Plataforma de Gestión de Recursos de TI

Documento de Justificación Técnica

Luis Diego Coghlan Loredo

28 de Julio, 2025

1. Arquitectura del Sistema

1.1. Tecnologías Seleccionadas

Frontend: React 18 con TypeScript y Vite como build tool.

- **React**: Framework maduro con amplio ecosistema, componentes reutilizables y excelente rendimiento
- **TypeScript**: Tipado estático que reduce errores en tiempo de desarrollo y mejora la mantenibilidad
- **Vite**: Build tool moderno con hot reload rápido y optimización automática para producción

Backend: FastAPI con SQLAlchemy ORM y PostgreSQL.

- **FastAPI**: Framework Python moderno con documentación automática, validación de tipos integrada y alto rendimiento
- **SQLAlchemy**: ORM robusto que permite trabajar con modelos Python mientras mantiene control sobre las consultas SQL
- **PostgreSQL**: Base de datos relacional confiable con soporte completo para integridad referencial y consultas complejas

Justificación: Esta combinación proporciona desarrollo rápido, tipado fuerte en ambos extremos, documentación automática de API, y escalabilidad para entornos empresariales.

2. Diseño de Base de Datos

2.1. Diagrama Entidad-Relación

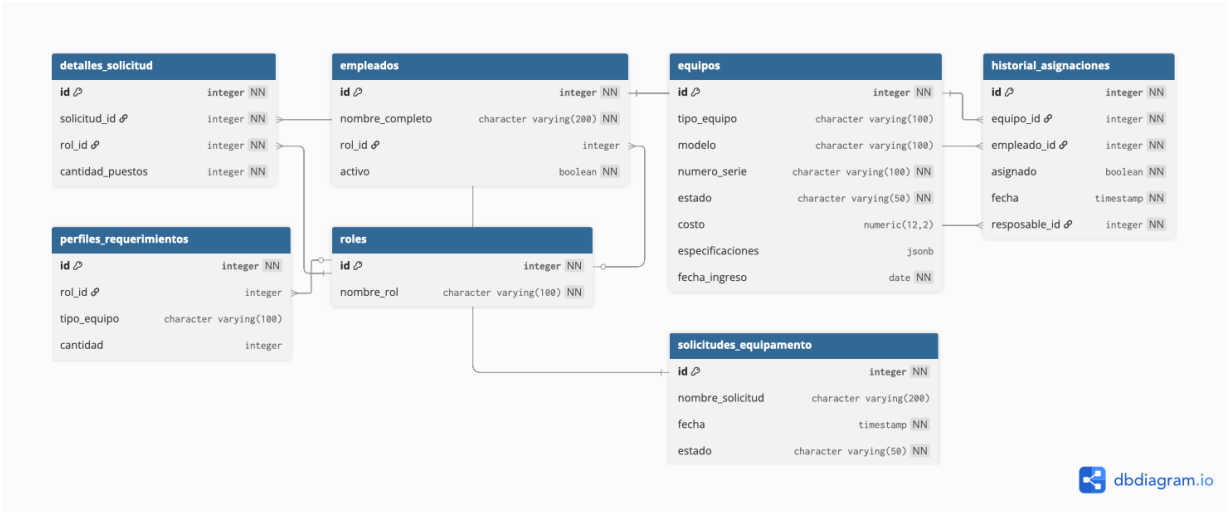

Figura 1: Diagrama Entidad-Relación de la Base de Datos

2.2. Descripción de Entidades

El esquema relacional normalizado incluye las siguientes entidades principales:

- **Equipos**: Inventario con id, tipo_equipo, modelo, numero_serie, estado, costo
- **Roles**: Definición de roles laborales del sistema
- **Perfiles_Requerimientos**: Mapeo crítico rol-equipo para optimización
- **Solicitudes_Equipamento**: Solicitudes principales con metadata
- **Detalles_Solicitud**: Especifica roles y cantidades por solicitud
- **Empleados**: Información de empleados y asignaciones
- **Historial_Asignaciones**: Trazabilidad completa de cambios

3. Algoritmo de Optimización

3.1. Criterio de Optimalidad

El criterio de optimalidad elegido es la **minimización del costo total** de equipos asignados. Esta decisión surge del análisis de múltiples alternativas posibles:

Alternativas Consideradas:

- **Costo Total Más Bajo**: Seleccionar siempre los equipos más económicos disponibles
- **Mayor Rendimiento**: Priorizar equipos con mejores especificaciones técnicas
- **Balance Costo-Rendimiento**: Optimización multiobjetivo considerando ambos factores
- **Antigüedad**: Preferir equipos más nuevos o más antiguos según política de deprecación

Justificación de la Elección: La minimización del costo total fue seleccionada como criterio óptimo por las siguientes razones:

1. **Métrica Objetiva**: El costo es una medida cuantificable y unívoca que elimina ambigüedades en la toma de decisiones
2. **Impacto Presupuestario**: En entornos empresariales, la eficiencia económica es crítica para la sustentabilidad de proyectos
3. **Simplicidad Computacional**: Permite implementación eficiente con complejidad predecible
4. **Auditabilidad**: Los resultados son fácilmente verificables y justificables ante stakeholders
5. **Escalabilidad**: El criterio mantiene relevancia independiente del tamaño del inventario o solicitud

3.2. Funcionamiento del Algoritmo

El algoritmo implementa una estrategia **greedy (codiciosa)** que funciona de la siguiente manera:
Paso 1: Validar la existencia de la solicitud y obtener sus detalles (roles y cantidades)
Paso 2: Para cada rol solicitado:

- Verificar que el rol existe en el sistema
- Obtener los requerimientos de equipos definidos para ese rol
- Calcular la cantidad total de equipos necesarios multiplicando requerimientos por cantidad de puestos

Paso 3: Para cada tipo de equipo requerido:

- Consultar todos los equipos disponibles de ese tipo
- Ordenar los equipos por costo en orden ascendente (decisión greedy)
- Seleccionar la cantidad necesaria de equipos más baratos
- Marcar equipos como asignados para evitar doble asignación

Paso 4: Generar respuesta con asignaciones realizadas, costo total y mensajes de estado

Algorithm 1 Algoritmo de Asignación Óptima

```
1: procedure PropuestaOptima(solicitud_id)
2:     Validar solicitud y obtener detalles
3:     Inicializar propuesta, costo_total = 0, mensajes_error = []
4:     for cada detalle en solicitud do
5:         Validar rol y obtener requerimientos
6:         for cada requerimiento do
7:             Calcular equipos necesarios = cantidad × puestos
8:             Buscar equipos disponibles ordenados por costo
9:             Seleccionar equipos más baratos
10:            if equipos insuficientes then
11:                Registrar faltantes, generar propuesta parcial
12:            end if
13:            Actualizar costo total
14:        end for
15:    end for
16:    return propuesta con costo y mensajes de estado
17: end procedure
```

3.3. Justificación de la Solución

Esta solución es óptima por las siguientes razones:

1. **Minimización Garantizada**: Al seleccionar siempre los equipos de menor costo disponibles, se garantiza el mínimo costo total posible para cada asignación
2. **Factibilidad**: Solo asigna equipos que realmente están disponibles, evitando conflictos
3. **Completitud**: Procesa todos los roles solicitados sin excepciones
4. **Determinismo**: Produce resultados consistentes para las mismas entradas
5. **Manejo de Restricciones**: Genera propuestas parciales cuando no hay suficiente inventario, proporcionando la mejor asignación posible

3.4. Complejidad Computacional (Big O)

Complejidad Temporal: $O(n \times r \times e \log e)$
Donde:

- n = número de roles diferentes en la solicitud
- r = número promedio de tipos de equipos requeridos por rol
- e = número promedio de equipos disponibles por tipo

Desglose:

- Iteración sobre roles: $O(n)$
- Iteración sobre requerimientos por rol: $O(r)$
- Consulta de equipos disponibles: $O(e)$
- Ordenamiento por costo (operación dominante): $O(e \log e)$

Complejidad Espacial: $O(n \times r \times k)$ donde k es el número promedio de equipos seleccionados por tipo.
Esta complejidad es eficiente para casos de uso empresariales típicos y escala bien con el crecimiento del inventario.