

Utiliser un CMS Headless pour stocker les données

- CMS Headless ???
- Un exemple
- Une api toute faite

CMS Headless

- Un CMS Headless : un CMS sans frontend
- Un backoffice pour structurer et saisir du contenu stocké dans une base de données
- Une API pour accéder à ce contenu à l'aide de requêtes REST ou GraphQL
- En général disponible en version cloud ou self-hosted

directus

- Cms headless écrit en node.js
- Disponible en cloud ou en version auto-hébergée
- Image docker disponible
- Un backoffice pour définir la structure du contenu et insérer des données
- Une api REST + GraphQL

architecture

- Une application web
 - Création de la structure
 - Gestion des droits
 - Création du contenu
- Une base de données SQL
 - postgres, mysql, mariaDB, Oracle, SQLite, S3 ...
- Capable d'utiliser une base existante et d'exploiter son contenu

Installation - documentation

- Avec docker compose

<https://docs.directus.io/self-hosted/docker-guide.html>

- Documentation

<https://docs.directus.io/>

Le modèle des données

- Une collection = 1 table avec un identifiant (PK)

The image shows the Directus Data Model interface. On the left, a sidebar contains various navigation options. An orange arrow points to the 'Data Model' option in the sidebar. The main area displays 'Settings Data Model' with a search bar and a 'No Collections' message. A large orange arrow points from the 'Create Collection' button to the 'Creating New Collection' dialog. The dialog has a 'Name' field with 'blog_post' and a 'Primary Key Field' field with 'id'. The 'Singleton' section is expanded, showing options for the primary key field, with 'Auto-incremented integer' selected.

Directus Data Model

Search Collection...

No Collections
You don't have any Collections. Click the button below to get started.

Create Collection

Creating New Collection

Optional Fields

Name *
blog_post
Collection names are case sensitive

Primary Key Field
id

Singleton
☐ Treat as single object
Auto-incremented integer
Auto-incremented big integer
Generated UUID
Manually entered string
Auto-incremented integer

■ Des champs de différents types

Blog Post

Fields & Layout **Saves Automatically**

id

status

date_created

user_created

Create Field

Create Field in Advanced Mode

New Field (blog_post) Search Field...

Text & Numbers

Input

Autocomplete Input (...)

Block Editor

Code

Textarea

WYSIWYG

Markdown

Tags

Selection

Toggle

Datetime

Repeater

Map

Color

Dropdown

Icon

Checkboxes

Checkboxes (Tree)

Dropdown (Multiple)

Radio Buttons

Relational

■ Une interface de saisie

Data Model


Blog Post

Fields & Layout **Saves Automatically**

id	
status	user_created
date_created	
titre*	
contenu	

Create Field

Create Field in Advanced Mode



Blog Post

Creating Item in Blog Post

Status

Draft

Titre *

titre

Contenu

B *I* U H1 H2 H3

■ Et une api pour accéder au contenu

GET api-1 x +

GET http://localhost:8055/items/blog_post

Query Body Headers Auth Vars Script

Assert Tests Docs

Name	Value
------	-------

+ Add Param

Response Headers Timeline Tests

200 OK 19ms 742B

```
1 {
2   "data": [
3     {
4       "id": 3,
5       "status": "draft",
6       "user_created": "b57e2d58-0567-4de7-b75d-2f43f1e19eaa",
7       "date_created": "2023-12-20T13:59:59.108Z",
8       "titre": "Mon premier Post",
9       "contenu": "<p>Salut les loulous,</p>\n<p>Ceci est mon premier post sur ce blog consacré à la vie des grenouilles sur les bords du lac de Chantebleu. C'est un thème qui me passionne :</p>\n<ul>\n<li>j'adore les grenouilles,</li>\n<li>je suis fan de l'écologie,</li>\n<li>j'habite moi-même près du lac.</li>\n</ul>\n<p>Bref, je vous raconte tout ça dans la suite ....</p>\n<p><br></p>",
10      "description": "le premier post qui plante le décor"
11    }
12  ]
13 }
```

