

TD3 - CMS headless Directus - Compte rendu

Auteur : Carette Robin

Depot : <https://github.com/CaretteRobin/nouveaux-paradigmes-TD3-4.git>

Introduction

Ce compte rendu presente la mise en place d'un CMS headless avec Directus pour un domaine "praticiens de sante". L'objectif est de construire un backoffice permettant de gerer les donnees, puis de les exposer via l'API REST de Directus. Les captures ecran documentent chaque etape cle et servent de preuves de realisation.

Objectifs du TD

- Installer Directus en Docker.
- Construire le modele de donnees (collections + relations).
- Importer un jeu de donnees (CSV) en respectant l'ordre des dependances.
- Verifier les donnees via le backoffice.
- Consommer l'API REST pour obtenir des donnees simples et imbriquees.

Environnement et demarrage

Le service est deploye avec Docker Compose. Le conteneur Directus est expose sur le port 8055.

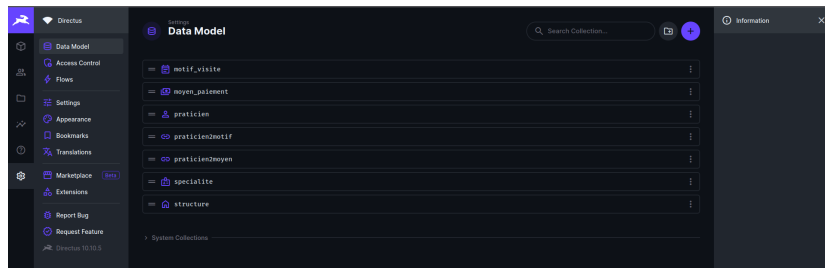
```
ml@mlrobin-drtinater: /dev/nouveaux-paradigmes/CMS$ docker compose up -d
Unable to get image 'directus/directus:10.10.5': Cannot connect to the Docker daemon at unix:///home/mlrobin/.docker/desktop/docker.sock. Is the docker daemon running?
ml@mlrobin-drtinater: /dev/nouveaux-paradigmes/CMS$ docker compose up -d
Container cms-db-1 started
Container cms-directus-1 started
ml@mlrobin-drtinater: /dev/nouveaux-paradigmes/CMS$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
51c6dd08e48   mysql                                "docker-entrypoint.s..." 23 hours ago   Restarting (1) 1 second ago   3306/tcp        automatisa-tion-exercice-2-td-database-1
8f4e8f289d84   directus/directus:10.10.5          "docker-entrypoint.s..." 7 days ago     Up 5 seconds   0.0.0.0:8055->8055/tcp        cms-directus-1
514ebae5c3     postgres:16                        "docker-entrypoint.s..." 7 days ago     Up 5 seconds   5432/tcp                        cms-db-1
ml@mlrobin-drtinater: /dev/nouveaux-paradigmes/CMS$
```

Ce ter-

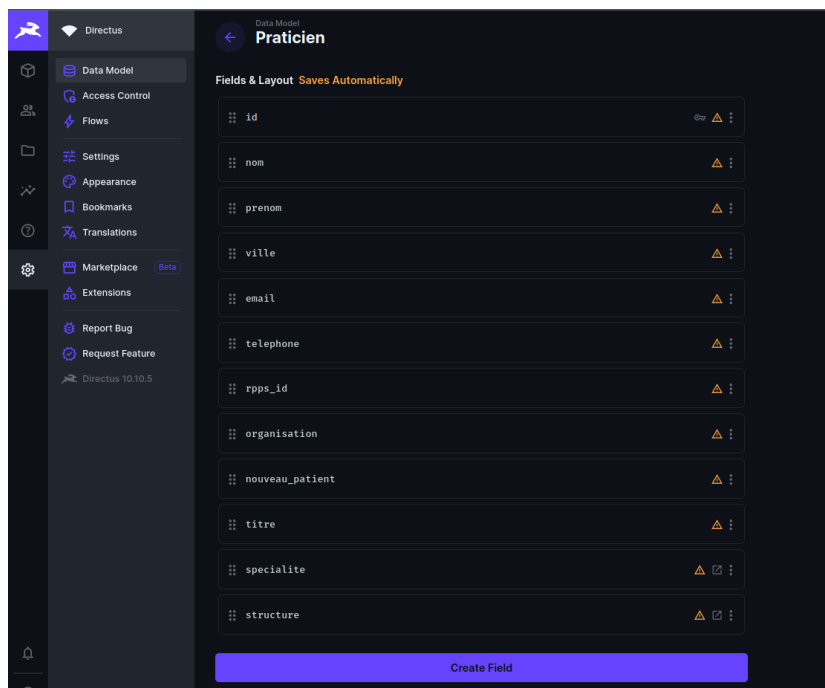
minial montre le lancement via `docker compose up -d` puis la verification avec `docker ps`. On constate que Directus est actif (port 8055) ainsi que la base de donnees. Une tentative initiale a signale un daemon Docker non disponible, puis le demarrage a reussi.

