

SYMFONY

SYMFONY



Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Table des matières

Liste des compétences du module :	6
1 Présentation du Framework :	7
1.1 Présentation Symfony :	7
2 Installation et prérequis du Framework :	7
2.1 Prérequis :	7
2.2 Installation de scoop :	7
2.3 Installation de CLI :	8
2.4 Installation de composer :	8
2.5 Exemple création de projets en ligne de commande :	8
2.6 Utilisation du serveur interne de Symfony :	9
3 Structure des projets :	11
3.1 Présentation de la structure des projets :	11
4 Routing avec Symfony :	12
4.1 Création de route avec routes.yaml :	12
4.2 Création de route avec les annotations :	14
4.3 Création de route avec maker-bundle :	15
4.4 Exercice :	18
5 Les templates Twig :	18
5.1 Block de code HTML :	18
5.2 Tester son interface :	20
5.3 Intégrer des assets (Bootstrap) à un projet :	21
5.4 Liaison fichier Bootstrap à base.html.twig :	21
5.5 Exercice :	21
6 ORM et Doctrine :	22
6.1 Création d'une base de données avec Symfony :	22

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

6.2 Modifier les entités (classes) et ajout des contraintes :	25
6.3 Configuration de l'accès à la base de données :	27
6.4 Effectuer une Migration :	28
6.5 Exercice :	29
7 ORM-Fixtures et Faker :	31
7.1 Installation de l'outil ORM-Fixtures :	31
7.2 Création de la classe AppFixtures.php :	31
7.3 Installation de la librairie Faker :	32
7.4 Intégration de Faker :	33
7.5 Exercice :	34
7.6 Sauvegarder des données en BDD :	35
8 Création d'une API (Micro-services) :	36
8.1 MCD MLD du projet task :	36
8.2 Définitions :	37
8.3 Structure de l'API :	37
8.4 Création des controllers :	37
8.5 Créer les entités :	38
8.6 Configurer l'accès à la base de données :	39
8.7 Créer la base de données :	39
8.8 Paramétrage et migration :	40
8.9 Exercice :	41
9 Création d'une API suite (outils et Méthodes) :	41
9.1 Installation de Postman :	41
9.2 Edition des controllers :	42
9.3 Exercice :	43
9.4 Encodage en Json :	44

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

9.5 Sériailisation :	45
9.6 Exercice :	49
10 API POST ajouter des enregistrements en BDD :	49
10.1 Ajout de la méthode create en POST :	49
10.2 Exercice :	54
11 Création d'une web-App :	54
11.1 Création d'un projet Web-App :	54
11.2 Configuration et intégration des dépendances du projet :	54
11.3 Ajouter une structure de données (entités, classe) :	55
11.4 Créer et migrer la base de données :	55
11.5 Exercice :	56
11.6 Ajoutez des données de tests :	57
11.7 Exercice :	57
12 Construire une page dynamique :	57
12.1 Génération d'un controller :	57
12.2 Modification de la méthode index :	59
12.2 Modification de l'interface twig :	59
12.3 Bloc FOR twig :	60
12.4 Exercice :	61
12.5 Intégration d'attribut et filtrage :	61
12.6 Afficher un seul enregistrement dans une interface :	62
12.7 Liaison bouton vers une route :	64
12.8 Exercice :	64
13 Création d'un panel Administrateur :	65
13.1 installation et création du Dashboard :	65
13.2 Configuration intégration des entités (CRUD) :	66

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

13.3 Exercice :	67
13.4 Configuration intégration des entités (CRUD) (suite) :	67
14 Créer un formulaire avec Symfony :	69
14.1 Générer un formulaire :	69
14.2 Configurer un formulaire ajouter des attributs non mappés :	70
14.3 intégrer le formulaire à un controller :	71
14.4 Ajouter le formulaire dans la vue :	71
14.5 Ajouter le champ submit :	72
14.5 Récupérer les données :	72
14.6 Sauvegarder les données en BDD :	73
15 Inscription et connexion sécurisé :	75
15.1 Création de l'entité User avec la commande suivante :	75
15.2 Mise en place de l'authentification avec la commande suivante :	76
15.3 Configuration de l'inscription :	78
15.4 Création du controller et de la méthode addUser :	80
15.5 Edition de la vue :	82
15.6 Connexion :	83
15.7 Redirection à la connexion :	84
15.8 Protection des routes :	84

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Liste des compétences du module :

Etre capable de créer des pages Web Dynamique,

Etre capable de mettre en place un système d'API,

Etre capable de connecter une application serveur à une base de données côté Back-end,

Etre capable de gérer des requêtes HTTP d'interaction côté Back-end.

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

1 Présentation du Framework :

1.1 Présentation Symfony :

Symfony est un ensemble de composants **PHP**, un **Framework** d'application Web, une philosophie et une communauté.

Un **Framework** permet aux développeurs de gagner du temps en réutilisant des modules génériques pour se concentrer sur d'autres domaines. Il va nous aider à développer mieux et plus vite !

Un **Framework** vous apporte la certitude que l'on développe une application parfaitement conforme aux règles métier, structurée, maintenable et évolutive.

C'est un ensemble de composants découplés et réutilisables sur lesquels sont construites les applications PHP : **Drupal**, **Prestashop** et **Laravel**.

2 Installation et prérequis du Framework :

2.1 Prérequis :

Windows 10,

Terminal (PowerShell, Gitbash),

Composer,

Scoop (gestionnaire de package),

Editeur de code Vscode,

Xampp (Wamp, Mamp), apache, php et MySQL.

Navigateur web (Chrome, Edge, Firefox etc....)

2.2 Installation de scoop :

-Ouvrir le terminal PowerShell et saisir la commande suivante :

```
ïex (new-object net.webclient).downloadstring('https://get.scoop.sh')
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

NB : si vous avez une erreur **SSL TLS** exécutez la commande suivante :

```
[Net.ServicePointManager]::SecurityProtocol =
[Net.SecurityProtocolType]::Tls12
Set-ExecutionPolicy RemoteSigned -scope CurrentUser
iwr -useb get.scoop.sh | iex
```

Lien issues : <https://github.com/ScoopInstaller/Scoop/issues/4002>

2.3 Installation de CLI :

Toujours depuis PowerShell saisir la commande ci-dessous :

```
scoop install symfony-cli
```

NB : avec Symfony nous n'avons pas à utiliser le serveur local (**Xampp, wamp**), il dispose de son propre serveur intégré.

2.4 Installation de composer :

Pour installer composer téléchargez le package **Composer-Setup.exe** à l'adresse ci-dessous :

<https://getcomposer.org/download/>

2.5 Exemple création de projets en ligne de commande :

2.5.1 Création d'un micro service en CLI :

Depuis PowerShell saisir la commande suivante :

```
symfony new micro_service
```

```
PS C:\> cd .\symphony\
PS C:\symphony> symfony new micro_service2
* Creating a new Symfony project with Composer
(running C:\ProgramData\ComposerSetup\bin\composer.phar create-project symfony/skeleton C:\symphony\micro_service2 --no-interaction)

* Setting up the project under Git version control
(running git init C:\symphony\micro_service2)

[OK] Your project is now ready in C:\symphony\micro_service2
```

2.5.2 Création d'une Web App en CLI :

Depuis PowerShell saisir la commande suivante :

```
symfony new website --webapp
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordnatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordnatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
PS C:\symphony> symfony new website --webapp
* Creating a new Symfony project with Composer
(running C:\ProgramData\ComposerSetup\bin\composer.phar create-project symfony/skeleton C:\symphony\website --no-interaction)

* Setting up the project under Git version control
(running git init C:\symphony\website)

(running C:\ProgramData\ComposerSetup\bin\composer.phar require webapp --no-interaction)

[OK] Your project is now ready in C:\symphony\website
```

2.5.3 Création d'un micro service avec composer :

Depuis PowerShell ou Gitbash saisir la commande suivante :

```
composer create-project symfony/skeleton microservice_composer
```

2.5.4 Création d'une Webapp avec composer :

