This application will manage data about Jobs, Companies and Job Applications. This page contains details of how to set up the Strapi backend for your application.

The fields in each type of data are as follows.

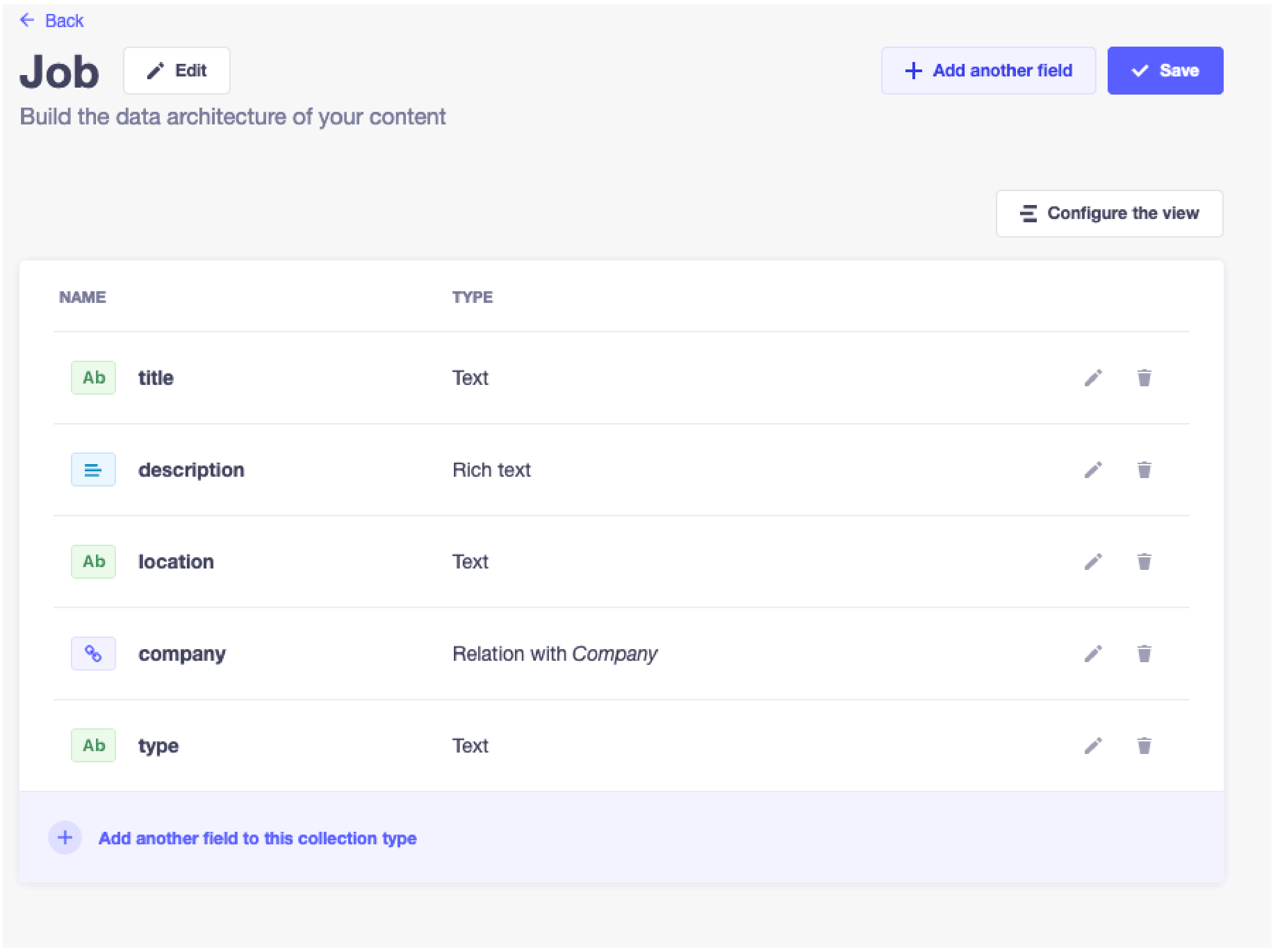
Job

A Job is a listing on the jobs board describing a job opening for a particular company.

These fields are created by Strapi, so you don't add them to your model.

