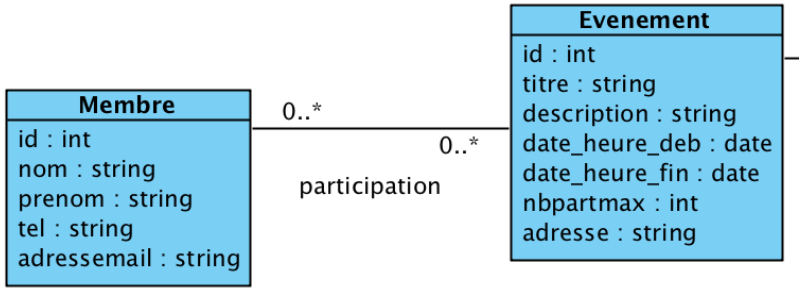
Module **PWS3** **ORM Symfony suite**

Etape 2 : Mise en place d'une relation Many to Many

Nous allons gérer les participations des membres des jeuxdicode aux évènements.



Ce qui se traduirait en schéma relationnel:
EVENEMENT (id, titre, description, date_heure_deb, date_heure_fin, nbpartmax, adresse)
Clé primaire : id
MEMBRE (id, nom, prenom, tel, adressemail)
Clé primaire : id
PARTICIPATION (id_evenement,id_membre)
Clé primaire : id_evene
ment, id_membre
Dépendance de référence : PARTICIPATION.id_evenement Ref EVENEMENT.id
Dépendance de référence : PARTICIPATION.id_evenement Ref MEMBRE.id

Cette table intermédiaire PARTICIPATION ne donne pas lieu à une entité ; c'est doctrine qui va s'occuper de sa création et de sa gestion.
Nous allons considérer dans un premier temps que l'entité EVENEMENT est l'entité propriétaire et que l'entité MEMBRE est l'entité inverse car on veut en priorité récupérer les participants (membres) à un évènement.

Créez l'entité MEMBRE php bin/console make:entity

```
Class name of the entity to create or update (e.g. FiercePopsicle):
> Membre

created: src/Entity/Membre.php
created: src/Repository/MembreRepository.php

Entity generated! Now let's add some fields!
You can always add more fields later manually or by re-running this command.

New property name (press <return> to stop adding fields):
> nom

Field type (enter ? to see all types) [string]:
>

Field length [255]:
> 80

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/Membre.php

Add another property? Enter the property name (or press <return> to stop adding fields):
> prenom

Field type (enter ? to see all types) [string]:
>

Field length [255]:
> 80
```

(Extrait de la création de l'entité)

Modifiez l'entité EVENEMENT php bin/console make:entity

```
root@442aa41ce81a:/var/www/html/jeuxdicode# php bin/console make:entity

 Class name of the entity to create or update (e.g. GentleElephant):
 > Evenement

 Your entity already exists! So let's add some new fields!

 New property name (press <return> to stop adding fields):
 > membres

 Field type (enter ? to see all types) [string]:
 > relation

 What class should this entity be related to?:
 > Membre

What type of relationship is this?
 ------------ ---------------------------------------------------------------------
  Type         Description
 ------------ ---------------------------------------------------------------------
  ManyToOne    Each Evenement relates to (has) one Membre.
               Each Membre can relate to (can have) many Evenement objects

  OneToMany    Each Evenement can relate to (can have) many Membre objects.
               Each Membre relates to (has) one Evenement

  ManyToMany   Each Evenement can relate to (can have) many Membre objects.
               Each Membre can also relate to (can also have) many Evenement objects

  OneToOne     Each Evenement relates to (has) exactly one Membre.
               Each Membre also relates to (has) exactly one Evenement.
 ------------ ---------------------------------------------------------------------

 Relation type? [ManyToOne, OneToMany, ManyToMany, OneToOne]:
 > ManyToMany

 Do you want to add a new property to Membre so that you can access/update Evenement objects from it - e.g. $membre->getEvenements()? (yes/no) [yes]:
 > no
```

L'attribut $membres contiendra une liste d'objets membre.

Étudions ce code :
Comme l'attribut $membres contient une liste d'objets membre, doctrine doit utiliser des Collection qui sont des formes particulières de tableaux.
L'attribut $membres est donc définit comme tel dans le constructeur de la classe qui a été rajouté.
L'accesseur getMembres() est classique.
L'accesseur setMembres() diffère du fait que $membres est une liste, il faut une méthode addMembre() qui ajoutera un membre à la liste et une méthode removeMembre() pour supprimer un membre de la liste.

Créez et appliquez les migrations

Comment s'appelle la table correspondant à la relation PARTICIPATION et les dépendances de références prévues ont-elles été créés?

Mettez à jour la vue detailevt pour qu'elle puisse faire également apparaître la liste des participants à l'événement.

Vous pouvez utiliser le profiler pour voir les requêtes effectuées par doctrine pour constituer la réponse attendue.