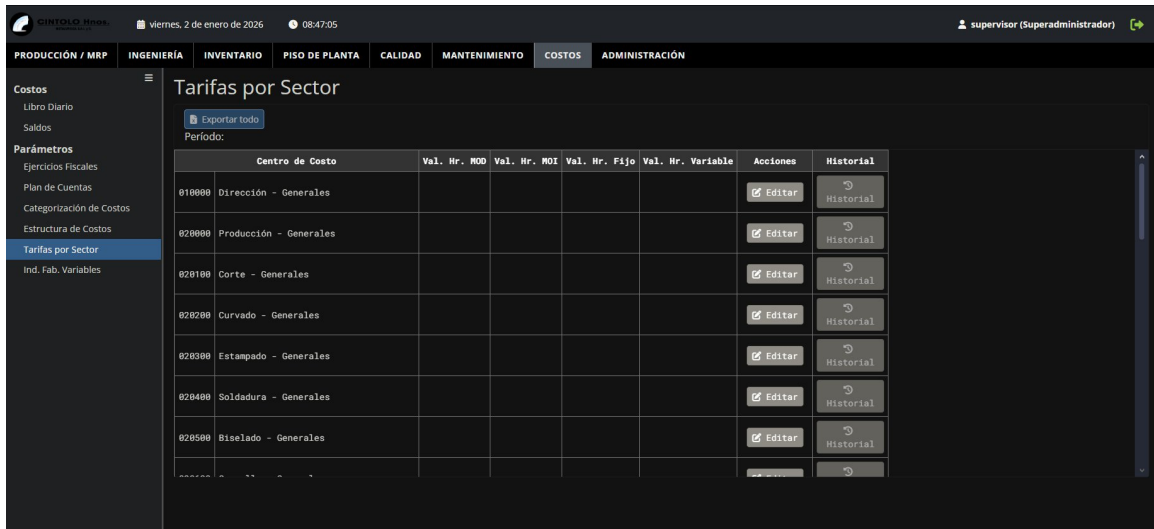


Diagrama de flujo de navegación y datos

Tarifas Component

Componente principal que administra la vista y carga inicial de las tarifas por sector.



Centro de Costo	Val. Hr. MOD	Val. Hr. MOI	Val. Hr. Fijo	Val. Hr. Variable	Acciones	Historial
010000 Dirección - Generales					Editar	Historial
020000 Producción - Generales					Editar	Historial
020100 Corte - Generales					Editar	Historial
020200 Curvado - Generales					Editar	Historial
020300 Estampado - Generales					Editar	Historial
020400 Soldadura - Generales					Editar	Historial
020500 Biselado - Generales					Editar	Historial

Figura 1 – Pantalla principal de Tarifas por Sector con listado y acciones de alta, historial y exportación.

Endpoints utilizados

- **GET costos tarifas:** obtiene el listado de tarifas para el período actual.
- **GET costos tarifas exportar:** descarga un Excel con todas las tarifas (invocado por `useDescargarExcel`).

Estado y hooks

- `centrosDeCosto` (estado): lista de registros `centroDeCostoType[]`.
- `contadorReiniciar` (estado): desencadena recarga de datos.
- Hooks personalizados:
- `useTraerConfiguracion`: controla cargando.
- `useDescargarExcel`: gestiona exportación a Excel.
- `useMostrarError`: muestra errores de API.

Flujo de datos

1. Al montarse, ejecuta `hacerPeticiónApi()` para GET.
2. Si hay datos, actualiza `centrosDeCosto`.
3. Muestra Cargando mientras `config.cargando` es true.
4. Renderiza `TarifasTabla` con los datos obtenidos.

Posibles fallas en la lógica

- El botón **Exportar todo** aparece siempre deshabilitado (disabled sin condición dinámica).
- La lógica para mostrar el modal de agregar tarifas está comentada y no se activa.
- Falta manejar fallos de red antes de `config.setCargando(false)`.

tsx

```
// Extracción relevante de la llamada inicial
useEffect(() => {
  hacerPeticionApi().then(respuesta => {
    config.setCargando(false);
    const dt = respuesta.datos?.datosTarifas as centroDeCostoType[] | undefined;
    if (!dt) {
      mostrarError(respuesta);
      return;
    }
    setCentrosDeCosto(dt);
  });
}, [contadorReiniciar]);
```

TarifasTabla Component ⚙️

Tabla interactiva que muestra, edita y guarda tarifas por centro de costo.

Endpoints utilizados

- **POST costos tarifas**: actualiza una tarifa existente con nuevos valores.
- **Navigate historial**: usa `useAcceder` para redirigir a `/costos/tarifas/historial?id={cc.id}`.

Props

- `centrosDeCosto`: lista de `centroDeCostoType[]` o `null`.
- `setCentrosDeCosto`: actualiza la lista.
- `setContadorReiniciar`: incrementa para forzar recarga.

Estado interno

- `guardandoCcId`: ID del registro que se está guardando.
- `idEditar`: ID del registro en modo edición.

Validaciones y edición

- Al hacer clic en **Editar**, activa inputs numéricos para modificar valores.
- El botón **Aceptar**:
- Muestra un spinner mientras `guardandoCcId === cc.id`.
- Se habilita solo si `valorMod > 0`, `valorMoi > 0` y valores fijos/variables son válidos.