* `id` - a unique integer identifier for the job
* `publishedAt` - timestamp recording when the job was published

* `title` - text title for the job
* `description` - html formatted long-form description of the job
* `location` - textual location of the job, eg. Sydney, Work from home
* `type` - job type, eg. Part Time, Full Time
* `company` - relation with Company, the company offering this job (Company has many jobs)



Job

Build the data architecture of your content

NAME	TYPE
title	Text
description	Rich text
location	Text
company	Relation with *Company*
type	Text

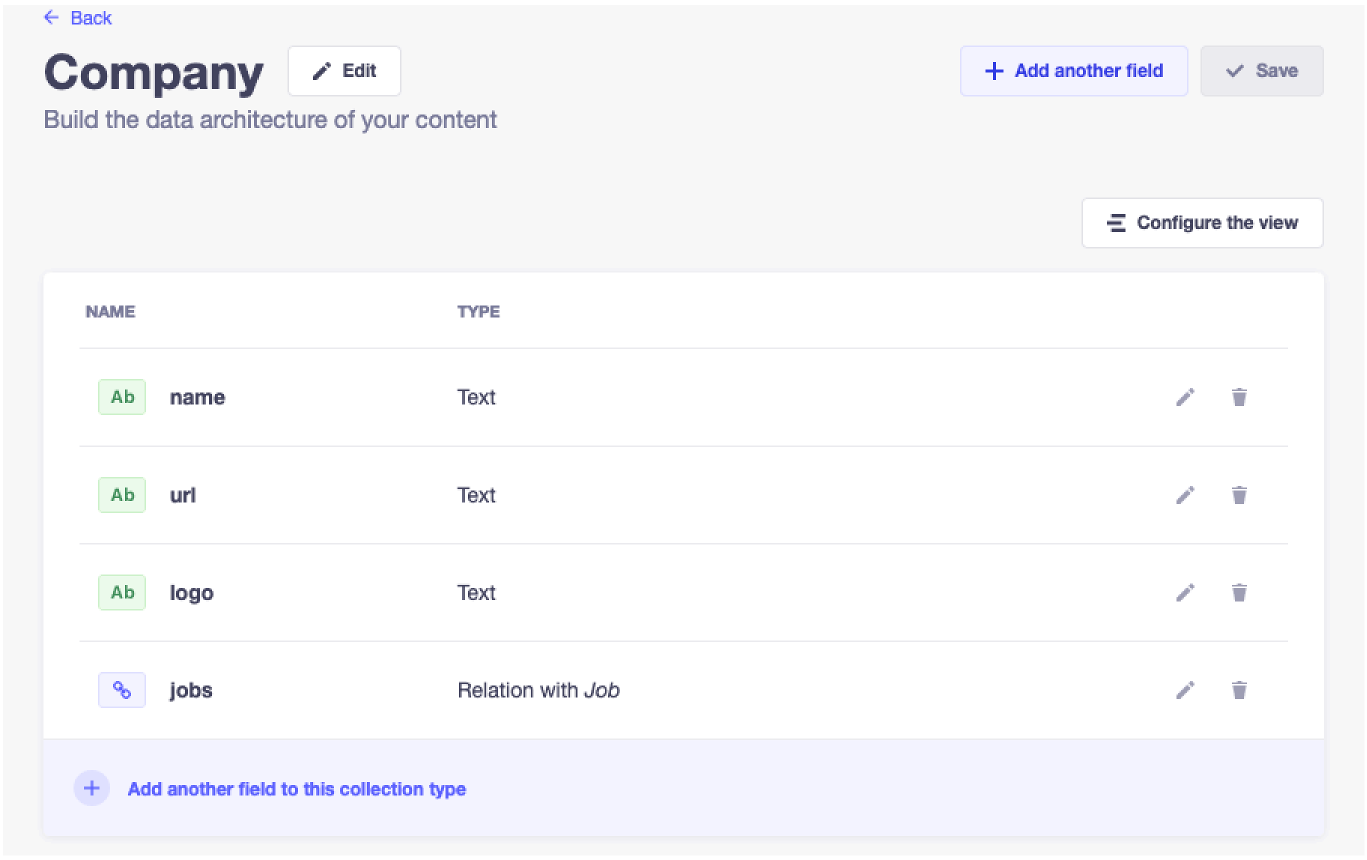
Company

A company is an organisation offering one or more jobs.

