALUES ('Geometría', 2);
49 •     INSERT INTO subcategoria (nombre_subcategoria, id_categoria) VALUES ('Álgebra', 2);
50
51 •     INSERT INTO docente (nombre_completo, email) VALUES ('Andrés Rojas', 'andres.rojas@academia.com');
52 •     INSERT INTO docente (nombre_completo, email) VALUES ('Camila Herrera', 'camila.herrera@academia.com');
53 •     INSERT INTO docente (nombre_completo, email) VALUES ('Ricardo Torres', 'ricardo.torres@academia.com');
54 •     INSERT INTO docente (nombre_completo, email) VALUES ('Daniela Flores', 'daniela.flores@academia.com');
55 •     INSERT INTO docente (nombre_completo, email) VALUES ('Santiago Vargas', 'santiago.vargas@academia.com');
56 •     INSERT INTO docente (nombre_completo, email) VALUES ('Natalia Cruz', 'natalia.cruz@academia.com');

55 •     INSERT INTO cursos (titulo, descripcion, fecha_inicio, fecha_fin, duracion_horas, precio, id_subcategoria, id_docente) VALUES
56     ('Python Básico', 'Fundamentos de Python y estructuras de datos', '2023-11-01', '2023-11-30', 40, 59.99, 1, 1);
57 •     INSERT INTO cursos (titulo, descripcion, fecha_inicio, fecha_fin, duracion_horas, precio, id_subcategoria, id_docente) VALUES
58     ('SQL Avanzado', 'Optimización de consultas y gestión de BBDD relacionales', '2023-10-15', '2023-12-01', 50, 149.99, 2, 2);
59
60 •     SELECT
61         c.id_curso,
62         c.titulo,
63         c.fecha_inicio,
64         c.fecha_fin,
65         c.precio,
66         scat.nombre_subcategoria,
67         cat.nombre_categoria,
68
68         d.nombre_completo AS Docente
69     FROM
70         cursos c
71     JOIN
72         subcategoria scat ON c.id_subcategoria = scat.id_subcategoria
73     JOIN
74         categoria cat ON scat.id_categoria = cat.id_categoria
75     JOIN
76         docente d ON c.id_docente = d.id_docente;
77
78 •     SELECT * FROM categoria;
79 •     SELECT * FROM docente;
```



SELECT * FROM docentes;

79 • **SELECT * FROM docente;**

Result Grid			
	id_docente	nombre_completo	email
▶	1	Ana Torres	ana.torres@academia.com
	2	Juan Pérez	juan.perez@academia.com
	3	María López	maria.lopez@academia.com
*	NULL	NULL	NULL

SELECT * FROM categoría;

78 • **SELECT * FROM categoria;**

79 • **SELECT * FROM docente;**

Result Grid		
	id_categoria	nombre_categoria
▶	3	Idiomas
	1	Informática
	2	Matemáticas
*	NULL	NULL

```
59
60 • SELECT
61     c.id_curso,
62     c.titulo,
63     c.fecha_inicio,
64     c.fecha_fin,
65     c.precio,
66     scat.nombre_subcategoria,
67     cat.nombre_categoria,
68     d.nombre_completo AS Docente
69 FROM
70     cursos c
71 JOIN
72     subcategoria scat ON c.id_subcategoria = scat.id_subcategoria
```

Result Grid								
	id_curso	titulo	fecha_inicio	fecha_fin	precio	nombre_subcategoria	nombre_categoria	Docente
▶	1	Python Avanzado	2023-11-01	2023-11-30	79.99	Lenguajes P.	Informática	Ana Torres
	2	SQL Avanzado	2023-10-15	2023-12-01	149.99	BD	Informática	Juan Pérez
	3	Matemáticas	2025-10-03	2025-10-27	200.00	Lenguajes P.	Informática	María López



SENAI

Trabajo Final

Gestión de Cursos

Categorías SubCategorías

Título del Curso

Descripción Básica del Curso

Fecha de Inicio del Curso dd/mm/aaaa Fecha del Fin del Curso dd/mm/aaaa Docentes Seleccionar

Precio del Curso Duración del Curso en Horas

Cancelar Guardar

Fecha de Inicio del Curso dd/mm/aaaa Fecha del Fin del Curso dd/mm/aaaa Docentes Seleccionar

Precio del Curso Duración del Curso en Horas

Cancelar Guardar

ID	Título del Curso	Categoría	Subcategoría	Docente	Inicio	Fin	Dur. (h)	Precio	Acciones
2	SQL Avanzado	Informática	BD	Camila Herrera	15/10/2023	01/12/2023	2	150.00	Editar Eliminar
1	Python Básico	Informática	Lenguajes P.	Andrés Rojas	01/11/2023	30/11/2023	3	250.00	Editar Eliminar

Eliminar

Fecha de Inicio del Curso dd/mm/aaaa Fecha del Fin del Curso dd/mm/aaaa Docentes Seleccionar

Precio del Curso

Cancelar Guardar

!

