# CrudTableComponent – Custom Angular Web Component

## 1) Overview

*CrudTableComponent* is a web component developed in Angular to storage data about anything. It contains a table where the data are displayed and above the table is displayed a form input field through we can filter the data from the table by a substring which is searched through all columns. The table provide also the crud operations on data (create, read, update, delete) and sort header to sort the data by each column. Also, the component provide the possibility to obtained the modified data to prelucrate after if it you wish.

## 2) Utility

The component can be used in various scenarios (sites for booksellers, online shops, forums, etc) to help users easy finding what they want. It also helps programmers in terms of reducing the time spent developing similar components that exist in most websites nowadays. Also, the stored entity into the table can be customized externally (you don't have to modify anything inside the component only the input properties you give to it) to the component as the programmer wish.

## 3) Start

The development server of the project is maintained on Github and for use it you must:

- download the repository from ***https://github.com/andragabrielapufu99/my-table-component.git***
- run ***npm install*** to install dependencies
- run ***ng serve*** to start the server
- navigate in web browser to ***http://localhost:4200***

## 4) Integrate

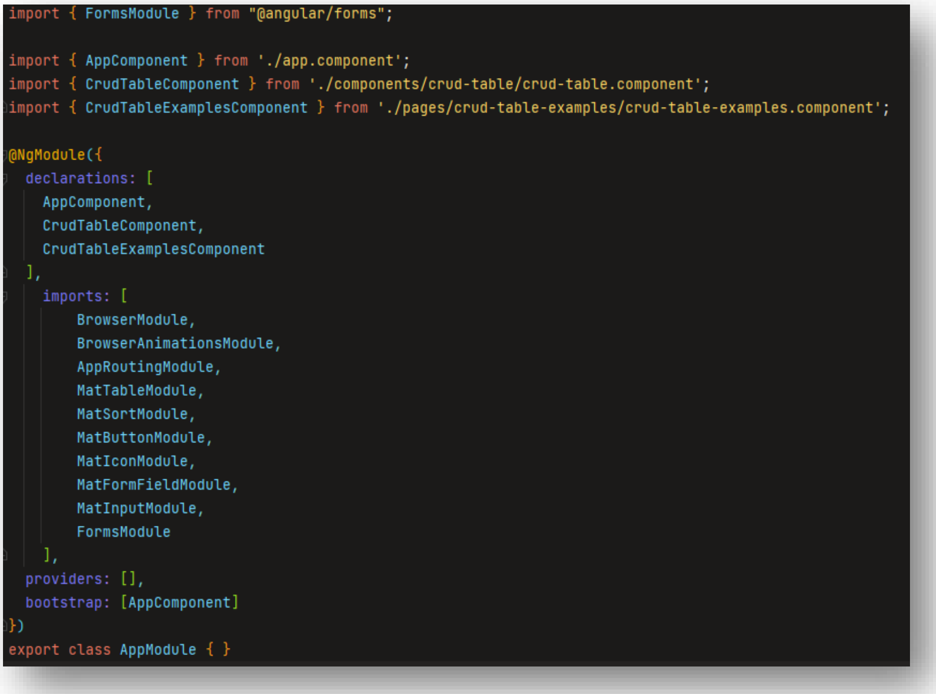
By default all ng modules dependencies and component declarations are add in the file app.module.ts (you must run npm install first) but in case you have compile errors you have to check that you have the next imports and declarations as in Fig 1.

Next, inside the project you can find a demo component *CrudTableExamplesComponent* which integrate many examples of CrudTableComponent (the path is *my-table-component/src/app/pages/crud-table-examples/*). You can use it as template.

In fact, all you need to do is to define a *TableData (see DataFormat section)* object which will be give as an input property (data) to CrudTableComponent in the HTML tag (see Fig. 2).

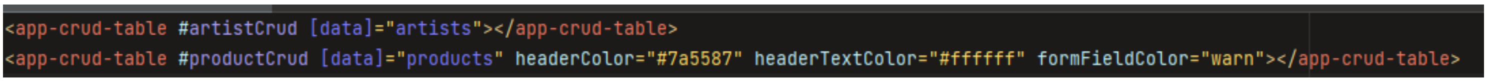
Optional, to customize the table design you can add as properties a header color, a header text color (both must be a hexa color code string) and a ThemePalette color for the filter form on click event (the possible values are "primary", "warn" , "accent" and are the predefinite colors for the current style theme you use). A customized html tag component can be see below (see Fig. 2).