This field is created by Strapi, so you don't add it to your model.

- `id` - a unique integer identifier for the company

- `name` - text field name of the company
- `url` - URL of the company website
- `logo` - URL of the company logo (if available)
- `jobs` - relation with Job, the jobs offered by the company (Company belongs to many Jobs)



Company

Build the data architecture of your content

NAME	TYPE
name	Text
url	Text
logo	Text
jobs	Relation with *Job*

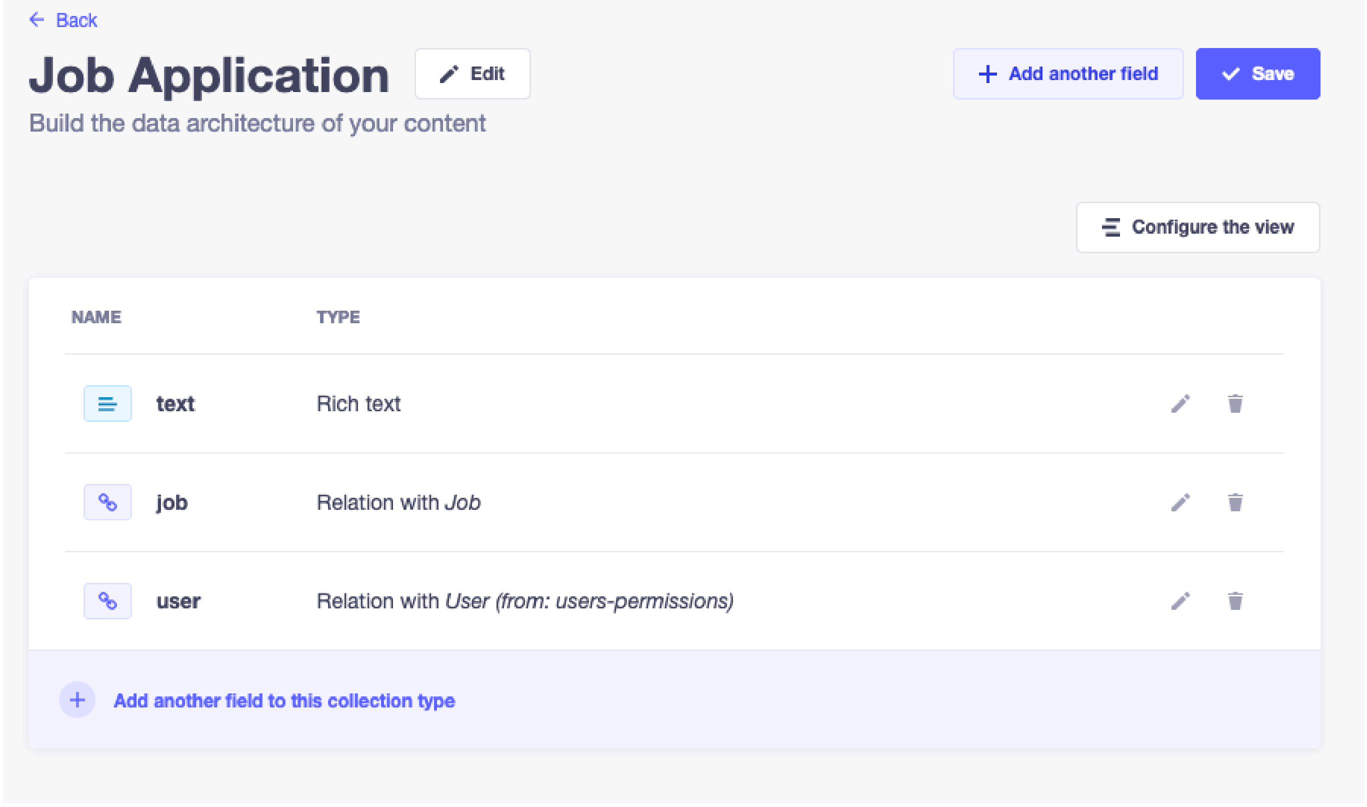
Job Application

A job application is completed by a User to apply for a particular Job.

This field is created by Strapi, so you don't add it to your model.

