x

Multitenancy in Directus

Learnings from a real-life application

Overview

What is multitenancy?

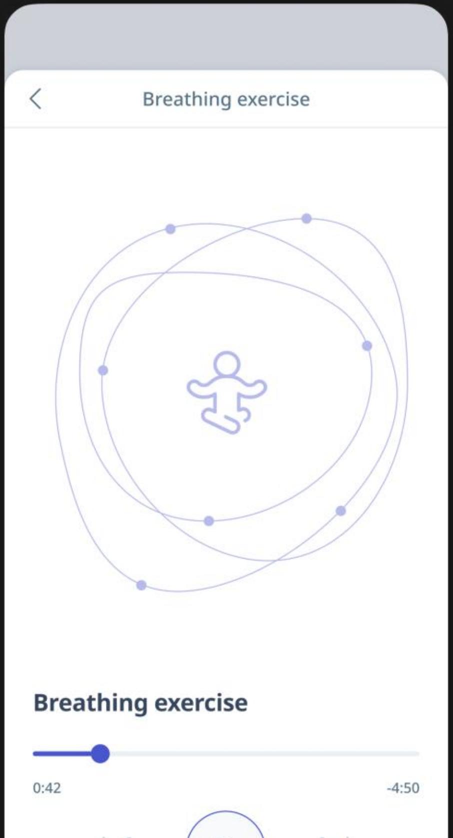
"In multi-tenant software architecture — also called software multitenancy — a single instance of a software application (and its underlying database and hardware) serves multiple tenants (or user accounts)"

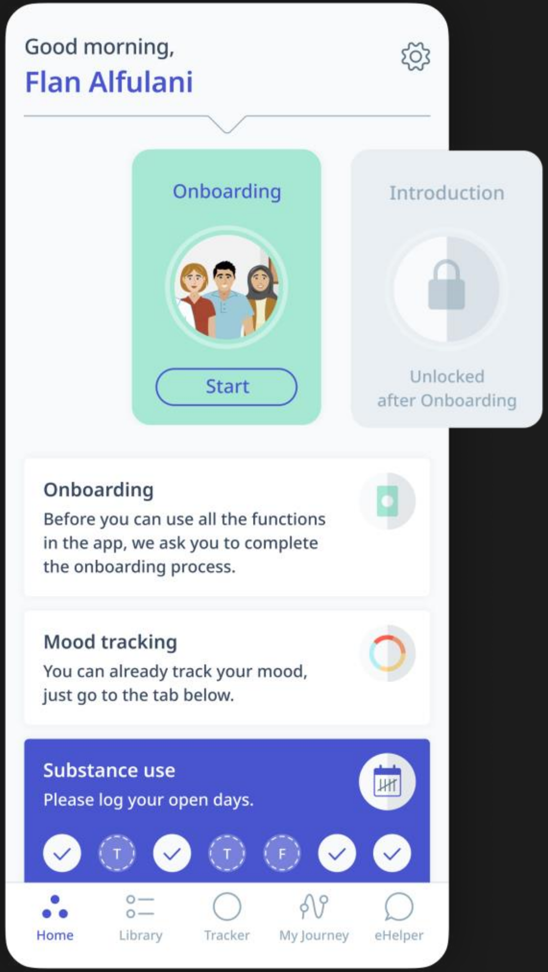
- IBM

Why you might need multitenancy

- To lower costs ↙
- To make the barrier of entry for your product lower by eliminating infrastructure setup from the equation ↙
- To make it easier to aggregate data

