The omni:us frontend challenge

Please read the entire challenge before proceeding. It needs to be built in Angular 8+, TypeScript 3+, SCSS, HTML 5 and webpack. No other languages, frameworks, libraries, tools or technologies are allowed. Please submit your solution in the form of an arbitrary publicly accessible GIT repository of your choice.

If any part of the challenge is unclear, we encourage you to ask questions on frontend-challenge@omnius.com. We prefer you asking questions, than submitting an unfinished or even invalid solution.

Good luck ☺

The task

Create a generic editable table component. It needs to be extendable in a way that it can consume any API which supports pagination, searching, sorting and filtering. Implement support for one specific API which takes the aforementioned parameters in the following format:

```
/api/endpoint?
        pageSize=pageSize&
        cursor=cursor&
        search=search&
        sort=[{field:string,method:string}]&
        filter=[{field:string,method:string,parameters:string|number[]}]
```

Where:

pageSize	can be any number between including 1 and including 100
cursor	can be any primary key from within the full data set
search	can be any value from within the full data set
sort	can be a list of sorts of any field and method, where method can be *ascending* or *descending* (any column can have at most one sort order)
filter	can be a list of filters of any field, method and parameter(s), where method can be *equality* (any value) or *range* (two numbers) (any column can have any amount of unique filters)

Technical requirements

- the component needs to be 100% customisable
- the component needs to be modular, so not everything is to be done within it
- the component needs to support direct data mapping and visualisation for simple types like strings (text), booleans (check box), currency (formatted text), date (formatted text), time (formatted text) and date-time (formatted text), as well as mapping on custom components injected in runtime

- the component needs to support edit mode; once in edit mode, leaving a cell persists the changes to the provided URL
- the component needs to be fully stateful
- the component needs to support events like on data load started, on data load succeded, on data load failed, on searching, on sorting, on filtering, etc.
- besides the tests, put the component into a view, add at least one custom component injected in runtime and use a mock JSON from the assets directory instead of the API
- the entire solution needs to support customisation via a configuration module (custom service, not what Angular is shipped with), so all the configuration values need to come out of external well structured JSON repositorie(s)
- the entire solution needs to support i18n via a i18n module (custom service, not what Angular is shipped with), so all the labels need to come out of external well structured JSON repositorie(s); only English needs to be supported, but the i18n module needs to allow for simple adoption of new cultures
- the entire solution needs to have a decent amount of coverage (checking for correctness of URL construction, loaded and rendered data, edit mode, filter rendering, etc.)
- the SCSS needs to be scalable and customisable (i.e. show us how you can use SCSS variables and maps, define design systems, write DRY mixins, etc.)

UX/UI requirements

- pagination needs to be supported via a drop down, which offers 5, 10, 25, 50 and 100 by default; the defaults can be overwritten as the component configuration parameter
- the default page size is 25; it can be overwritten as the component configuration parameter
- searching needs to be supported via a text box; it can be toggled as the component configuration parameter
- sorting is supported via an icon in every column header; first click transitions from no sort to ascending, second click from ascending to descending and third click from descending back to no sort; it can be toggled and configured as the component configuration parameter
- filtering is supported via an icon in every column header; it opens up a drop down menu that offers customizable filters of equality (check box) and range (two number boxes); it can be toggled and configured as the component configuration parameter
- columns need to support setting up the layout in terms of width and alignment via the configuration
- columns need to be toggleable

The solution is going to be evaluated based on requirement implementation (fullness and correctness), modularisation, separation of concerns, code DRYness, abstraction, generics, encapsulation, code cleanliness and readability, coverage, the UX/UI smoothness, as well as general cleverness and maturity of the architecture.