In the Fig.3 we can see how we can obtain the references of web components to access them later if we want.



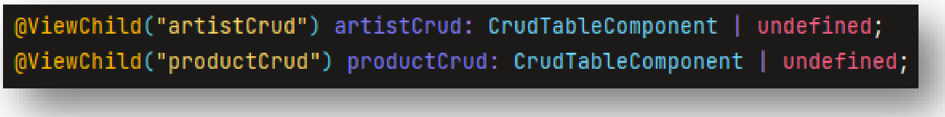
```
import { FormsModule } from "@angular/forms";

import { AppComponent } from './app.component';
import { CrudTableComponent } from './components/crud-table/crud-table.component';
import { CrudTableExamplesComponent } from './pages/crud-table-examples/crud-table-examples.component';

@NgModule({
  declarations: [
    AppComponent,
    CrudTableComponent,
    CrudTableExamplesComponent
  ],
    imports: [
        BrowserModule,
        BrowserAnimationsModule,
        AppRoutingModule,
        MatTableModule,
        MatSortModule,
        MatButtonModule,
        MatIconModule,
        MatFormFieldModule,
        MatInputModule,
        FormsModule
    ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Fig. 1: Required ng imports and dependencies



```
<app-crud-table #artistCrud [data]="artists"></app-crud-table>
<app-crud-table #productCrud [data]="products" headerColor="#7a5587" headerTextColor="#ffffff" formFieldColor="warn"></app-crud-table>
```

Fig. 2: HTML tags for a default and a customized

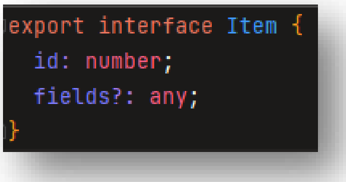


```
@ViewChild("artistCrud") artistCrud: CrudTableComponent | undefined;
@ViewChild("productCrud") productCrud: CrudTableComponent | undefined;
```

Fig. 3: References to html tags of the components

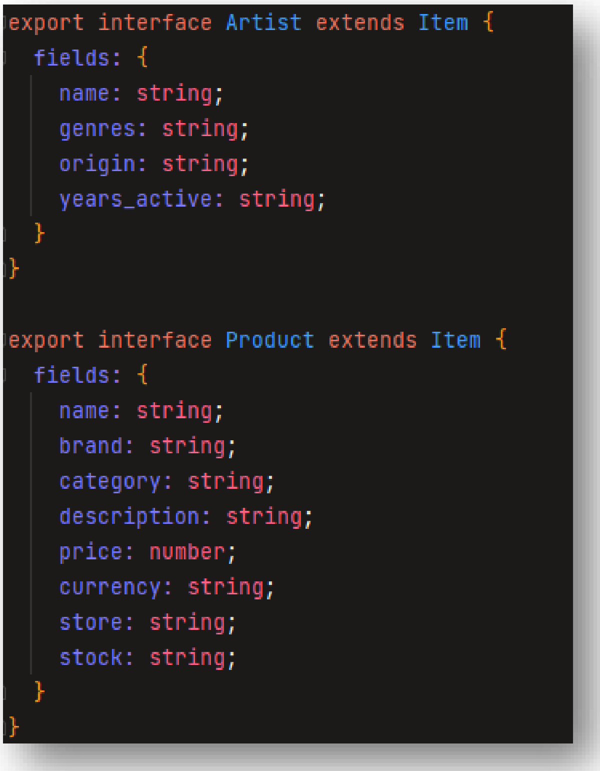
## 5) Data Format

***<u>Item interface (see Fig.4.1)</u>***



