



DATOS DEL ESTUDIANTE

Apellidos y Nombres:	Alcala Meneses Luis Orlando		ID:	001463684
Dirección Zonal/CFP:	Zonal Ica-Ayacucho-Filial Chincha			
Carrera:	Ingenieria de Software con IA	Semestre	5	
Curso/ Mód. Formativo:	FULLSTACK DEVELOPER SOFTWARE			
Tema de Trabajo Final:	EDUPLUS			

1. INFORMACIÓN

- Identifica la problemática del caso práctico propuesto.

La botica "nova salud", ubicada en una zona con alta demanda, ha experimentado un crecimiento en su clientela que ha generado dificultades en la gestión del inventario y en la atención al cliente. actualmente, todos los procesos se realizan de forma manual, lo que provoca errores frecuentes, desabastecimientos y tiempos de espera prolongados. esta situación afecta la eficiencia del negocio y la satisfacción de los clientes. por ello, se propone implementar un software web que permita automatizar y centralizar el control de stock, ventas y atención, con el fin de optimizar la gestión interna y mejorar la competitividad de la botica.

Propuesta de solución

Implementar un software web centralizado que automatice la gestión del inventario, las ventas y la atención al cliente. Este sistema incluirá funcionalidades como actualización en tiempo real del stock, alertas automáticas para reposición de productos, registro de ventas y una interfaz amigable para mejorar el proceso de atención al cliente.

Objetivo:

Desarrollar una aplicación web que permita la gestión centralizada del inventario la automatización del registro de ventas y la optimización del proceso de atención al cliente.

- Respuestas a preguntas guía

Durante el análisis y estudio del caso práctico, debes obtener las respuestas a las interrogantes:

Pregunta

01: *¿Cuáles son los principales desafíos operativos que enfrenta la botica "FarmaPlus" debido a la falta de un sistema de gestión de inventarios automatizado?*

Errores en conteo de stock, ventas no registradas correctamente, desabastecimientos frecuentes, demoras en atención al cliente y ausencia de reportes confiables.

Pregunta

02: *¿Qué características específicas debe incluir el software web para atender las necesidades de control de stock y atención al cliente en "FarmaPlus"?*

Control automatizado del stock, registro y seguimiento de ventas, interfaz fácil de usar, alertas de stock bajo y reportes exportables.

Pregunta

03: *¿De qué manera la implementación de un software web para la gestión de inventarios puede impactar en la reducción del desabastecimiento y la mejora del servicio al cliente?*

Aumenta la eficiencia, reduce errores humanos, mejora el control del inventario y permite una atención más rápida y ordenada al cliente.

Pregunta 04: *¿Qué beneficios adicionales, aparte del control de inventarios, podría ofrecer el software para incrementar la competitividad de "FarmaPlus" en su mercado?*

Estadísticas para toma de decisiones, seguimiento del rendimiento de empleados, historial de ventas, escalabilidad para futuras funciones como pagos electrónicos.

Pregunta 05: *¿Qué métricas se pueden utilizar para evaluar la efectividad del software en mejorar los tiempos de respuesta al cliente y reducir los errores en la gestión de inventario?*


Tiempo promedio de atención, porcentaje de productos sin stock, reducción de errores en inventario, número de ventas sin fallas y satisfacción del cliente.

2. PLANIFICACIÓN DEL TRABAJO

▪ Cronograma de actividades:

N°	ACTIVIDADES	CRONOGRAMA					
		Lun	Mar	Mie	Ju	Vi	Sab
1	Conoce y aplica prácticas de desarrollo de software colaborativo	x					
2	Usa entorno de ejecución backend con JavaScript		x				
3	Usa tecnología frontend con JavaScript			x			
4	Diseña y crea servicios API RESTful				x		

▪ Lista de recursos necesarios:

1. MÁQUINAS Y EQUIPOS	
Descripción	Cantidad
 Laptop	1

2. HERRAMIENTAS E INSTRUMENTOS	
Descripción	Cantidad
❖ Visual Studio code	1
❖ Node.js	
❖ GitHub	

3. MATERIALES E INSUMOS	
Descripción	Cantidad
💧 Energía	1
💧 Documentación de SENATI	1
💧 Internet	

3. DECIDIR PROPUESTA

- Describe la propuesta determinada para la solución del caso práctico

PROPUESTA DE SOLUCIÓN

La propuesta consiste en implementar Eduplus, una plataforma web (Node.js + MySQL) para gestionar categorías, subcategorías, profesores y cursos.

1.Objetivo

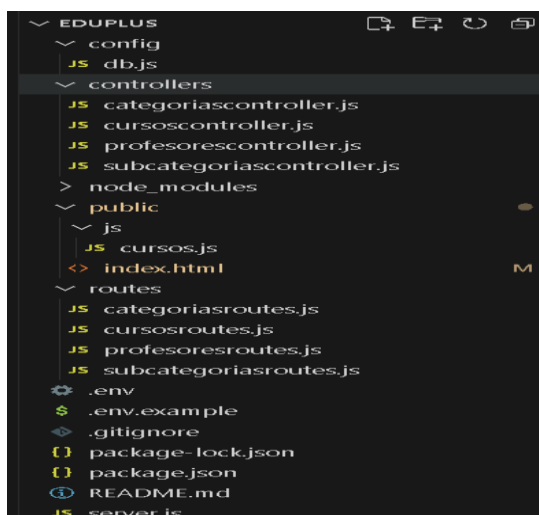
Implementar una aplicación web para gestionar categorías, subcategorías, profesores y cursos con un backend REST (Node.js + Express + MySQL) y un frontend ligero con Bootstrap y Font Awesome, mejorando la presentación y usabilidad del panel administrativo.