```
composer create-project symfony/website-skeleton website_composer
```

NB : à la question docker taper **n** puis **entrée**.

2.6 Utilisation du serveur interne de Symfony :

Pour lancer un projet avec le serveur interne de Symfony :

Ouvrir un terminal et se positionner dans le dossier du site à exécuter (par ex website),

Saisir la commande suivante :

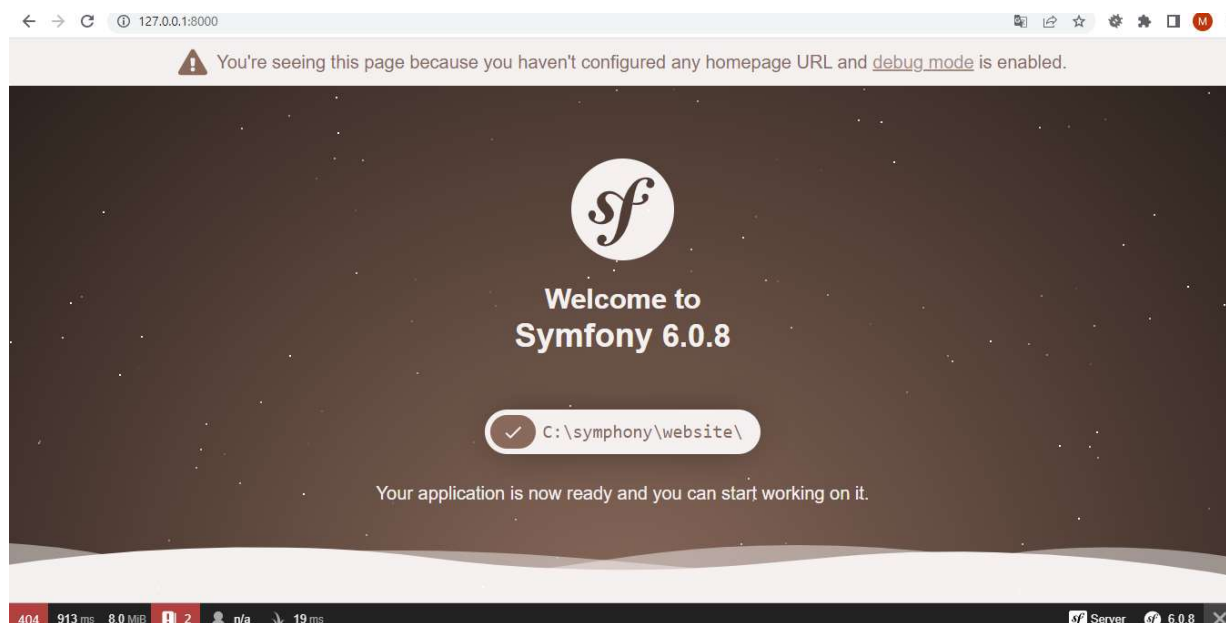
```
symfony server:start -d
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Le site sera accessible dans le navigateur web à l'adresse suivante :

<http://127.0.0.1:8000>



Pour arrêter le serveur saisir la commande suivante :

`symfony server:stop`

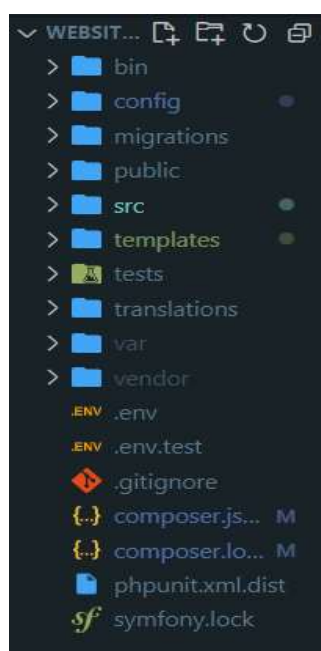
Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

3 Structure des projets :

3.1 Présentation de la structure des projets :

Les projets symfony sont structurés comme ci-dessous :



Dossier **bin** contient 2 fichiers :

Console (exécutable pour manager le projet),

Phpunit (exécutable pour le testing de l'application).

Dossier **config** (contient les informations de configuration du projet),

Dossier **migrations** (contient le code SQL du projet),

Dossier **public** (contient les fichiers et dossiers accessible au public),

Dossier **src** (contient le code source du projet, fichiers et dossier non accessible au public),

Dossier **templates** (contient le templates HTML des pages),

Dossier **test** (contient les tests du projet),

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Dossier **translations** (contient les différentes traductions (langues) du projet),

Dossier **var** (contient les variables et les fichiers temporaires),

Dossier **vendor** (contient les dépendances du projet),

Fichier **.env** (contient les variables d'environnement).

Fichier **.env.test** (contient les variables des tests),

Fichier **.gitignore** (fichier git pour exclure les fichiers et dossier),

Fichier **composer.json** (fichier qui contient les dépendances du projet),

Fichier **composer.lock** (fichier qui contient les dépendances du projet),

Fichier **phpunit.xml.dist** (fichier de configuration des tests du projet),

Fichier **symfony.lock** (fichier de configuration des dépendances installées sur le projet).

4 Routing avec Symfony

Un des principes de base de Symfony, va être de rattacher nos **méthodes** (qui sont contenues dans nos classes **controller**) à des **routes** (à la différence de ce que l'on a vu précédemment avec PHP où nous rattachions les **routes** à des **controllers**).

Exemple :

Création d'une fonction qui va afficher bonjour dans une page web, elle sera accessible depuis la **route /bonjour**.

4.1 Création de route avec **routes.yaml**

Création d'un nouveau controller dans le répertoire **src/controller**

Intégrer le code ci-dessous :

```
<?php
```

```
namespace App\Controller;
```

```
use Symfony\Component\HttpFoundation\Response;
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
class HelloCtrl{
```

```
    public function sayHello():Response{
```

```
        return new Response("Bonjour");
```

```
    }
```

```
    public function sayHelloUtil($name):Response{
```

```
        return new Response("Bonjour ".$name);
```

```
    }
```

```
}
```

```
?>
```

Paramétrage de la route dans le fichier **routes.yaml**.

Ajouter le code comme ci-dessous :

HelloController:

```
path: /bonjour
```

```
controller : App\Controller\HelloCtrl::sayHello
```

HelloControllerName:

```
path: /bonjour/{toto}
```

```
controller : App\Controller\HelloCtrl::sayHelloUtil
```

Pour tester les routes veuillez saisir :

http://127.0.0.1/bonjour

Affiche *Bonjour* (dans la page)

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

<http://127.0.0.1//bonjour/Adrar>

Affiche *Bonjour Adrar* (dans la page)

NB : le serveur web intégré de **symfony** doit être lancé préalablement avec la commande :

```
symfony server:start -d
```

4.2 Création de route avec les annotations :

Symfony permet de **générer automatiquement** les routes avec le système **d'annotations**.
 (Commentaire sur une fonction).

Pour pouvoir utiliser cette fonctionnalité nous allons devoir installer la **bibliothèque** avec la commande suivante (dans un **terminal**) :

```
composer require --dev annotations
```

Créer un nouveau contrôleur dans le dossier src/controller

```
<?php
```

```
//src/Controller/NameController.php
```

```
namespace App\Controller;
```

```
use Symfony\Component\HttpFoundation\Response;
```

```
use Symfony\Component\Routing\Annotation\Route;
```

```
class NameController
```

```
{
```

```
/**
```

```
 * @Route("/hello")
```

```
 */
```

```
public function Hello(): Response
```

```
{
```

```
return new Response(
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
'<html><body>Hello World</body></html>'

);

}

//fonction avec Paramètre (route/param)

/**
 * @Route("/hello/{name}")
 */

public function HelloUser($name): Response
{
    return new Response(
        '<html><body>Hello '.$name.'</body></html>'
    );
}

}
```

Pour générer les routes nous utilisons la syntaxe suivante :

-@Route("/nom_route") pour une route simple,

-@Route("/nom_route/{param}") pour une route avec un paramètre.

4.3 Création de route avec **maker-bundle** :

Installer la bibliothèque **maker-bundle** avec la commande suivante (dans un terminal) :

symfony console make:controller

Nous devons saisir un nom de **controller** (exemple Accueil) :

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
$ symfony console make:controller

Choose a name for your controller class (e.g. TinyPopsicleController):
> Accueil|
```

La commande va générer automatiquement le controller Accueil et un template HTML :

```
created: src/Controller/AccueilController.php
created: templates/accueil/index.html.twig
```

Success!

Fichier **AccueilController** (src/controller):

```
<?php
```

```
namespace App\Controller;
```

```
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
```

```
use Symfony\Component\HttpFoundation\Response;
```

```
use Symfony\Component\Routing\Annotation\Route;
```

```
class AccueilController extends AbstractController
```

```
{
```

```
    #[Route('/accueil', name: 'app_accueil')]
```

```
    public function index(): Response
```

```
    {
```

```
        return $this->render('accueil/index.html.twig', [
```

```
            'controller_name' => 'AccueilController',
```

```
        ]);
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMPHONY

```
}
```

```
}
```

Le fichier **AccueilController.php** contient : une méthode **index** qui va générer la **vue** avec la **fonction render** (qui possède **2 paramètres** en entrée : un **Template html twig** et un **tableau**).