```
export interface Item {
  id: number;
  fields?: any;
}
```

Fig. 4.1: Item



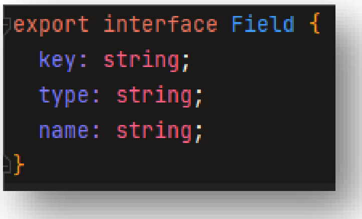
```
export interface Artist extends Item {
  fields: {
    name: string;
    genres: string;
    origin: string;
    years_active: string;
  }
}

export interface Product extends Item {
  fields: {
    name: string;
    brand: string;
    category: string;
    description: string;
    price: number;
    currency: string;
    store: string;
    stock: string;
  }
}
```

Fig. 4.2: Examples

This interface must be extended by all entities the programmer wants to store inside the *CrudTableComponent*. It has a required property **id** which is necessary for identifying of each element of type Item. There are also an optional property **fields** inside of which the programmer can define other properties for the desired entity. Below we can observe some examples of entities which are extends the Item interface (see Fig. 4.2).

### *Field interface (see Fig. 5)*

This interface represents the structure which a column of the table must have. Here we have the properties **key** (the id of the column), **name** (the title of the column which will appear in the header of the table) and the **type** (can be text or number).



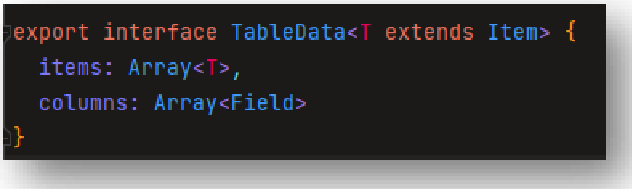
```
export interface Field {
  key: string;
  type: string;
  name: string;
}
```

Fig. 5: Field interface

### *TableData interface (see Fig. 6)*

This is the interface which define the whole structure of entity required by the *CrudTableComponent*. It is a parametrized interface and the **T type** must be an interface which extends the Item interface. The contained properties are:

- **items** : an array of **T** type elements
- **columns**: an array of **Field** type elements

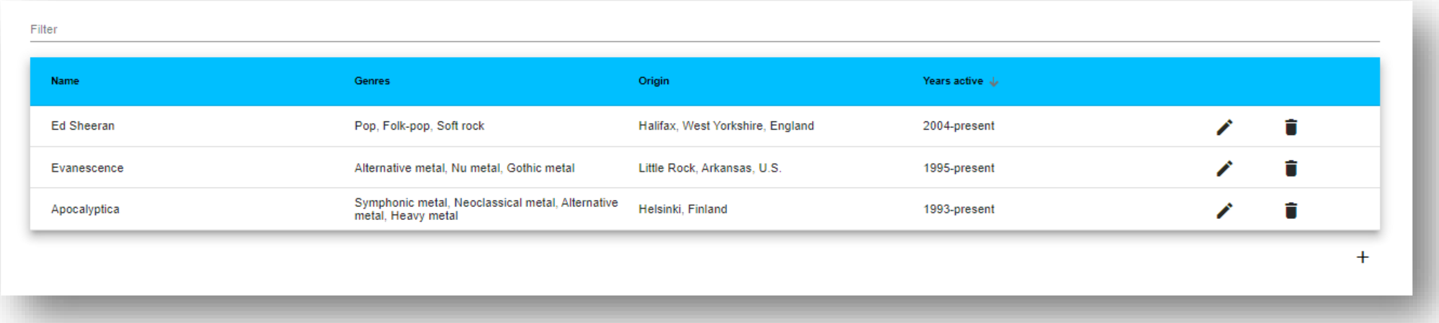


