Diseños Backend
SPRINT 0

PROYECTO APLICACIONES DE BIOMETRÍA Y MEDIO AMBIENTE

Nombre del Alumno: Alan Guevara Martínez

Enlace Git: https://github.com/ALANGMupv/ProyectoBiometria2025.git

Enlace trello:
https://trello.com/invite/b/68c9273bf08b76ab09c96593/ATTI19e0d5a059994ed24df6c6455b577bae3E1C4F40/gtiproyectobiometria

Curso: 2025/2026

ÍNDICE / TABLA DE CONTENIDOS

TABLA DE ILUSTRACIONES / DISEÑOS

Introducción

En este documento se presentan los diseños y la documentación del backend en **Node.js con Express** y del **front-end en JavaScript (cliente)** que consume la API REST. El flujo completo conecta los dispositivos BLE → móvil → API REST → lógica de negocio → base de datos → interfaz web (Todo el flujo subido al **Plesk**).

Flujo simplificado:

1. El cliente hace una petición → API REST.
2. ReglasREST valida la entrada → (controlador).
3. Llama a Logica.guardarMedida() o Logica.listarMedidas() → (lógica de negocio).
4. La lógica ejecuta SQL sobre MySQL.
5. Devuelve los datos al cliente en JSON.

Los diagramas completos de diseño pueden consultarse en el archivo de Figma correspondiente a **Diseño Backend y Frontend**:
https://www.figma.com/board/OhXGiBzKXEmR9dCqy3RwvB/Dise%C3%B1o-Backend-y-Frontend?node-id=0-1&t=TAxnJyZzxQ48UPCf-1

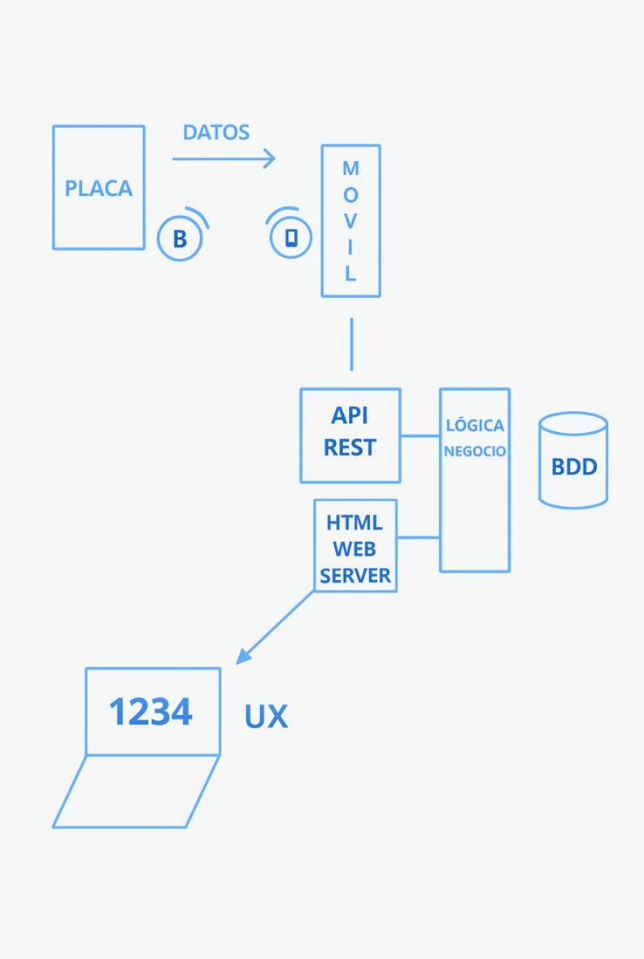

Ilustración 1: Diseño Flujo Completo

mainServidorREST.js (punto de entrada del servidor)

Archivo principal que configura y arranca el servidor REST con Express.
Se encarga de inicializar la infraestructura, aplicar middlewares y montar las rutas.

Responsabilidades principales:

- Configura el servidor Express.
- Aplica middlewares (express.json(), cors()).
- Instancia la lógica de negocio (Logica).
- Monta las rutas definidas en ReglasREST.
- Arranca el servidor en el puerto definido (en Plesk lo asigna el sistema).

ReglasREST.js (controladores / endpoints)

Archivo que define los endpoints REST de la API.
No accede directamente a la base de datos, sino que delega en la lógica de negocio.

Endpoints principales:

- **POST /medida:**
  - Recibe un JSON con { uuid, gas, valor, contador }.
  - Valida los campos obligatorios.
  - Llama a logica.guardarMedida(...).
  - Devuelve confirmación en JSON con la medida guardada o error.

- **GET /medidas:**
  - Permite obtener las últimas medidas registradas (?limit=N).
  - Llama a logica.listarMedidas(limit).
  - Devuelve un array JSON con las medidas.

El diseño se encuentra representado en la página que sigue.

