

## Diagrama de flujo de navegación y datos

Este diagrama muestra la interacción entre la UI, los hooks personalizados, las peticiones API y los componentes de tabla. Ilustra el flujo de datos y la navegación en las pantallas de **Plan de Cuentas** y **SalDOS**.

### PlanDeCuentasTabla.tsx

Este componente recibe cuentas contables y página actual. Aplica estilos, paginación y ordena por código.

- **Props**
- `cuentasContables: cuentaContableType[]`
- `pagina: number`
- **Servicios utilizados**
- `generarEstilosDeCabezaDeTabla(titulos: string[])`
- `paginado(lista, pagina): cuentaContableType[]`
- **Flujo de datos**
- Inyecta estilos de cabeza de tabla.
- Aplica paginado y `.sort` por código.
- Renderiza filas o `<NoHayElementos />`.
- **Endpoints**
- No realiza peticiones; recibe datos desde **PlanDeCuentas**.

### Fragmento clave: sangrado según nivel de código

tsx

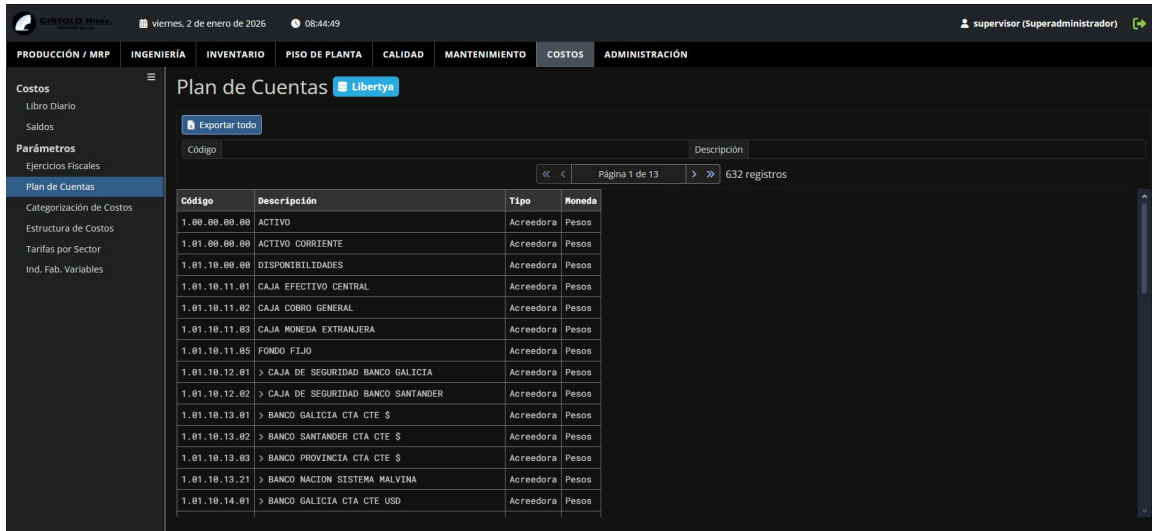
```
const estiloMargen = (codigo: number): ReactNode => {
  const largo = codigo.toString().length;
  let rep = 0;
  if (largo === 2) rep = 1;
  else if (largo === 3) rep = 2;
  else if (largo === 4) rep = 3;
  else if (largo === 6) rep = 4;
  else if ([7,8,9,10,11,12].includes(largo)) rep = 5;
  return rep === 0
    ? <Fragment />
    : <span className='text-muted' style={{fontSize:'0.75em'}}>
      {Array(rep).fill(<i className='fas fa-chevron-right me-2' />)}
    </span>;
};
```

⚠ Falla en la lógica

El sangrado se calcula con código: `number`, pero el componente usa `estiloMargen(l.id)` en vez de `l.codigo`. Esto aplica el margen sobre el **ID** en lugar del **código contable**.

## PlanDeCuentas.tsx

Este componente aglutina filtros, paginación, descarga de Excel y entrega datos a la tabla de Plan de Cuentas.



Código	Descripción	Tipo	Moneda
1.00.00.00.00	ACTIVO	Acreedora	Pesos
1.01.00.00.00	ACTIVO CORRIENTE	Acreedora	Pesos
1.01.10.00.00	DISPONIBILIDADES	Acreedora	Pesos
1.01.10.11.01	CAJA EFECTIVO CENTRAL	Acreedora	Pesos
1.01.10.11.02	CAJA COBRO GENERAL	Acreedora	Pesos
1.01.10.11.03	CAJA MONEDA EXTRANJERA	Acreedora	Pesos
1.01.10.11.05	FONDO FIJO	Acreedora	Pesos
1.01.10.12.01	> CAJA DE SEGURIDAD BANCO GALICIA	Acreedora	Pesos
1.01.10.12.02	> CAJA DE SEGURIDAD BANCO SANTANDER	Acreedora	Pesos
1.01.10.13.01	> BANCO GALICIA CTA CTE \$	Acreedora	Pesos
1.01.10.13.02	> BANCO SANTANDER CTA CTE \$	Acreedora	Pesos
1.01.10.13.03	> BANCO PROVINCIA CTA CTE \$	Acreedora	Pesos
1.01.10.13.21	> BANCO NACION SISTEMA MALVINA	Acreedora	Pesos
1.01.10.14.01	> BANCO GALICIA CTA CTE USD	Acreedora	Pesos