DIRECT - A case study

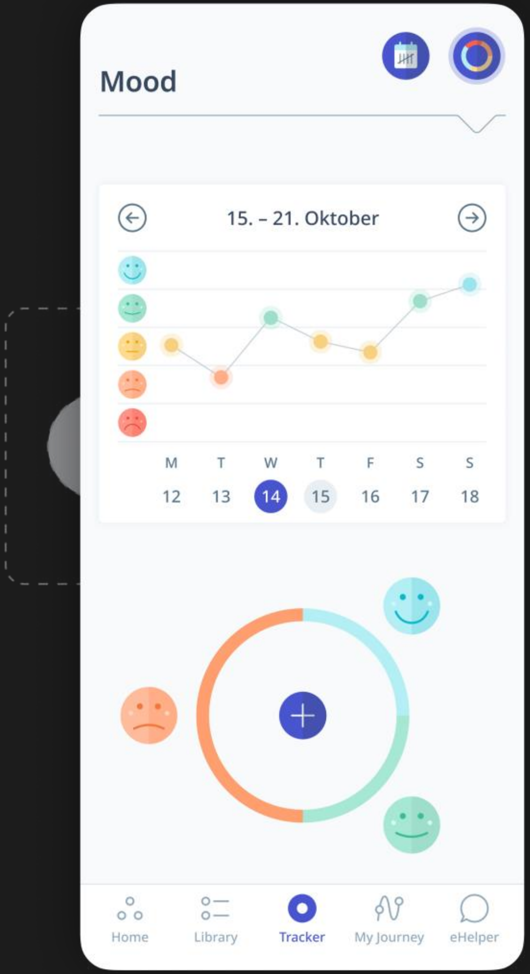



Good morning,
Flan Alfulani

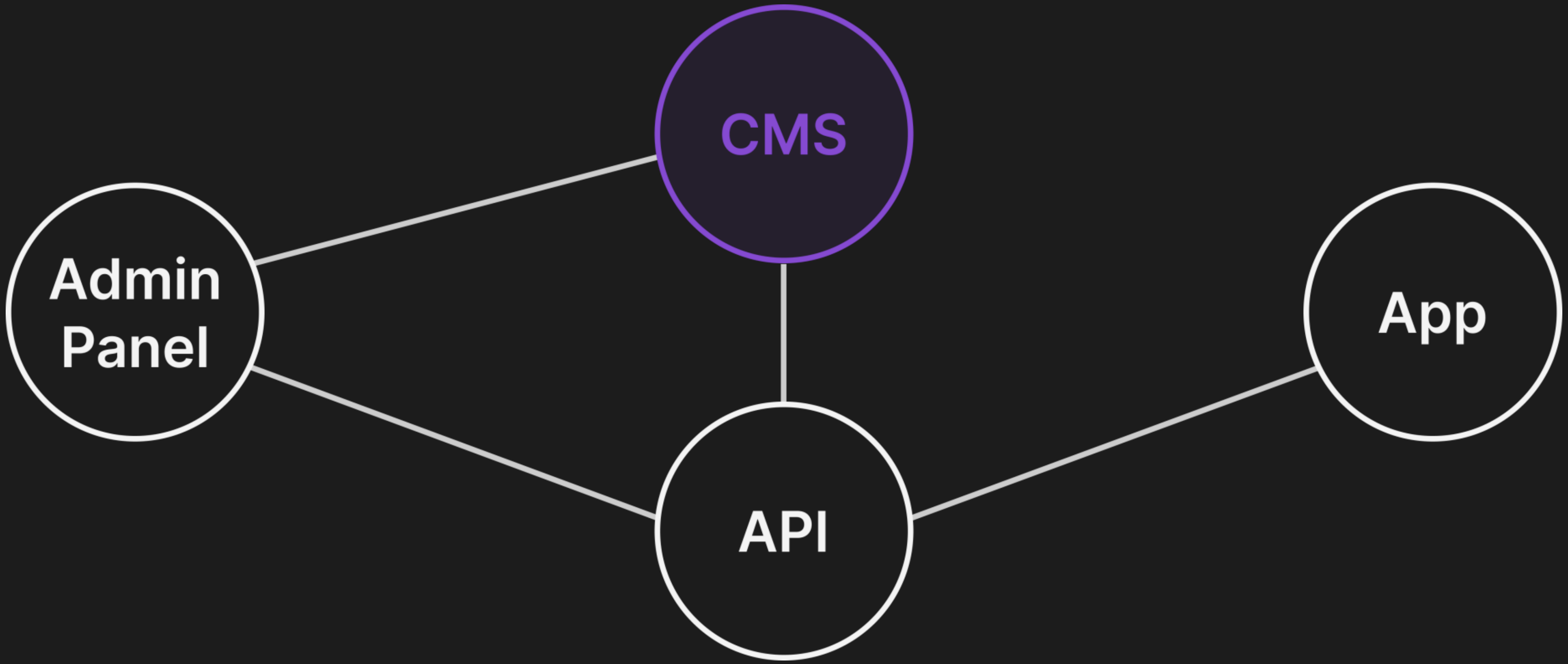
Onboarding
Before you can use all the functions in the app, we ask you to complete the onboarding process.