Posibles fallas en la lógica

- **Validación incorrecta** de campos fijos y variables:

js

```
!(cc.gastosFijosXHora || 0 > 0)
```

debería ser

js

```
!( (cc.gastosFijosXHora || 0) > 0 )
```

- parsearDecimales devuelve **string**, pero el tipo centroDeCostoType espera number.
- El botón **Historial** se activa aun cuando el valor es "0.00", por la comprobación de truthiness en lugar de comparación numérica.

tsx

```
<tbody>
  {!!centrosDeCosto?.length
    ? centrosDeCosto.map(cc => (
      <tr key={cc.id}>
        {/* ... columnas de texto o input según idEditar ... */}
        {idEditar === cc.id ? (
          <button
            className='btn btn-success btn-sm w-100'
            onClick={() => aceptar(cc)}
            disabled={
              guardandoCcId === cc.id ||
              !(cc.valorMod > 0) ||
              !(cc.valorMoi > 0) ||
              !(cc.gastosFijosXHora || 0 > 0) || // lógica defectuosa
              !(cc.gastosVarXHora || 0 > 0)
            }
          >
            {guardandoCcId === cc.id
              ? <span className="spinner-border spinner-border-sm" />
              : 'Aceptar'}
          </button>
        ) : (
          <button className='btn btn-secondary btn-sm'
            onClick={() => setIdEditar(cc.id)}>
            <i className='fas fa-edit me-2' />Editar
          </button>
        )}
      </tr>
    )
  ) : <NoHayElementos />
}
</tbody>
```

TarifasAgregar Component

Modal destinado a la **creación** de nuevas tarifas por sector. Actualmente **sin lógica implementada**.

Endpoints utilizados

- **Ninguno** (todo el bloque de selección y envío está comentado).

Descripción breve

- Renderiza un modal con encabezado, cuerpo y pie de página.

- Incluye botones **Agregar** (sin funcionalidad) y **Cancelar** (cierra modal).

Posibles fallas en la lógica

- No captura ni envía datos de insumos.
- Botón **Agregar** ejecuta solo código comentado.
- Falta integrarlo con `hacerPeticionApi` y actualizar la tabla.

tsx

```
<button
  className='btn btn-primary btn-md'
  onClick={() => {
    // TODO: implementar lógica de alta de tarifa
  }}
  // disabled={!insumoId > 0}}
>
  <i className='fas fa-plus' /> Agregar
</button>
```

HistorialTarifas Component

Controlador de la vista de historial de tarifas para un centro de costo específico.

Endpoints utilizados

- **GET costos tarifas historial**: recupera todas las versiones históricas de una tarifa.
- **GET costos tarifas historial exportar?id={id}**: descarga Excel del historial (invocado por `useDescargarExcel`).

Estado y hooks

- `historialTarifasSector`: lista de `centroDeCostoType[]` o `null`.
- Hooks:
- `useTraerConfiguracion` controla cargando.
- `useMostrarError` para errores.
- `useNavegar` para volver a la lista principal.
- `useDescargarExcel` para exportar.

Flujo de datos

1. Al montarse, llama a `hacerPeticionApi()` para GET historial.
2. Si no hay datos, muestra error.
3. Presenta Cargando mientras `config.cargando` es `true`.
4. Renderiza `HistorialTarifasTabla`.

Posibles fallas en la lógica

- Usa siempre `historialTarifasSector?.[0].id` para exportar; si el array está vacío, `id` es `undefined`.
- No ofrece mensaje al usuario si el historial llega vacío (solo tabla sin datos).

tsx

```

<button
  className='btn btn-primary btn-sm'
  onClick={e => descargarExcel(e, `?id=${historialTarifasSector?.[0].id}`)}
  disabled={!historialTarifasSector?.length}
>
  <i className='fas fa-file-excel me-2' />Exportar todo
</button>

```

HistorialTarifasTabla Component

Tabla que muestra cada versión histórica de una tarifa.

Endpoints utilizados

- **Ninguno** (renderiza datos ya obtenidos por el padre).

Descripción

- Formatea el período con `formatearPeriodoFiscal`.
- Genera estilos de cabecera con `generarEstilosDeCabezaDeTabla`.
- Muestra columnas: Fecha, Usuario, y valores monetarios.

Posibles observaciones

- No valida internamente la presencia de datos; confía en `NoHayElementos` del padre.
- Está optimizado para lectura, sin interactividad.

tsx

```

<tbody>
  {!!historialTarifasSector?.length
  ? historialTarifasSector.map(cc => (
    <tr key={cc.tarifaId}>
      <td>{formatearPeriodoFiscal(cc.periodo)}</td>
      <td>{cc.tarifaAltaUsuarioUsername}</td>
      <td className='text-end'>${cc.valorMod}</td>
      <td className='text-end'>${cc.valorMoi}</td>
      <td className='text-end'>${cc.gastosFijosXHora}</td>
      <td className='text-end'>${cc.gastosVarXHora}</td>
    </tr>
  ))
  : <NoHayElementos />
}
</tbody>

```

Conclusión:

El conjunto de componentes ofrece CRUD parcial de tarifas y consulta de historial. Se recomienda corregir validaciones numéricas, completar la lógica de creación y pulir la UX del exportar Excel.