



# Access Management orienté métier avec ReBAC

Infuser du métier dans vos autorisations



# Présentation de l'équipe



**Pauline Jamin**

Software engineer @Agicap  
Grenoble 🏔️



**Geoffroy Braun**

Software engineer @Agicap  
Meistratzheim 🥨



# Sommaire

1. Découvrir le Relation-Based Access Control
2. Découper un scénario utilisateur
3. Découvrir l'Access Management
4. Implémenter des scénarii utilisateur
5. Take aways



# Découvrir le Relation-Based Access Control

Concept inventé par Google

Papier blanc publié en 2019

Tout est *type* ...

Tout est *objet* ...

Tout est *relation* ...

Tout est modélisable ...

L'ensemble s'appelle un tuple

USER	user:Randy
RELATION	owner
OBJECT	car:SUV



# Découper un scénario utilisateur

Tout est une question de structure :

- quels sont les types ?
- quels sont les objets ?
- quelles sont les relations ?



# Découper un scénario utilisateur

Tout est une question de structure :

- quels sont les types ?
- quels sont les objets ?
- quelles sont les relations ?

Given a user Randy

When user Randy buys a car SUV

Then Randy is owner of the car SUV



# Découper un scénario utilisateur

Tout est une question de structure :

- quels sont les types ?
- quels sont les objets ?
- quelles sont les relations ?

Given a user:Randy Object

When user Randy buys a car:SUV Object

Then Randy is owner of the car SUV  
Relation



# Découper un scénario utilisateur

Tout est une question de structure :

- quels sont les types ? *Logique métier*
- quels sont les objets ?
- quelles sont les relations ?

Given a user:Randy *Object*

~~When user Randy buys a~~ car:SUV *Object*

Then Randy is owner of the car SUV  
*Relation*





# Découvrir l'Access Management

OpenFGA

Implémentation libre de  
Google Zanzibar par Auth0

Interface en ligne appelée  
*Playground*

The screenshot displays the Auth0 OpenFGA Playground interface. The top navigation bar includes the Auth0 logo, the text "fine grained authorization", a dropdown menu set to "github", and links for "JOIN THE COMMUNITY", "DOCS", "TAKE A TOUR", "NEW STORE", and "SIGN UP".

The main interface is divided into three sections:

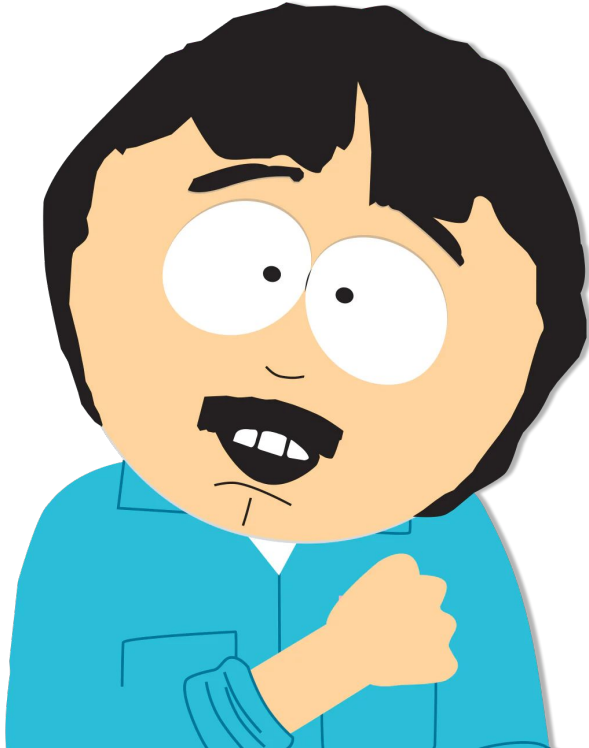
- AUTHORIZATION MODEL (4):** A code editor showing the OpenFGA model in DSL. The model defines types for user, team, repo, and organization, and their relationships. The code is as follows:

```
1 model
2   schema 1.1
3   type user
4   type team
5   relations
6     define member: [user,team#member]
7   type repo
8   relations
9     define admin: [user,team#member] or repo_admin from owner
10    define maintainer: [user,team#member] or admin
11    define owner: [organization]
12    define reader: [user,team#member] or triager or repo_reader from
13  owner
14  define triager: [user,team#member] or writer
15  define writer: [user,team#member] or maintainer or repo_writer
16  from owner
17  type organization
18  relations
19    define member: [user] or owner
20    define owner: [user]
```
- Tuples (9):** A list of assertions (tuples) for the "github" store. The tuples are:
  - USER: user:erik, RELATION: member, OBJECT: organization:openfga
  - USER: organization:openfga#member, RELATION: repo\_admin, OBJECT: organization:openfga
  - USER: team:openfga/core#member, RELATION: admin, OBJECT: repo:openfga/openfga
  - USER: organization:openfga, RELATION: owner, OBJECT: repo:openfga/openfga
- TYPES PREVIEW:** A graph visualization showing the relationships between types. The graph includes nodes for user, team, repo, organization, owner, admin, maintainer, triager, reader, repo\_reader, repo\_admin, repo\_writer, and member. The graph shows how these types are related to each other based on the model and tuples.



