Prompt para el Agente de IA - Desarrollo de Aplicación Web de Inventario y Activos IT

Rol y Contexto Principal:

Actúa como un Ingeniero de Sistemas Senior, con profunda experiencia en desarrollo full-stack utilizando **Node.js** para el backend, **React** para el frontend, **JavaScript** (ES6+), y bases de datos **SQL Server**. Posees también una sólida experiencia en diseño de interfaces de usuario (UI) y experiencia de usuario (UX), enfocado en aplicaciones de gestión interna.

Objetivo del Proyecto:

Desarrollar una aplicación web robusta y fácil de usar para la gestión de inventario y control de activos fijos del sector de Soporte IT de una empresa. Es crucial entender que este sector **no realiza compras**, sino que gestiona los activos desde su recepción hasta su asignación, reparación y eventual baja. La aplicación será utilizada internamente por un equipo de 8-10 personas.

Requisitos Funcionales Detallados:

1. **Gestión de Usuarios y Permisos:**
    - Implementar un sistema de autenticación basado en usuarios y contraseñas.
    - Cada miembro del equipo (8-10 personas) tendrá una cuenta de usuario estándar.
    - Debe existir al menos un rol de **Administrador** con permisos totales sobre la aplicación (gestión de usuarios, configuración, gestión de catálogo de productos, etc.).
    - Cada acción crítica (entrada, salida, modificación, inicio de reparación, finalización de reparación, baja) debe registrar qué usuario la realizó y la fecha/hora. Esta información debe ser claramente visible en los historiales y detalles del activo.
    - Los usuarios deben poder ver su perfil y cambiar su propia contraseña. Incluir una sección de "Recordatorios" o "Tareas Pendientes" en el perfil podría ser útil (considerar viabilidad).
2. **Gestión de Activos y Catálogo:**
    - **Categorización:** Implementar un sistema de categorización jerárquica para los activos. La propuesta inicial es:
        - **1. Computadoras:** Desktops, Notebooks, Raspberry Pi
        - **2. Celulares:** (Sin subcategorías)
        - **3. Periféricos:** Teclados, Mouse, Kit Teclado/Mouse, Auriculares, Webcams, Monitores, Televisores, etc.
        - **4. Consumibles:** Cables (diversos tipos), Pilas, Toner, Drum (Unidad de Imagen), Cargadores (Celular, Notebook), etc.

        - **5. Componentes:** Memorias RAM, Discos Externos, Discos SSD/NVMe, Placas Sending (especificar tipo si es necesario), Placas de Video, Motherboards, Adaptadores USB Varios, etc.
        - **Tarea para la IA:** Revisa esta categorización. ¿Es lógica y completa? ¿Sugieres alguna mejora, fusión o adición de categorías/subcategorías para optimizar la gestión y los reportes?
    - **Catálogo de Productos:** El Administrador debe tener un módulo específico para dar de alta y gestionar los *tipos* de productos que pueden ingresar al inventario (ej. "Notebook Dell Latitude 7400", "Toner HP 85A"). Debe poder definir marca, modelo, descripción y a qué categoría/subcategoría pertenece.
    - **Entrada de Stock (Alta):**
        - Formulario para registrar la entrada de nuevos activos al inventario.
        - Los usuarios seleccionarán el producto del catálogo predefinido por el Admin.
        - Se debe poder registrar la cantidad (para consumibles o periféricos idénticos) o números de serie individuales (para computadoras, celulares, monitores, etc.).
        - Registrar fecha de recepción.
        - **Operaciones en Lote:** Permitir el alta de múltiples unidades idénticas (ej. 20 mouses iguales) o múltiples activos diferentes en una sola operación para agilizar la carga inicial o grandes recepciones.
    - **Salida de Stock (Asignación):**
        - Formulario para registrar la salida/asignación de activos.
        - Debe indicarse el destino: Empleado específico, Sucursal, Sector, Obra Nueva.
        - **Seguimiento Especial - Notebooks:** Al asignar una Notebook a un *empleado*, se debe registrar obligatoriamente una **contraseña de encriptación** (ej. Bitlocker). Esta contraseña debe ser visible en los detalles del activo asignado *únicamente* para los usuarios de la aplicación (personal de IT).
        - **Seguimiento Especial - Celulares:** Al asignar un Celular (a *empleado*, *sucursal* o *sector*), se debe registrar obligatoriamente: Número de Teléfono asociado, Cuenta de Gmail, Contraseña de Gmail, Código de Verificación de 2 Pasos de WhatsApp (6 dígitos). Estos datos deben ser visibles en los detalles del activo asignado *únicamente* para los usuarios de la aplicación.
        - Debe quedar un historial claro de asignaciones para cada activo serializable.

        - **Operaciones en Lote:** Permitir la asignación de múltiples activos (ej. 5 monitores a una nueva sucursal) en una sola operación.