Mood tracking
You can already track your mood, just go to the tab below.

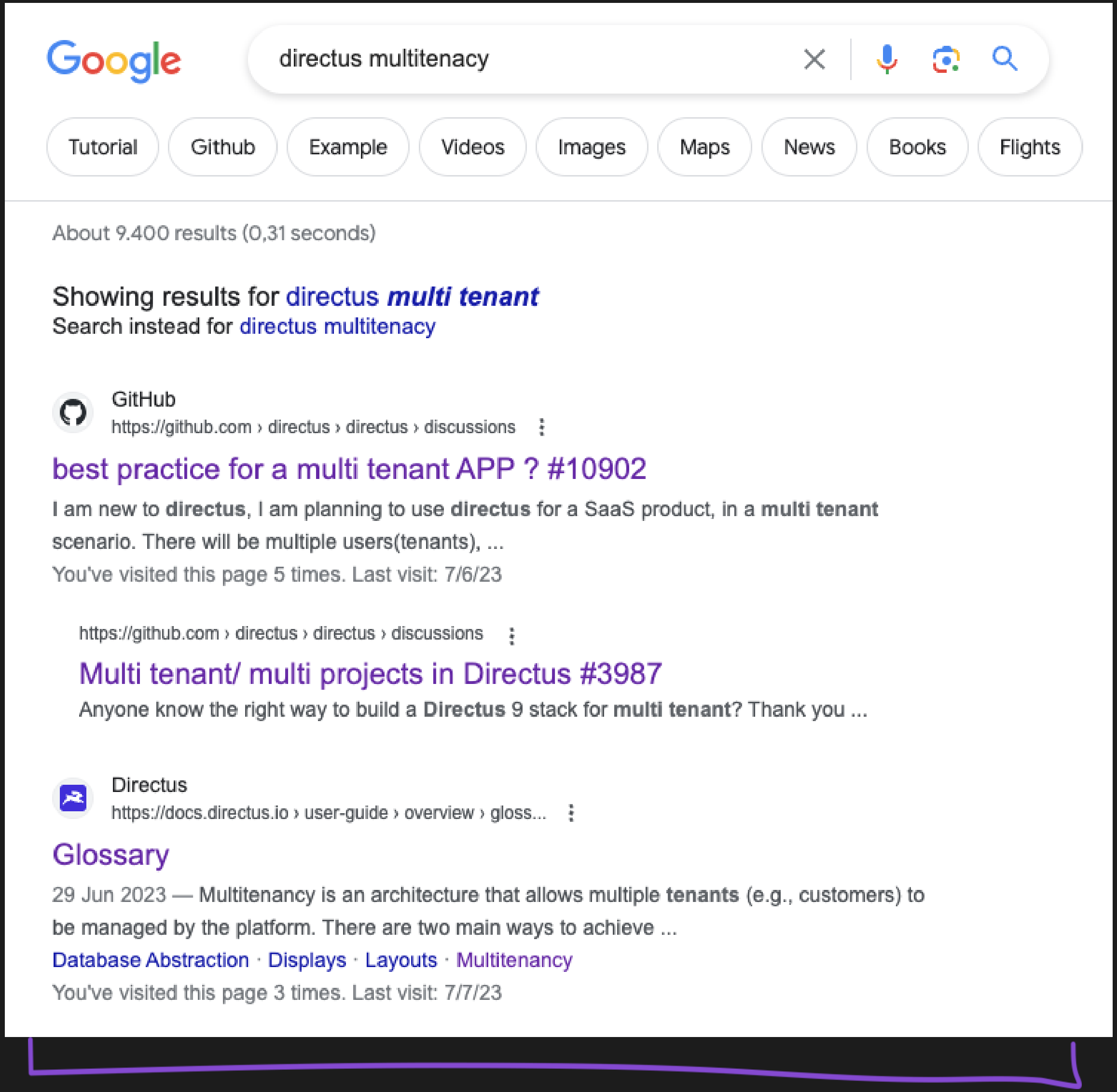
Substance use
Please log your open days.