Modele de donnees

Le modele suit le schema du domaine "praticiens de sante". Les collections principales sont : - `praticien` - `specialite` - `structure` - `motif_visite` - `moyen_paiement` - tables de jonction `praticien2motif` et `praticien2moyen`



Cette vue liste toutes les collections définies dans Directus, confirmant l'existence du modèle complet et la préparation des relations.



Les champs principaux du praticien sont visibles, ainsi que les relations vers specialite et structure. Cela valide la modélisation M2O attendue.

Chargement des données (CSV)

Les données sont importées depuis des fichiers CSV avec séparateur ; afin de reprendre un jeu existant.

```
id;nom;prenom;ville;email;telephone;rpps_id;organisation;nouveau_patient;titre;specialite;structure
1;Martin;Claire;Nancy;claire.martin@example.org;0610101010;RPPS1001;false;true;Dr;1;1
2;Durand;Luc;Villers-sur-Mer;luc.durand@example.org;0620202020;RPPS1002;false;false;Dr;2;2
3;Petit;Sarah;Luneville;sarah.petit@example.org;0630303030;RPPS1003;true;true;Dr;2;3
4;Bernard;Alex;Nancy;alex.bernard@example.org;0640404040;RPPS1004;false;true;Dr;3;1
```

L'extrait montre les colonnes attendues (nom, prenom, ville, email, tele-

phone, etc.) et les clés d'association (specialite, structure). Cela garantit la compatibilité avec les relations du modèle.

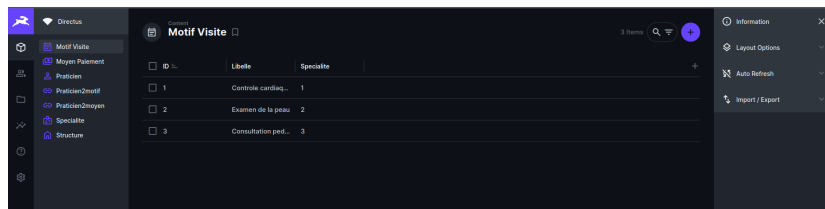


ID	Nom	Prenom	Ville
1	Martin	Claire	Nancy
2	Durand	Luc	Villars-sur-Mer
3	Petit	Sarah	Lunelville
4	Bernard	Alex	Nancy
5	Dupont	Alice	Paris

La liste des praticiens confirme que l'import a été effectué correctement et que les enregistrements sont consultables dans l'interface.

Verification dans le backoffice

Après import, les collections sont consultables et navigables directement via l'interface.



ID	Libelle	Specialite
1	Contrôle cardiaque	1
2	Examen de la peau	2
3	Consultation ped...	3

Cet écran illustre l'accès aux collections et aux items, confirmant la disponibilité des données dans le backoffice.

API REST (exposition des données)

Base URL (exemple local) : `http://localhost:8055`

Si un token est utilisé :

Authorization: Bearer <TOKEN>

Requetes REST réalisées (conformes aux exigences du TD) :

1) Liste des praticiens

```
curl -H "Authorization: Bearer <TOKEN>" \
  "http://localhost:8055/items/praticien?fields=id,nom,prenom,ville"
```

2) Specialite d'ID 2

```
curl -H "Authorization: Bearer <TOKEN>" \
  "http://localhost:8055/items/specialite/2"
```

3) Specialite d'ID 2 avec uniquement le libelle

```
curl -H "Authorization: Bearer <TOKEN>" \
  "http://localhost:8055/items/specialite/2?fields=libelle"
```

4) Un praticien avec sa specialite (libelle)

```
curl -H "Authorization: Bearer <TOKEN>" \
  "http://localhost:8055/items/praticien/<ID>?fields=nom,prenom,specialite.libelle"
```