Alcance

- Conexión y consultas a MySQL correctas.
- Endpoints REST para categorías, subcategorías, profesores y cursos.
- Panel en public/index.html con:
 - Listado de categorías y cursos.
 - Formularios en modales para crear registros.
 - Estilos con Bootstrap y iconos con Font Awesome (agregados).
- Estructura del proyecto organizada (config, controllers, routes, public).

2) Arquitectura

Patrón MVC ligero / por capas



3) Configuración:

Variables de entorno (.env)

```
.env
1 DB_HOST=localhost
2 DB_USER=root
3 DB_PASSWORD=
4 DB_DATABASE=eduplus
5 DB_PORT=3306
6 DB_POOL_SIZE=10
```

4) Diseño de la base de datos:EDUPLUS

SQLyog Enterprise - MySQL GUI - [New Connection - root@localhost*]

File Edit Favorites DB Table Objects Tools Powertools Window Help

eduplus

root@localhost

- biblioteca
- eduplus**
- farmaplus
- information_schema
- mysql
- performance_schema
- phpmyadmin
- test
- videoclub

Query Query Builder Schema Designer

Autocomplete: [Tab]->Next Tag. [Ctrl+Space]->List Matching Tags. [Ctrl+Enter]->List All Tags.

```
1 -- Crear BD con UTF-8
2 CREATE DATABASE IF NOT EXISTS eduplus
3 CHARACTER SET utf8mb4
4 COLLATE utf8mb4_general_ci;
5 USE eduplus;
6 CREATE TABLE categorias (
7   id_cat INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
8   nombre VARCHAR(80) NOT NULL UNIQUE
9 ) ENGINE=INNODB;
10 CREATE TABLE subcategorias (
11   id_sub INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
12   id_cat INT UNSIGNED NOT NULL,
13   nombre VARCHAR(80) NOT NULL,
14   UNIQUE KEY uq_subcat (id_cat, nombre),
15   CONSTRAINT fk_subcat_cat FOREIGN KEY (id_cat)
16     REFERENCES categorias(id_cat)
17 ) ENGINE=INNODB;
```

Subcategorias:

SQLyog Enterprise - MySQL GUI - [New Connection - root@localhost*]

File Edit Favorites DB Table Objects Tools Powertools Window Help

eduplus

root@localhost

- biblioteca
- eduplus
 - Tables
 - categorias
 - cursos
 - profesores
 - subcategorias**
 - Views
 - Stored Procs
 - Functions
 - Triggers
 - Events
- farmaplus
- information_schema
- mysql
- performance_schema
- phpmyadmin
- test
- videoclub

Query Query Builder Schema Designer

Autocomplete: [Tab]->Next Tag. [Ctrl+Space]->List Matching Tags. [Ctrl+Enter]->List All Tags.

```
10 CREATE TABLE subcategorias (
11   id_sub INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
12   id_cat INT UNSIGNED NOT NULL,
13   nombre VARCHAR(80) NOT NULL,
14   UNIQUE KEY uq_subcat (id_cat, nombre),
15   CONSTRAINT fk_subcat_cat FOREIGN KEY (id_cat)
16     REFERENCES categorias(id_cat)
17     ON UPDATE CASCADE ON DELETE RESTRICT
18 ) ENGINE=INNODB;
19 CREATE TABLE profesores (
20   id_doc INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
21   nombres VARCHAR(100) NOT NULL,
22   apellidos VARCHAR(100) NOT NULL,
23   email VARCHAR(120),
24   UNIQUE KEY uq_doc_email (email)
25 ) ENGINE=INNODB;
```

1 Messages 2 Table Data 3 Objects 4 History

All Rows Rows in a Range First Row: 0 No. of Rows: 50 Refresh

id_sub	id_cat	nombre
2	1	BD
1	1	Lenguajes
3	2	Geometría
*	(NULL)	(NULL)

Profesores:

SQLyog Enterprise - MySQL GUI - [New Connection - root@localhost*]

File Edit Favorites DB Table Objects Tools Powertools Window Help

eduplus

root@localhost

- biblioteca
- eduplus
 - Tables
 - categorias
 - cursos
 - profesores
 - subcategorias
 - Views
 - Stored Procs
 - Functions
 - Triggers
 - Events
- farmaplus
- information_schema
- mysql
- performance_schema
- phpmyadmin
- test
- videoclub

