




Flujo de Trabajo y Dependencias - Krypton Inventory

Este es el orden lógico para abordar el desarrollo. Una tarea no puede comenzar hasta que la tarea de la que depende (su "bloqueador") esté completada.





Fase 1: Cimientos del Backend y UI Básica (Tareas Urgentes)

Esta es la base de todo. Sin la base de datos y la autenticación, no hay aplicación.

1.  **(BLOQUEADOR INICIAL) Login de Trabajador (Backend)**
 - **Descripción:** Se debe crear la lógica para que un usuario pueda iniciar sesión. Esto implica conectar a la base de datos y verificar credenciales.
 - **¿Por qué es primero?** Porque sin poder iniciar sesión, ninguna otra funcionalidad interna (dashboard, perfil, tareas) puede ser probada o utilizada.
2.  **(Depende de #1) Manejo de Sesiones (Backend)**
 - **Descripción:** Una vez que el usuario inicia sesión, el sistema debe "recordarlo" mientras navega.
 - **¿Por qué sigue?** Para proteger las páginas como worker.php o dashboard.php y asegurar que solo usuarios autenticados puedan acceder.
3.  **(Depende de #2) Sistema de Notificaciones (JS)**
 - **Descripción:** Crear la función JS para mostrar mensajes de "éxito" o "error".
 - **¿Por qué ahora?** El frontend necesita una forma de comunicar al usuario el resultado del login (ej: "Credenciales incorrectas" o "Bienvenido"). Esta es la primera interacción real entre backend y frontend.

Fase 2: El Corazón de la Aplicación - Gestión de Tareas (Tareas y)

Ahora que el trabajador puede entrar, le damos su herramienta principal: el tablero de tareas.

4.  **(BLOQUEADOR) Cargar Tareas en el Tablero (Backend)**
 - **Descripción:** Crear el script PHP que lee las tareas de la base de datos y las devuelve.
 - **¿Por qué es primero en esta fase?** El desarrollador de frontend necesita recibir datos reales para poder construir la interfaz que las muestra. No se puede pintar una tarjeta de tarea si no hay datos de la tarea.
5.  **(Depende de #4) Cargar Tareas en el Tablero (Frontend)**
 - **Descripción:** Usar JS para llamar al script PHP del paso anterior y dibujar dinámicamente las tarjetas de tareas en el tablero Kanban.
6.  **(Depende de #3) Sistema de Modales (JS)**
 - **Descripción:** Crear la función para mostrar y ocultar modales.
 - **¿Por qué ahora?** Se necesita para poder abrir el formulario de "Nueva Tarea".
7.  **(Depende de #6) Crear y Guardar Tareas (Backend)**
 - **Descripción:** Script PHP que recibe los datos del modal y los guarda en la base de datos.

- **¿Por qué depende del modal?** Porque el backend necesita recibir los datos que el modal del frontend le enviará.
- 8. 🚀 **(Depende de #7) Crear y Guardar Tareas (Frontend)**
 - **Descripción:** Conectar el formulario del modal para que, al hacer clic en "Guardar", envíe los datos al script PHP del paso anterior y muestre un toast de confirmación.
- 9. 🐘 **(Depende de #8) Editar y Eliminar Tareas (Backend)**
 - **Descripción:** Scripts PHP para actualizar o borrar una tarea en la base de datos.
- 10. 🚀 **(Depende de #9) Editar y Eliminar Tareas (Frontend)**
 - **Descripción:** Lógica JS para los botones de cada tarjeta, que abren el modal con la información de la tarea y se comunican con el backend.
- 11. 🐘 **(Depende de #10) Arrastrar y Soltar (Backend)**
 - **Descripción:** Un script PHP específico que solo actualiza el **estado** de una tarea (pendiente, en progreso, etc.).
- 12. 🚀 **(Depende de #11) Arrastrar y Soltar (Frontend)**
 - **Descripción:** Implementar la lógica visual de drag & drop y, al soltar una tarea, llamar al script PHP del paso anterior para guardar el cambio.

Fase 3: Funcionalidades Complementarias (Tareas 🟡 y 🟢)

Con el núcleo funcionando, ahora podemos construir las demás páginas que dependen de los datos generados.

- 13. 🚀 **(BLOQUEADOR) Captura de Firmas (JS)**
 - **Descripción:** Lógica del <canvas> para dibujar.
- 14. 🐘 **(Depende de #13) Registro de Equipo (Backend)**
 - **Descripción:** Guardar toda la información del dashboard.php.
 - **¿Por qué depende de las firmas?** Porque el backend debe estar preparado para recibir los datos del canvas que el frontend le enviará.
- 15. 🐘 **(Depende de #12 y #14) Visualización Dinámica de Estado (PHP)**
 - **Descripción:** La página estado.php debe leer la información del equipo y el historial de tareas.
 - **¿Por qué al final?** Porque esta página es de "solo lectura". Depende de que todas las demás funcionalidades (registro de equipos, creación y actualización de tareas) ya existan para tener datos que mostrar.