# Front-end Companies grid

The grid front-end will load all companies, visible to the user, according to the the implemented distribution partner logic from previous building block.

That means, a user cannot see more results than his role (distribution partner id) permits.

### Preferred implementation of list-configuration

Every grid we make has a json configuration, with the column configuration, the filter options, the sort options, a title.

See included example (companies.json).

The purpose of a configuration here is that we can reuse the same configuration options if we want to build an export function for the grid.

We can create a config/grids/ subfolder which contains jsons for every list.

The CompanyController loads the configuration, processes the columns we require and adds the filter options and sort options.

If “html”:true is set, the column is presented in the display of the grid using HTML (front end)

If “export”: true is set, the column is presented in the exporter

The export function is not within the scope of this building block.

We will create 2 grid-configurations for the companies grid. The first one is presented only to the root distribution partner users. The second one is only presented to the non-root distribution partner users.

So, in the Controller we will check if the user belongs to a root distribution partner or not, if so, load a the config for root users (like companies-root.conf). In the other situation, load companies.conf.

### Root distribution partner config

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Number** | **Allocation code** | **Segmentation** | **Company name** | **Client type** | **Owner** |
| 1000 | 123456 | Large account | Saint Gobain | Distribution partner | H. Geluk |
| 1001 | 4567890 | Medium account | Collactive B.V. | Distribution partner | H. Geluk |
| 1002 | 999999 | Small account | Saval B.V. | Client | H. Geluk |

Filterbox: search on the following columns:

* Number
* Company name
* Allocation code
* Owner

### Mapping to database

Number: company.number

Allocation code: company.number\_distribution\_partner

Segmentation: segmentation.description

Company name: company.name

Client type: distribution\_partner.type (Label translated to description above)

Owner: contact.initials + contact.infixes + contact.lastname

### Sorting options

Order by company.number

Order by company.number\_distribution\_partner Order by company.name ( DEFAULT).

Method: by clicking on arrow icons in sortable column

### Non-root distribution partner config

|  |  |  |  |
| --- | --- | --- | --- |
| **Allocation code** | **Segmentation** | **Company name** | **Owner** |
| 123456 | Large account | Saint Gobain | H. Geluk |
| 4567890 | Medium account | Collactive B.V. | H. Geluk |
| 999999 | Small account | Saval B.V. | H. Geluk |

Filterbox: search on the following columns:

* Company name
* Allocation code
* Owner

### Mapping to database

Allocation code: company.number\_distribution\_partner

Segmentation: segmentation.description

Company name: company.name

Owner: contact.initials + contact.infixes + contact.lastname

### Sorting options

Order by company.number\_distribution\_partner

Order by company.name ( DEFAULT).

Method: by clicking on arrow icons in sortable column

### Clickable rows

When you click on an item in the grid, the browser redirects to /company/<ID>

Right now, this route is not implemented. The grid on click can already be implemented though!

### Filter options in API

Filter options are missing in API, so we need to implement these for this grid.

Ideally, the arguments are passed in the query parameters of the route.

Like:

/get/companies/?term=<My generic search term>

Term argument will search on all searchable columns.

/get/companies/?name=<Company name>&number<Company number>

These arguments filter more specific on the according database fields

Advised is to create a generic API handling for this, so to not have to filter inputs manually for each thinkable argument, but instead create a generic flow:

* For every api request define an expandable list of query arguments that can be passed
* For every query argument define if it needs to be filtered (validate numbers, dates

We want to prevent to create structures like this:

If(isset($args[‘<my query argument>’]))

If(preg\_match(‘~MY DATE VALIDATOR~’, $args[‘my query argument’])

We are looking for a more streamlined approach with reusable coding blocks and configurable argument arrays, which we can extend easily.

The developer can create a proposal how to implement this, we will discuss this plan and work on a final implementation.