Query

Autocomplete: [Tab]->Next Tag. [Ctrl+Space]->List Matching Tags. [Ctrl+Enter]->List All Tags.

```

11 id_sub INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
12 id_cat INT UNSIGNED NOT NULL,
13 nombre VARCHAR(80) NOT NULL,
14 UNIQUE KEY uq_subcat (id_cat, nombre),
15 CONSTRAINT fk_subcat_cat FOREIGN KEY (id_cat)
16 REFERENCES categorias(id_cat)
17 ON UPDATE CASCADE ON DELETE RESTRICT
18 ) ENGINE=INNODB;
19 CREATE TABLE profesores (
20 id_doc INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
21 nombres VARCHAR(100) NOT NULL,
22 apellidos VARCHAR(100) NOT NULL,
23 email VARCHAR(120),
24 UNIQUE KEY uq_doc_email (email)
25 ) ENGINE=INNODB;
26 CREATE TABLE cursos (

```

1 Messages 2 Table Data 3 Objects 4 History

All Rows Rows in a Range First Row: 0 No. of Rows: 50 Refresh

	id_doc	nombres	apellidos	email
	1	Juan	Tandaña	juan@eduplus.pe
*	(NULL)	(NULL)	(NULL)	(NULL)

Cursos:

SQLyog Enterprise - MySQL GUI - [New Connection - root@localhost*]

File Edit Favorites DB Table Objects Tools Powertools Window Help

eduplus

root@localhost

- biblioteca
- eduplus
 - Tables
 - categorias
 - cursos
 - profesores
 - subcategorias
 - Views
 - Stored Procs
 - Functions
 - Triggers
 - Events
- farmaplus
- information_schema
- mysql
- performance_schema
- phpmyadmin
- test
- videoclub

Query

Autocomplete: [Tab]->Next Tag. [Ctrl+Space]->List Matching Tags. [Ctrl+Enter]->List All Tags.

```

26 CREATE TABLE cursos (
27 id_curso INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
28 titulo VARCHAR(120) NOT NULL,
29 descripcion TEXT,
30 fecha_inicio DATE,
31 fecha_fin DATE,
32 duracion_h INT UNSIGNED,
33 precio DECIMAL(10,2) NOT NULL DEFAULT 0.00,
34 id_sub INT UNSIGNED NOT NULL,
35 id_doc INT UNSIGNED NULL,
36 UNIQUE KEY uq_curso (id_sub, titulo),
37 KEY idx_curso_sub (id_sub),
38 KEY idx_curso_doc (id_doc),
39 CONSTRAINT fk_curso_sub FOREIGN KEY (id_sub)
40 REFERENCES subcategorias(id_sub)
41 ON UPDATE CASCADE ON DELETE RESTRICT

```

1 Messages 2 Table Data 3 Objects 4 History

All Rows Rows in a Range First Row: 0 No. of Rows: 50 Refresh

	id_curso	titulo	descripcion	fecha_inicio	fecha_fin	duracion_h	precio	id_sub	id_doc
	1	Python Básico	Intro a Python	2025-11-10	2025-12-20	40	100.00	1	
	2	SQLite	Bases de datos...	2025-11-15	2025-12-15	30	60.00	2	
	3	Geometría I	Ángulos y triángulos	2025-11-05	2025-12-05	32	40.00	3	

5) Ejecución:

npm init -y

npm install express mysql2 dotenv

Levantar:

nodemon server.js

6) Configuración del Servidor Backend

-Archivo db.js — Conexión con la Base de Datos

Este archivo permite establecer la conexión entre el servidor backend y la base de datos MySQL usando un pool de conexiones. Esto mejora el rendimiento porque mantiene conexiones abiertas y reutilizables en lugar de abrir una nueva cada vez que se hace una consulta.

```
JS db.js  X
config > JS db.js > ...
1  require('dotenv').config();
2
3  //Administrar la BD(promesesas=proceso en curso..)
4  const mysql = require('mysql2/promise');
5
6  //Pool de conexiones = acceso
7  const pool = mysql.createPool({
8    host: process.env.DB_HOST || '127.0.0.1',
9    user: process.env.DB_USER || 'root',
10   password: process.env.DB_PASSWORD || '',
11   database: process.env.DB_DATABASE || 'eduplus',
12   port: Number(process.env.DB_PORT || 3306),
13   waitForConnections: true,
14   connectionLimit: Number(process.env.DB_POOL_SIZE || 10),
15   queueLimit: 0,
16   multipleStatements: false, // evita SQL injection por lotes
17   dateStrings: true         // fechas como string (sin líos de TZ)
18 });
19
20 //Aprovechar el recurso en otra parte de la App
21 module.exports = pool;
```