Multitenancy beyond Directus



Getting started!



About 9.400 results (0,31 seconds)

Showing results for directus ***multi tenant***
Search instead for directus multitenacy

GitHub
https://github.com › directus › directus › discussions

best practice for a multi tenant APP ? #10902

I am new to **directus**, I am planning to use **directus** for a SaaS product, in a **multi tenant** scenario. There will be multiple users(tenants), ...
You've visited this page 5 times. Last visit: 7/6/23

https://github.com › directus › directus › discussions

Multi tenant/ multi projects in Directus #3987

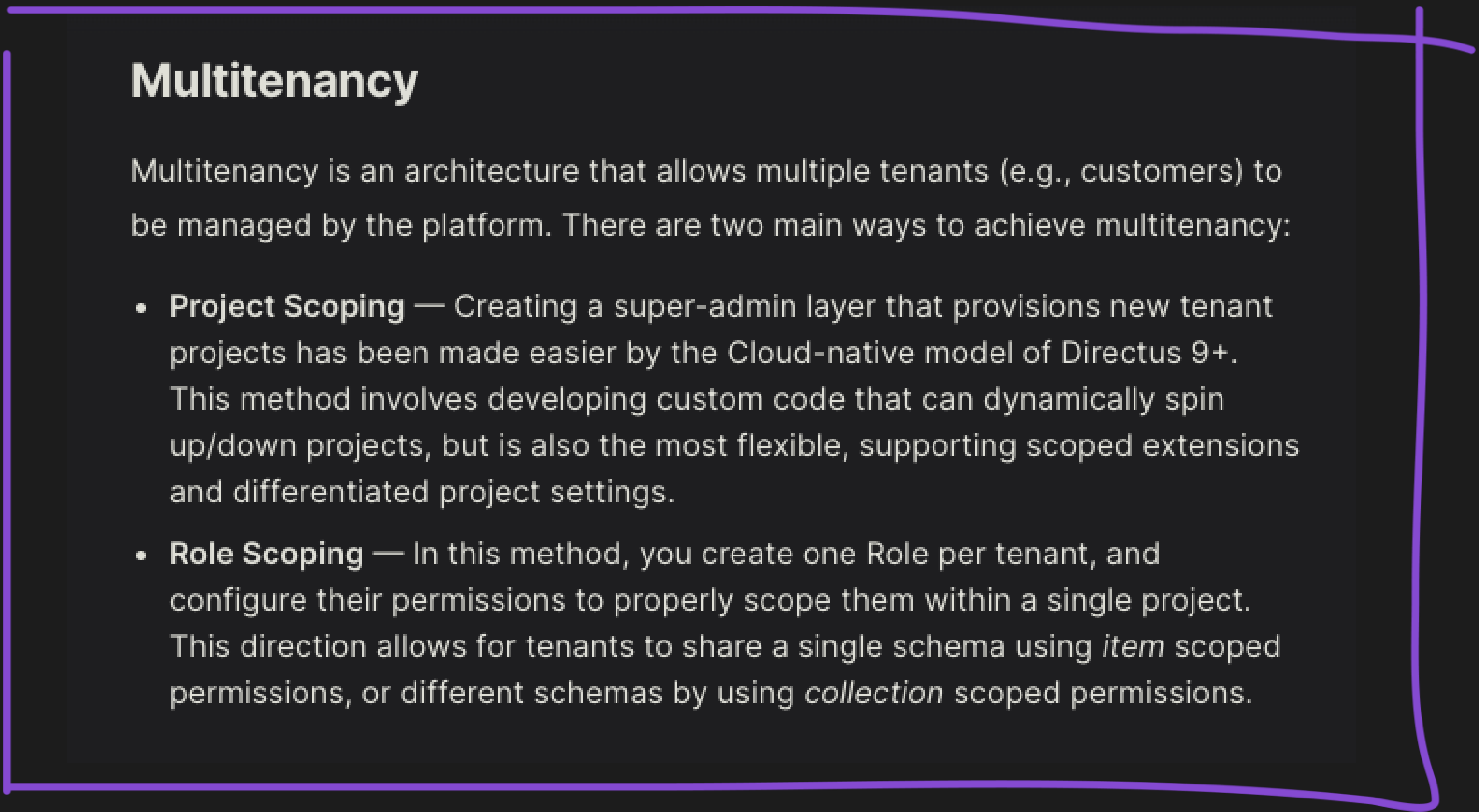
Anyone know the right way to build a **Directus** 9 stack for **multi tenant**? Thank you ...

