# Documentación de componentes de MRP

Esta guía describe cuatro componentes React para gestionar el **Registro MRP**.

Incluye su propósito, flujo de datos, hooks y endpoints utilizados.

## Diagrama de flujo de navegación y datos

Este diagrama muestra cómo interactúan los componentes y las llamadas a la API.

Refleja la navegación desde el listado al detalle y el flujo de datos entre capas.

## RegistrosMrp.tsx

Este componente gestiona y muestra el **listado de registros MRP**.

Carga datos, controla estado de carga y maneja navegación.

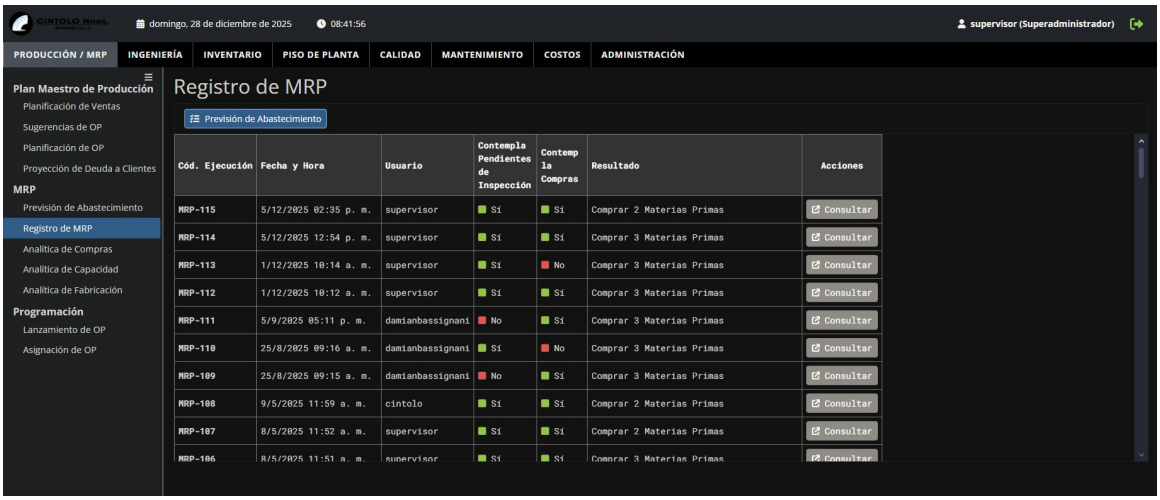


Figura 1 – Pantalla principal de Registro de MRP con listado de registros MRP.

## Principales responsabilidades

- Obtener la lista de registros con `hacerPeticionApi`.
- Controlar estado de carga (`config.cargando`).
- Mostrar errores con `useMostrarError`.
- Navegar a la **Previsión de Abastecimiento** si el usuario tiene acceso.

## Hooks y utilidades usadas

- **useState** para `datosRegistroMrp`.
- **useEffect** para la petición inicial.
- **useAcceder** para validación y navegación.
- **useTraerConfiguracion** para estado global de carga.
- **useMostrarError** para alertas.

## Endpoints utilizados

| Endpoint | Método | Descripción |
|---|---|---|
| GET /produccion/previsionabastecimiento | GET | Obtiene arreglo `datosRegistrosMrp`. |

## Flujo de datos

- Al montarse, dispara `hacerPeticionApi()`.
- Desactiva el spinner: `config.setCargando(false)`.
- Extrae `respuesta.datos.datosRegistrosMrp`.
- Si no hay datos, invoca `mostrarError(respuesta)`.
- Al recibir datos, actualiza `setDatosRegistroMrp`.

## Fragmento de código clave

**tsx**

```
useEffect(() => {
  hacerPeticionApi().then(respuesta => {
    config.setCargando(false);
    const drmrp = respuesta.datos?.datosRegistrosMrp as registroMrpType[];
    if (!drmrp) {
      mostrarError(respuesta);
      return;
    }
    setDatosRegistroMrp(drmrp);
  });
}, []);
```

# RegistrosMrpTabla.tsx

Este componente presenta la **tabla de registros MRP** recibidos.

No realiza peticiones; solo despliega la información formateada.

## Principales responsabilidades

- Recibir `datosRegistroMrp` como prop.
- Ordenar registros por ID descendente.
- Formatear fecha con `formatearFecha`.
- Aplicar estilos al encabezado con `generarEstilosDeCabezaDeTabla`.
- Ofrecer botón de **Consultar** si el usuario tiene acceso.

## Hooks y utilidades usadas

- **useAcceder** para controlar el acceso a detalles.
- **formatearFecha** para mostrar fechas legibles.
- **generarEstilosDeCabezaDeTabla** para estilos CSS dinámicos.

