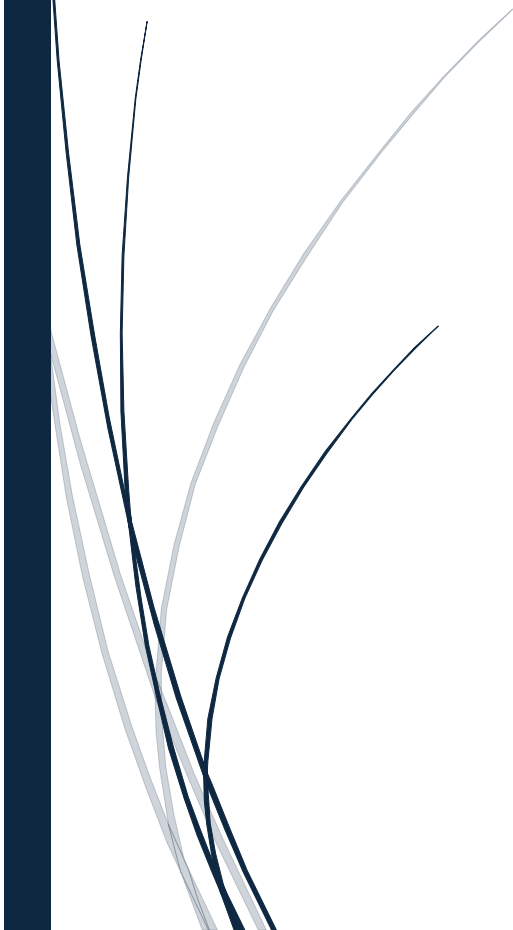




21-4-2025

Documentación prueba técnica Totalum



Rafael Pedrosa Castelo
TOTALUM

Contenido

Introducción	2
Despliegue	2
Componente principal.....	2
Inputs	2
Funcionalidades	3
Métodos principales	3
Formato de fechas	3
Formato de euros	3
Base de datos	4

Introducción

Este proyecto es una prueba técnica desarrollada para Totalum. Consiste en una aplicación web construida con Angular que permite gestionar entidades genéricas a través de una interfaz reutilizable. La aplicación se conecta a una API REST para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre diferentes entidades.

Despliegue

1. Clona el repositorio.

```
git clone https://github.com/RafaPeCas/prueba_tecnica_totalum.git
cd prueba_tecnica_totalum
```

2. Instala las dependencias.

```
npm install
```

3. Introduce la API adjunta en el correo perteneciente al Totalum “Rafael-prueba-tecnica”.

La ruta es “src/enviroments/enviroments.ts.”

4. Ejecuta la aplicación.

```
ng serve
```

Componente principal

El componente “EntityTableComponent” está diseñado para ser reutilizable con cualquier entidad, permitiendo operaciones completas **CRUD** en una tabla editable con **paginación, búsqueda y creación vía modal**.

Inputs

- ❖ **tableName** es una cadena obligatoria que indica el nombre de la tabla o colección en la base de datos de Totalum. Este valor se utiliza internamente para realizar las operaciones CRUD contra la API.
- ❖ **title** es una cadena de texto que se muestra como encabezado de la tabla. Permite contextualizar visualmente qué tipo de entidad se está mostrando o editando.
- ❖ **items** es un array de objetos que contiene los datos que se van a mostrar en la tabla. Normalmente se obtiene desde el servicio que conecta con la API.

- ❖ **columns** es un array de strings que define los nombres de las propiedades que se deben mostrar como columnas en la tabla. El orden del array determina el orden visual de las columnas.
- ❖ **editable** es un booleano que define si los registros pueden editarse o eliminarse desde la interfaz. Si se establece en false, los botones de acción desaparecen y la tabla queda en modo solo lectura.
- ❖ **searchField** es el nombre del campo sobre el que se aplica la funcionalidad de búsqueda en tiempo real. Este campo debe estar presente en cada uno de los objetos del array items.

Funcionalidades

- ❖ Visualización de datos paginados.
- ❖ Búsqueda dinámica sobre un campo especificado.
- ❖ Edición en línea.
- ❖ Creación de registros mediante modal.
- ❖ Eliminación directa desde la tabla.

Métodos principales

- ❖ loadItems() → Carga los datos desde la API.
- ❖ addItem(newItem) → Añade un nuevo registro.
- ❖ updateItem(item) → Actualiza un registro.
- ❖ deleteItem(id) → Elimina un registro.
- ❖ filteredItems() → Devuelve los datos filtrados según searchQuery.
- ❖ getInputType(column) → Determina el tipo de input (text o date).
- ❖ nextPage() / previousPage() → Control de paginación.
- ❖ saveEdit() / cancelEdit() → Control del modo de edición.

Formato de fechas

Cualquier columna que incluya la palabra "fecha" automáticamente:

- ❖ Se muestra con formato dd/MM/yyyy.
- ❖ Se edita mediante un <input type="date">.

Formato de euros

Pipe personalizado que se utiliza para **formatear valores monetarios almacenados en céntimos** y mostrarlos correctamente como euros con formato local (por ejemplo, 12345 se renderiza como 123,45 €).

He decidido guardar los valores monetarios en céntimos, que siempre son números enteros, en la base de datos, ya que son mucho más estables que los números flotantes.

Base de datos

El servicio **TotalumApiService** actúa como puente entre el frontend Angular y la base de datos gestionada a través de la SDK de **Totalum API**.

Contiene todos los métodos de las *CRUDS*, la paginación y el buscador, que vienen dados en la misma documentación de **Totalum**.

La API se recoge en *src/enviroments/enviroments.ts*:

```
export const environment = {  
  production: true,  
  totalumApiKey: 'key'  
};
```