¿Estás seguro?

Estás a punto de eliminar el curso: "SQL Avanzado". ¡Esta acción no se puede revertir!

No, cancelar Sí, ¡Eliminar!

ID	Título del Curso	Categoría	Subcategoría	Docente	Inicio	Fin	Dur. (h)	Precio	Acciones
2	SQL Avanzado	Informática	BD	Camila Herrera	15/10/2023	01/12/2023	2	150.00	Editar Eliminar
1	Python Básico	Informática	Lenguajes P.	Andrés Rojas	01/11/2023	30/11/2023	3	250.00	Editar Eliminar

Guardar Registro

A3

Fecha de Inicio del Curso

07/10/2025

Fecha del Fin del Curso

07/11/2025

Docentes

Ricardo Torres

Precio del Curso

300

Duración del Curso en Horas

3

[Cancelar](#) [Guardar](#)

ID	Título del Curso	Categoría	Subcategoría	Docente	Inicio	Fin	Dur. (h)	Precio	Acciones
2	SQL Avanzado	Informática	BD	Camila Herrera	15/10/2023	01/12/2023	2	150.00	Editar Eliminar
1	Python Básico	Informática	Lenguajes P.	Andrés Rojas	01/11/2023	30/11/2023	3	250.00	Editar Eliminar

Guardado exitosamente

[Cancelar](#) [Guardar](#)

ID	Título del Curso	Categoría	Subcategoría	Docente	Inicio	Fin	Dur. (h)	Precio	Acciones
3	Números	Matemáticas	Geometría	Ricardo Torres	07/10/2025	07/11/2025	3	300.00	Editar Eliminar
2	SQL Avanzado	Informática	BD	Camila Herrera	15/10/2023	01/12/2023	2	150.00	Editar Eliminar
1	Python Básico	Informática	Lenguajes P.	Andrés Rojas	01/11/2023	30/11/2023	3	250.00	Editar Eliminar



[ENTRTEGABLE_01]

YATACO SARAVIA LUIS ALONSO

[ESCALA]

5. CONTROLAR

- Verificar el cumplimiento de los procesos desarrollados en la propuesta de solución del caso práctico.

EVIDENCIAS	CUMPLE	NO CUMPLE
• ¿Se identificó claramente la problemática del caso práctico?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
• ¿Se desarrolló las condiciones de los requerimientos solicitados?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
• ¿Se formularon respuestas claras y fundamentadas a todas las preguntas guía?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
• ¿Se elaboró un cronograma claro de actividades a ejecutar?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
• ¿Se identificaron y listaron los recursos (máquinas, equipos, herramientas, materiales) necesarios para ejecutar la propuesta?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
• ¿Se ejecutó la propuesta de acuerdo con la planificación y cronograma establecidos?	<input type="checkbox"/>	<input type="checkbox"/>
• ¿Se describieron todas las operaciones y pasos seguidos para garantizar la correcta ejecución?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
• ¿Se consideran las normativas técnicas, de seguridad y medio ambiente en la propuesta de solución?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
• ¿La propuesta es pertinente con los requerimientos solicitados?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
• ¿Se evaluó la viabilidad de la propuesta para un contexto real?	<input checked="" type="checkbox"/>	<input type="checkbox"/>



6. VALORAR

- Califica el impacto que representa la propuesta de solución ante la situación planteada en el caso práctico.

CRITERIO DE EVALUACIÓN	DESCRIPCIÓN DEL CRITERIO	PUNTUACIÓN MÁXIMA	PUNTAJE CALIFICADO POR EL ESTUDIANTE
Identificación del problema	Claridad en la identificación del problema planteado.	3	
Relevancia de la propuesta de solución	La propuesta responde adecuadamente al problema planteado y es relevante para el contexto del caso práctico.	8	
Viabilidad técnica	La solución es técnicamente factible, tomando en cuenta los recursos y conocimientos disponibles.	6	
Cumplimiento de Normas	La solución cumple con todas las normas técnicas de seguridad, higiene y medio ambiente.	3	
PUNTAJE TOTAL		20	