-Archivo controller-categoriascontroller.js — Lógica del CRUD

Este archivo contiene la lógica de negocio de la aplicación, es decir, las funciones que interactúan directamente con la base de datos MySQL para crear, listar, buscar, actualizar y eliminar registros de la tabla Eduplus.

```
JS categoriascontroller.js X
controllers > JS categoriascontroller.js > ...
1 // Categorías - CRUD
2 const pool = require('../config/db')
3 // Crear
4 exports.crearCategoria = async (req,res)=>{
5   const { nombre } = req.body
6   if(!nombre) return res.status(400).json({mensaje:'nombre requerido'})
7   try{
8     const [r] = await pool.query('INSERT INTO categorias(nombre) VALUES (?)',[nombre])
9     res.status(201).json({ id_cat:r.insertId, mensaje:'Categoría creada' })
10  }catch(e){ if(e.code==='ER_DUP_ENTRY') return res.status(409).json({mensaje:'categoría ya existe'}); res.sta
11  }
12
13 // Listar
14 exports.obtenerCategorias = async (_req,res)=>{
15   try{ const [rows]=await pool.query('SELECT id_cat,nombre FROM categorias ORDER BY nombre'); res.json(rows) }
16   catch(e){ res.status(500).json({mensaje:'Error interno'}) } }
17 }
18
19 // Obtener por ID
20 > exports.obtenerCategoriaPorId = async (req,res)=>{ ...
26 }
27
28 // Actualizar
29 > exports.actualizarCategoria = async (req,res)=>{ ...
37 }
38
39 // Eliminar
40 > exports.eliminarCategoria = async (req,res)=>{ ...
46 }
47
```

-Archivo controller-cursoscontroller.js — Lógica del CRUD

```
JS cursoscontroller.js ×
controllers > JS cursoscontroller.js > ...
  1 // Cursos - CRUD
  2 const pool = require('../config/db')
  3
  4 > const SELECT = `...
13 LEFT JOIN profesores p ON p.id_doc=c.id_doc`
14
15 // Crear
16 > exports.crearCurso = async (req,res)=>{ ...
30 }
31
32 // Listar
33 > exports.obtenerCursos = async (_req,res)=>{ ...
36 }
37
38 // Obtener por ID
39 > exports.obtenerCursoPorId = async (req,res)=>{ ...
45 }
46
47 // Actualizar (parcial)
48 > exports.actualizarCurso = async (req,res)=>{ ...
66 }
67
68 // Eliminar
69 > exports.eliminarCurso = async (req,res)=>{ ...
75 }
```

Archivo controller-profesorerescontroller.js — Lógica del CRUD

```
JS profesorescontroller.js X
controllers > JS profesorescontroller.js > ...
1 // Profesores - CRUD
2 const pool = require('../config/db')
3
4 // Crear
5 exports.crearProfesor = async (req,res)=>{
6   const { nombres, apellidos, email } = req.body
7   if(!nombres || !apellidos) return res.status(400).json({mensaje:'nombres y apellidos requeridos'})
8   try{
9     const [r]=await pool.query('INSERT INTO profesores(nombres,apellidos,email) VALUES (?,?,:)',[nombres,apell
10     res.status(201).json({ id_doc:r.insertId, mensaje:'Profesor creado' })
11   }catch(e){ if(e.code==='ER_DUP_ENTRY') return res.status(409).json({mensaje:'email ya existe'}); res.status(
12   }
13
14 // Listar
15 > exports.obtenerProfesores = async (_req,res)=>{ ...
18 }
19
20 // Obtener por ID
21 > exports.obtenerProfesorPorId = async (req,res)=>{ ...
27 }
28
29 // Actualizar
30 > exports.actualizarProfesor = async (req,res)=>{ ...
38 }
39
40 // Eliminar
41 > exports.eliminarProfesor = async (req,res)=>{ ...
47 }
48
```

Archivo controller-subcategoriascontroller.js — Lógica del CRUD

```

JS subcategoriascontroller.js X
controllers > JS subcategoriascontroller.js > ...
1 // Subcategorías - CRUD
2 const pool = require('../config/db')
3
4 // Crear
5 exports.crearSubcategoria = async (req,res)=>{
6   const { nombre, id_cat } = req.body
7   if(!nombre || !id_cat) return res.status(400).json({mensaje:'nombre e id_cat requeridos'})
8   try{
9     const [r]=await pool.query('INSERT INTO subcategorias(nombre,id_cat) VALUES (?,?)',[nombre,id_cat])
10    res.status(201).json({ id_sub:r.insertId, mensaje:'Subcategoría creada' })
11  }catch(e){ if(e.code==='ER_DUP_ENTRY') return res.status(409).json({mensaje:'duplicada en la misma categoría'})
12 }
13
14 // Listar
15 > exports.obtenerSubcategorias = async (_req,res)=>{ ...
20 }
21
22 // Obtener por ID
23 > exports.obtenerSubcategoriaPorId = async (req,res)=>{ ...
32 }
33
34 // Actualizar
35 > exports.actualizarSubcategoria = async (req,res)=>{ ...
43 }
44
45 // Eliminar
46 > exports.eliminarSubcategoria = async (req,res)=>{ ...
52 }
53

```

Routes

Archivo categoriaroutes.js — Definición de Rutas de la API

Este archivo define las rutas principales que permiten interactuar con el backend de Eduplus. Cada ruta está asociada a una función específica del controlador (categoriascontroller.js), lo que facilita la organización del código siguiendo el patrón MVC (Modelo – Vista – Controlador).

```

JS categoriaroutes.js ●
routes > JS categoriaroutes.js > ...
1 const express = require('express');
2 const router = express.Router();
3 const categoriaController = require('../controllers/categoriascontroller');
4
5 // Crear categoría
6 router.post('/', categoriaController.crearCategoria);
7
8 // Listar todas las categorías
9 router.get('/', categoriaController.obtenerCategorias);
10
11 // Obtener categoría por ID
12 router.get('/:id', categoriaController.obtenerCategoriaPorId);
13
14 // Actualizar categoría
15 router.put('/:id', categoriaController.actualizarCategoria);
16
17 // Eliminar categoría
18 router.delete('/:id', categoriaController.eliminarCategoria);
19
20 // Exportar el router
21 module.exports = router;

```

Archivo server.js — Configuración del Servidor Backend

Este archivo es el corazón del proyecto. Se encarga de levantar el servidor con Express, conectarlo a la base de datos y registrar las rutas de la API.