- `id` - a unique integer identifier for the application

- `user` - relation with User (from: `users-permissions`), the user applying for the job (Job Application has one User)
- `job` - relation with Job, the job applied for (Job Application has one Job)
- `text` - the text of the job application letter


← Back

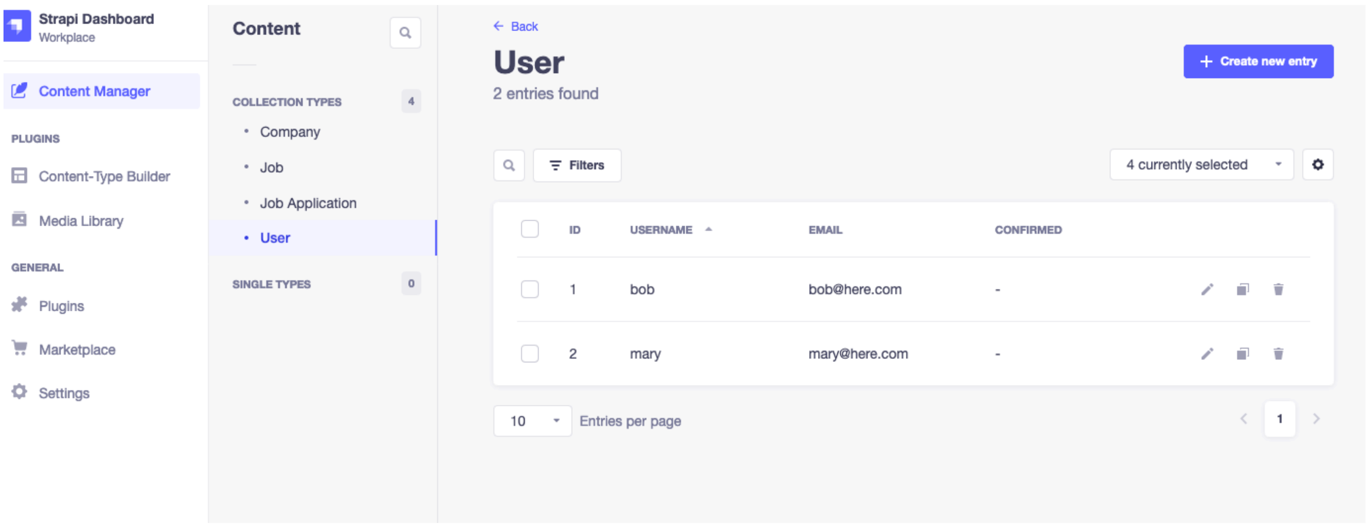
Job Application

Build the data architecture of your content

NAME	TYPE
text	Rich text
job	Relation with *Job*
user	Relation with *User (from: users-permissions)*

Users

You should create at least one user with username 'bob' and password 'bobalooba' (you can create others if you wish).