```
export interface TableData<T extends Item> {
  items: Array<T>,
  columns: Array<Field>
}
```

Fig. 6: TableData interface

## 6) Description

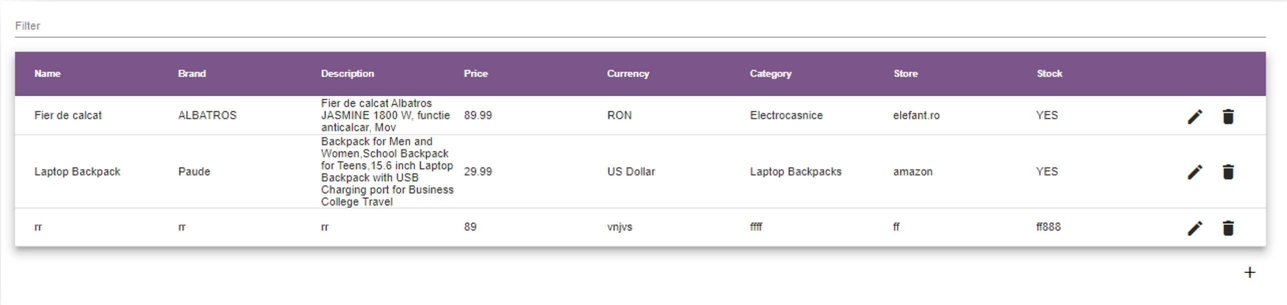
Below we can see how the component looks like. The first example (see Fig. 7.1) is the component with the default style properties if the programmer doesn't change them and the second one is an customized example (see Fig. 7.2).



Filter

| Name | Genres | Origin | Years active ↓ | | |
|---|---|---|---|---|---|
| Ed Sheeran | Pop, Folk-pop, Soft rock | Halifax, West Yorkshire, England | 2004-present | ✎ | 🗑 |
| Evanescence | Alternative metal, Nu metal, Gothic metal | Little Rock, Arkansas, U.S. | 1995-present | ✎ | 🗑 |
| Apocalyptica | Symphonic metal, Neoclassical metal, Alternative metal, Heavy metal | Helsinki, Finland | 1993-present | ✎ | 🗑 |

Fig. 7.1: Default component display

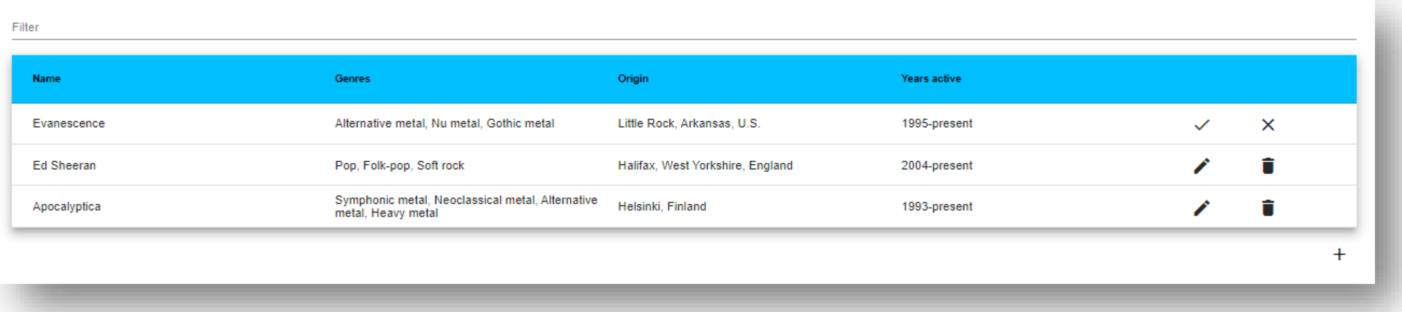


Filter

| Name | Brand | Description | Price | Currency | Category | Store | Stock | | |
|---|---|---|---|---|---|---|---|---|---|
| Fier de calcat | ALBATROS | Fier de calcat Albatros JASMINE 1800 W, functie anticalcar, Mov | 89.99 | RON | Electrocasnice | elefant.ro | YES | ✎ | 🗑 |
| Laptop Backpack | Paude | Backpack for Men and Women,School Backpack for Teens,15.6 inch Laptop Backpack with USB Charging port for Business College Travel | 29.99 | US Dollar | Laptop Backpacks | amazon | YES | ✎ | 🗑 |
| rr | rr | rr | 89 | vnjvs | ffff | ff | ff888 | ✎ | 🗑 |

Fig. 7.2: Customized component display