# Implémenter des scénarii utilisateur

## 1. Devenir propriétaire d'un véhicule



Given a user Randy

When user Randy buys a car SUV

Then Randy is owner of car SUV



# Implémenter des scenarii utilisateur

## 1. Devenir propriétaire d'un véhicule

*Given a user Randy*

*When user Randy buys a car SUV*

*Then Randy is owner of car SUV*

Déclarez les types `user` et `car`

model

schema 1.1

type `user`

type `car`

relations

define `owner`: [`user`]



# Implémenter des scenarii utilisateur

## 1. Devenir propriétaire d'un véhicule

*Given a user Randy*

*When user Randy buys a car SUV*

*Then Randy is owner of car SUV*

Déclarez les types **user** et **car**

Ajoutez un test ...

USER	user:Randy
RELATION	owner
OBJECT	car:SUV
ALLOWED	true



# Implémenter des scenarii utilisateur

## 1. Devenir propriétaire d'un véhicule

*Given a user Randy*

*When user Randy buys a car SUV*

*Then Randy is owner of car SUV*

Déclarez les types **user** et **car**

Ajoutez un test ...

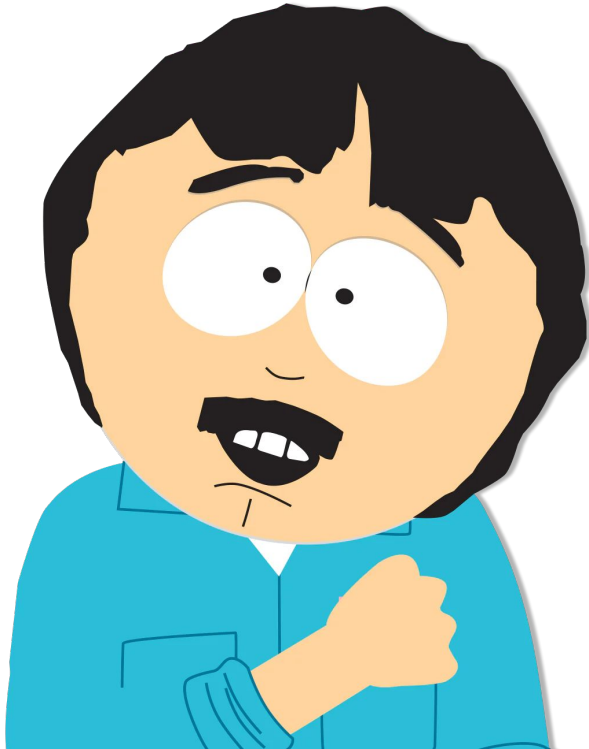
... Faites-le passer *au vert*

USER	user:Randy
RELATION	owner
OBJECT	car:SUV



# Implémenter des scénarii utilisateur

## 2. Conduire une voiture



Given a user Randy

When user Randy buys a car SUV

Then Randy is owner of car SUV

And Randy *de facto* is driver of car SUV



# Implémenter des scenarii utilisateur

## 2. Conduire une voiture

*Given a user Randy*

*When user Randy buys a car SUV*

*Then Randy is owner of car SUV*

*And Randy de facto is driver of car SUV*

Utilisez une relation **concentrique**

model

schema 1.1

type **user**

type **car**

relations

define **owner**: [**user**]

