

Diagrama de Flujo de Navegación y Datos

Este diagrama muestra cómo fluyen la navegación y los datos entre los componentes principales y la API.

EstructuraDeCostos.tsx

Componente principal que carga y muestra la lista de estructuras de costos.

Utiliza hooks personalizados para configuración, usuario, descarga y errores.

The screenshot shows a web application interface for managing cost structures. At the top, there's a navigation bar with tabs: PRODUCCIÓN / MRP, INGENIERIA, INVENTARIO, PISO DE PLANTA, CALIDAD, MANTENIMIENTO, COSTOS (which is currently selected), and ADMINISTRACIÓN. Below the navigation bar, there's a sidebar with links like Costos, Libro Diario, Saldos, Parámetros, Ejercicios Fiscales, Plan de Cuentas, Categorización de Costos, and Estructura de Costos. The main content area is titled "Estructura de Costos" and contains a table with the following data:

Descripción	Costo Fab.	Origen	Valor
Materia Prima	SE	Costo STD	FT / Cant. MP
Mano de Obra Directa	SE	Costo STD	FT / Cant. MP
Mano de Obra Indirecta	SE	Tarifas por Sector	FT / Horas 100%
CIF Fijos	SE	Tarifas por Sector	FT / Horas 100%
CIF Variables	SE	Tarifas por Sector	FT / Horas 20%
Placas	SE	Ind. Fab. Variables	FT / Cant. Prod.
Rolado	SE	Ind. Fab. Variables	FT / Peso Neto
Porcentaje de Administración 11%	No	Porcentaje	11%
Porcentaje de Comercialización 18%	No	Porcentaje	18%
Porcentaje de Comercialización 30%	No	Porcentaje	30%

Figura 1 – Pantalla principal de Estructura de Costos con listado y acciones de alta y exportación.

- **Hooks usados:** useState, useEffect, useTraerConfiguracion, useTraerUsuario, useDescargarExcel, useMostrarError.
- **Componentes hijos:** EstructuraDeCostosTabla, CrearCategoria (modal).
- **Flujo:** monta → fetch datos → render tabla → acciones de UI.

tsx

```
useEffect(() => {
  hacerPeticionApi().then(respuesta => {
    config.setCargando(false);
    const cv = respuesta.datos?.estructurasDeCostos as typeof estructurasCostos;
    if (!cv) {
      mostrarError(respuesta);
      return;
    }
    setEstructurasCostos(cv);
  });
});
```

```
}, []);
```

Endpoints utilizados

- **GET /estructuras-costos** (obtención de lista).

GET /estructuras-costos

⚠ Fallas en la lógica

- No refresca la lista tras crear una nueva categoría.
- El botón **Editar** está siempre deshabilitado y sin handler.
- Se pasan 3 títulos a generarEstilosDeCabezaDeTabla, pero la tabla tiene 5 columnas.
- No maneja errores de red antes de ocultar el spinner.
- Falta cancelar la petición en el desmontaje del componente.

CrearCategoria.tsx

Modal funcional para crear una nueva categoría.

Controla el estado del campo **descripcion** y llama a la API al enviar el formulario.

Descripción	Costo Fab.	Origen	Valor
Materia Prima	SI	Costo STD	FT / Cant. MP
Mano de Obra Directa	SI	Costo STD	FT / Cant. MP
Mano de Obra Indirecta	SI	Tarifas por Sector	FT / Horas 100%
CIF Fijos	SI	Tarifas por Sector	FT / Horas 100%
CIF Variables	SI	Tarifas por Sector	FT / Horas 2%
Placas	SI	Ind. Fab. Variables	FT / Cant. Prod.
Rolado	SI	Ind. Fab. Variables	FT / Peso Neto
Porcentaje de Administración 11%	NO	Porcentaje	11%
Porcentaje de Comercialización 18%	NO	Porcentaje	18%
Porcentaje de Comercialización 30%	NO	Porcentaje	30%

Figura 2 – Formulario de alta de una nueva Estructura de Costos con carga de Descripción.

- **Props:**

Prop	Tipo	Descripción
onClose	() => void	Cierra el modal de creación

- **Estado interno:** descripcion: string.
- **Hook usado:** useMostrarError para notificar fallos.

tsx

```

const handleSubmit = async (e: FormEvent<HTMLFormElement>) => {
  e.preventDefault();
  const response = await hacerPeticionApi({
    method: 'POST',
    url: '/categorias',
    body: { descripcion, estructuras: [] }
  });
  if (response.error) {
    mostrarError(response);
    return;
  }
  onClose();
}

```

Endpoints utilizados

- **POST /categorias** (crea una nueva categoría).

POST /categorias

⚠ Fallas en la lógica

- No muestra un **loading** al enviar el formulario.
- No valida duplicados o longitud mínima de descripción.
- No comunica al padre que recargue la lista tras creación.

EstructuraDeCostosTabla.tsx

Presenta los datos de estructurasCostos en una tabla.

Permite mostrar un badge según esFabril y un botón de edición según permisos.

- **Props:**

	Prop	Tipo	Descripción
estructurasCostos	`estructuraCostoTyp e[]`	null`	Datos a mostrar en la tabla

- **Hooks usados:** useTraerUsuario.
- **Servicios:** tienePermiso, generarEstilosDeCabezaDeTabla.
- **Componente fallback:** NoHayElementos si no hay datos.

tsx

```

{!!estructurasCostos?.length
? estructurasCostos.map(e => (
  <tr key={e.id}>
    <td className='text-center-d>
      {tienePermiso(usuario, e) && (
        <button className='btn btn-sm btn-secondary' disabled>
          <i className='fas fa-edit' />
        </button>
      )}
    </td>
    <td>{e.descripcion}</td>
    <td>
      {e.esFabril ? (

```

```
        <span className='badge bg-success'>Sí</span>
    ) : (
        <span className='badge bg-danger'>No</span>
    )
</td>
<td>{e.origen}</td>
<td>{e.valor}</td>
</tr>
))
: <NoHayElementos />}
```

Endpoints utilizados

Este componente no realiza llamadas directas a la API.

⚠ Fallas en la lógica

- **titulosDeTabla** (3 elementos) no coincide con las 5 columnas reales.
- El botón de **editar** siempre aparece deshabilitado; falta implementación.
- No ofrece funcionalidades de ordenación, filtrado o paginación.