Fichier **Template html twig** (**templates/accueil/index.html.twig**) :

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

```
{% block title %}Hello AccueilController!{% 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>Hello {{ controller_name }}! ✓</h1>
```

```
This friendly message is coming from:
```

```
<ul>
```

```
<li>Your controller at <code><a href="{{  
'C:/symphony/website_composer/src/Controller/AccueilController.php'|file_link(0) }}">src/Controller/AccueilController.php</a></code></li>
```

```
<li>Your template at <code><a href="{{  
'C:/symphony/website_composer/templates/accueil/index.html.twig'|file_link(0) }}">templates/accueil/index.html.twig</a></code></li>
```

```
</ul>
```

```
</div>
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
{% endblock %}
```

Le fichier **index.html.twig** (correspond à la **vue** dans le model **MVC**) contient le style **CSS** au sein d'une balise (**style**), les scripts **JS** (balise **script**) et le code **HTML**.

4.4 Exercice :

Dans la class **AccueilController**, en vous inspirant de la méthode **index** généré par **make-bundle**, ajoutez une nouvelle méthode **index2** qui va :

- prendre en paramètre (un nom).
- Récupérer le paramètre dans la fonction render (Vous allez devoir pour cela modifier le tableau en paramètre et ajouter à celui-ci une entrée **'test_name' => \$name**).
- Modifier le **template html** pour gérer l'affichage du nom dans la page.
- Créez une **route** pour appeler la fonction.

Le nom devra être récupéré dans la route comme ceci : **/accueil/nom**.

5 Les templates Twig :

Les templates Twig sont des fichiers qui vont contenir le code HTML et CSS de nos vues. Si nous avons généré un projet complet nous avons dans le dossier templates un fichier **base.html.twig**, celui-ci va contenir le code HTML.

Si le **package twig** n'est pas présent dans notre projet nous allons l'intégrer avec la commande suivante (dans un **terminal**) :

```
composer require --dev twig
```

5.1 Block de code HTML :

Les pages HTML contiennent des blocs de code avec la syntaxe suivante :

```
{% block title %}titre de La page{% endblock %}
```

Ces blocs (encadrés par des accolades) fonctionnent comme des variables, nous allons pouvoir les modifier en fonction de nos besoins en les surchargeant (réécriture de ceux-ci).

Dans la partie précédente (**routing**) nous avons généré une page avec **make-bundle (accueil)**.

Nous allons remplacer son contenu afin de comprendre l'utilisation des blocs.

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Auteur(s)	Relu, validé et visé par :		Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE :		25/09/2022	
	Resp. Secteur Numérique Occitanie			
	Florence CALMETTES :		Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Coordinatrice Filière Syst. & Réseaux			
	Sophie POULAKOS :			
	Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT			

SYMFONY

Balise extends :

La balise **extends** sert à indiquer depuis quel fichier elle doit hériter la structure html de base dans notre cas nous allons nous servir du fichier **base.html.twig** qui se trouve à la racine de templates.

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

Balise title :

La balise title permet de redéfinir le titre de notre page (comme en HTML). Elle s'ouvre avec **block** et se referme avec **endblock**.

```
{% block title %} Accueil {% endblock %}
```

Balise body :

La balise body permet de redéfinir le corps de la page (body en HTML). Elle s'ouvre avec **block** et se referme avec **endblock**. (Elle va **surcharger** le body de **base.html.twig**)

```
{% block body %}
```

```
<h1>
```

```
Bienvenue sur La page Accueil
```

```
</h1>
```

```
{% endblock %}
```

5.2 Tester son interface :

Pour tester notre **interface** nous allons revenir dans le **controller accueil** et réécrire la méthode **index** comme ci-dessous :

```
/*Attention !!! pensez à renommer le name (app_accueil en app_accueil2) si vous avez plusieurs routes dans votre controller */
```

```
#[Route('/home', name: 'app_accueil2')]
```

```
public function index(): Response
```

```
{
```

```
return $this->render('accueil/index.html.twig');
```

```
}
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

La page est accessible à l'adresse suivante :

localhost :8000/home ou **localhost :8000/accueil**

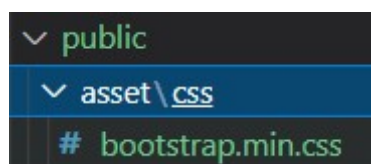
5.3 Intégrer des assets (Bootstrap) à un projet :

Pour utiliser des **thèmes Bootstrap** dans un projet nous allons les télécharger depuis le site ci-dessous :

<https://bootswatch.com/>

Nous allons créer à l'intérieur du répertoire **public** (à la **racine** du projet) un dossier **asset->css**

Nous déplacerons le fichier **bootstrap.min.css** téléchargé précédemment à l'intérieur de celui-ci :



5.4 Liaison fichier Bootstrap à base.html.twig :

Pour **lier** le fichier **style CSS** à notre page nous allons intégrer le code ci-dessous :

```
{% block stylesheets %}
```

```
<link rel="stylesheet" href="../asset/css/bootstrap.min.css">
```

```
{% endblock %}
```

NB : Le fichier **style CSS** se trouve dans le répertoire **public/asset/css/**.

5.5 Exercice :

Recréer la vue **index.html.twig** en intégrant :

-Un Menu Bootstrap, (Menu, liens (Accueil, Présentation, Prix et A propos))

-Un titre H1 avec Bienvenue sur la page d'accueil

Un titre H1 avec Bienvenue sur la page d'accueil **prénom** (version qui récupère le prénom dans la **route**)

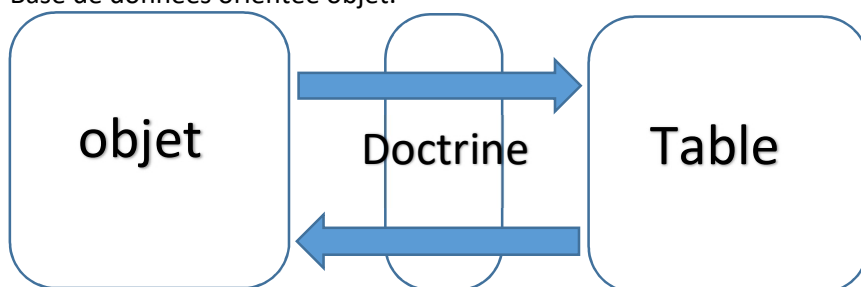
Réaliser les 2 versions (avec et sans le prénom : 2 méthodes dans la classe **AccueilController**)

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

6 ORM et Doctrine :

Un **ORM (Object-Relationnal Mapping)** est un programme qui va servir d'interface entre une base de données relationnelle (Nous utilisons **MySQL** ou **MariaDB**) et une application afin de **simuler** une Base de données orientée objet.



6.1 Création d'une base de données avec Symfony :

Symfony va nous permettre de créer les bases de données de façon automatique.

Pour se faire nous allons utiliser l'outil **make-bundle** avec la commande suivante (depuis un **terminal**) :

symfony console make:entity

L'outil (**make-bundle**) va nous demander de choisir un nom pour l'entité (**table**), nous allons saisir **Article**.

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

created: src/Entity/Article.php
created: src/Repository/ArticleRepository.php

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

Il va nous demander ensuite d'ajouter les attributs et leurs types, nous allons saisir **title** à la question :

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

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordnatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordnatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Nous allons ensuite définir le **type** de l'**attribut** :

NB : Pour connaître les différents **types** il suffit de saisir **?** et **entrée** pour afficher la liste de celle-ci.

```

Main types
* string
* text
* boolean
* integer (or smallint, bigint)
* float

Relationships / Associations
* relation (a wizard will help you build the relation)
*ManyToOne
*OneToMany
*ManyToMany
*OneToOne

Array/Object Types
* array (or simple_array)
* json
* object
* binary
* blob

Date/Time Types
* datetime (or datetime_immutable)
* datetimetz (or datetimetz_immutable)
* date (or date_immutable)
* time (or time_immutable)
* dateinterval

Other Types
* ascii_string
* decimal
* guid
  
```

Nous allons saisir **string** pour le type, **100** pour la **longueur** du champ et **no** (not null)

```

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

Field length [255]:
> 100

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

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Nous allons ajouter les champs supplémentaires suivants :

-**contenu** (nom attribut), **text** (type), **no** (not null),

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

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

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

-**image** (pas de **type**, **longueur** et **not null**), Symfony va utiliser les attributs par défaut (**string**, **255** et **no**) en tapant juste sur la touche **entrée**.