define **driver**: **owner**



# Implémenter des scénarii utilisateur

## 2. Conduire une voiture

*Given a user Randy*

*When user Randy buys a car SUV*

*Then Randy is owner of car SUV*

*And Randy de facto is driver of car SUV*

Utilisez une relation **concentrique**

Ajoutez un test ...

USER	user:Randy
RELATION	driver
OBJECT	car:SUV
ALLOWED	true





# Implémenter des scenarii utilisateur

## 2. Conduire une voiture

*Given a user Randy*  
*When user Randy buys a car SUV*  
*Then Randy is owner of car SUV*  
*And Randy de facto is driver of car SUV*

Utilisez une relation **concentrique**

Ajoutez un test ...

... Faites-le passer *au vert*

<del>USER</del>	<del>user:Randy</del>
<del>RELATION</del>	<del>driver</del>
<del>OBJECT</del>	<del>car:SUV</del>



# Implémenter des scenarii utilisateur

## 3. Laisser sa femme conduire la voiture



Given a user Randy

And Randy is owner of the car SUV

And a user Sharon

When user Randy marries user Sharon

Then they both are members of family Marsh

And Sharon *therefore* is driver of car SUV



# Implémenter des scenarii utilisateur

## 3. Laisser sa femme conduire la voiture

*Given a user Randy*

*And Randy is owner of the car SUV*

*And a user Sharon*

*When user Randy marries user Sharon*

*Then they both are members of family Marsh*

*And Sharon therefore is driver of car SUV*

Déclarez le type **family** comme **driver**

model

schema 1.1

type **user**

type **car**

relations

define **owner**: [**user**]

define **driver**: [**family**] or **owner**

type **family**

relations

define **member**: [**user**]



# Implémenter des scenarii utilisateur

## 3. Laisser sa femme conduire la voiture

*Given a user Randy*

*And Randy is owner of the car SUV*

*And a user Sharon*

*When user Randy marries user Sharon*

*Then they both are members of family Marsh*

*And Sharon therefore is driver of car SUV*

Déclarez le type **family** comme **driver**

Ajoutez un test ...

USER	user:Sharon
RELATION	driver
OBJECT	car:SUV
ALLOWED	true



# Implémenter des scenarii utilisateur

## 3. Laisser sa femme conduire la voiture

*Given a user Randy*

*And Randy is owner of the car SUV*

*And a user Sharon*

*When user Randy marries user Sharon*

*Then they both are members of family Marsh*

*And Sharon therefore is driver of car SUV*

Déclarez le type **family** comme **driver**

Ajoutez un test ...

... Faites-le passer *au vert*

USER	user:Randy
RELATION	member
OBJECT	family:Marsh

USER	user:Sharon
RELATION	member
OBJECT	family:Marsh



# Implémenter des scenarii utilisateur

## 3. Laisser sa femme conduire la voiture

*Given a user Randy*

*And Randy is owner of the car SUV*

*And a user Sharon*

*When user Randy marries user Sharon*

*Then they both are members of family Marsh*

*And Sharon therefore is driver of car SUV*

Déclarez les **member** du type **family**

```
model
  schema 1.1
```

```
type user
```

```
type car
  relations
    define owner: [user]
    define driver: [family#member] or owner
```

```
type family
  relations
    define member: [user]
```



# Implémenter des scenarii utilisateur

## 3. Laisser sa femme conduire la voiture

*Given a user Randy*

*And Randy is owner of the car SUV*

*And a user Sharon*

*When user Randy marries user Sharon*

*Then they both are members of family Marsh*

*And Sharon therefore is driver of car SUV*

Déclarez les **member** du type **family**

Ajoutez un test ...