Figura 1 – Pantalla principal de Plan de Cuentas con listado, filtros y acciones de exportación.

- **Hooks personalizados**
- `useTraerConfiguracion()` → controla estado de carga
- `useDescargarExcel()` → descarga datos en Excel
- `useFiltrar()` → gestiona filtros de texto
- `useMostrarError()` → muestra errores de API
- **Componentes internos**
- `<Cargando />`
- `<ControlPaginas />`
- `<PlanDeCuentasTabla />`
- **Endpoints**
- **GET /api/cuentasContables** → `hacerPeticionApi()` devuelve { datos: { cuentasContables } }
- **GET /api/exportar-cuentasContables** → disparado por `descargarExcel`

## Fragmento clave: filtrado y reinicio de página

tsx

```
const muestra: cuentaContableType[] = useMemo(() => {
  setPagina(1);
  return cuentasContables
})
```

```
?.filter(st =>
  st.descripcion.toLowerCase()
    .includes(filtrar.filtro.descripcion2.toLowerCase())
    && st.id.toString()
      .includes(filtrar.filtro.descripcion1.toLowerCase())
) || [];
}, [cuentasContables, filtrar.filtro]);
```

### ⚠ Falla en la lógica

Llamar a `setPagina(1)` dentro de `useMemo` introduce un efecto secundario durante el render. Debería moverse a un `useEffect` para evitar posibles bucles de render.

## SaldosCostos.tsx

Ofrece un informe de saldos por periodo y cuenta, con filtros de texto y fecha. Permite exportar resultados.



Figura 1 – Pantalla principal de Saldos con listado, filtros y acción de exportación.

- **Hooks personalizados**
- `useTraerConfiguracion()`
- `useDescargarExcel()`
- `useFiltrar()`
- `useMostrarError()`
- **Componentes internos**
- `<Cargando />`
- `react-datepicker`
- `<SaldosCostosTabla />`
- **Endpoints**
- **GET /api/saldosContables** → `hacerPeticionApi()` devuelve { datos: { saldosContables } }
- **GET /api/exportar-saldosContables?...** → `descargarExcel(e, buildQueryString())`

## Fragmento clave: generación de query string

tsx

```
const buildQueryString = () => {
  const params = new URLSearchParams();
```

```
const desc1 = filtrar.filtro.descripcion1?.trim();
if (desc1) {
  const alphanumeric = /^[a-z0-9]+$/.test(desc1);
  if (alphanumeric) params.append('descripcion', desc1);
  else params.append('cuentacontable', desc1);
}
if (filtrar.filtro.descripcion2)
  params.append('periodoDesde', filtrar.filtro.descripcion2);
if (filtrar.filtro.descripcion3)
  params.append('periodoHasta', filtrar.filtro.descripcion3);
return params.toString();
};
```

### ⚠ Falla en la lógica

La prueba alfanumérico (/^[a-z0-9]+\$/) clasifica entradas numéricas como **descripcion**, no como **cuentacontable**. Esto invierte la intención de ciertos parámetros.

## SaldosCostosTabla.tsx

Muestra el detalle de saldos ordenados por saldo final. Aplica formato monetario y estilos dinámicos.

- **Props**
- saldosContables: cuentaContableType[]
- **Servicios utilizados**
- generarEstilosDeCabezaDeTabla(titles: string[])
- formatoMoneda(valor: number)
- **Estructura de columnas**

Key	Título	Width	Alineación
codigo	Nro. Cuenta	12%	left
descripcion	Desc. Cuenta	35%	left
ejercicioFiscal	Ejercicio Fiscal	10%	center
centroCostos	Centro de Costos	28%	left
saldo	Saldo	15%	right

- **Flujo de render**
- Inyecta CSS para cabecera según columnas.
- Ordena por saldoFinal descendente.
- Renderiza filas o <NoHayElementos />.
- **Endpoints**
- No realiza peticiones; depende de **SaldosCostos**.

### ⚠ Observación

No se detectan errores lógicos críticos. El componente es conciso y mantiene la separación de preocupaciones.