DISEÑO API REST

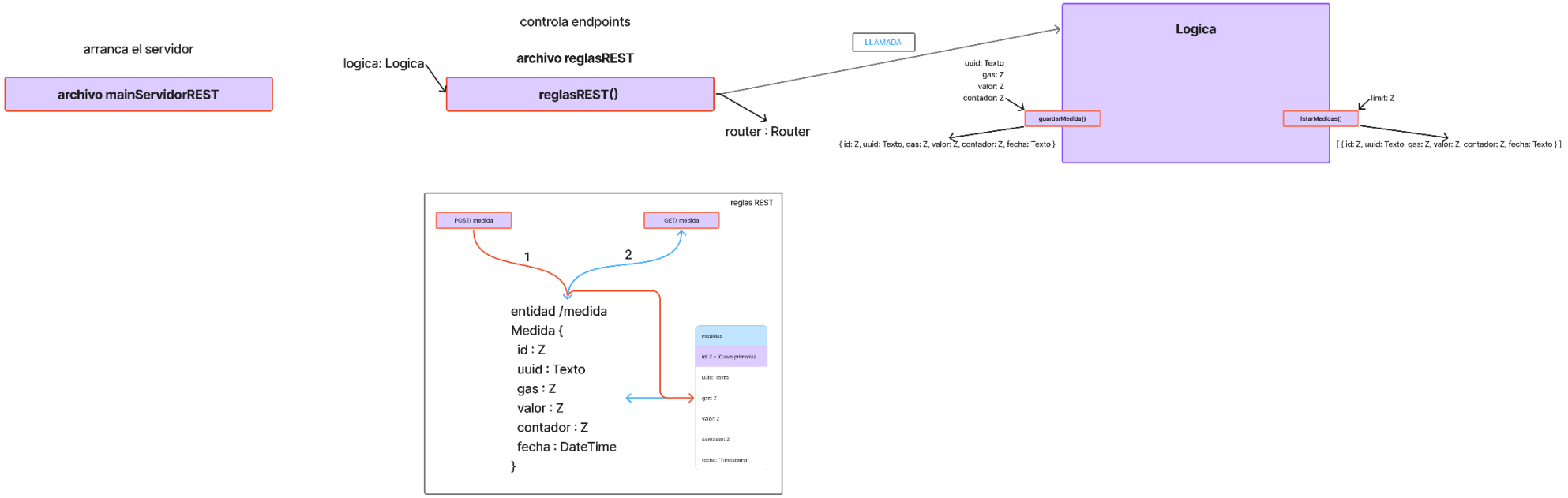

Ilustración 2: Diseño completo Backend

Logica.js (capa de negocio)

Clase que implementa la lógica de negocio de la aplicación, conectando con la base de datos MySQL.
Gestiona el pool de conexiones y ejecuta consultas SQL.

Funciones principales:

- **guardarMedida(uuid, gas, valor, contador):**
  Inserta una nueva medida en la tabla medidas y devuelve la fila insertada para feedback inmediato del cliente de la API.

- **listarMedidas(limit = 50):**
  Devuelve un array con las últimas medidas desde la base de datos, teniendo como parámetro un límite de 50, por si en el frontend no se pidiera nada.

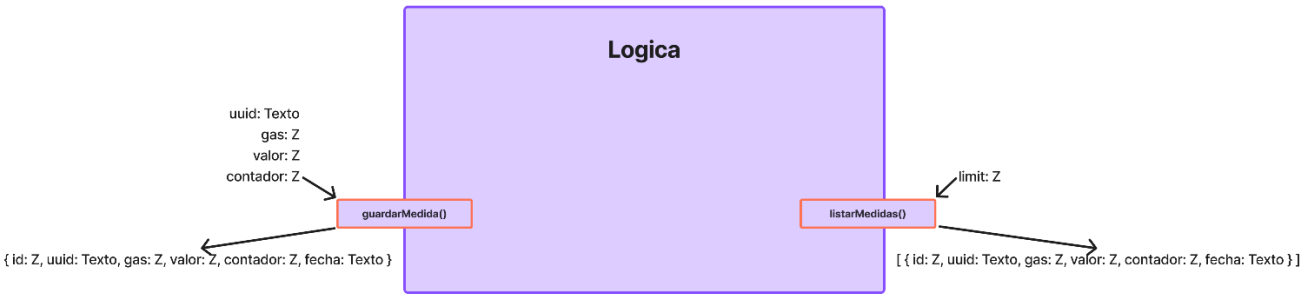

Ilustración 3: Diseño Logica.js

Base de datos (aguemar_proyecto_biometria, tabla medidas)

Estructura en MySQL que almacena las mediciones recibidas desde la API. En esta primera versión solo tenemos una tabla para lo que se ha pedido, está alojada en phpMyAdmin en Plesk. "Quizás con más datos de los necesarios para el Sprint 0".

Tabla medidas:

- **id (Z):** clave primaria, autoincremental.
- **uuid (Texto):** identificador único del dispositivo/sensor.
- **gas (Z):** código numérico del tipo de gas.
- **valor (Z):** valor de la medida.
- **contador (Z):** número de secuencia de la medida.
- **fecha (Timestamp):** momento en que se almacenó la medición.