5) Une structure (nom, ville) avec la liste de ses praticiens

```
curl -H "Authorization: Bearer <TOKEN>" \
  "http://localhost:8055/items/structure/<ID>?fields=nom,ville,praticiens.nom,praticiens.pre"
```

6) Idem avec le libelle de la specialite des praticiens

```
curl -H "Authorization: Bearer <TOKEN>" \
  "http://localhost:8055/items/structure/<ID>?fields=nom,ville,praticiens.nom,praticiens.pre"
```

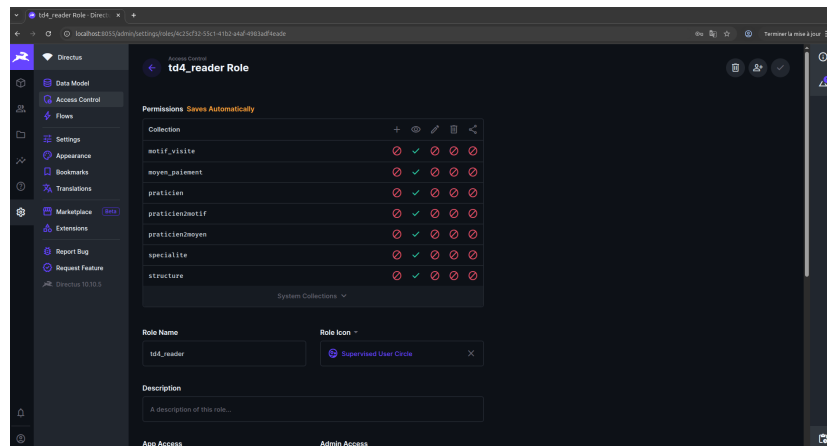
7) Structures dont la ville contient “sur”, avec la liste des praticiens

```
curl -H "Authorization: Bearer <TOKEN>" \
  "http://localhost:8055/items/structure?filter[ville][_contains]=sur&fields=nom,ville,prati"
```

Ces requetes valident l'exposition REST attendue, la selection de champs (fields) pour limiter les donnees retournees et la navigation dans les relations.

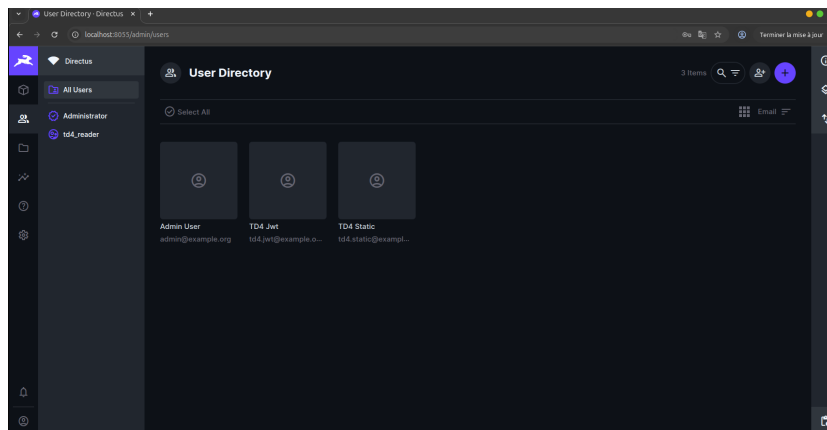
Annexes - Securite et GraphQL (preparation TD4)

Les captures suivantes illustrent des actions realisees en parallele pour preparer l'API GraphQL et les droits d'accès (hors perimetre strict du TD3, mais utiles pour la suite).