3. **Gestión de Reparaciones:**
    - Formulario para enviar un activo a reparación.
    - Campos requeridos: Fecha de envío a reparación, Proveedor/Lugar de reparación, Descripción detallada del problema/falla.
    - El estado del activo debe cambiar a "En Reparación".
    - Al regresar de reparación, se debe actualizar el estado:
        - **Reparado:** El activo vuelve al stock general disponible (no necesariamente al usuario/lugar original). Registrar fecha de retorno y breve descripción de la reparación efectuada.
        - **Sin Reparación / Baja:** Marcar el activo como irreparable o dado de baja. Registrar fecha y motivo.
    - **Historial de Reparaciones:** Cada activo serializable debe tener un historial completo de sus reparaciones (fechas, problemas, soluciones, proveedor). Esto es vital para detectar fallos recurrentes.
4. **Gestión de Consumibles:**
    - **Umbrales de Stock Mínimo:** Para cada tipo de consumible en el catálogo (definido por el Admin), permitir establecer un umbral de stock mínimo.
    - **Alertas:** Cuando la cantidad disponible de un consumible caiga por debajo de su umbral, generar una alerta visible en el Dashboard o en una sección específica de notificaciones.
5. **Búsqueda y Filtrado:**
    - Implementar una funcionalidad de búsqueda potente en todas las vistas de listado de activos (Inventario general, Asignados, En Reparación, etc.).
    - Permitir buscar por: Número de serie, Categoría, Subcategoría, Marca, Modelo, Estado (Stock, Asignado, En Reparación, Baja), Empleado asignado, Sucursal/Sector asignado.
    - Los filtros deben poder combinarse (ej. buscar Notebooks Dell en estado "Stock").
6. **Reportes:**
    - Generar reportes personalizables sobre el inventario. Ejemplos:
        - Listado completo de activos con todos sus detalles.
        - Activos por categoría/subcategoría.
        - Activos asignados por empleado/sucursal/sector.
        - Activos en reparación.

        - Activos dados de baja (en un período).
        - Historial de un activo específico.
        - Niveles de stock de consumibles (resaltando los que están bajo el mínimo).
    - Permitir la descarga de estos reportes y de cualquier vista de listado en formatos **PDF, CSV y Excel**.
7. **Dashboard Principal:**
    - Crear una pantalla de inicio (Dashboard) que muestre información clave de un vistazo:
        - Widgets con: Cantidad total de activos, número de activos por categoría principal, activos asignados, activos en reparación.
        - Listado/Alerta de consumibles por debajo del umbral mínimo.
        - Quizás accesos directos a acciones comunes (Nueva Entrada, Nueva Salida, Enviar a Reparación).
        - Gráficos simples (ej. Torta de activos por categoría) si es viable.

Requisitos No Funcionales:

1. **Tecnología:**
    - Backend: **Node.js** (Especificar versión si es relevante, ej. LTS actual).
    - Frontend: **React** (Especificar versión si es relevante, ej. v18+).
    - Lenguaje: **JavaScript** (ES6+).
    - Base de Datos: **Microsoft SQL Server**.
    - Considerar librerías estándar y bien mantenidas para funcionalidades clave (ej. gestión de estado, routing, UI components, generación de PDF/Excel).
2. **Diseño Visual (UI):**
    - El diseño visual específico (colores, tipografía, layout general, logo) se proporcionará en un documento/guía de estilo aparte. **Es mandatorio adherirse estrictamente a estas especificaciones visuales.**
    - La aplicación debe ser intuitiva, limpia y fácil de navegar para usuarios técnicos (el equipo de IT). Priorizar la funcionalidad y claridad sobre la complejidad visual innecesaria.
3. **Seguridad:**
    - Asegurar la autenticación y proteger las rutas.
    - Aunque se solicita almacenar contraseñas (Gmail, Encriptación) en texto visible para el equipo IT interno, implementar las mejores prácticas posibles dentro de este requisito (ej. asegurar que el acceso a la base de

datos y a la aplicación esté fuertemente restringido). **Validar que este requisito es absoluto, ya que almacenar credenciales en texto plano es una práctica de seguridad muy riesgosa.**

- Prevenir vulnerabilidades comunes (XSS, CSRF, SQL Injection).

4. **Usabilidad:**
    - La aplicación debe ser responsiva (adaptarse a diferentes tamaños de pantalla, al menos en escritorio).
    - Los formularios deben tener validaciones claras (campos obligatorios, formatos correctos).
    - Mensajes de error y éxito claros para el usuario.
5. **Mantenibilidad:**
    - Generar código limpio, bien comentado y estructurado siguiendo buenas prácticas de desarrollo en Node.js y React.

Entregables Esperados:

- Código fuente completo del backend y frontend.
- Scripts SQL para la creación de la base de datos y tablas.
- Instrucciones claras para la configuración y despliegue de la aplicación.

Preguntas / Aclaraciones:

Si algún requisito no está claro o si tienes sugerencias para mejorar la funcionalidad o el enfoque técnico, por favor, indícalo.