

BaseComponent<T>

[←Regresar](#)

Clase abstracta reutilizable para facilitar la creación de componentes que usen formularios, tablas, modales y alertas. Provee métodos y propiedades comunes que se pueden extender y personalizar según las necesidades del componente específico.

Objetivo

Esta clase permite centralizar funciones comunes como:

- Inicialización de formularios.
- Configuración y carga de datos en tablas.
- Manejo de paginación.
- Registro de acciones en tablas.
- Control de selección de filas.
- Navegación entre rutas.
- Configuración de modales y alertas visuales.

Inyecciones de Dependencias

- **Router** (`@angular/router`): permite navegar programáticamente entre rutas.
- **FormBuilder** (`@angular/forms`): crea formularios reactivos fácilmente.
- **BreadcrumbService**: establece el breadcrumb de navegación (migas de pan).

Propiedades

- **title / subTitle**: Título para encabezados del componente.
- **form**: Instancia del formulario reactivo principal del componente.
- **tableConfig**: Configuración de comportamiento y estilo de la tabla.
- **columns**: Lista de columnas visibles de la tabla.
- **paginationConfig**: Configuración específica para la paginación.
- **dataSource**: Arreglo de datos que alimenta la tabla.
- **page / size**: Controlan la página actual y el tamaño de la paginación.
- **selectedItems**: Elementos seleccionados en la tabla.
- **showMessage**: Bandera que controla si se muestra o no un mensaje.
- **alertOptions**: Configuración para mostrar un mensaje con SweetAlert.
- **modalConfig**: Configuración de un modal genérico.
- **actionHandlers**: Mapa que asocia nombres de acciones con funciones que las manejan.

Métodos Públicos y Protegidos

initForm()

Inicializa el formulario base con un campo `filter`.

```
this.form = this._fb.group({
  filter: ['']
});
```

setBreadcrumb(items: BreadcrumbItem[])

Configura el breadcrumb que se mostrará en la parte superior del componente, útil para indicar la ruta actual.

```
this._breadcrumbService.setItems(items);
```

initTable()

Inicializa la tabla cargando columnas, configuración y datos. Solo funciona si los métodos `getTableColumns()`, `getTableConfig()` y `loadDataTable()` están implementados en la subclase.

getTableColumns(), getTableConfig(), loadDataTable()

Son métodos opcionales que deben ser sobrescritos por los componentes que extiendan `BaseComponent`.

```
protected getTableColumns(): TableListColumns[] {
  return [...]; // retorna las columnas
}
```

navigateTo(route: routeParams[])

Permite redirigir a otra ruta usando el Router de Angular.

```
this.navigateTo(['/ruta', id]);
```

actionsTable(event: ITableEmitterAction<T>)

Ejecuta una función asociada a una acción en la tabla. Se debe haber registrado antes usando `registerAction()`.

```
this.registerAction('editar', (item) => this.editarItem(item));
```

handlePaginator(event: IPagination)

Actualiza page y size con los valores del evento y recarga los datos.

onSelectRow(event: SelectedItemTable<T>)

Controla la selección de filas en la tabla. Si una fila es seleccionada, se agrega a selectedItems, de lo contrario, se elimina.

registerAction(action: string, handler: (item: T) => void)

Asocia una acción de la tabla con una función manejadora. Esto permite ejecutar lógica personalizada cuando se dispara una acción desde la tabla.

onAlertClosed(result: SweetAlertResult)

Maneja el cierre de una alerta. Si el usuario confirma, se oculta el mensaje.

initSweetAlert(title, text, icon, confirmButtonText)

Configura los parámetros para mostrar una alerta con SweetAlert. Es reutilizable y puede personalizarse fácilmente.

```
this.initSweetAlert('Guardado', 'Los datos han sido guardados', 'success');
```

initModal(title, width, visible, closeButton)

Inicializa la configuración de un modal que puede abrirse en el componente.

```
this.initModal('Editar usuario', '30rem', true, true);
```

Ejemplo de Uso

```
export class MiComponente extends BaseComponent<Usuario> {  
  protected override getTableColumns(): TableListColumns[] {  
    return [  
      { field: 'nombre', header: 'Nombre' },  
      { field: 'email', header: 'Correo' }  
    ];  
  }  
}
```

```
protected override getTableConfig(): ConfigTable {  
    return {  
        actions: true,  
        selectable: true  
    };  
}  
  
protected override loadDataTable(): void {  
    // Aquí haces una llamada a un servicio para llenar dataSource  
}  
  
ngOnInit() {  
    this.setBreadcrumb([ { label: 'Usuarios' } ]);  
    this.initTable();  
}  
}
```

Beneficios de esta clase

- Evita repetir lógica en múltiples componentes.
- Mejora la mantenibilidad del código.
- Estandariza el uso de formularios, tablas y modales.
- Facilita pruebas y futuras refactorizaciones.

[←Regresar](#)

Documento actualizado: 25/06/2025

Elaborado por: Maicol Arroyave - Desarrollador

From:
<http://wiki.adacsc.co/> - Wiki

Permanent link:
<http://wiki.adacsc.co/doku.php?id=ada:howto:sicoferp:factory:new-migracion-sicoferp:front:base-component>

Last update: 2025/06/25 19:21