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

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

Field length [255]:
>

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

-**createThe** (nom attribut), **datetime** (type) et **no** (not null) :

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

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

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

Pour terminer la création de la **table** nous allons simplement utiliser la touche **entrée**.

```
updated: src/Entity/Article.php

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

Success!

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

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

L'outil **make-bundle** crée automatiquement la classe dans le répertoire **src/Entity/Article.php** (car nous avons donné comme nom à la table **Article**).

Le fichier contient la classe **Article**, les **attributs** et les **Getter** et **Setter**.

NB : Notons que nous n'avons pas besoin de créer l'attribut **id** l'outil (**make-bundle**) le fait pour nous

Exercice :

En utilisant la commande suivante (dans un terminal) :

```
symfony console make:entity
```

Ajoutez les entités suivantes :

User (name, firstName, mail, password et createAt),

Category (title, description, createAt).

NB : Nous devons répéter la commande **make:entity** pour chaque entité (taper sur la touche **entrée** quand l'outil demande d'ajouter un attribut). Nous devons nous retrouver avec les 3 classes suivantes dans le répertoire **src/Entity/** : **Article**, **User** et **Category**.

6.2 Modifier les entités (classes) et ajout des contraintes :

Avec **Symfony** plutôt que de gérer nos tables d'association et les clés étrangères nous allons utiliser l'outil de génération d'entités (make-bundle).

Pour cela nous allons réutiliser la commande suivante :

```
symfony console make:entity
```

A l'invite de la commande nous allons saisir le nom d'une entité existante à laquelle nous souhaitons en connecter une autre (Article et Category) pour pouvoir lister les Articles par catégories.

Nous allons saisir Article :

```
mathieumithridate@FORMATEURS17-19 MINGW64 /c/symphony/website_composer (master)
$ symfony console make:entity

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

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

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

L'outil demande si nous souhaitons ajouter des nouveaux champs (**Field**) à l'entité.

Nous allons saisir le nom du champ pour les relier, dans ce cas-ci **Category** (qui va représenter la liaison avec l'entité **Category**) :

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

Dans l'invite de commande pour le choix du **type** nous allons choisir **relation** (la liste complète est accessible en cliquant sur **?** et **entrée**) :

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

Elle va ensuite nous demander la classe avec (**entity**) à laquelle nous souhaitons la relier. Nous allons choisir **Category** (à ne pas confondre avec le champ créé qui lui se nomme **category**):

```
What class should this entity be related to?:
> Category
```

Il va nous falloir définir le type de **relation** entre les **Classes** :

Many To One : Chaque **objet Article** possède un **objet Category**, un **objet Category** possède plusieurs **objets Article** (représente une relation **0.1 0.N**),

One To Many : Chaque **objet Article** possède plusieurs **objets Category**, un **objet Category** pointe vers un seul **objet Article** (représente une relation **0.N 0.1**),

Many To Many : Chaque **objet Article** possède plusieurs **objets Category**, chaque **objet Category** possède plusieurs **objets Article** (représente une relation **0.N 0.N** -> **table d'association**),

One To One : Les **objets Article** sont associés à un **seul objet Category**, les **objets Category** sont associé à un **seul objet Article** (représente une relation **0.1 0.1**).

Dans notre cas nous allons choisir la relation **Many To Many**.

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

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

A la question ajouter une nouvelle propriété à la classe **Category** nous cliquons sur **entrée** pour accepter (**make:entity** va effectuer la même action que dans **Article** au sein de la classe **Category**).

6.3 Configuration de l'accès à la base de données :

Afin de créer la base de données sur notre serveur MySQL nous allons configurer le fichier **.env** qui se trouve à la racine du projet **Symfony**.

6.3.1 Editer le fichier .env :

Dans ce fichier nous trouvons à la fin de celui-ci 3 modes de connexions (**PostgreSQL**, **Sqlite** ou **MySQL**).

```
29 # DATABASE_URL="sqlite:///kernel.project_dir%/var/data.db"
30 # DATABASE_URL="mysql://db_user:db_password@127.0.0.1:3306/db_name?serverVersion=5.7&charset=utf8mb4"
31 DATABASE_URL="postgresql://symfony:ChangeMe@127.0.0.1:5432/app?serverVersion=13&charset=utf8"
```

Nous allons commenter avec **#** les lignes **PostgreSQL** et **Sqlite** et dé commenter **MySQL**.

La connexion à la base de données et les divers paramètres sont représentés de cette façon :

mysql://nom_compte:mot_de_passe@url_serveur_bdd/nom_bdd?version&langue_base

Exemple :

```
DATABASE_URL="mysql://root:@127.0.0.1:3306/symfonyBDD?serverVersion=5.7&charset=utf8mb4"
```

6.3.2 Créer la base avec la Doctrine :

Pour créer la base de données sur notre serveur **MySQL** nous n'avons pas besoin de le faire à la main par avance. **Symfony** va la créer pour nous, en utilisant la commande suivante (dans un **terminal**) :

symfony console doctrine:database:create

NB : attention à bien dé commenter la ligne **extension=pdo_mysql** (enlever le **;**) dans le fichier **php.ini** de votre serveur web (**Xampp**, **Wamp** etc...).

Si les informations de connexion sont valides Symfony va créer la base sur notre serveur. Celle-ci sera par contre vide (elle ne contient pas les tables), nous allons avoir besoin d'utiliser l'outil **make-bundle** pour les ajouter.

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

6.4 Effectuer une Migration :

Pour transformer nos **classes** (générées avec **make-bundle**) en **table** dans une base de données, nous allons effectuer l'opération de **migration**.

Une **migration** est une **classe** du projet qui va décrire les changements à apporter, pour mettre à jour un schéma de base de données, de son état actuel (**vide**) vers le nouveau (**tables ajoutées**), en fonction des **attributs** des différentes **entités (classe)**. La migration va alors consister à créer les 3 **tables** au sein de la base de données (**Article, Category, User**).

6.4.1 Créer le fichier de migration :

Nous allons dans un premier temps générer un fichier pour la migration qui va être nommé comme ceci :

VersionDatedujourNumaleatoire.php : **Version2022051821161564.php**

Il se trouvera dans le répertoire **/migrations** du projet.

Pour le créer nous utiliserons la commande suivante (dans un **terminal**) :

symfony console make:migration

Si la migration se déroule comme il faut, vous aurez un message tel que :

```
mathieumithridate@FORMATEURS17-19 MINGW64 /c/symphony/website_composer (master)
$ symfony console make:migration

Success!

Next: Review the new migration "migrations/Version20220518125246.php"
Then: Run the migration with php bin/console doctrine:migrations:migrate
See https://symfony.com/doc/current/bundles/DoctrineMigrationsBundle/index.html
```

NB : Il faut vérifier que le fichier **version.php** a bien été ajouté dans le répertoire **/migrations** du projet.

Le fichier **version.php** contient une **classe** et différentes **méthodes**. La fonction qui va nous intéresser est : **up**, elle contient le code SQL de création des différentes **tables**. Au besoin nous pouvons éditer le code pour ajouter, modifier ou supprimer des **attributs** (attention que les **attributs** existent dans les **classes** correspondantes).

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

6.4.2 Effectuer la migration :

Pour créer les tables dans notre base de données nous allons utiliser le service de **migration** de **doctrine** en utilisant la commande suivante (dans un **terminal**) :

`symfony console doctrine:migrations:migrate`

Il faut répondre **Yes** pour continuer. Attention la doctrine migration va supprimer la totalité des tables et les données.

```
$ symfony console doctrine:migrations:migrate
WARNING! You are about to execute a migration in database "symfonybdd" that could result
in schema changes and data loss. Are you sure you wish to continue? (yes/no) [yes]:
> |
```

Si tout s'est bien déroulé nous aurons un message tel que :

```
[notice] Migrating up to DoctrineMigrations\Version20220518125246
[notice] finished in 1214.2ms, used 20M memory, 1 migrations executed, 6 sql queries
```

Nous pouvons vérifier au niveau de la base de données (avec par ex : **PhpMyAdmin**)

- ☐ article
- ☐ article_category
- ☐ category
- ☐ doctrine_migration_versions
- ☐ user

Nous pouvons remarquer qu'il y a une table qui ne correspond pas à notre structure :

doctrine_migration_versions.

Cette table va stocker les différents enregistrements des **doctrines** de **migration** (fichier **version.php**).

6.5 Exercice :

Mettez en pratique les notions abordées jusqu'à présent (Chapitre **6 Doctrine et ORM**), en ajoutant la relation entre **User** et **Article**.