User

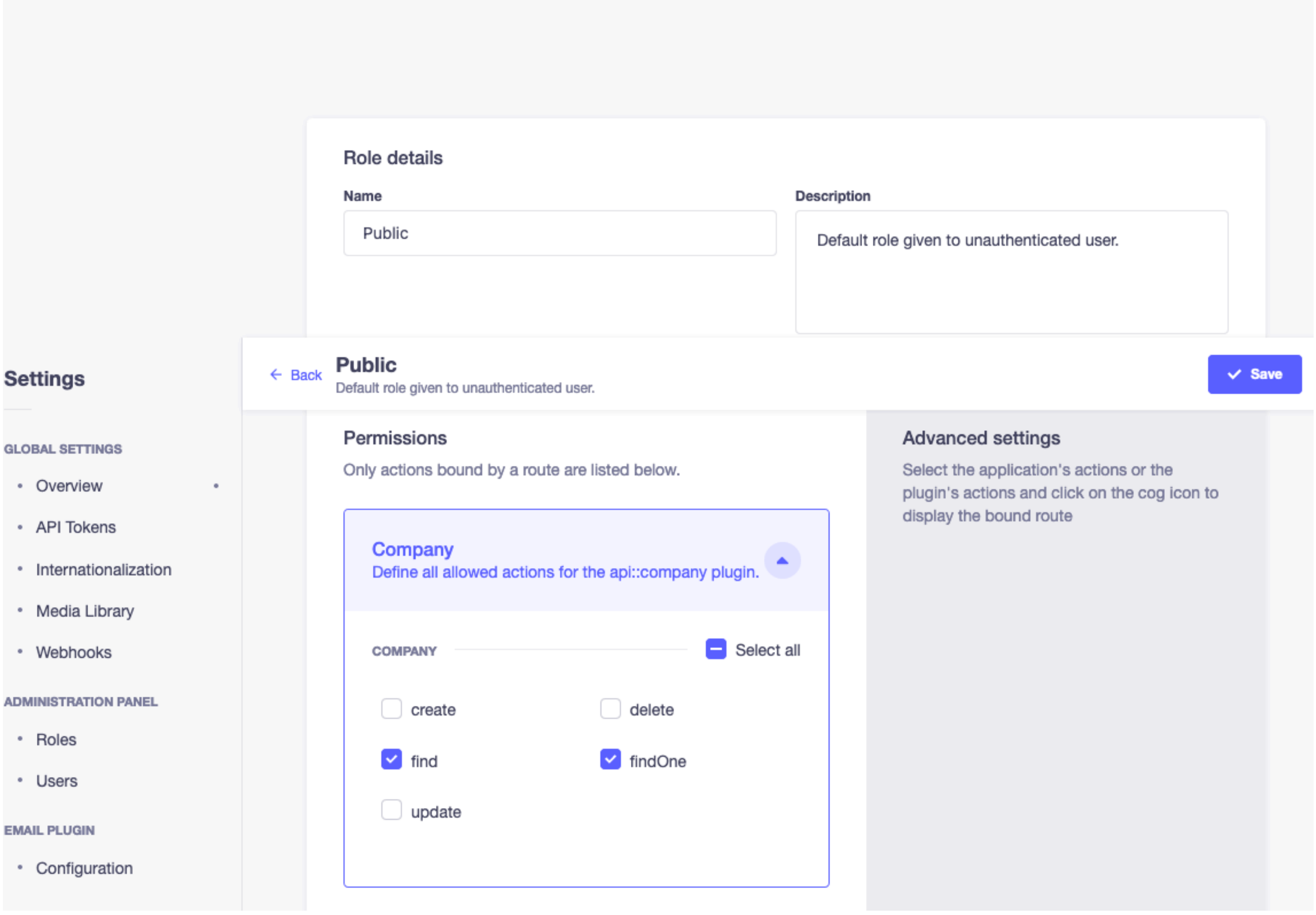
2 entries found

ID	USERNAME	EMAIL	CONFIRMED
1	bob	bob@here.com	-
2	mary	mary@here.com	-

Permissions

Set up the permissions on all of your models to allow access by the Public role. Click on 'Settings' and then select 'Roles' under the Users and Permissions Plugin.

For the Public role, enable the permissions for `find` and `findOne`for the Jobs, Company and Job Application models.


Role details

Name: Public

Description: Default role given to unauthenticated user.

Public
Default role given to unauthenticated user.

Permissions
Only actions bound by a route are listed below.