Directus
https://docs.directus.io › user-guide › overview › gloss...

Glossary

29 Jun 2023 — Multitenancy is an architecture that allows multiple **tenants** (e.g., customers) to be managed by the platform. There are two main ways to achieve ...
Database Abstraction · Displays · Layouts · Multitenancy
You've visited this page 3 times. Last visit: 7/7/23

Multitenancy as defined by Directus



Multitenancy

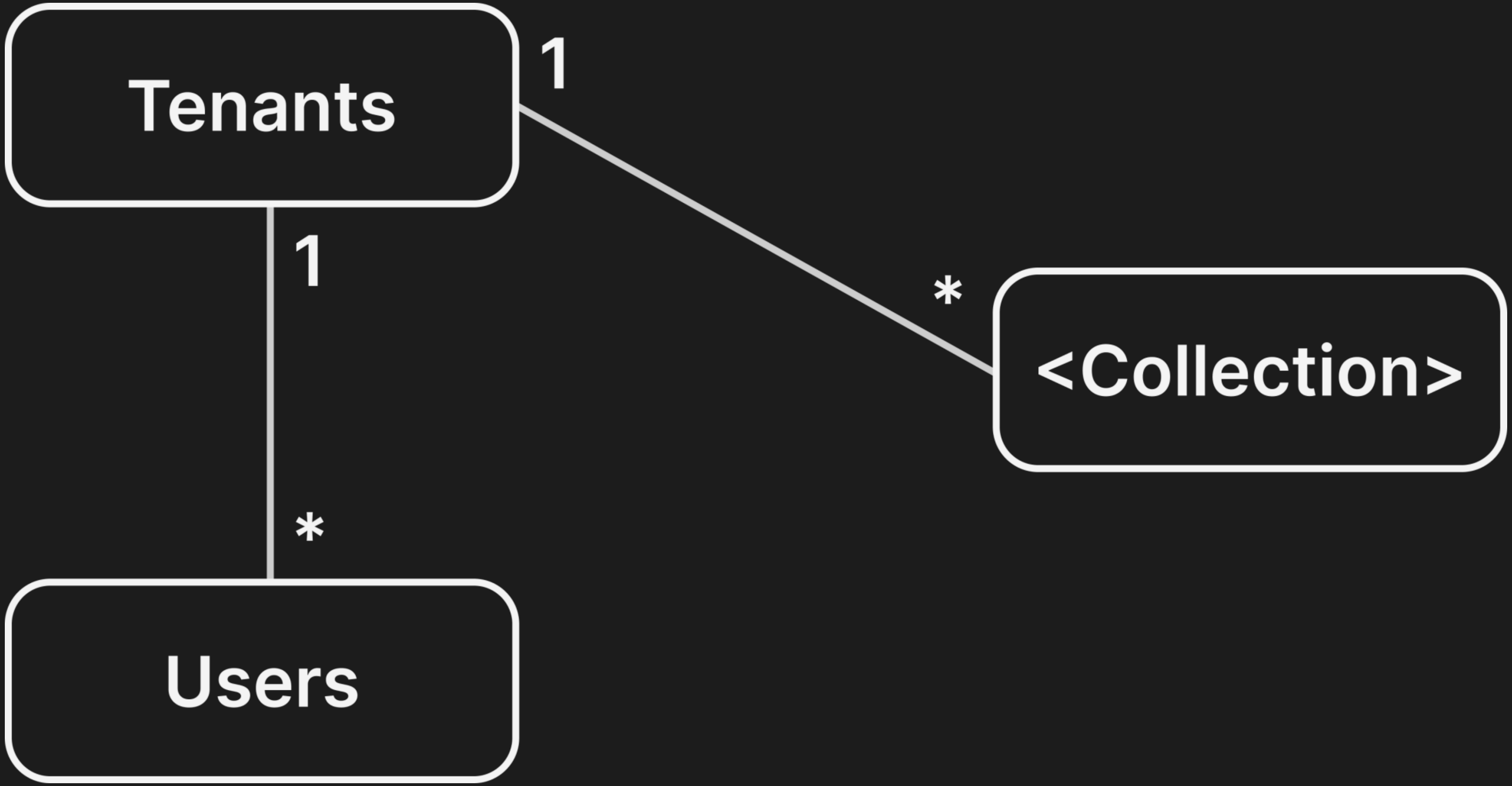
Multitenancy is an architecture that allows multiple tenants (e.g., customers) to be managed by the platform. There are two main ways to achieve multitenancy:

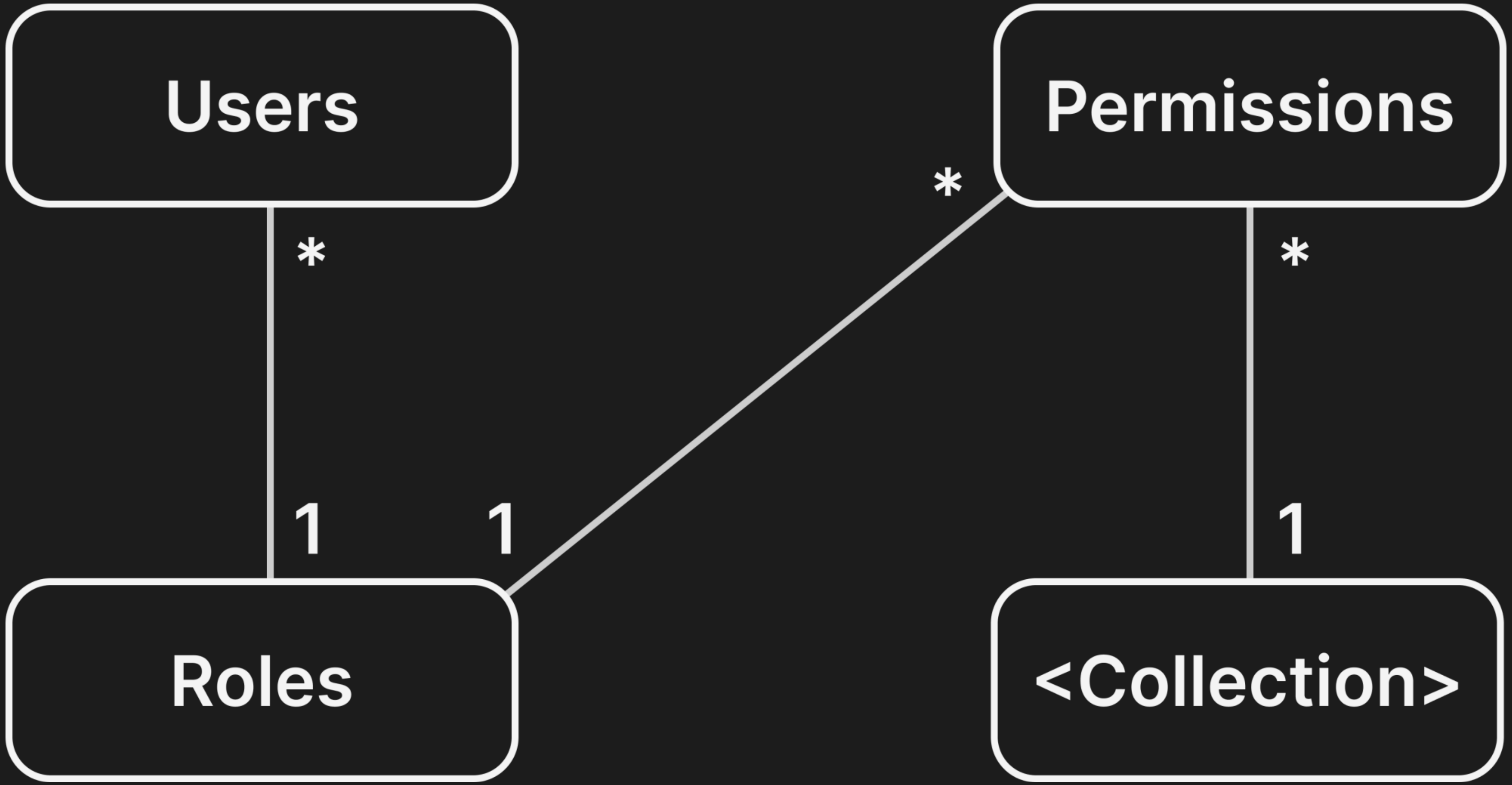
- **Project Scoping** — Creating a super-admin layer that provisions new tenant projects has been made easier by the Cloud-native model of Directus 9+. This method involves developing custom code that can dynamically spin up/down projects, but is also the most flexible, supporting scoped extensions and differentiated project settings.
- **Role Scoping** — In this method, you create one Role per tenant, and configure their permissions to properly scope them within a single project. This direction allows for tenants to share a single schema using *item* scoped permissions, or different schemas by using *collection* scoped permissions.