As we can see the data is displayed in the table on rows and for each row there are two buttons, first for edit the entity and the second for delete it. After we click on edit button the icons will be changed (see Fig. 7.3) and they represent save changes and cancel changes. Also the whole row will became editable. After click of one of them the row will return to its original state.

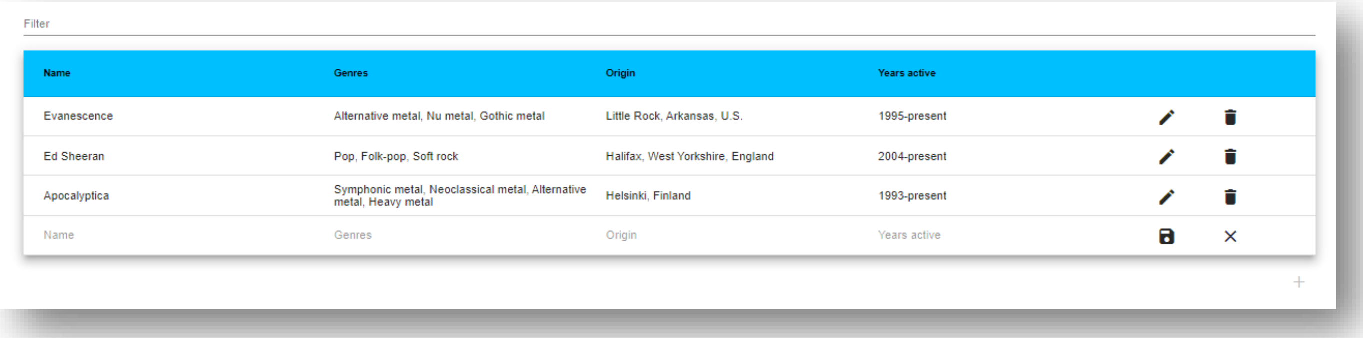


Filter

| Name | Genres | Origin | Years active | | |
|---|---|---|---|---|---|
| Evanescence | Alternative metal, Nu metal, Gothic metal | Little Rock, Arkansas, U.S. | 1995-present | ✓ | ✕ |
| Ed Sheeran | Pop, Folk-pop, Soft rock | Halifax, West Yorkshire, England | 2004-present | ✎ | 🗑 |
| Apocalyptica | Symphonic metal, Neoclassical metal, Alternative metal, Heavy metal | Helsinki, Finland | 1993-present | ✎ | 🗑 |

Fig. 7.3: Edit mode

If we want to add a new element in the table we can click on the "+" button icon from the right bottom corner of the table and after click a new editable row will be added to table, the user must complete the fields and then click of one of two buttons (save changes or cancel them) and the table will be updated (see Fig. 7.4).



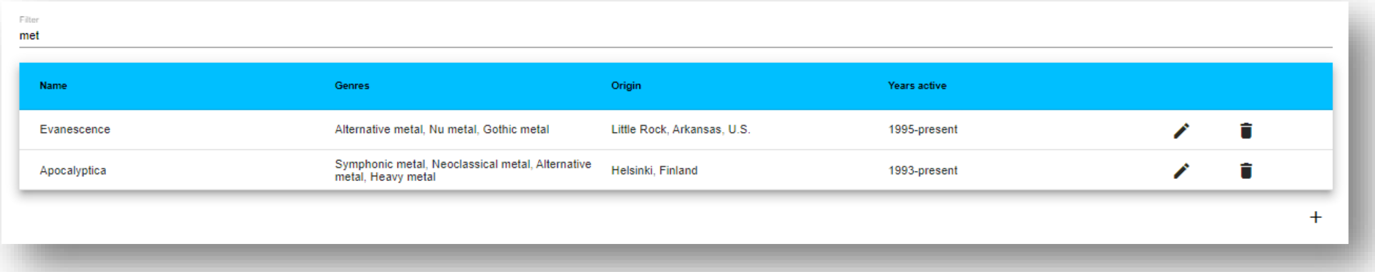
Filter

| Name | Genres | Origin | Years active | | |
|---|---|---|---|---|---|
| Evanescence | Alternative metal, Nu metal, Gothic metal | Little Rock, Arkansas, U.S. | 1995-present | ✏ | 🗑 |
| Ed Sheeran | Pop, Folk-pop, Soft rock | Halifax, West Yorkshire, England | 2004-present | ✏ | 🗑 |
| Apocalyptica | Symphonic metal, Neoclassical metal, Alternative metal, Heavy metal | Helsinki, Finland | 1993-present | ✏ | 🗑 |