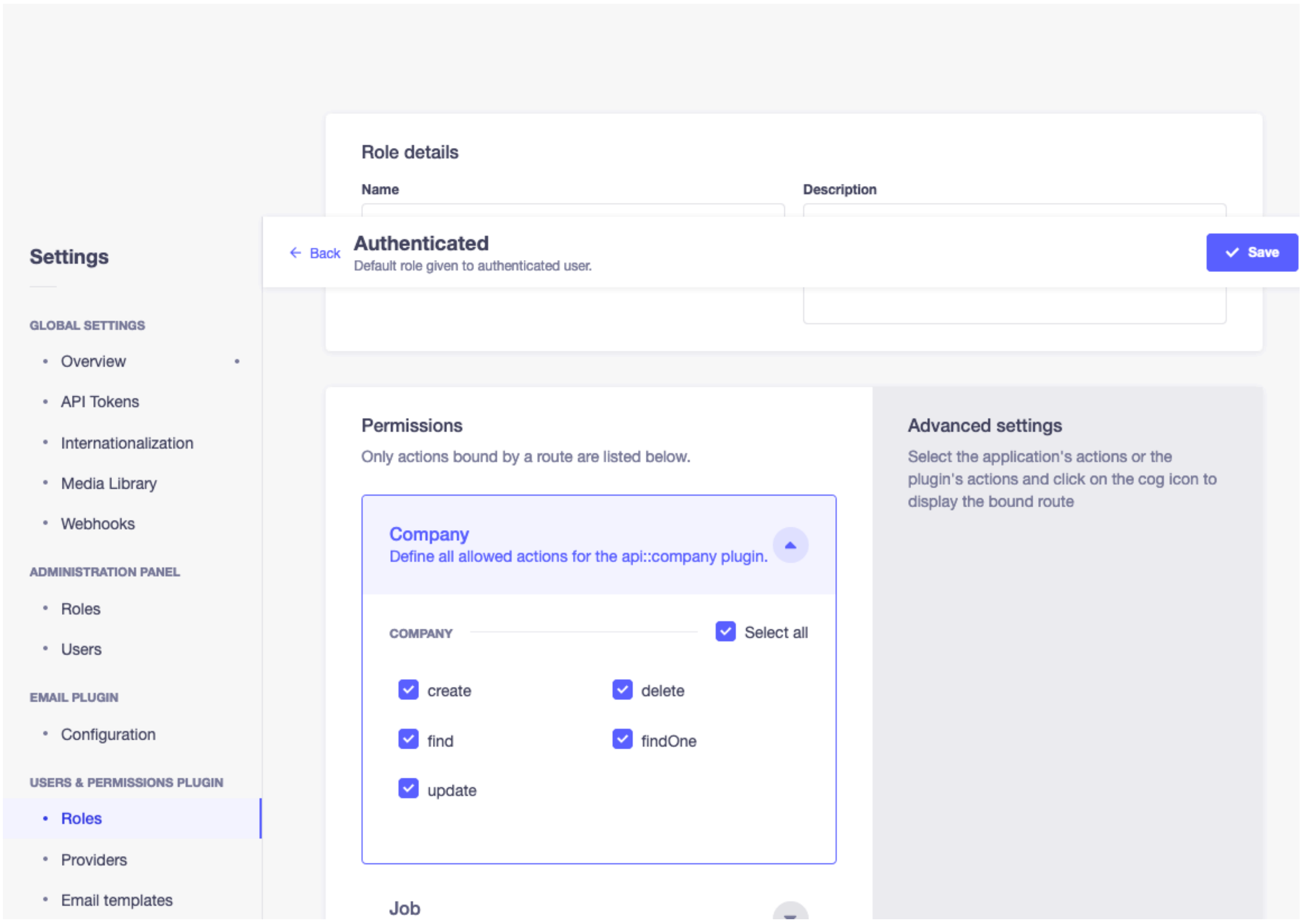
Company
Define all allowed actions for the api::company plugin.

COMPANY — Select all

- [ ] create
- [ ] delete
- [x] find
- [x] findOne
- [ ] update

Advanced settings
Select the application's actions or the plugin's actions and click on the cog icon to display the bound route

For the Authenticated Role, enable __all__ permissions for the Jobs, Company and Job Application models.


Role details

Name

Description

Authenticated
Default role given to authenticated user.

Permissions
Only actions bound by a route are listed below.

Company
Define all allowed actions for the api::company plugin.