USER	user:Sharon
RELATION	driver
OBJECT	car:SUV
ALLOWED	true



# Implémenter des scenarii utilisateur

## 3. Laisser sa femme conduire la voiture

*Given a user Randy*

*And Randy is owner of the car SUV*

*And a user Sharon*

*When user Randy marries user Sharon*

*Then they both are members of family Marsh*

*And Sharon therefore is driver of car SUV*

Déclarez les **member** du type **family**

Ajoutez un test ...

... Faites-le passer *au vert*

USER	family:Marsh#member
RELATION	driver
OBJECT	car:SUV





# Implémenter des scénarii utilisateur

## 4. Interdire à son fils de conduire la voiture



Given a user Randy

And Randy is owner of the car SUV

And a user Sharon

And user Randy marries user Sharon

When users Randy and Sharon have a child Stan

Then Stan is member of family Marsh

And Stan is not driver of car SUV



# Implémenter des scenarii utilisateur

## 5. Interdire à son fils de conduire la voiture

*Given a user Randy*

*And Randy is owner of the car SUV*

*And a user Sharon*

*And user Randy marries user Sharon*

*When users Randy and Sharon have a child*

*Stan*

*Then Stan is member of family Marsh*

*And Stan is not driver of car SUV*

Etendez la relation **member** du type **family**

```
model
  schema 1.1
```

```
type user
```

```
type car
```

```
  relations
```

```
    define owner: [user]
```

```
    define driver: [family#member] or
                  [family#parent] or owner
```

```
type family
```

```
  relations
```

```
    define member: [user] or parent or child
```

```
    define parent: [user]
```

```
    define child: [user]
```



# Implémenter des scenarii utilisateur

## 5. Interdire à son fils de conduire la voiture

*Given a user Randy*  
*And Randy is owner of the car SUV*  
*And a user Sharon*  
*And user Randy marries user Sharon*  
*When users Randy and Sharon have a child Stan*  
*Then Stan is member of family Marsh*  
*And Stan is not driver of car SUV*

Etendez la relation **member** du type **family**

Ajouter un test ...

USER	user:Stan
RELATION	driver
OBJECT	car:SUV
ALLOWED	false



# Implémenter des scenarii utilisateur

## 5. Interdire à son fils de conduire la voiture

*Given a user Randy*  
*And Randy is owner of the car SUV*  
*And a user Sharon*  
*And user Randy marries user Sharon*  
*When users Randy and Sharon have a child Stan*  
*Then Stan is member of family Marsh*  
*And Stan is not driver of car SUV*

Etendez la relation **member** du type **family**

Ajouter un test ...

... Faites-le passer *au vert*

USER	user:Randy
RELATION	parent
OBJECT	family:Marsh

USER	user:Sharon
RELATION	parent
OBJECT	family:Marsh



# Implémenter des scenarii utilisateur

## 5. Interdire à son fils de conduire la voiture

*Given a user Randy*  
*And Randy is owner of the car SUV*  
*And a user Sharon*  
*And user Randy marries user Sharon*  
*When users Randy and Sharon have a child*  
*Stan*  
*Then Stan is member of family Marsh*  
*And Stan is not driver of car SUV*

Etendez la relation **member** du type **family**

Ajouter un test ...