Ilustración 4: Diseño BBDD

index.js (front-end web – "extra")

Archivo JavaScript que corre en el navegador y consume la API REST.
Actualiza dinámicamente la tabla de mediciones en la página web.

Funciones principales:

- **construirURL():** construye la URL de petición a la API en base al límite seleccionado.
- **cargarMedidas():** obtiene medidas desde la API y actualiza la tabla HTML.
- **pintarFilas(filas):** renderiza las filas de la tabla a partir del array de medidas.
- **mostrarError(msg):** muestra mensajes de error en pantalla.
- **gasATexto(gas):** convierte el código de gas a un texto descriptivo (ej. 11 → "CO_2").
- **formatearFecha(iso):** transforma la fecha ISO en formato dd/mm/yyyy hh:mm:ss.
- **escaparHTML(s):** protege la salida HTML contra inyecciones (XSS).

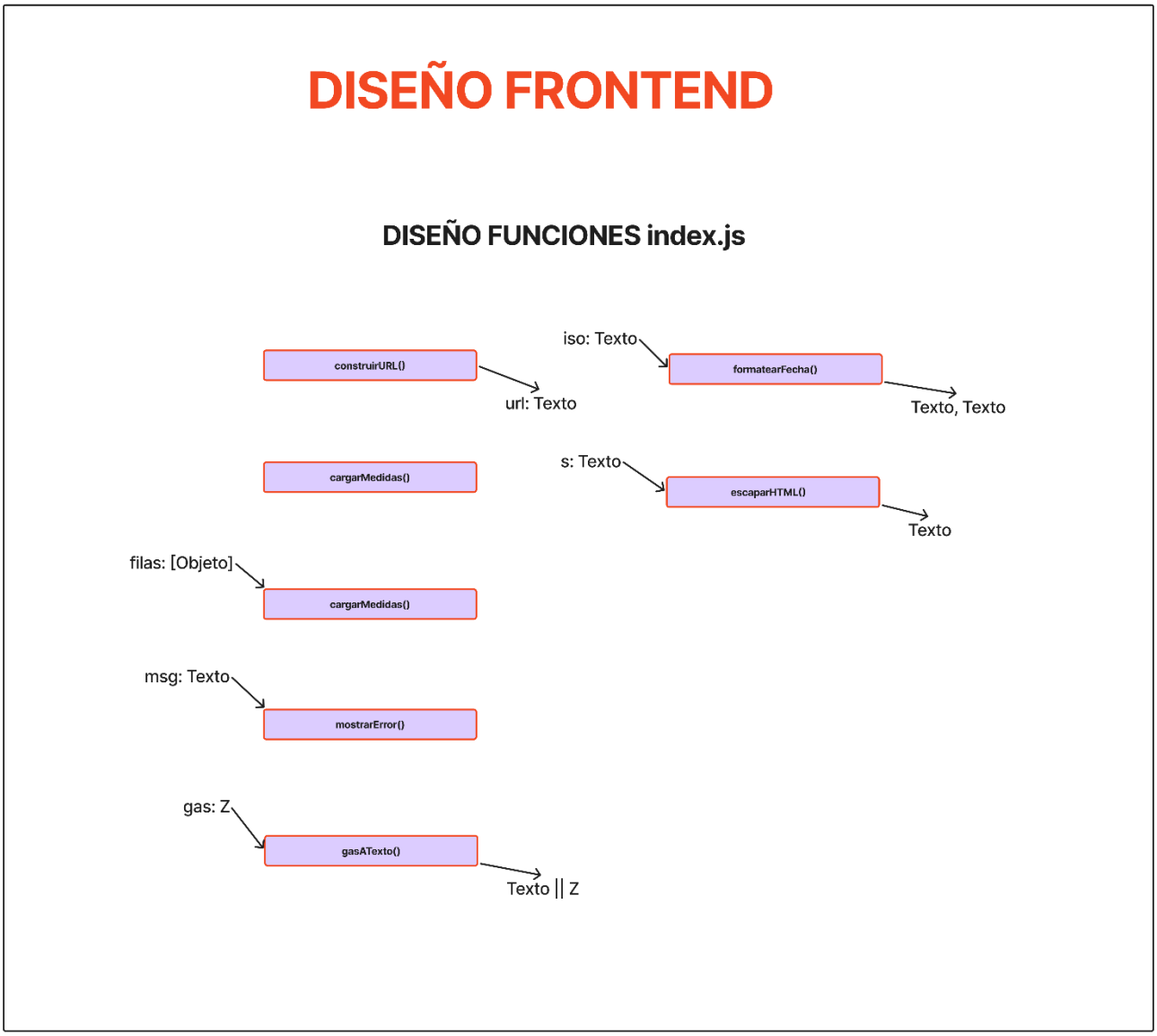

Ilustración 5: Diseño Frontend Web