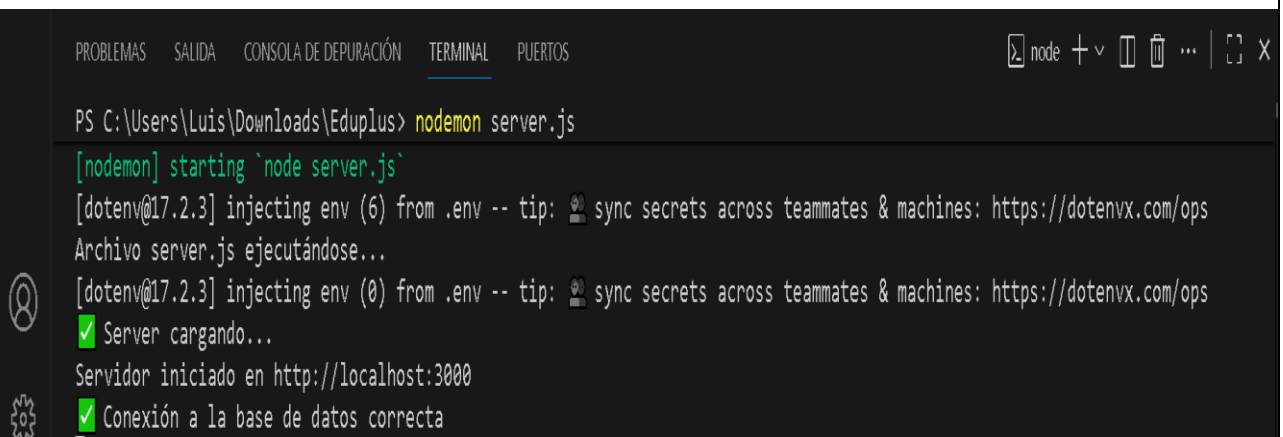
```

JS server.js  X
JS server.js > ...
1  // Cargar variables de entorno
2  require('dotenv').config()
3
4  // Mensaje de control al iniciar
5  console.log('Archivo server.js ejecutándose...')
6
7  // Importar dependencias principales
8  const express = require('express')
9
10 // Importar rutas de Eduplus
11 const categoriasRoutes = require('./routes/categoriasroutes')
12 const subcategoriasRoutes = require('./routes/subcategoriasroutes')
13 const profesoresRoutes = require('./routes/profesoresroutes')
14 const cursosRoutes = require('./routes/cursosroutes')
15
16 // Importar pool de conexión a la BD
17 const pool = require('./config/db')
18
19 // Inicializar la aplicación Express
20 const app = express()
21
22 // Definir el puerto del servidor (por .env o por defecto 3000)
23 const PORT = process.env.PORT || 3000
24

```

Consola: Evidencia en consola de la correcta conexión del servidor backend con la base de datos MySQL. El mensaje “ Conexión a la base de datos correcta” confirma que el pool establecido en db.js funciona correctamente.

