INFORME DEL MÓDULO CLIENTES

Proyecto: CODEASY — Sistema Gestor de Proyectos

Integrante responsable: Andres Elias Arroyo

Entrega: Sprint 1 — Módulo de Clientes

Tecnologías: Node.js, TypeScript, Fastify, PostgreSQL

1. Contexto general del sistema

El sistema CODEASY es una aplicación de gestión integral de proyectos, clientes, consultores y tareas. Opera bajo una arquitectura hexagonal (Ports & Adapters), lo que permite una separación clara entre el dominio, la aplicación, la infraestructura y la capa de presentación. El módulo Clientes se encarga de administrar toda la información de los clientes registrados en la organización, ofreciendo operaciones CRUD completas y manejo centralizado de errores y validaciones.

2. Objetivo del módulo de clientes

El objetivo principal del módulo es registrar, consultar, actualizar y eliminar clientes de manera segura, validando los datos ingresados antes de su persistencia en la base de datos PostgreSQL.

3. Arquitectura utilizada

Arquitectura: Hexagonal (Ports & Adapters)

Dominio: core/dominio/cliente — Define la entidad Cliente y la interfaz ICliente.
Aplicación: core/aplicacion/casos-uso/ClienteCasosUso — Contiene la lógica de negocio y flujos.
Infraestructura: core/infraestructura/postgres/ClienteRepositorio.ts — Implementa los métodos CRUD conectados a PostgreSQL.
Presentación: presentacion/controladores/ClienteControlador.ts — Controla rutas HTTP y validaciones con Zod.

4. Base de datos

Archivo 001-tabla-Clientes.sql en Carpeta Migraciones

Tabla: clientes
id_cliente SERIAL PRIMARY KEY
nombre VARCHAR(100) NOT NULL
identificacion VARCHAR(50) UNIQUE NOT NULL
email VARCHAR(100) UNIQUE NOT NULL
telefono VARCHAR(20) NULL
estatus VARCHAR(10) DEFAULT 'Activo'

Reglas: los clientes eliminados se marcan como 'Eliminado' y solo se listan los activos.

5. Validaciones implementadas Zod.

El controlador del módulo **Cliente** valida los datos con **Zod**, asegurando que los campos sean correctos antes de procesarlos.
Si ocurre un error, usa ZodError.flatten() para generar mensajes claros y el método manejarError() para devolver respuestas coherentes (400, 404 o 409) según el tipo de fallo.

Ejemplos:

- "nombre: El nombre debe tener al menos 3 caracteres"
- "email: El formato del correo no es válido"
- "identificacion: Este campo es obligatorio"
- "telefono: Debe contener al menos 7 dígitos y no más de 15"
- "identificacion: Ya existe un cliente con esta identificación"

6. Endpoints del módulo

POST /api/cliente — Crea un nuevo cliente
GET /api/cliente — Lista todos los clientes activos
GET /api/cliente/:idCliente — Obtiene un cliente por su ID
PUT /api/cliente/:idCliente — Actualiza un cliente existente
DELETE /api/cliente/:idCliente — Elimina lógicamente un cliente

7. Ejecución del módulo

1. Configurar entorno (.env):
PUERTO=3000

PGHOST=localhost
PGPORT=5432
PGUSER=postgres
PGPASSWORD=admin
PGDBNAME=codeasy_db

2. Crear base de datos y tabla clientes.
3. Ejecutar servidor: npm run dev

Consola esperada: Conectado correctamente a PostgreSQL y servidor corriendo en http://localhost:3000

Pruebas realizadas

Usando el Cliente Bruno con las operaciones CRUD definidas.

• Crear cliente con todos los campos → ✅ Cliente creado exitosamente
• Crear cliente con email inválido → ❌ email: El formato del correo no es válido
• Actualizar cliente → ✅ Cliente actualizado exitosamente
• Buscar cliente inexistente → ❌ Cliente no encontrado
• Eliminar cliente → ✅ Cliente eliminado correctamente
• Listar clientes → ✅ Solo muestra los activos