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMPHONY

NB : Si après votre migration vos routes ne fonctionnent plus, il faut éditer le fichier **.env** et remplacer la connexion à la BDD avec le code suivant :

```
DATABASE_URL=mysql://root:@127.0.0.1:3306/testtest?serverVersion=mariadb-10.4.11
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

7 ORM-Fixtures et Faker :

Régulièrement quand on développe un projet, nous avons besoin d'ajouter des données factices afin de tester nos fonctionnalités. Nous allons voir un outil, intégrable à Symfony qui va générer pour nous des données de test aléatoires.

7.1 Installation de l'outil ORM-Fixtures :

Pour installer la dépendance nous allons saisir la commande suivante (dans un terminal) :

```
composer require --dev orm-fixtures
```

```
Symfony operations: 1 recipe (0db395d46a72ba46fda430aeec74375)
- Configuring doctrine/doctrine-fixtures-bundle (>=3.0): From
s:main
Executing script cache:clear [OK]
Executing script assets:install public [OK]
What's next?
```

7.2 Création de la classe AppFixtures.php :

Nous allons éditer dans le répertoire **/src/DataFixtures** le fichier php suivant :

AppFixtures.php va contenir le code qui va nous permettre de générer des jeux de données dans notre BDD.

Nous allons éditer le fichier avec le code ci-dessous (**partie 1**) :

```
<?php
```

```
namespace App\DataFixtures;
```

```
use App\Entity\User;
```

```
use Doctrine\Bundle\FixturesBundle\Fixture;
```

```
use Doctrine\Persistence\ObjectManager;
```

```
use Faker;
```

```
class AppFixtures extends Fixture
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
{
    public function load(ObjectManager $manager): void
    {
        //Tableau vide qui va stocker les utilisateurs que je génère
        $users = [];

        //Boucle qui va itérer 20 utilisateurs factices
        for($i=0; $i<20; $i++){
            $user = new User();
            $user->setName();
        }

        $manager->flush();
    }
}
```

7.3 Installation de la librairie Faker :

Pour intégrer la librairie Faker qui va générer des données aléatoires au projet nous allons saisir la commande suivante (dans un terminal) :

```
composer require fakerphp/faker
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
$ composer require fakerphp/faker
Info from https://repo.packagist.org: #StandwithUkraine
Using version ^1.19 for fakerphp/faker
./composer.json has been updated
Running composer update fakerphp/faker
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
- Locking fakerphp/faker (v1.19.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
- Downloading fakerphp/faker (v1.19.0)
- Installing fakerphp/faker (v1.19.0): Extracting archive
Generating optimized autoload files
109 packages you are using are looking for funding.
Use the `composer fund` command to find out more!

Run composer recipes at any time to see the status of your Symfony recipes.

Executing script cache:clear [OK]
Executing script assets:install public [OK]
```

7.4 Intégration de Faker :

Nous allons éditer le fichier **AppFixtures.php** qui se trouve dans le répertoire **/src/DataFixtures** en remplaçant le contenu par le code suivant :

<?php

namespace App\DataFixtures;

use Faker\Factory;

use App\Entity\User;

use Doctrine\Bundle\FixturesBundle\Fixture;

use Doctrine\Persistence\ObjectManager;

use Faker;

class AppFixtures extends Fixture

{

public function load(ObjectManager \$manager): void

{

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
//création d'une variable qui va contenir
$faker = Faker\Factory::create();

//Tableau vide qui va stocker les utilisateurs que l'on génère
$users = [];

//Boucle qui va itérer 20 utilisateurs factices
for($i=0; $i<20; $i++){
    $user = new User();

    //génération d'un utilisateur factice
    $user->setName($faker->name());
    $user->setFirstName($faker->firstname());
    $user->setMail($faker->email());
    $user->setPassword($faker->password());
    $user->setCreatedAt(new \DateTime());

    //stockage dans le manager
    $manager->persist($user);

    $users[] = $user;
}

$manager->flush();
}
```

7.5 Exercice :

Utiliser Faker pour générer 10 **Category** et 100 **Article**, réaliser le code correspondant dans la fonction **load** du fichier **AppFixtures.php**.

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

7.6 Sauvegarder des données en BDD :

Maintenant que nous avons généré des enregistrements factices nous allons les sauvegarder dans la base de données **MySQL**.

Pour lancer l'ajout des enregistrements en **BDD** nous allons utiliser la commande suivante (dans un **terminal**) :

symfony console doctrine:fixtures:load

```
$ symfony console doctrine:fixtures:load

Careful, database "symfonybdd" will be purged. Do you want to continue? (yes/no) [no]:
> yes

> purging database
> loading App\DataFixtures\AppFixtures
```

NB : attention **Symfony** va nous indiquer qu'il va vider la base de données (saisir **yes**).

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

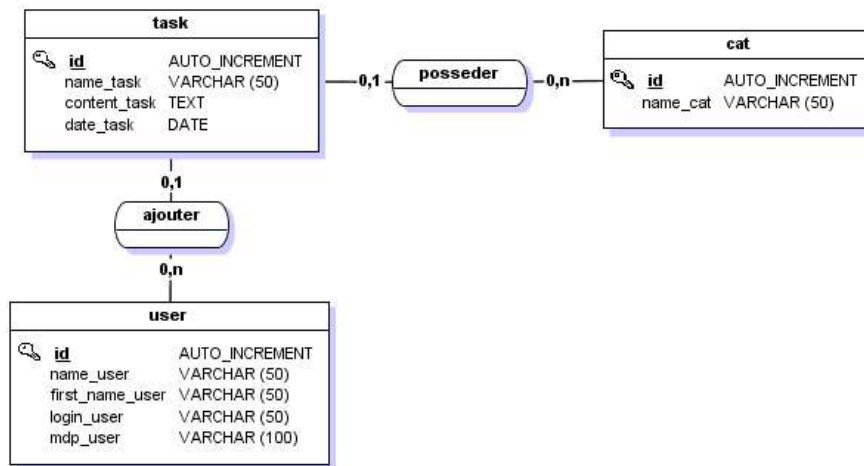
SYMFONY

8 Création d'une API (Micro-services) :

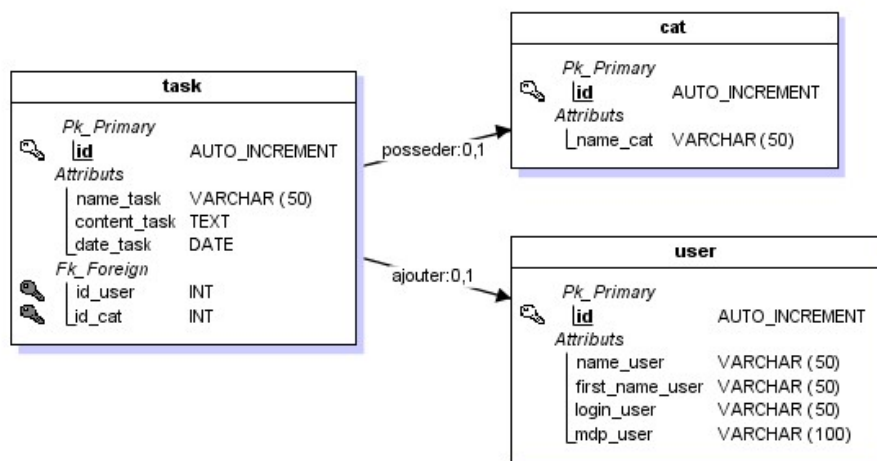
Dans cette partie nous allons voir comment créer une **API** en utilisant les **micro-services**. Nous allons reconstruire l'**API** du projet **task** que nous avons développé dans le module **PHP**.

8.1 MCD MLD du projet task :

MCD :



MLD :



Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

8.2 Définitions :

Un **micro-services** est un logiciel qui ne va faire qu'une seule tâche mais de façon optimale (par exemple renvoyer une liste d'article).

Une **API** est une interface de programmation qui va permettre d'interagir avec un ou plusieurs **micro-services** pour accéder, ajouter, modifier, supprimer des enregistrements en base de données, en passant par une **URL** et des paramètres

(Exemple d'API : <https://adrardev.fr/task/api/task.php?task=1>).