... Faites-le passer *au vert*

USER	user:Stan
RELATION	child
OBJECT	family:Marsh

USER	family:Marsh#parent
RELATION	driver
OBJECT	car:SUV



# Implémenter des scenarii utilisateur

## 5. Interdire à son fils de conduire la voiture

*Given a user Randy*  
*And Randy is owner of the car SUV*  
*And a user Sharon*  
*And user Randy marries user Sharon*  
*When users Randy and Sharon have a child*  
*Stan*  
*Then Stan is member of family Marsh*  
*And Stan is not driver of car SUV*

Pensez à nettoyer votre modèle ...

```
model
  schema 1.1

  type user

  type car
    relations
      define owner: [user]
      define driver: [family#member] or
        [family#parent] or owner

  type family
    relations
      define member: [user] or parent or child
      define parent: [user]
      define child: [user]
```



# Implémenter des scenarii utilisateur

## 5. Interdire à son fils de conduire la voiture

*Given a user Randy*  
*And Randy is owner of the car SUV*  
*And a user Sharon*  
*And user Randy marries user Sharon*  
*When users Randy and Sharon have a child*  
*Stan*  
*Then Stan is member of family Marsh*  
*And Stan is not driver of car SUV*

Pensez à nettoyer votre modèle ...

```
model
  schema 1.1

  type user

  type car
    relations
      define owner: [user]
      define driver: [family#parent] or owner

  type family
    relations
      define parent: [user]
      define child: [user]
```



# Implémenter des scenarii utilisateur

## 5. Interdire à son fils de conduire la voiture

*Given a user Randy*  
*And Randy is owner of the car SUV*  
*And a user Sharon*  
*And user Randy marries user Sharon*  
*When users Randy and Sharon have a child Stan*  
*Then Stan is member of family Marsh*  
*And Stan is not driver of car SUV*

Pensez à nettoyer votre modèle ...

... Ainsi que vos données !

USER	user:Randy
RELATION	member
OBJECT	family:Marsh

USER	user:Sharon
RELATION	member
OBJECT	family:Marsh





# Implémenter des scenarii utilisateur

## 5. Interdire à son fils de conduire la voiture

*Given a user Randy*  
*And Randy is owner of the car SUV*  
*And a user Sharon*  
*And user Randy marries user Sharon*  
*When users Randy and Sharon have a child Stan*  
*Then Stan is member of family Marsh*  
*And Stan is not driver of car SUV*

Pensez à nettoyer votre modèle ...

... Ainsi que vos données !

USER	user:Stan
RELATION	member
OBJECT	family:Marsh

USER	family:Marsh#member
RELATION	driver
OBJECT	car:SUV



# Implémenter des scenarii utilisateur

## 5. Empêcher son ex de conduire la voiture



Given a user Randy

And Randy is owner of the car SUV

And a user Sharon

And user Randy marries user Sharon

And users Randy and Sharon have a child Stan

When Randy divorces Sharon

Then Randy is no longer parent of family Marsh

And Sharon is not driver of car SUV



# Implémenter des scenarii utilisateur

## 5. Empêcher son ex de conduire la voiture

*Given a user Randy*

*And Randy is owner of the car SUV*

*And a user Sharon*

*And user Randy marries user Sharon*

*And users Randy and Sharon have a child Stan*

*When Randy divorces Sharon*

*Then Randy is no longer parent of family Marsh*

*And Sharon is not driver of car SUV*

Rien ne change dans le modèle

```
model
  schema 1.1
```

```
type user
```

```
type car
  relations
    define owner: [user]
    define driver: [family#parent] or owner
```

```
type family
  relations
    define parent: [user]
    define child: [user]
```



# Implémenter des scenarii utilisateur

## 5. Empêcher son ex de conduire la voiture

*Given a user Randy*

*And Randy is owner of the car SUV*

*And a user Sharon*

*And user Randy marries user Sharon*

*And users Randy and Sharon have a child Stan*

*When Randy divorces Sharon*

*Then Randy is no longer parent of family Marsh*

*And Sharon is not driver of car SUV*

Rien ne change dans le modèle

Mettez à jour vos tests ...

USER	user:Sharon
RELATION	driver
OBJECT	car:SUV
ALLOWED	false



# Implémenter des scenarii utilisateur

## 5. Empêcher son ex de conduire la voiture

*Given a user Randy*  
*And Randy is owner of the car SUV*  
*And a user Sharon*  
*And user Randy marries user Sharon*  
*And users Randy and Sharon have a child Stan*  
*When Randy divorces Sharon*  
*Then Randy is no longer parent of family Marsh*  
*And Sharon is not driver of car SUV*

Rien ne change dans le modèle

Mettez à jour vos tests ...