| Name | Genres | Origin | Years active | 💾 | ✕ |

+

Fig. 7.4: Add mode

To filter data you must enter some text in form text field above the table and during your typing process the data will be automatically filtered (Fig. 7.5). For sorting data you must click on column header that you wish the sort be made by. The first click will sort ascending and the second one descending (see Fig. 7.6, Fig. 7.7).

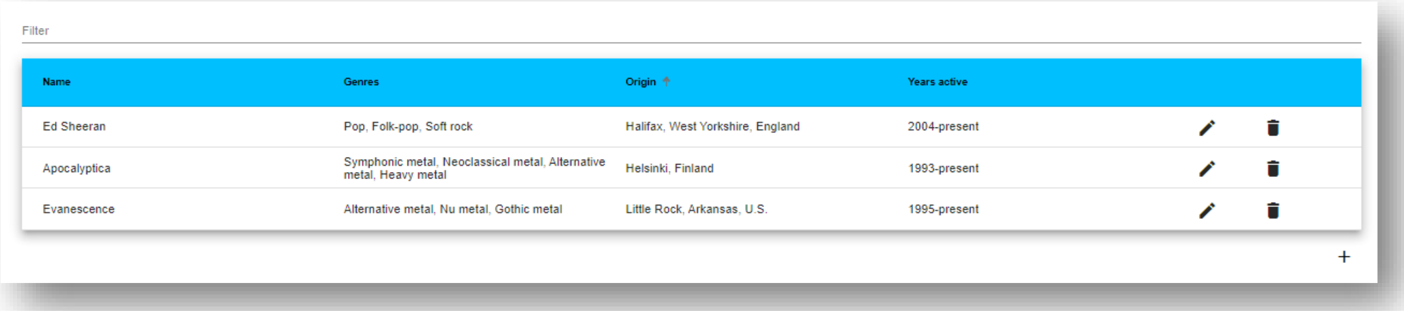


Filter
met

| Name | Genres | Origin | Years active | | |
|---|---|---|---|---|---|
| Evanescence | Alternative metal, Nu metal, Gothic metal | Little Rock, Arkansas, U.S. | 1995-present | ✏ | 🗑 |
| Apocalyptica | Symphonic metal, Neoclassical metal, Alternative metal, Heavy metal | Helsinki, Finland | 1993-present | ✏ | 🗑 |

+

Fig. 7.5: Filter mode

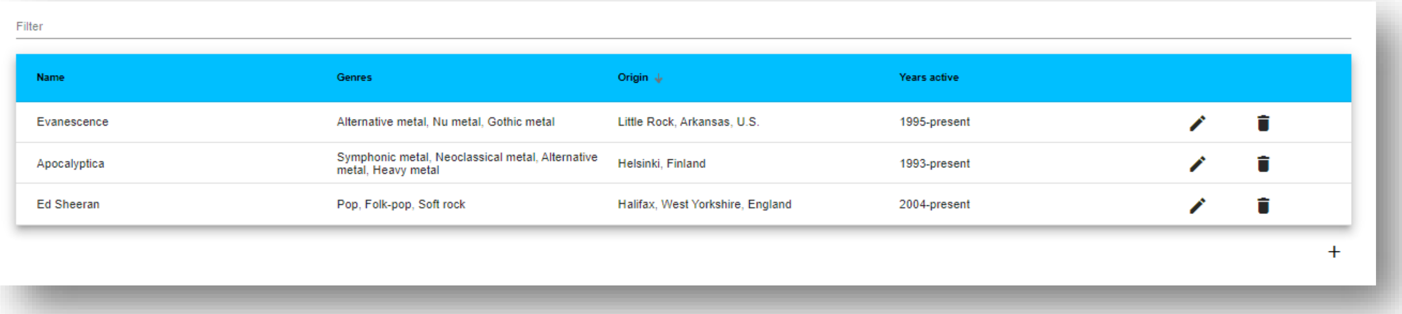


Filter

| Name | Genres | Origin ↑ | Years active | | |
|---|---|---|---|---|---|
| Ed Sheeran | Pop, Folk-pop, Soft rock | Halifax, West Yorkshire, England | 2004-present | ✏ | 🗑 |
| Apocalyptica | Symphonic metal, Neoclassical metal, Alternative metal, Heavy metal | Helsinki, Finland | 1993-present | ✏ | 🗑 |
| Evanescence | Alternative metal, Nu metal, Gothic metal | Little Rock, Arkansas, U.S. | 1995-present | ✏ | 🗑 |

+

Fig. 7.6: Sorting ascending mode



Filter

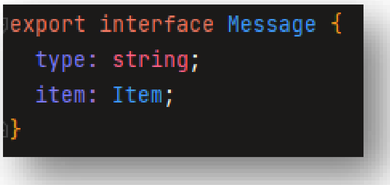
| Name | Genres | Origin ↓ | Years active | | |
|---|---|---|---|---|---|
| Evanescence | Alternative metal, Nu metal, Gothic metal | Little Rock, Arkansas, U.S. | 1995-present | ✎ | 🗑 |
