

Flujo de navegación y datos

Este diagrama muestra cómo interactúan los componentes de Matricería entre sí y con la API. Refleja el flujo de datos desde la carga inicial, la visualización en tabla y la creación de nuevas Órdenes de Producción.

OPsMatriceria.tsx

Componente principal que orquesta la **carga**, **filtro** y **navegación** entre el listado de OPS y el formulario de creación.

- Usa hooks de configuración, usuario y filtrado.
- Controla estado de:
- centrosDeCosto
- datosLanzamientoDeOp
- matrices
- tiposMatrices
- mostrarAgrega
- Renderiza:
- Botón para alternar el formulario (si nivel de usuario >1)
- Inputs de filtro para OP y código de matriz
- Tabla de resultados (OPsMatriceriaTabla)
- Formulario de creación (OPsMatriceriaAgregar)
- Gestiona estado de carga con Cargando.



Figura 1 – Pantalla principal de OP de Matricería con listado, filtros y acciones de Lanzar.

Uso de endpoints

GET Listado inicial de Matricería

Lógica y posibles mejoras

- El filtro de descripcion3 (Producto) está siempre deshabilitado. Evaluar si es necesario o eliminarlo.

- `useEffect` sin dependencias carga datos solo al montar. Adecuado si no hay recarga dinámica.
- Filtrado en `useMemo` convierte `descripcion2` a minúsculas. No afecta IDs numéricos.

OPsMatriceriaTabla.tsx

Componente de presentación que muestra la lista de Órdenes de Producción en una tabla.

- Propiedades:
- `datosLanzamientoDeOp`: array o `null`.
- Funcionalidades:
- Genera estilos de encabezado si hay datos.
- Ordena por `id` descendente.
- Formatea fecha con `formatearFecha`.
- Muestra código de máquina con `codMaquina1`.
- Indica estado con iconos y colores:
- **Terminada** (verde) si `estadoId === 3`
- **Detenida** (rojo) si `estadoId === 2`
- **No terminada** (amarillo) en otro caso
- Si no hay elementos, renderiza `<NoHayElementos />`.

Lógica y posibles mejoras

- El `map` no incluye acciones ni checkbox, pese a que `titulosDeTabla` sugiere “Seleccionar” y “Acciones”. Revisar si se debe implementar.
- Código comentado para detalle modal puede removerse si no se usa.
- Considerar memoización si la lista es muy grande.

OPsMatriceriaAgregar.tsx

Formulario para **crear** una nueva Orden de Producción de Matricería.

- Props:
- `centrosDeCosto`
- `matrices`
- `tiposMatrices`
- `setDatosPlanificacionOp`
- `setMostrarAgregar`
- Estados internos:
- `centroDeCostoId`, `tipoId`, `numero`, `descripcion`, `marca`
- `fechaObjetivo`
- `guardando` (flag de envío)
- Validaciones con `useMemo`:

- yaExiste: verifica que no haya otra matriz con mismo tipoId y numero.
- esValido: campos obligatorios y !yaExiste.
- Usa hook useTeclaEnter para disparar crearOp al presionar Enter.

Figura 2 – Formulario de creación de una nueva Orden de Producción de Matricería con carga de Centro de Costo, Tipo de matriz, Número, Descripción, Marca y fecha objetivo.

Uso de endpoint

POST Crear nueva OP Matricería

Lógica y posibles mejoras

- El umbral `tipoId > 55` para considerar válido puede excluir tipos con ID menores. Revisar si corresponde.
- `numero` parseado con `parseInt` puede generar NaN en entrada vacía; validar con `isNaN`.
- Buena práctica: manejar mensajes de error específicos junto a cada campo.

Nota

Asegurar uniformidad en IDs de tipo de matriz antes de validar con thresholds numéricos.

Resumen de posibles fallas detectadas:

- Clave de encabezados de tabla (Seleccionar, Acciones) no implementada en `OPsMatriceriaTabla`.
- Lógica de validación en `OPsMatriceriaAgregar` podría descartar opciones legítimas por condición `tipoId > 55`.
- Input de Producto (`descripcion3`) sin uso en filtros.
- Falta manejo de errores de validación de número (`parseInt` no controlado).

Con estas observaciones, el flujo y las interacciones quedan bien documentados y se identifican puntos de mejora para optimizar la UX y la robustez del código.