## Esquema de columnas

| Columna | Descripción |
|---|---|
| Cód. Ejecución | Muestra `MRP-{id}` |
| Fecha y Hora | Fecha de alta (día, hora). |
| Usuario | Usuario que creó el registro. |
| Contempla Pendientes de Inspección | Indicador Sí/No. |
| Contempla Compras | Indicador Sí/No. |
| Resultado | Texto con sugerencia de compra. |
| Acciones | Botón **Consultar** hacia detalle. |

## Ejemplo de fila

**tsx**

```
<tr key={r.id}>
  <td className='fw-bold'>MRP-{r.id}</td>
  <td>{formatearFecha(r.altaFechaHora, true, true)}</td>
  <td>{r.altaUsuarioUsername}</td>
  <td>
    {r.contemplaPendientesInspeccion
      ? <><i className='fas fa-square text-success me-2'/>Sí</>
      : <><i className='fas fa-square text-danger me-2'/>No</>}
  </td>
  {/* ... */}
  <td className='text-center-d'>
    {acceder.tieneAcceso() && (
      <button
        className='btn btn-sm btn-secondary'
        onClick={e => acceder.navegarSiTieneAcceso(`detalle?id=${r.id}`, e)}
      >
        <i className='fas fa-external-link me-2'/>Consultar
      </button>
    )}
  </td>
</tr>
```

# RegistroMrp.tsx

Este componente muestra el **detalle de un registro MRP** específico.

Carga datos individuales, ofrece volver atrás e impresión de tabla.

## Principales responsabilidades

- Obtener `registroMrp` según `id` en query params.
- Controlar carga con `useTraerConfiguracion`.
- Invocar `useImprimirTabla` para función de impresión.

- Navegar con `useNavegar` al listado.
- Mostrar errores si no encuentra datos.

## Hooks y utilidades usadas

- **useRef** para referenciar la tabla a imprimir.
- **useState** para almacenar `mrp`.
- **useEffect** para petición de detalle.
- **useImprimirTabla** para exportar la tabla.
- **useMostrarError** para alertas.
- **useNavegar** para retorno al listado.

## Endpoints utilizados

| Endpoint | Método | Descripción |
|---|---|---|
| GET /produccion/previsionabastecimiento/detalle?id={id} | GET | Obtiene objeto `registroMrp`. |

## Datos mostrados

- **Fecha** y **Hora** de alta.
- **Usuario** que generó el registro.
- **Stock** a contemplar y **Previsión de Compras**.
- Tabla de líneas con detalle de materias primas.

## Fragmento de código clave

**tsx**

```
useEffect(() => {
  hacerPeticionApi().then(respuesta => {
    config.setCargando(false);
    const rmrp = respuesta.datos?.registroMrp as typeof mrp;
    if (!rmrp) {
      mostrarError(respuesta);
      return;
    }
    setMrp(rmrp);
  });
}, []);
```

Nota: El componente se encuentra definido a nivel de frontend, pero su funcionamiento es parcial y no permite la visualización efectiva del detalle.

# RegistroMrpTabla.tsx

Se encarga de renderizar la **tabla de líneas** de un registro MRP.

Filtra solo líneas con `cantidadDemandada > 0` y aplica estilos.

## Principales responsabilidades

- Recibir `mrp` y `tablaRef` como props.
- Generar estilos para el encabezado.
- Mostrar datos de cada línea: artículo, cantidades y sugerida.
- Mostrar `NoHayElementos` si no hay líneas.

## Hooks y utilidades usadas

- **generarEstilosDeCabezaDeTabla** para insertar CSS.
- **NoHayElementos** para placeholder de tabla vacía.

## Columnas detalladas

| Columna | Campo asociado |
| --- | --- |
| Materia Prima | `articuloId`, `articuloDescripcion`, `articuloCod` |
| Cantidad demandada | `cantidadDemandada` |
| Cantidad disponible | `cantidadDisponible` |
| Cantidad pendiente de inspeccion | `cantidadPendienteInspeccion` |
| Cantidad pendiente de recepcion | `cantidadPendienteRecepcion` |
| Cantidad sugerida | `cantidadSugerida` |

## Ejemplo de mapeo

**tsx**

```
mrp.lineas
  .filter(x => x.cantidadDemandada > 0)
  .map(ab => (
    <tr key={ab.id}>
      <td className='text-nowrap'>
        {ab.articuloId} {ab.articuloDescripcion}<br/>
        {ab.articuloCod}
      </td>
      <td className='text-end-d'>{ab.cantidadDemandada}</td>
      {/* ... */}
      <td className='text-end-d'>{ab.cantidadSugerida}</td>
    </tr>
  ))
```

### Manejo de Errores

Cada componente muestra errores con useMostrarError si la respuesta no incluye datos válidos.