El mensaje “ Servidor iniciado en http://localhost:3000” indica que el servidor Express se está ejecutando sin errores y está listo para recibir peticiones en el puerto configurado.



```

PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
node + v  [icon]  [icon]  [icon]  [icon]  [icon]  [icon]  [icon]  [icon]  [icon]  [icon]

PS C:\Users\Luis\Downloads\Eduplus> nodemon server.js

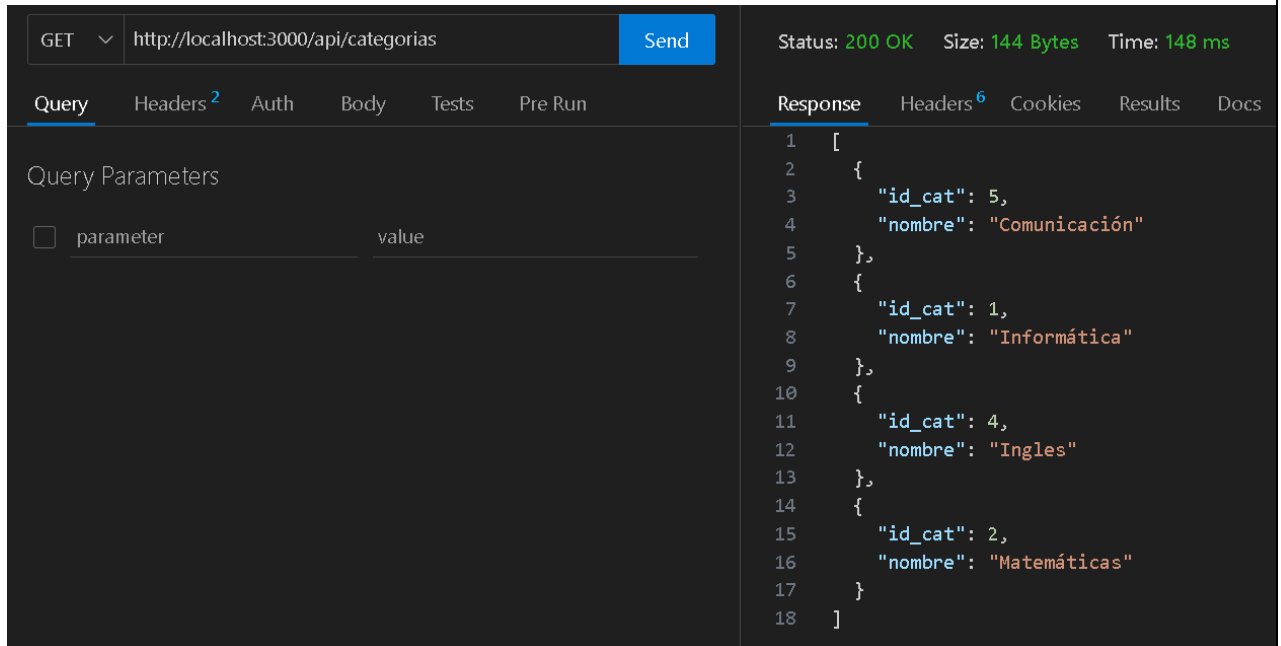
[nodemon] starting `node server.js`
[dotenv@17.2.3] injecting env (6) from .env -- tip: 📖 sync secrets across teammates & machines: https://dotenvx.com/ops
Archivo server.js ejecutándose...
[dotenv@17.2.3] injecting env (0) from .env -- tip: 📖 sync secrets across teammates & machines: https://dotenvx.com/ops
✅ Server cargando...
Servidor iniciado en http://localhost:3000
✅ Conexión a la base de datos correcta

```

Endpoints REST Implementado

GET /api/categorias→ listar categorías.

Este endpoint permite obtener un listado completo de todas los Eduplus almacenadas en la base de datos. Cuando se realiza una solicitud GET a esta ruta, el servidor responde con un arreglo en formato JSON que contiene cada categoría registrada, incluyendo sus atributos: id, nombre.



The screenshot shows a REST client interface. The request is a GET to `http://localhost:3000/api/categorias`. The status is **200 OK**, size is **144 Bytes**, and time is **148 ms**. The response is a JSON array of three category objects:

```

1  [
2    {
3      "id_cat": 5,
4      "nombre": "Comunicación"
5    },
6    {
7      "id_cat": 1,
8      "nombre": "Informática"
9    },
10   {
11     "id_cat": 4,
12     "nombre": "Inglés"
13   },
14   {
15     "id_cat": 2,
16     "nombre": "Matemáticas"
17   }
18 ]

```

6) Resultados

- CRUD funcional para las entidades definidas.
- Interfaz con Bootstrap y Font Awesome mejorada visualmente.
- Conexión a BD estable (pool MySQL).

El proyecto Eduplus logró un backend REST confiable (Node.js + Express + MySQL) y un panel web usable con Bootstrap y Font Awesome, cumpliendo el CRUD de categorías, subcategorías, profesores y cursos. La arquitectura modular (config–controllers–routes–public) facilita el mantenimiento y la escalabilidad, y deja preparado el sistema para incorporar mejoras sin romper lo construido. La base de datos está correctamente normalizada y la conexión estable asegura integridad en las operaciones

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.

