Prueba Técnica Full-Stack: Plataforma de Gestión de Recursos de TI

Para: Candidato a Desarrollador Full-Stack
De: Unidad de Sistemas e Informática
Plazo de Entrega: El plazo oficial es de 2 días (48 horas). Entendemos que pueden surgir imprevistos, por lo que se aceptarán entregas hasta un máximo de 3 días (72 horas) tras la recepción de esta prueba.

Introducción

¡Hola! Bienvenido al desafío técnico para la posición de Desarrollador Full-Stack. Tu misión es diseñar y construir una aplicación web completa (frontend y backend) para la gestión y asignación inteligente de nuestro inventario de TI.

Esta prueba evaluará tu habilidad para crear una aplicación funcional y bien estructurada, desde el diseño de la base de datos y la API, hasta la implementación de una interfaz de usuario interactiva y la resolución de un problema logístico complejo mediante un algoritmo.

Paso 1: Diseño del Backend y la Base de Datos (Ampliado)

Esta es la fundación de tu aplicación. Debes crear una API RESTful robusta y un esquema de base de datos bien diseñado.

A. Diseño de la Base de Datos:

Diseña e implementa un esquema de base de datos relacional que incluya, como mínimo, las siguientes tablas:

- Equipos: Almacena cada pieza de hardware (id, tipo_equipo, modelo, numero_serie, estado, costo, especificaciones en JSON, etc.).
- Empleados: Registra a los usuarios del sistema (id, nombre_completo, rol_actual, etc.).
- Roles: Define los diferentes puestos de trabajo en la empresa (id, nombre_rol).
- PerfilesRequerimientos: Tabla que mapea qué Rol necesita qué tipo de Equipo y en qué cantidad (ej: Rol 'Diseñador' necesita 1 'Laptop' y 1 'Monitor').
- SolicitudesEquipamiento: Registra una solicitud formal hecha por un gerente (id, nombre_solicitud, fecha, estado [ej: 'pendiente', 'resuelta']).
- DetallesSolicitud: Tabla intermedia que detalla cuántos puestos de cada Rol se pidieron en una SolicitudEquipamiento.
- HistorialAsignaciones: Un log inmutable que registra cada vez que un Equipo es asignado o desasignado a un Empleado, guardando la fecha y el responsable.

B. Endpoints de la API:

Implementa los siguientes endpoints. Deben estar bien documentados y seguir las convenciones REST.

- **Gestión de Inventario:**
  - GET /api/equipos: Lista todos los equipos con filtros (por estado, por tipo).
  - POST /api/equipos: Agrega un nuevo equipo al inventario.
- **Gestión de Solicitudes:**
  - POST /api/solicitudes: Crea una nueva solicitud de equipamiento. Recibe un nombre para la solicitud y una lista de roles con sus cantidades (ej: [{rol_id: 1, cantidad: 2}, {rol_id: 2, cantidad: 1}]).
  - GET /api/solicitudes: Lista todas las solicitudes hechas.
  - GET /api/solicitudes/{id}: Obtiene los detalles de una solicitud específica.
- **El Endpoint Clave (Algoritmo):**
  - GET /api/solicitudes/{id}/propuesta-optima: **Este es el endpoint más importante.** No recibe cuerpo. Al ser llamado, debe ejecutar tu algoritmo de optimización sobre la solicitud {id} y devolver la mejor propuesta de asignación posible basada en el inventario disponible. La estructura de la respuesta debe ser clara (ver Paso 3).

Paso 2: Desarrollo de la Interfaz de Usuario (Frontend)

Debes crear una aplicación de una sola página (SPA) que consuma la API que construiste. Puedes usar el framework que prefieras (React, Vue, Angular, etc.).

Vistas Requeridas:

1. **Dashboard de Inventario:**
   - Muestra una tabla con todos los equipos del inventario.
   - Debe ser posible filtrar los equipos por estado ('disponible', 'asignado') y por tipo ('Laptop', 'Monitor', etc.).
   - La tabla debe mostrar claramente a quién está asignado un equipo, si aplica.
2. **Página de Solicitudes:**
   - Un formulario para crear una nueva "Solicitud de Equipamiento". El usuario debe poder dar un nombre a la solicitud y agregar dinámicamente los roles que necesita y la cantidad de personas por cada rol.
   - Una lista que muestre todas las solicitudes existentes y su estado ('pendiente', 'resuelta').
3. **Vista de Propuesta de Asignación:**
   - Al hacer clic en una solicitud 'pendiente', el usuario debe ser llevado a esta vista.

- La vista debe llamar automáticamente al endpoint GET /api/solicitudes/{id}/propuesta-optima.
- Debe mostrar la propuesta de asignación de forma clara y agrupada por rol, indicando qué equipo específico se sugiere para cada puesto.
- Debe mostrar el costo total estimado de la propuesta.
- Si la propuesta no se puede completar, debe mostrar un mensaje de error claro indicando qué equipos faltan.

Paso 3: El Desafío Central - Tu Lógica de Decisión

El corazón de la prueba reside en la lógica del endpoint GET /api/solicitudes/{id}/propuesta-optima.

- **Tu Algoritmo:** Debes diseñar e implementar un algoritmo que, dados los requerimientos de una solicitud y el inventario disponible, encuentre la "mejor" combinación de equipos.
- **Define "Mejor":** No hay una única respuesta correcta. Tú debes decidir el criterio de optimización y justificarlo. ¿Es el costo total más bajo? ¿Es asignar los equipos de mayor rendimiento posible? ¿Un balance entre ambos?
- **Manejo de Casos Borde:** Tu algoritmo debe manejar elegantemente el caso en que no haya suficientes equipos para satisfacer la demanda, informando exactamente qué falta.

Paso 4: Documentación y Entrega

Deberás entregar:

1. **Código Fuente Completo:** Dos carpetas separadas: /frontend y /backend.
2. **Repositorio Git:** Un único repositorio con todo el código, mostrando un historial de commits claro.
3. **Instrucciones de Ejecución:** Un archivo README.md con instrucciones claras sobre cómo instalar las dependencias y ejecutar tanto el backend como el frontend localmente.
4. **Documento de Justificación (PDF, 2-3 páginas):**
   - **Arquitectura:** Explica las tecnologías que elegiste para el frontend y el backend y por qué. Incluye un diagrama de tu esquema de base de datos.
   - **Algoritmo de Optimización: Esta es la sección más importante.** Describe en detalle el criterio de "optimalidad" que elegiste. Explica cómo funciona tu algoritmo paso a paso, por qué lo consideras una buena solución y cuál es su complejidad computacional (Big O).