| Apocalyptica | Symphonic metal, Neoclassical metal, Alternative metal, Heavy metal | Helsinki, Finland | 1993-present | ✎ | 🗑 |
| Ed Sheeran | Pop, Folk-pop, Soft rock | Halifax, West Yorkshire, England | 2004-present | ✎ | 🗑 |

+

Fig. 7.7: Sorting descending mode

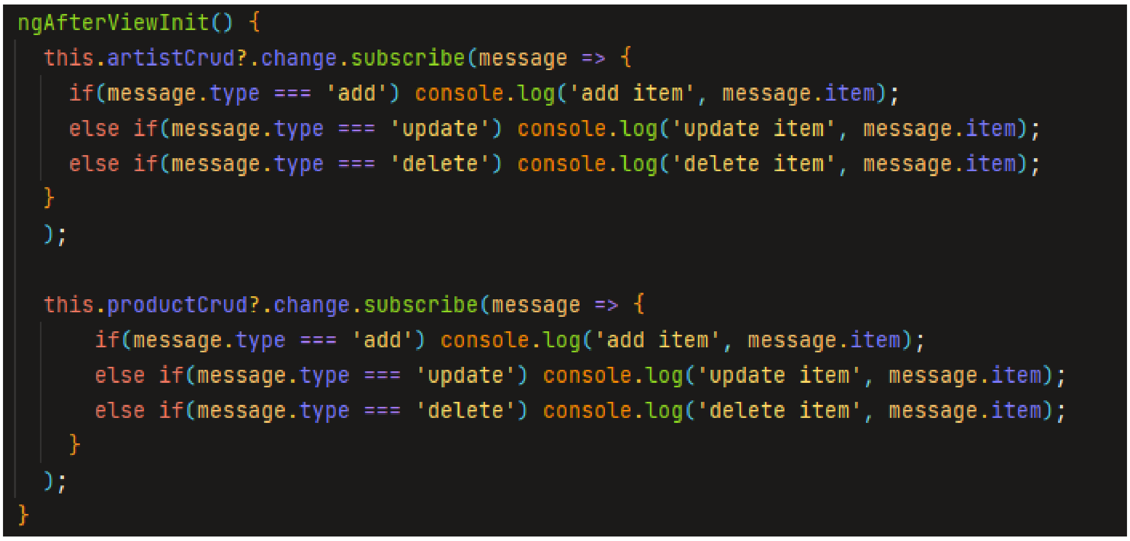
## 7) Code

For listening externally the changes, we can access the EventEmitter object defined inside the CrudTableComponent which will emit an event for each succesfull crud operation (add, update, delete). The message send is a Message object, where Message is an exported interface from the CrudTableComponent (see Fig. 8.1). At each event we'll check what is the type of made operation and depending on the case the programmer will decide the next steps (see Fig. 8.2).



```
export interface Message {
  type: string;
  item: Item;
}
```

Fig. 8.1: Message interface



```
ngAfterViewInit() {
  this.artistCrud?.change.subscribe(message => {
    if(message.type === 'add') console.log('add item', message.item);
    else if(message.type === 'update') console.log('update item', message.item);
    else if(message.type === 'delete') console.log('delete item', message.item);
  }
  );

  this.productCrud?.change.subscribe(message => {
      if(message.type === 'add') console.log('add item', message.item);
      else if(message.type === 'update') console.log('update item', message.item);
      else if(message.type === 'delete') console.log('delete item', message.item);
    }
  );
}
```

Fig. 8.2: Subscribe events