... Faites-les passer *au vert*

<del>USER</del>	<del>user:Randy</del>
<del>RELATION</del>	<del>parent</del>
<del>OBJECT</del>	<del>family:Marsh</del>

<del>USER</del>	<del>family:Marsh#parent</del>
<del>RELATION</del>	<del>driver</del>
<del>OBJECT</del>	<del>car:SUV</del>



## Implémenter des scénarii utilisateur

### 6. Pavaner avec sa voiture



Given a user Randy

When user Randy buys a car SUV

Then any user is viewer of car SUV



# Implémenter des scenarii utilisateur

## 6. Pavaner avec sa voiture

*Given a user Randy*

*When user Randy buys a car SUV*

*Then any user is viewer of car SUV*

Ajoutez la relation **viewer** au type **car**

```
model
  schema 1.1
```

```
type user
```

```
type car
```

```
  relations
```

```
    define owner: [user]
```

```
    define driver: [family#parent] or owner
```

```
    define viewer: [user:*] or driver
```

```
type family
```

```
  relations
```

```
    define parent: [user]
```

```
    define child: [user]
```



# Implémenter des scenarii utilisateur

## 6. Pavaner avec sa voiture

*Given a user Randy*

*When user Randy buys a car SUV*

*Then any user is viewer of car SUV*

Ajoutez la relation **viewer** au type **car**

Ajoutez un test ...

USER	user:Gerald
RELATION	viewer
OBJECT	car:SUV
ALLOWED	true





# Implémenter des scenarii utilisateur

## 6. Pavaner avec sa voiture

*Given a user Randy*

*When user Randy buys a car SUV*

*Then any user is viewer of car SUV*

Ajoutez la relation **viewer** au type **car**

Ajoutez un test ...

... Faites-le passer *au vert*

USER	user:*
RELATION	viewer
OBJECT	car:SUV



# Implémenter des scenarii utilisateur

## 7. Empêcher les grand-parents de conduire



Given a user Randy

And user Grandpa is user Randy parent

When user Randy buys a car SUV

Then user Grandpa is not driver of car SUV



# Implémenter des scenarii utilisateur

## 7. Empêcher les grand-parents de conduire

*Given a user Randy*

*And user Grandpa is user Randy parent*

*When user Randy buys a car SUV*

*Then user Grandpa is not driver of car SUV*

Ajoutez la relation **parent** au type **user**

```
model
  schema 1.1
```

```
type user
  relations
    define parent: [user]
```

```
type car
  relations
    define owner: [user]
    define driver: ([family#parent] or owner)
    but not parent from owner
    define viewer: [user:*] or driver
```

```
type family
  relations
    define parent: [user]
    define child: [user]
```



# Implémenter des scenarii utilisateur

## 7. Empêcher les grand-parents de conduire

*Given a user Randy*

*When user Randy buys a car SUV*

*Then any user is viewer of car SUV*

Ajoutez la relation **parent** au type **user**

Ajoutez un test ...

USER	user:Grandpa
RELATION	driver
OBJECT	car:SUV
ALLOWED	false



# Implémenter des scenarii utilisateur

## 7. Empêcher les grand-parents de conduire

*Given a user Randy*

*When user Randy buys a car SUV*

*Then any user is viewer of car SUV*

Ajoutez la relation **parent** au type **user**

Ajoutez un test ...