8.3 Structure de l'API :

Nous allons utiliser **composer** pour créer la base du projet sous la forme de micro-services, pour se faire saisir la commande suivante (depuis un **terminal**) :

```
composer create-project symfony/skeleton microservice_composer
```

8.4 Création des controllers :

Comme vu précédemment nous allons créer nos controller avec make-Bundle, saisir la commande suivante (dans un **terminal**) :

```
symfony console make:controller
```

Nous allons rencontrer l'erreur suivante :

```
There are no commands defined in the "make" namespace.

You may be looking for a command provided by the "MakerBundle" which is currently not installed. Try running "composer require symfony/maker-bundle --dev".
```

Celle-ci est normale car nous avons créé un projet sous forme de **micro-services**, **Symfony** n'installe alors que les composants de base.

Symfony nous propose pour installer l'outil la commande suivante à saisir (dans un **terminal**) :

```
composer require symfony/maker-bundle --dev
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Quand nous allons tenter de relancer la commande précédente (**`symfony console make:controller`**), nous pouvons avoir une nouvelle erreur :

```
[ERROR] Missing package: to use the make:controller command, run:
composer require doctrine/annotations
```

Symfony nous propose pour installer l'outil la commande suivante à saisir (dans un **terminal**) :

`composer require doctrine/annotations`

Cela va nous rajouter l'outil **d'annotations** pour générer nos routes.

Nous pouvons maintenant générer nos controllers :

Task, **Cat** et **User**, **make-bundle** va générer les 3 controllers **TaskController**, **CatController** et **UserController** dans le répertoire **src/Controller**.

Nous devons relancer la commande **`symfony console make:controller`** 3 fois.

Pour vérifier le bon fonctionnement nous pouvons tester dans le navigateur internet les URL suivantes :

localhost :8000/task, localhost :8000/cat, localhost :8000/user, elles doivent renvoyer un json comme ci-dessous :

```
{"message":"Welcome to your new controller!","path":"src\\Controller\\CatController.php"}
```

NB : Si jamais cela ne fonctionne pas il faut arrêter et relancer le serveur, si vous avez déjà un site qui tourne sur le serveur, le nouveau (api) prendra le port 8001 (avec les commandes : **`symfony server:stop`** et **`symfony server:start -d`**).

8.5 Créer les entités :

En utilisant les notions du **chapitre 6 Doctrine et ORM**, nous allons créer les différentes **entités** correspondantes à nos 3 **controllers**. Avec l'aide de l'outil **make-bundle** et de la commande suivante à saisir (dans un **terminal**) :

`symfony console make:entity`

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Nous allons devoir créer les **entités** comme défini dans les **diagrammes MCD MLD** (qui se trouvent plus haut dans ce chapitre). Pour reproduire les **Foreign Key** nous utiliserons les **relations (many to many, one to many, etc...)**.

Elles seront les suivantes (**relations**) :

Task many to one avec **Cat**,

Task many to one avec **User**,

User one to many avec **Task**,

Cat one to many avec **Task**.

NB : Nous pouvons voir apparaître l'erreur suivante :

```
[ERROR] Missing package: to use the make:entity command, run:
composer require orm
```

Dans ce cas-là, nous allons simplement devoir installer ORM (répondre **n** à la question **docker module** puis **entrée**) avec la commande suivante à saisir (dans un **terminal**) :

```
composer require orm
```

8.6 Configurer l'accès à la base de données :

Nous allons devoir configurer le fichier **.env** avec les informations de connexion à la base de données.

```
DATABASE_URL="mysql://root:@127.0.0.1:3306/taskapi?serverVersion=5.7&charset=utf8mb4"
```

NB : Nous éditerons le fichier comme vu dans la partie 6.3.1, pensez à commenter les lignes avec un **PostgreSQL** et **Sqlite** avec un **#**.

8.7 Créer la base de données :

Nous allons créer la base en utilisant la commande suivante (dans un **terminal**)

```
symfony console doctrine:database:create
```

NB : En cas d'erreurs, vérifier si les informations dans le fichier **.env** sont correctes, que vous n'ayez pas une base qui porte le même nom sur votre serveur, enfin que le serveur **MySQL** soit bien lancé (port 3306-3307 **MariaDB**).

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

8.8 Paramétrage et migration :

Créer le fichier de migration **version.php** en utilisant la commande suivante (dans un **terminal**) :

symfony console make:migration

```
$ symfony console make:migration

Success!

Next: Review the new migration "migrations/Version20220519125536.php"
Then: Run the migration with php bin/console doctrine:migrations:migrate
See https://symfony.com/doc/current/bundles/DoctrineMigrationsBundle/index.html
```

Nous pouvons vérifier que celle-ci c'est bien déroulé en allant dans le répertoire **/migrations**

Si vous avez bien notre fichier **version....php**

Ensuite nous allons effectuer la migration en utilisant la commande suivante (dans un **terminal**) :

symfony console doctrine:migrations:migrate

Si la **procédure** s'est bien déroulé, nous devons avoir un **message** de la sorte dans le **terminal** :

```
$ symfony console doctrine:migrations:migrate

WARNING! You are about to execute a migration in database "taskapi" that could result in
schema changes and data loss. Are you sure you wish to continue? (yes/no) [yes]:
>

[OK] Already at the latest version ("DoctrineMigrations\Version20220519125536")
```

Et les 3 tables (**Task**, **Cat** et **User**) ainsi que la table de **migration** sur notre serveur **MySQL** (**Wamp**, **Xampp**, **Mamp** etc...)

- ☐ cat
- ☒ doctrine_migration_versions
- ☐ task
- ☐ user

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordnatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordnatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

8.9 Exercice :

En utilisant les notions abordées au **chapitre 7 (Fixtures et Faker)**, vous allez générer des données factices : 100 utilisateurs, 400 tâches et 50 catégories. Editez le fichier AppFixtures.php qui se trouve dans le répertoire : **/src/DataFixtures/**.

NB : Vous allez avoir besoin d'installer **Fixtures** et **Faker**. Pensez à bien importer vos **différentes entités** (exemple : **use App\Entity\Task**) et la librairie **Faker** (**use Faker**).

NB : Vous trouverez à l'adresse ci-dessous des exemples d'utilisation de **Faker** :

<https://fakerphp.github.io/formatters/>

9 Création d'une API suite (outils et Méthodes) :

Dans ce chapitre nous allons voir comment utiliser des outils de test et comment créer des méthodes qui vont nous retourner des fichiers JSON.

9.1 Installation de Postman :

Afin de tester et d'afficher les retours de notre **API (GET, POST, PUT, PATCH, DELETE, etc...)**, nous allons installer **Postman**.

L'outil se télécharge à l'adresse ci-dessous :

<https://www.postman.com/downloads/>

Il fonctionne sous **Windows, Mac OS** ou **LINUX**, nous pouvons également utiliser une **extension de Google Chrome** qui se nomme **Advanced Rest Client**.

Elle est disponible à l'adresse ci-dessous :

<https://chrome.google.com/webstore/detail/advanced-rest-client/hgmlloofddfdnphfgcellkdfbfjeloo?hl=fr>

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

9.2 Edition des controllers :

Afin de vérifier si nous récupérons bien les données depuis la base de données, nous allons éditer le **TaskController.php** qui se trouve dans le répertoire **src/controller** comme ci-dessous :

```
<?php

namespace App\Controller;

use App\Entity\Task;
use App\Repository\CatRepository;
use App\Repository\TaskRepository;
use App\Repository\UserRepository;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\Serializer\SerializerInterface;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

class TaskController extends AbstractController
{
    // je rajoute à la route la méthode associée GET
    #[Route('/task', name: 'app_task_index', methods: 'GET')]
    public function index(
        TaskRepository $taskRepository
    ): Response {

        //récupération de toutes les taches
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
$tasks = $taskRepository->findAll();

dd($tasks);

}

}
```

La méthode **findAll()** est une des **méthodes** auto générée par **Symfony**, elle se trouve dans le fichier **TaskRepository.php** du répertoire **src/repository**. Elle va nous retourner un **tableau d'objets** de toutes les tâches en **BDD**.

La méthode **dd()** est un **var_dump()** amélioré intégré à **Symfony** qui va nous permettre de visualisé le contenu de nos **variables**. Elle est plus rapide, plus efficace et surtout nous permet d'avoir un affichage beaucoup plus lisible de nos **variables**.

Si nous exécutons la route **localhost :8000/task** nous allons avoir un affichage comme ci-dessous :

