Partie 2 - Nous allons maintenant faire le lien avec la partie Base de données.

Avant de pouvoir utiliser une entité, on doit créer la table correspondante dans la base de données.

3 étapes pour créer ou modifier une table associée à une entité

1- créer ou modifier l'entité

2- créer une migration => générer le script sql

3- appliquer la migration => exécuter le script sql

D'ailleurs quand nous avons créé notre entité Evenement, symfony nous a proposé :

Next: When you're ready, create a migration with php bin/console make:migration

Et quand c'est fait vous pouvez aller voir le fichier créé :

Next: Review the new migration "migrations/Version20201104204301.php" par exemple ici

```
        // this up() migration is auto-generated, please modify it to your needs
        $this->addSql( sql: 'CREATE TABLE evenement (id INT AUTO_INCREMENT NOT NULL, titre VARCHAR(60) NOT NULL, date_heure_debut DATETIME NOT NULL,
date_heure_fin DATETIME NOT NULL, nbpartmax INT DEFAULT NULL, adresse VARCHAR(150) NOT NULL, PRIMARY KEY(id)) DEFAULT CHARACTER SET utf8mb4 COLLATE `utf8mb4_unicode_ci`
ENGINE = InnoDB');
    }
```

Then: Run the migration with php bin/console doctrine:migrations:migrate

La documentation de la notion de migration :

https://symfony.com/doc/current/bundles/DoctrineMigrationsBundle/index.html

Se connecter à la base de données via phpstorm et vérifiez que la table evenement a bien été créée :

Modifier une entité, pour cela il faut créer ou modifier un attribut et lui attacher ou modifier l'annotation correspondante.

Modifiez la classe evenement en lui ajoutant un attribut $evtcourant de type booléen qui indique si l'événement est l'événement courant à venir.

Ensuite, soit vous écrivez les accesseurs manuellement, soit vous utilisez le générateur : **php bin/console make:entity --regenerate** qui génère les entités en fonction du mapping qu'il connait.

Mettez à jour la BD en fonction en créant une nouvelle migration.

Faites un refresh sur votre database dans phpstorm pour voir la modification.

Récupération des entités via le repository :

Un repository est un ensemble d'outils pour récupérer des instances d'entités très facilement.

Il a été généré lors de la création de l'entité, et il met à notre disposition un certain nombre de fonctions prédéfinies nous reviendrons plus dans le détail par la suite.

Le contrôleur va utiliser un repository pour accéder aux données.

Complétez votre contrôleur Évènement pour créer une methode "listevt" associée au template "listevt.html.twig" accessible par la route "/listevt" faisant apparaître la liste des évènements.

Il faut récupérer le repository dans la méthode listevt du contrôleur Evenement :

Une première méthode 'magique' pour récupérer l'ensemble des entités évènements à partir des nuplets de la base de données findAll().

```
/**
 * @Route("/listevt", name="listevt")
 */
public function listevt()
{
    $repository=$this->getDoctrine()->getManager()-
>getRepository('App\Entity\Evenement');
    $evenements=$repository->findAll();
    return $this->render('evenement/listevt.html.twig', [
        'controller_name' => 'EvenementController',
        'evenements'=> $evenements,
    ]);
}
```

Mais du coup il nous faut des données

Ajoutez des enregistrements dans la table évènement via la console mysql dans phpstorm et faites les apparaître dans la vue listevt.html.twig.

Côté de la vue :

```
{% extends 'base.html.twig' %}

{% block title %}Liste des évènements{% endblock %}

{% block body %}
    <style>
        .example-wrapper { margin: 1em auto; max-width: 800px; width: 95%;
font: 18px/1.5 sans-serif; }
        .example-wrapper code { background: #F5F5F5; padding: 2px 6px; }
    </style>

    <div class="example-wrapper">
        <h1>Liste des évènements</h1>
        <ul>
            {% for evt in evenements %}
                <li>{{ evt.titre }}</li>
            {% endfor %}
        </ul>
```

```
    </div>
{% endblock %}
```

A faire si vous avez terminé :

- Ajouter un contrôleur Membre qui gèrera les membres de l'association, on veut pouvoir consulter la liste des membres de l'association
- Pour un membre on aura son nom, son prénom, son tel et son adresse mail
- Gérer la partie entité correspondante, mettez à jour la BD en fonction
- Créer la page permettant l'affichage des membres dans un tableau responsive (bootstrap)
- On veut maintenant ajouter un menu dans le layout principal qui permettra d'obtenir la liste des évènements et la liste des membres