... Faites-le passer *au vert*

USER	user:Grandpa
RELATION	parent
OBJECT	user:Randy



# Implémenter des scenarii utilisateur

## 8. Laisser son fils majeur conduire la voiture



Given a user Randy

And a user Stan

And user Stan is driver of car SUV

When user Stan gets 18 years old

Then user Stan can drive car SUV



# Implémenter des scenarii utilisateur

## 8. Laisser son fils majeur conduire la voiture

*Given a user Randy*

*And a user Stan*

*And user Stan is driver of car SUV*

*When user Stan gets 18 years old*

*Then user Stan can drive car SUV*

Ajoutez une **relation conditionnelle**

```
model
  schema 1.1

  type user
    relations
      define parent: [user]

  type car
    relations
      define owner: [user]
      define driver: ([family#parent, user with
        allowed_age] or owner) but not parent from owner
      define viewer: [user:*] or driver

  type family
    relations
      define parent: [user]
      define child: [user]

  condition allowed_age(age: int) {
    age >= 18
  }
```



# Implémenter des scenarii utilisateur

## 8. Laisser son fils majeur conduire la voiture

```
model
  schema 1.1
```

```
type user
  relations
    define parent: [user]
```

```
type car
  relations
    define owner: [user]
    define driver: ([family#parent, user with
allowed_age] or owner) but not parent from owner
    define viewer: [user:*] or driver
```

```
type family
  relations
    define parent: [user]
    define child: [user]
```

```
condition allowed_age(age: int) {
  age >= 18
}
```

```
tuples:
```

```
- user: user:Randy
  relation: owner
  object: car:SUV
```

```
tests:
```

```
- name: conditional-relationship
  description: laisser son fils majeur conduire le SUV
```

```
tuples:
```

```
- user: user:Stan
  relation: driver
  object: car:SUV
  condition:
    name: allowed_age
    context:
      age: 18
```

```
check:
```

```
- user: user:Stan
  object: car:SUV
  assertions:
    driver: true
```





# Implémenter des scénarii utilisateur

## 8. Laisser son fils majeur conduire la voiture

- Créer un fichier `devoxx-fr-2024.fga` contenant le modèle d'autorisations
- Créer un fichier `tests-devoxx.yaml` contenant les tests
- Installer la [CLI d'OpenFGA](#)
- Exécuter la commande `fga model test --tests ./tests-devoxx.yaml`
  - ajouter l'option `--verbose` en fin de commande pour afficher le détail des tests



## Take aways

- Le modèle est au centre de tout
- Il impacte donc 100% de votre SI



# Take aways

- Le modèle est au centre de tout
- Il impacte donc 100% de votre SI
- Toute donnée validée est insérée
- Nettoyer et normaliser vos données



# Take aways

- Le modèle est au centre de tout
- Il impacte donc 100% de votre SI
- Toute donnée validée est insérée
- Nettoyer et normaliser vos données
- Le modèle représente votre métier
- Aligner le vocabulaire



# Take aways

- Le modèle est au centre de tout
- Il impacte donc 100% de votre SI
- Toute donnée validée est insérée
- Nettoyer et normaliser vos données
- Le modèle représente votre métier
- Aligner le vocabulaire
- User et abuser des tests
- Tout changement sera signalé



## Take aways

- Le modèle est au centre de tout
- Il impacte donc 100% de votre SI
- Toute donnée validée est insérée
- Nettoyer et normaliser vos données
- Le modèle représente votre métier
- Aligner le vocabulaire
- User et abuser des tests
- Tout changement sera signalé
- Respecter l'idempotence
- La continuité de service sera garantie



# Access Management orienté métier avec ReBAC

Merci pour votre attention



# Ressources

<https://zanzibar.academy/#!>

[https://storage.googleapis.com/pub-tools-public-publication-data/pdf/10683a8987dbf0c6d4edcaf  
b9b4f05cc9de5974a.pdf](https://storage.googleapis.com/pub-tools-public-publication-data/pdf/10683a8987dbf0c6d4edcaf<br/>b9b4f05cc9de5974a.pdf)

<https://openfga.dev/>

[https://authorizationinsoftware.auth0.com/public/49/Authorization-in-Software-f9b69587/7889b  
b9c](https://authorizationinsoftware.auth0.com/public/49/Authorization-in-Software-f9b69587/7889b<br/>b9c)

<https://www.aserto.com/blog/google-zanzibar-drive-rebac-authorization-model>

<https://www.osohq.com/post/zanzibar>