



Challenge Técnico – Developer SSR



Descripción

El objetivo de este ejercicio es desarrollar una aplicación web **full-stack** que permita gestionar un listado de **Assets** (ej: laptops, celulares, monitores, etc.).

La aplicación debe estar compuesta por:

- **Backend (API REST)** en Node.js.
 - **Base de datos** PostgreSQL.
 - **Frontend simple** en HTML + CSS + JS (puede usar Bootstrap).
-



Requerimientos funcionales

1. El usuario debe poder:
 - Crear un asset.
 - Listar todos los assets existentes.
 - Editar un asset.
 - Eliminar un asset.
2. Cada asset debe tener al menos los siguientes campos:
 - **id** (autogenerado).
 - **name** (string, obligatorio).
 - **type** (string: "Laptop", "Celular", "Monitor", etc.).
 - **owner** (string, persona asignada).
 - **created_at** (fecha automática).
3. En el **frontend** debe existir una interfaz que permita:

- Formulario para crear/editar un asset.
 - Tabla con los assets existentes.
 - Botones para editar y eliminar.
-

Requerimientos técnicos

- **Backend:**
 - Node.js (puede usar Express u otro framework).
 - CRUD completo expuesto como **API REST** (`/api/assets`).
 - Validaciones básicas (ejemplo: `name` no vacío).
 - Manejo de errores (404, 500).
 - **Base de datos:**
 - PostgreSQL con al menos 1 tabla `assets`.
 - Entregar script SQL de creación (`schema.sql`).
 - **Frontend:**
 - HTML + CSS + JS (se puede usar Bootstrap).
 - Uso de `fetch()` para consumir la API.
 - CRUD completo desde la interfaz gráfica.
-

Entregables

1. Código fuente en un repositorio **GitHub/GitLab**.
2. Archivo `README.md` con:
 - Instrucciones de instalación.

- Configuración de conexión a PostgreSQL.
 - Comandos para ejecutar el proyecto.
 - Cómo acceder a la interfaz web.
-

Tiempo estimado

Se espera que este challenge pueda resolverse en **6 a 8 horas de trabajo** (puede realizarse en varios días).

Bonus (no obligatorios, pero suman puntos)

- Uso de **Docker** para levantar la app y la base de datos.
 - Autenticación básica (ej: login con JWT).
 - Tests unitarios en el backend.
 - Uso de algún framework frontend moderno (React, Vue, Angular) en lugar de HTML plano.
 - Buen diseño visual de la interfaz.
-

 **Entrega final:** compartir el link al repositorio con el código y las instrucciones.