```
^ array:150 [▼
  0 => App\Enti...\Task {#249 ▼
    -id: 1
    -name_task: "A exercitationem ad omnis perspiciatis."
    -content_task: "Saepe omnis et totam eum modi quo. Nulla ipsam ut velit non suscipit. Aut distinctio saepe eos. Error et quae qui enim sint quos. Molestias dolor ipsam quos. Co ▶"
    -date_task: DateTime @1653419447 {#251 ▶}
    -users: Proxies...\User {#200 ▶}
    -cats: Proxies...\Cat {#218 ▶}
  }
  1 => App\Enti...\Task {#210 ▶}
  2 => App\Enti...\Task {#318 ▶}
  3 => App\Enti...\Task {#215 ▶}
  4 => App\Enti...\Task {#211 ▶}
  5 => App\Enti...\Task {#206 ▶}
  6 => App\Enti...\Task {#395 ▶}
  7 => App\Enti...\Task {#398 ▶}
  8 => App\Enti...\Task {#400 ▶}
```

9.3 Exercice :

Editez les controllers **CatController.php** et **UserController.php**, pour faire en sorte que les routes correspondantes à la méthode **index** retourne un affichage équivalent au controller **TaskController.php**. (En utilisant la méthode **dd()**). Affichez dans **Postman** ou **Advanced Rest Client** le résultat de la requête **GET : localhost :8000/task**

Bonus : Ajoutez des méthodes **index2()** et les **routes** correspondantes dans chaque **controller** afin qu'elles retournent un **var_dump** plutôt que **dd** et faites une capture d'écran pour comparer les résultats.

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

9.5 Sérialisation :

Nous allons installer la librairie **serialise** avec la commande ci-dessous (à lancer dans un **terminal**) :

```
composer require symfony/serializer-pack
```

Le processus de transformation est fait par **Normalizer**, il va faire passer d'un objet complexe à un tableau associatif simple, ce processus se nomme la **normalisation**.

Nous allons modifier la fonction index de **TaskController.php** comme ci-dessous :

```
<?php
```

```
namespace App\Controller;
```

```
use App\Entity\Task;
```

```
use App\Repository\CatRepository;
```

```
use App\Repository\TaskRepository;
```

```
use App\Repository\UserRepository;
```

```
use Doctrine\ORM\EntityManagerInterface;
```

```
use Symfony\Component\HttpFoundation\Request;
```

```
use Symfony\Component\HttpFoundation\Response;
```

```
use Symfony\Component\Routing\Annotation\Route;
```

```
use Symfony\Component\Serializer\SerializerInterface;
```

```
use Symfony\Component\Serializer\Normalizer\NormalizerInterface;
```

```
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
```

```
class TaskController extends AbstractController
```

```
{
```

```
// je rajoute à la route la méthode associée GET
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
#[Route('/task', name: 'app_task_index', methods: 'GET')]

public function index(
    TaskRepository $taskRepository,
    NormalizerInterface $normalizer
): Response {
    //récupération de toutes les taches
    $tasks = $taskRepository->findAll();

    //stockage des tasks passées dans la méthode normalize
    $tasksNormalize = $normalizer->normalize($tasks);

    dd($tasksNormalize);
}
```

Nous allons nous retrouver avec une erreur circulaire. Cette exception est appelée **circular reference** (référence circulaire). Une **référence circulaire** c'est quoi ? C'est tout simplement à cause de la relation entre task et cat. Une tâche fait référence à une catégorie qui fait référence à une tâche etc... Ils bouclent sur eux-mêmes.

CircularReferenceException

H

A circular reference has been detected when serializing the object of class "App\Entity\Task" (configured limit: 1).

Nous allons corriger cette erreur en utilisant les groupes :

Nous allons éditer le fichier **Task (entity)** comme ci-dessous en groupant à l'aide des annotations tous les attributs (or **users** et **cats**) comme ci-dessous :

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
<?php

namespace App\Entity;

use App\Repository\TaskRepository;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Serializer\Annotation\Groups;

#[ORM\Entity(repositoryClass: TaskRepository::class)]
class Task
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column(type: 'integer')]
    #[Groups('task:readAll')]
    private $id;

    #[ORM\Column(type: 'string', length: 255)]
    #[Groups('task:readAll')]
    private $name_task;

    #[ORM\Column(type: 'text')]
    #[Groups('task:readAll')]
    private $content_task;

    #[ORM\Column(type: 'datetime')]
    #[Groups('task:readAll')]
    private $date_task;
```

Nous allons ensuite éditer la fonction **index** de **TaskController.php** comme ci-dessous :

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
<?php

namespace App\Controller;

use App\Entity\Task;
use App\Repository\CatRepository;
use App\Repository\TaskRepository;
use App\Repository\UserRepository;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\Serializer\SerializerInterface;
use Symfony\Component\Serializer\Normalizer\NormalizerInterface;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

class TaskController extends AbstractController
{
    // je rajoute à la route la méthode associée GET
    #[Route('/task', name: 'app_task_index', methods: 'GET')]
    public function index(
        TaskRepository $taskRepository
    ): Response {
        return $this->json($taskRepository->findAll(), 200, [], ['groups' =>
            'task:readALL']);
    }
}
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
}
```

La fonction renvoie (return) toutes les tâches (méthode **findAll**, un code **200**, sous forme d'un **tableau**, uniquement les attributs groupés par **task:readAll**.

9.6 Exercice :

Editez les contrôleurs cat et user pour faire en sorte qu'ils retournent un fichier json. Inspirez-vous de l'exemple du cours (task). Testez vos routes dans Postman ou équivalent pour voir si vous récupérez bien du json.

10 API POST ajouter des enregistrements en BDD :

Dans ce chapitre nous allons voir comment faire en sorte de récupérer un nouvel enregistrement en mode POST.

10.1 Ajout de la méthode create en POST :

Nous allons implémenter une nouvelle méthode dans le contrôleur task (**TaskController.php**) comme ci-dessous :

Nous allons dans ajouter la ligne ci-dessous (pour pouvoir utiliser **request**) :

```
use Symfony\Component\HttpFoundation\Request;
```

Fonction à ajouter :

```
#[Route('/task', name: 'app_task_create', methods: 'POST')]
```

```
public function create_task(
```

```
    Request $request
```

```
): Response {
```

```
    $json_recup = $request->getContent();
```

```
    dd($json_recup) ;
```

```
}
```

Nous constatons que la route est la même que pour la récupération du json des toutes les tâches, le paramètre qui va différer est la méthode (**POST**), nous pouvons sur la même route avoir accès à des fonctions différentes par la méthode (**GET**, **POST** etc...).

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Afin de vérifier le bon fonctionnement de la fonction **create_task()** nous allons utiliser **Postman** (ou équivalent) pour essayer d'ajouter une tâche à la **BDD**. Comme ci-dessous :

`http://localhost:8001/task`



POST `http://localhost:8001/task`

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1  {
2    "name_task": "new task",
3    "content_task": "Ma nouvelle tache ajouté depuis Postman",
4    "users": {
5      "id": 14
6    },
7    "cats": {
8      "id": 5
9    }
10 }

```

Nous allons vérifier le retour dans **Postman**.

Dans la réponse **body** -> **preview** de **Postman** nous allons avoir ceci :

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize

```
{
  "name_task": "new task",
  "content_task": "Ma nouvelle tache ajouté depuis Postman",
  "users": {
    "id": 14
  },
  "cats": {
    "id": 5
  }
}
```

Nous allons dans un premier temps modifier notre méthode en effectuant une sérialisation comme ci-dessous :

```
#[Route('/task', name: 'app_task_create', methods: 'POST')]
```

```
public function create_task(
```

```
    Request $request,
```

```
    SerializerInterface $serializer,
```

```
    EntityManagerInterface $em
```

```
): Response {
```

```
    $task_recup = $request->getContent();
```

```
    //désérialisation de la tâche (json en POST)
```

```
    $task = $serializer->deserialize($task_recup, Task::class, 'json');
```

```
    //dump and die de $task
```

```
    dd($task);
```

```
}
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Postman nous retourne un résultat comme ci-dessous :

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize

```
^ App\Entity\Task {#2085
  -id: null
  -name_task: "new task"
  -content_task: "Ma nouvelle tache ajouté depuis Postman"
  -date_task: null
  -users: App\Entity\User {#294
    -id: null
    -name_user: null
    -first_name_user: null
    -login_user: null
    -mdp_user: null
    -tasks: Doctrine\ORM\ArrayCollection {#268
      -elements: []
    }
  }
  -cats: App\Entity\Cat {#303
    -id: null
    -name_cat: null
    -tasks: Doctrine\ORM\ArrayCollection {#309
      -elements: []
    }
  }
}
```

L'entité **Task** à bien été désérialisé mais **User** et **Cat** ne le sont pas, nous allons corriger le problème en éditant la fonction **create_task** comme ci-dessous :