■ Création d'associations entre collections

- Type

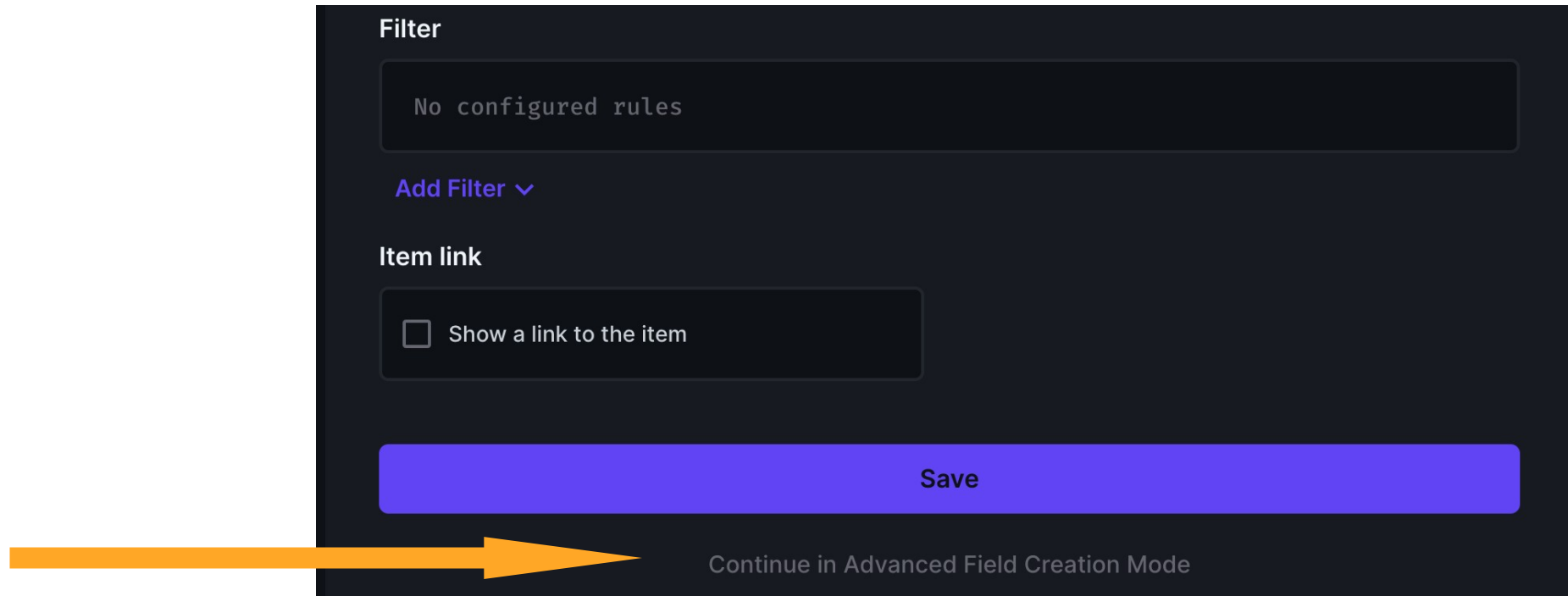
- Collection associée et FK

The screenshot shows the 'Posts (categorie)' form in a dark theme. At the top, there's a search bar and three tabs: 'Checkboxes (Tree)', 'Dropdown (Multiple)', and 'Radio Buttons'. Below these is the 'Relational' section, which contains eight icons representing different relationship types: File, Image, Files, Builder (M2A), Many to Many, One to Many, Tree View, and Many to One. An orange arrow points from the 'Type' bullet point to the 'One to Many' icon. Below the icons is a configuration panel with the following fields:

Key *	Type
posts	Alias
Default Value	Required
NULL	<input type="checkbox"/> Require value to be set on creation
Related Collection *	Foreign Key *
blog_post	cat_id

An orange arrow points from the 'Collection associée et FK' bullet point to the 'Related Collection' field, which contains 'blog_post'.

- Attention : pour créer l'association inverse, notamment dans la cas Many-To-Many, il faut passer en *Advanced Field Creation Mode* avant de sauvegarder



Filter

No configured rules

[Add Filter](#) ▾


Item link



☐ Show a link to the item

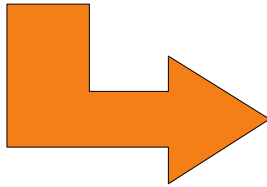
[Save](#)

[Continue in Advanced Field Creation Mode](#)

■ Des droits d'accès basés sur des rôles











Settings **Access Control** 

	Name	Users	Description
	Public	...	Controls what API data is available without aut
	Administrator	1	Initial administrative role with unrestricted App



Access Control **Public**

Permissions **Saves Automatically**

Collection					
blog_post					
categorie					

System Collections ▾

API REST

- Directus permet d'accéder au contenu au travers d'une API REST
- Un point d'entrée par modèle de données défini dans le backoffice
- Des paramètres pour sélectionner les champs, filtrer, trier ...

`http://host/items/spectacles`

`?fields=titre,duree`

`?filter={"titre":{"_eq": "stones"}}`

`?sort=duree`

récapitulatif

- Backend pour création d'une api d'accès à des données très rapide à mettre en place
- Types de données riches
- Très facile d'emploi pour des actions simples de type CRUD
- Possibilité d'ajouter du métier au travers de hooks et de flows (enchaînement d'actions sur événements)