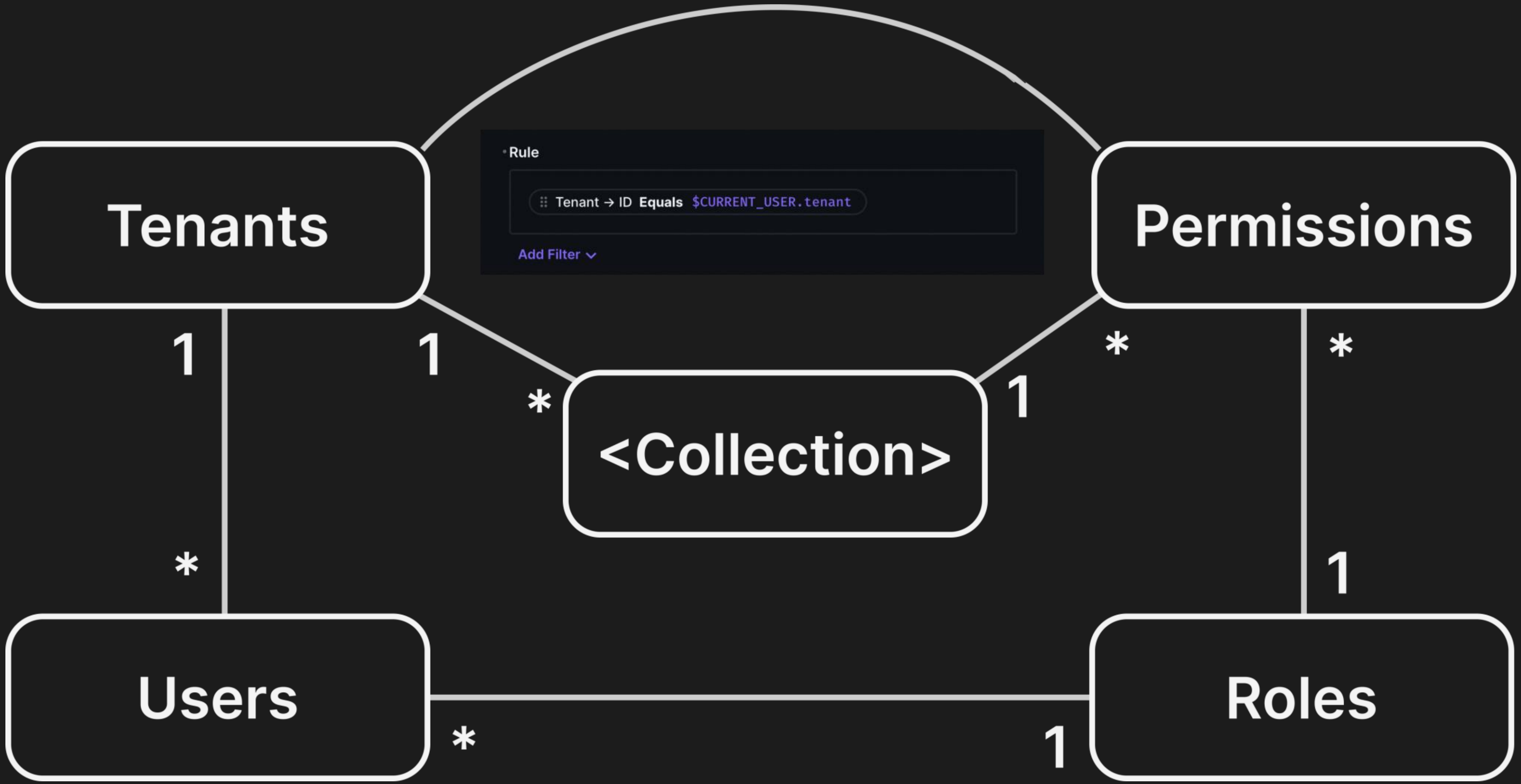
Implementing multitenancy

- In our approach, **tenant-based permissions** are the core of the logic;

- Custom permissions allow us to filter items in a collection based on the tenant assigned to the requesting user and the requested item.


Tenants
1
1
*
<Collection>
*
Users


Users
Permissions
*
*
*
1
1
1
Roles
<Collection>



```
Tenant → ID Equals $CURRENT_USER.tenant
```

Automating permissions

```
const tenantCollections = (await directus.fields.readAll())
  .filter((item) => item.field === 'tenant')
  .map((item) => item.collection);
```

Automating permissions

```
const permissions = [];

// ...


tenantCollections.forEach((collectionName: string) => {
  permissions.push({
    role: userRole,
    collection: collectionName,
    action: 'create',
    permissions: null,
    validation: null,
    presets: { tenant: { id: "$CURRENT_USER.tenant" } },
    fields: ['*']
  });
  // ...
  permissions.push({
    role: userRole,
    collection: collectionName,
    action: 'update',
    permissions: { _and: [{ tenant: { id: { _eq: "$CURRENT_USER.tenant" } } }] },
    validation: null,
    presets: { tenant: { id: "$CURRENT_USER.tenant" } },
    fields: ['*']
  });
});

// ...
// Make sure you're not creating duplicate permissions
// ...

await directus.permissions.createMany(permissions);
```

Automating permissions

```javascript
const permissions = [];

// ...

// directus_folders
permissions.push({
  role: userRole,
  collection: 'directus_folders',