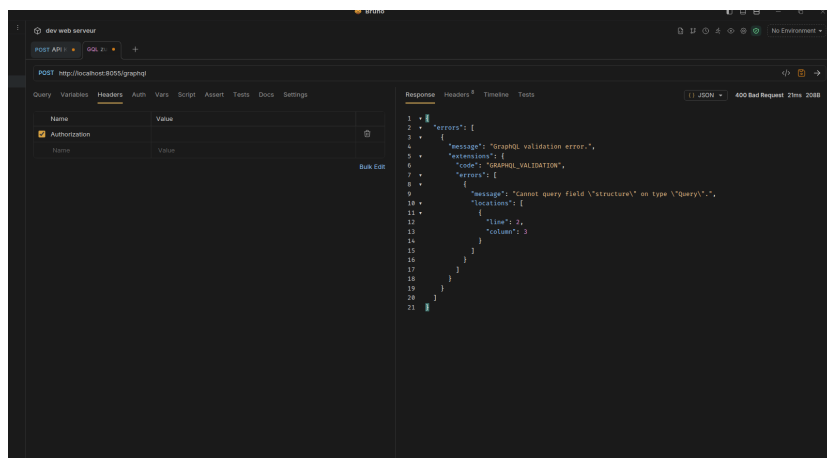


Cre-

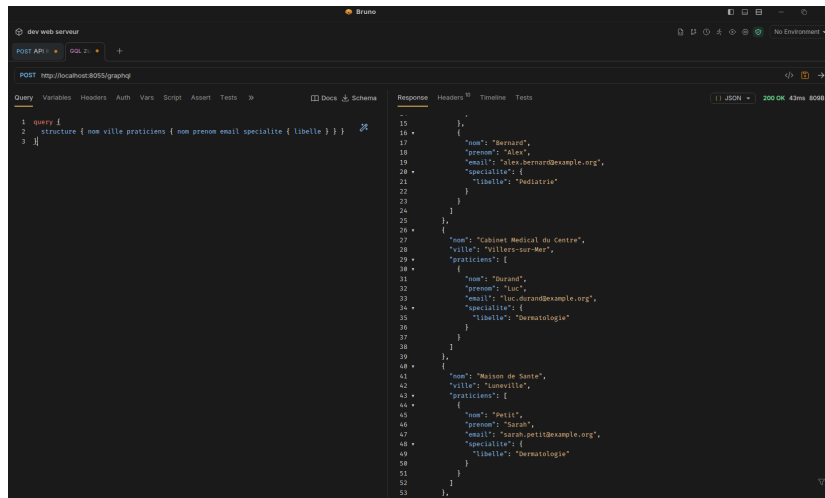
ation d'un role dedie avec les droits de lecture sur les collections necessaires. Cela sert a controler l'accès a l'API.



Deux comptes sont associés au rôle : un utilisateur avec token statique et un utilisateur pour JWT, afin de tester l'authentification.

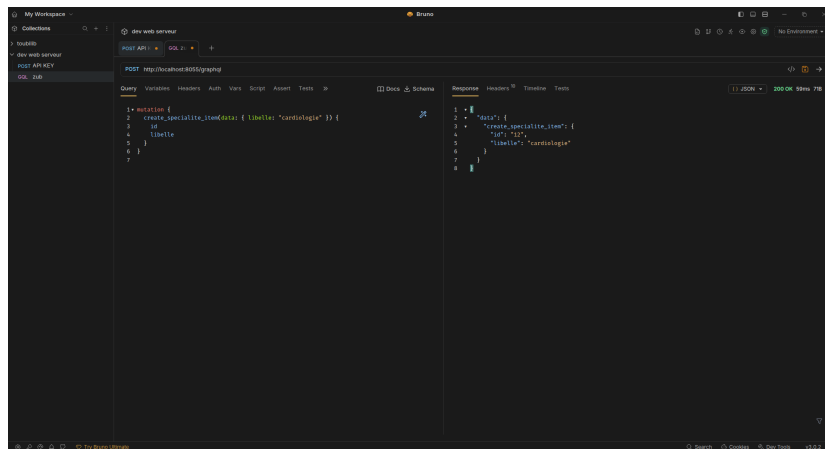


Test d'accès sans authentification : la requête échoue, ce qui confirme que les permissions publiques sont bien restreintes.



Verifi-

cation d'une requete GraphQL renvoyant des structures et leurs praticiens avec specialite. Cela valide la navigation dans les relations cote API.



Exem-

ple de mutation (creation d'une specialite) en tant qu'admin, pour verifier les operations d'écriture.

Conclusion

Le CMS headless Directus a ete deploye avec succes, le modele de donnees a ete construit et alimente via CSV, puis verifie dans le backoffice. L'API REST permet de consommer les donnees avec selection de champs et navigation relationnelle. Les captures annexes montrent en plus la mise en place des roles et l'utilisation de GraphQL pour preparer la suite du projet. L'ensemble constitue un rendu clair, exploitable et conforme aux objectifs du TD.