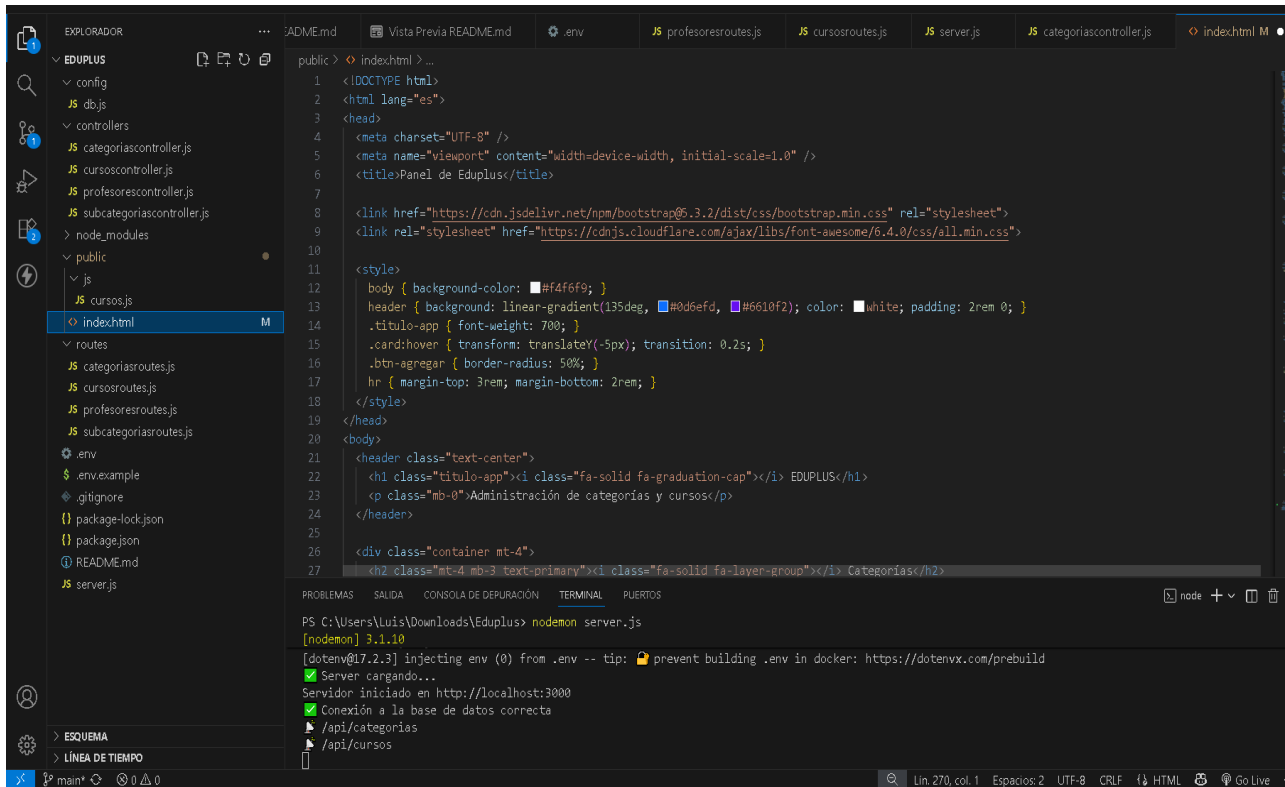
INSTRUCCIONES: Ser lo más explícito posible. Los gráficos ayudan a transmitir mejor las ideas. Tomar en cuenta los aspectos de calidad, medio ambiente y SHI.

OPERACIONES / PASOS / SUBPASOS	NORMAS TÉCNICAS - ESTANDARES / SEGURIDAD / MEDIO AMBIENTE
Paso 1: Se definió la BD eduplus en MySQL con charset utf8mb4.	ISO/IEC 23894: Gestión de riesgos en tecnologías de la información
Paso 2: Se crearon tablas: categorías, subcategorías, profesores y cursos con claves foráneas.	ISO/IEC 23053: Marco para el uso de aprendizaje automático (machine learning)
Paso 3: Se cargó semilla inicial (categorías, subcategorías, profesor y cursos).	ISO/IEC 24029: Evaluación de la robustez de sistemas de IA.
Paso 4: Se configuró .env con variables de conexión a MySQL y puerto.	ISO/IEC 27001: Gestión de la seguridad de la información.
Paso 5: Se instaló y configuró Express y mysql2 en el proyecto.	ISO/IEC 20000: Gestión de servicios de TI.
Paso 6: Se creó config/db.js con pool de conexiones y lectura de .env.	ISO/IEC 12207: Procesos del ciclo de vida del software.
Paso 7: Se implementaron rutas REST para categorías (GET/POST/DELETE).	ISO/IEC 25010: Calidad del producto software.
Paso 8: Se implementaron rutas REST para subcategorías (GET/POST).	ISO/IEC 9126: Calidad del software.
Paso 9: Se implementaron rutas REST para profesores (GET/POST).	ISO/IEC 27032: Seguridad en el ciberespacio.
Paso 10: Se implementaron rutas REST para cursos (GET/POST/DELETE).	ISO/IEC 27005: Gestión de riesgos de seguridad de la información.
Paso 11: Se configuró server.js: middlewares, registro de rutas y estáticos desde /public.	ISO/IEC 15408 : Criterios comunes para la evaluación de seguridad de productos y sistemas de TI.
Paso 12: Se construyó public/index.html como panel de administración.	ISO 9001: Sistemas de gestión de calidad.
Paso 13: Se integró Bootstrap y Font Awesome para estilos e íconos en el front.	ISO 21500: Gestión de proyectos.
Paso 14: Se programaron fetch en el front para listar/crear/eliminar (modales y botones).	ISO 14001: Gestión ambiental.
Paso 15: Se probó la API con Thunder Client y se verificó el CRUD funcionando.	ISO 50001: Gestión de la energía.
.	

DIBUJO / ESQUEMA / DIAGRAMA DE PROPUESTA

(Adicionar las páginas que sean necesarias)

DIBUJO 1:



The screenshot shows a VS Code editor with the following components:

- EXPLORADOR (Explorer):** Shows the project structure for 'EDUPLUS'. The 'public' folder is expanded, showing 'index.html' (selected) and 'js' folder. Other folders include 'config', 'controllers', 'node_modules', and 'routes'.
- index.html:** Contains the following code:


```

1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6   <title>Panel de Eduplus</title>
7
8   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
9   <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-free@6.4.0/css/all.min.css">
10
11   <style>
12     body { background-color: #f4f6f9; }
13     header { background: linear-gradient(135deg, #0d6efd, #6610f2); color: white; padding: 2rem 0; }
14     .titulo-app { font-weight: 700; }
15     .card:hover { transform: translateY(-5px); transition: 0.2s; }
16     .btn-agregar { border-radius: 50%; }
17     hr { margin-top: 3rem; margin-bottom: 2rem; }
18   </style>
19 </head>
20 <body>
21   <header class="text-center">
22     <h1 class="titulo-app"><i class="fa-solid fa-graduation-cap"></i> EDUPLUS</h1>
23     <p class="mb-0">Administración de categorías y cursos</p>
24   </header>
25
26   <div class="container mt-4">
27     <h2 class="mt-4 mb-3 text-primary"><i class="fa-solid fa-layer-group"></i> Categorías</h2>

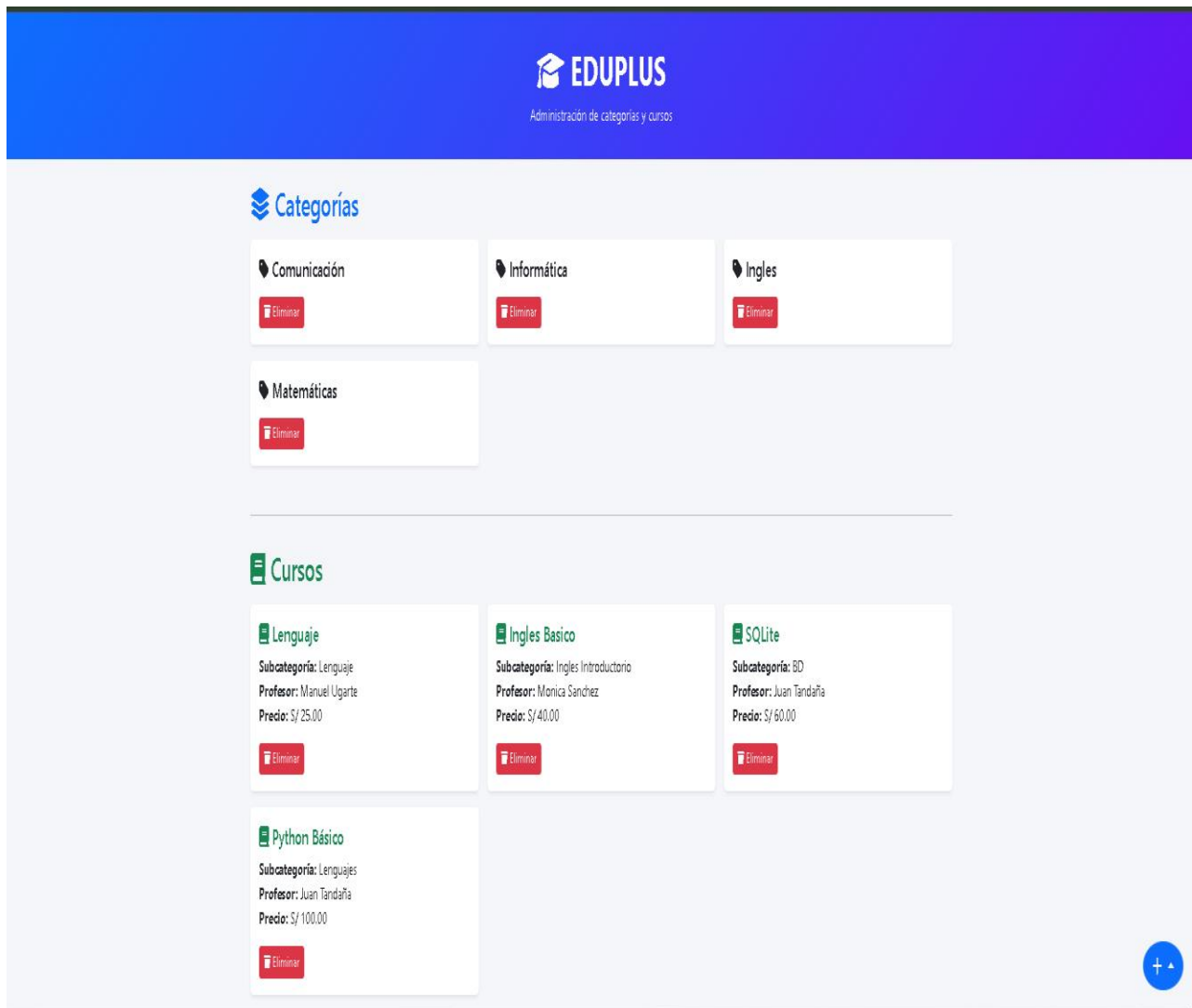
```
- TERMINAL:** Shows the command 'PS C:\Users\Luis\Downloads\Eduplus> node server.js' and the output:


```

[dotenv@17.2.3] injecting env (0) from .env -- tip: prevent building .env in docker: https://dotenvx.com/prebuild
[dotenv@17.2.3] Server cargando...
Servidor iniciado en http://localhost:3000
Conexión a la base de datos correcta
/api/categorias
/api/cursos

```

DIBUJO 2:



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 checked="" type="checkbox"/>	<input type="checkbox"/>
• ¿Se elaboró un cronograma claro de actividades a ejecutar?	<input checked="" type="checkbox"/>	<input 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 checked="" 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			