  action: 'create',
  permissions: null,
  validation: null,
  presets: { tenant: { id: "$CURRENT_USER.tenant" } },
  fields: ['*']
});
permissions.push({
  role: userRole,
  collection: 'directus_folders',
  action: 'update',
  permissions: { _and: [{ tenant: { id: { _eq: "$CURRENT_USER.tenant" } } }] },
  validation: null,
  presets: { tenant: { id: tenant } },
  fields: ['*']
});

// ...
// Make sure you're not creating duplicate permissions
// ...

await directus.permissions.createMany(permissions);
```

Downsides

- ↘ Complexity
- ↘ Data security

Other topics

- Guaranteeing uniqueness across tenants
- Tenancy outside of Directus
- ...

Other topics

- …
- Keeping permissions up to date
- Role-based vs. User-based tenancy
- …

Thank you!

mariana.costa@hybridheroes.de

Overview

References

- https://github.com/directus/directus/discussions/3987
- https://github.com/directus/directus/discussions/10902
- https://www.ibm.com/topics/multi-tenant
- https://www.youtube.com/watch?v=ruzkqxDIG-Y
- https://en.wikipedia.org/wiki/Multitenancy
- https://github.com/directus/directus/discussions/9682
- https://github.com/directus/directus/discussions/2687

Appendix A - A very brief guide to multitenancy

- Setup a “Tenants” collection
- Add `tenant` field to collections
- Add `tenant` field to users
- Add at least one role for which you can set its permissions
- Update the permissions to limit access to content by checking if the tenant of the requested item matches that of the requesting user