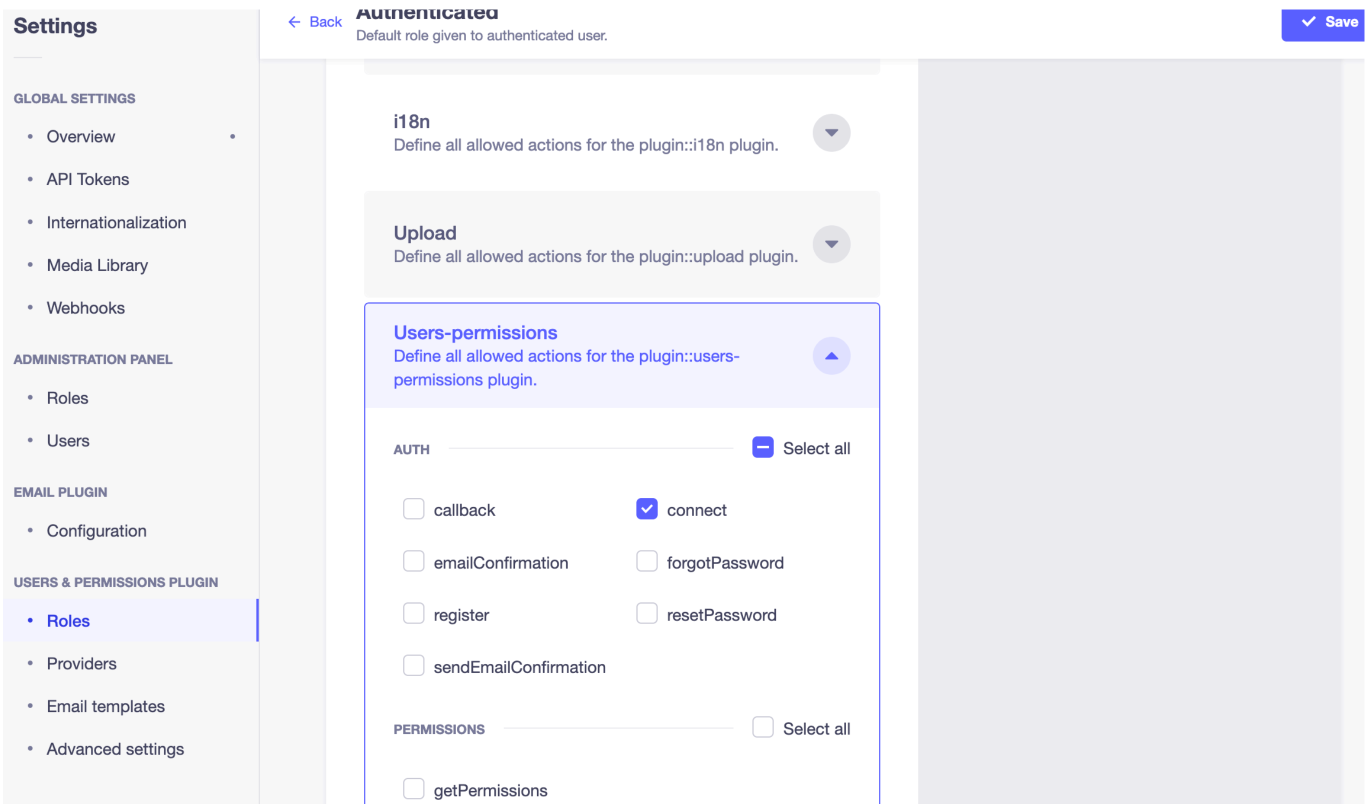
COMPANY — [x] Select all

- [x] create
- [x] delete
- [x] find
- [x] findOne
- [x] update

Job

Advanced settings
Select the application's actions or the plugin's actions and click on the cog icon to display the bound route

You also need to update the permissions for authenticated users on the Users-Permissions content type. This is so that we can populate the user field in a job application. You need to add the 'find' permission:


Settings

GLOBAL SETTINGS
- Overview
- API Tokens
- Internationalization
- Media Library
- Webhooks

ADMINISTRATION PANEL
- Roles
- Users

EMAIL PLUGIN
- Configuration

USERS & PERMISSIONS PLUGIN
- Roles
- Providers
- Email templates
- Advanced settings

Back Authenticated
Default role given to authenticated user.

Save

i18n
Define all allowed actions for the plugin::i18n plugin.

Upload
Define all allowed actions for the plugin::upload plugin.

Users-permissions
Define all allowed actions for the plugin::users-permissions plugin.

AUTH — Select all

- [ ] callback
- [x] connect
- [ ] emailConfirmation
- [ ] forgotPassword
- [ ] register
- [ ] resetPassword
- [ ] sendEmailConfirmation

PERMISSIONS — [ ] Select all

- [ ] getPermissions

Upload Data

Once you have this all configured, you should be able to run the data upload script:

```
npm run sampledata
```

This should create a sample database from the sample JSON data set. If you get errors here, check that you have followed all of the instructions above carefully.

Last modified: Monday, 9 May 2022, 10:33 AM

◄ Web Development Assignment

Jump to...

Tests for Level 3 and 4 ►