```
#[Route('/task', name: 'app_task_create', methods: 'POST')]
```

```
public function create_task(
```

```
    Request $request,
```

```
    UserRepository $userRepository,
```

```
    CatRepository $catRepository,
```

```
    SerializerInterface $serializer,
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
EntityManagerInterface $em
```

```
): Response {
```

```
    $task_recup = $request->getContent();
```

```
    //décodage du json récupéré
```

```
    $data = $serializer->decode($task_recup, 'json');
```

```
    //récupération du user par son id
```

```
    $user = $userRepository->find($data['users']['id']);
```

```
    //récupération de la catégorie par son id
```

```
    $cat = $catRepository->find($data['cats']['id']);
```

```
    //création d'une nouvelle tâche
```

```
    $task = new Task();
```

```
    $task->setNameTask($data['name_task']);
```

```
    $task->setContentTask($data['content_task']);
```

```
    $task->setDateTask(new \DateTimeImmutable);
```

```
    $task->setUsers($user);
```

```
    $task->setCats($cat);
```

```
    //on fait persister les données
```

```
    $em->persist($task);
```

```
    //envoi en BDD
```

```
    $em->flush();
```

```
    //dump and die de $task
```

```
    dd($task);
```

```
}
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Nous pouvons vérifier en BDD que la nouvelle tâche à bien été ajouté :

151	14	5 new task	Ma nouvelle tache ajouté depuis Postman	2022-05-25 02:54:15
-----	----	------------	---	---------------------

10.2 Exercice :

Créer les méthodes **create_cat** et **create_user** en vous inspirant de l'exemple précédent.

11 Création d'une web-App :

Dans ce chapitre nous allons aborder le développement d'une **Web-App** (site Blog), les différentes briques de code détaillés dans ce chapitre et les suivants vous permettra de pouvoir développer des sites web avec le Framework **Symfony**.

11.1 Création d'un projet Web-App :

Nous devons dans un premier temps lancer la création d'un nouveau projet en ligne de commande, soit avec l'exécutable **Symfony** ou le gestionnaire de paquet **composer**. Pour ce faire nous allons saisir les commandes suivantes (dans un **terminal**) :

Dans un le dossier des projets symfony, lancer un **terminal** (Git Bash).

`symfony new nom_site -webapp` ou

`composer create-project symfony/website-skeleton nom_site`

Cela va créer un nouveau projet **Web-App** (site web).

11.2 Configuration et intégration des dépendances du projet :

Afin de construire une **Web-App**, nous allons devoir intégrer différentes dépendances (**module**, **librairie**, **composant**). Avant tout nous devons nous ouvrir un terminal à la racine de notre projet :

`cd nom_site`

NB : la commande **change directory** de **Bash** nous permet de nous déplacer dans le répertoire du projet.

Nous allons ajouter les **dépendances** avec les commandes suivantes (dans le **terminal**) :

`composer require --dev annotations` (Routing)

Ou **`composer require doctrine/annotations`**

`composer require symfony/maker-bundle --dev` (Générateur)

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

`composer require --dev orm-fixtures` (Générateur de données de test)

`composer require fakerphp/faker` (Générateur de données factices)

`composer require orm` (facultatif)

`composer require symfony/serializer-pack` (Outil de sérialisation)

11.3 Ajouter une structure de données (entités, classe) :

Pour pouvoir interagir avec une **base de données**, nous devons générer, à l'aide de **l'ORM** la **structure** de nos données côté **code (classes et méthodes)**. Pour ce faire nous allons créer avec l'aide de l'outil make-bundle nos entités et les repository correspondants. **Les entités** vont contenir le code de chaque **table** (représentation **objet** des **tables** de la **BDD**). **Les repository** vont eux contenir les **fonctions** pour interagir avec la **BDD (requêtes SQL)**. Pour se faire nous allons saisir la commande suivante (dans un **terminal**) :

`symfony console make:entity`

NB : La commande devra être utilisée, autant de fois que nous avons des **tables** en **BDD** (sauf pour **les tables d'associations**, celle-ci seront gérées par les relations **ManyToMany**).

11.4 Créer et migrer la base de données :

Symfony, comme vu dans les chapitres précédents à la capacité d'automatiser la création de nos bases de données. Pour ce faire nous allons devoir suivre plusieurs étapes :

11.4.1 Créer une base vide (sans les tables) :

Avant de créer la base de données sur le serveur, nous allons devoir configurer l'accès à celle-ci, par le biais du fichier **.env**, qui se trouve à la racine du projet, comme ci-dessous :

`DATABASE_URL="mysql://Login:mdp@127.0.0.1:3306/nom_db?serverVersion=mariadb-10.4.11&charset=utf8mb4"`

NB : Les valeurs (**identifiant**, **MDP**, **URL** du serveur et **port**, nom de la **BDD**) sont à personnaliser avec les informations de votre serveur **MYSQL**. La commande suivante va créer la **base de données** (dans un **terminal**) :

`symfony console doctrine:database:create`

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

11.4.2 Migrer la structure de données :

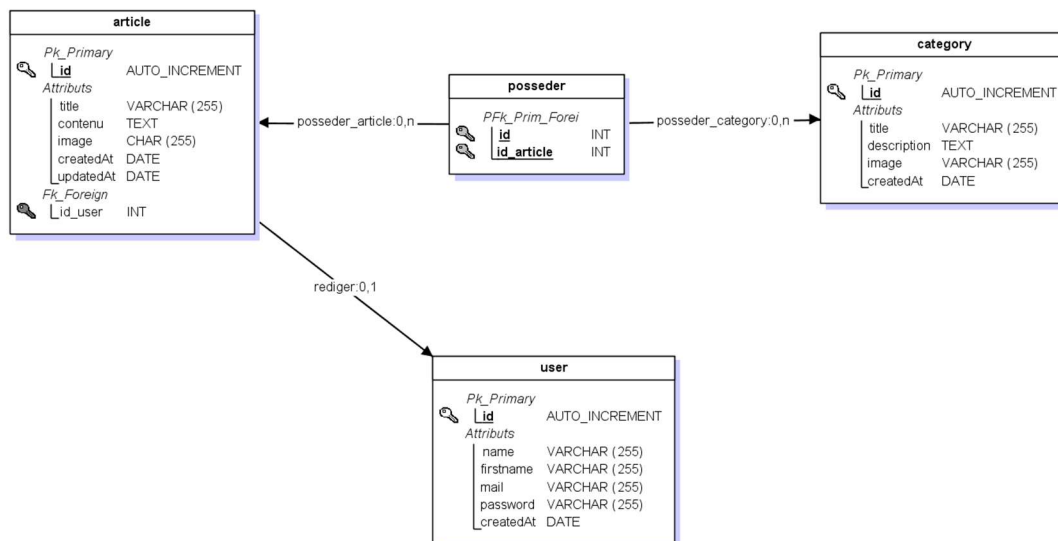
Pour effectuer la **migration** des données, nous allons créer un fichier de migration (contient les **requêtes SQL de structure**) et l'appliquer sur la base précédemment ajoutée au serveur **MYSQL**, en utilisant les commandes suivantes (dans le **terminal**) :

`symfony console make:migration` (créer le fichier **Version.php**),

`symfony console doctrine:migrations:migrate` (ajoute les **tables** dans la **BDD**).

11.5 Exercice :

Depuis le **MLD** ci-dessous, créer la structure de données et migrer la **base de données** du projet blog :



L'attribut **id_user** de l'entité **Article** (**Foreign Key**) sera nommé **writeBy** (classe **Article**) et la relation sera de type **ManyToOne** -> **User**,

Ajoutez un attribut **category** dans **Articles** avec comme relation **ManyToOne** -> **Category**,

Ajoutez un attribut **articles** dans **Category** avec comme relation **ManyToOne** -> **Article**.

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

11.6 Ajoutez des données de tests :

Pour pouvoir tester notre structure de données nous allons générer avec les outils **Fixture** et **Faker** des données factices, elles permettront d'alimenter nos projets et de vérifier le bon fonctionnement du site internet.

Pour se faire nous allons devoir éditer la fonction load du fichier `src\DataFixtures\AppFixtures.php`.

Pour ajouter à la **BDD** ces **données**, nous allons saisir la commande suivante (dans le **terminal**) :

```
symfony console doctrine:fixtures:load
```

Des exemples de méthodes pour **faker** sont disponibles dans la doc à cette adresse :

<https://fakerphp.github.io/formatters/>

11.7 Exercice :

Modifier la méthode load pour ajouter à la BDD :

200 articles,

50 catégories et

50 utilisateurs.

NB : Faites bien attention à utiliser les bon setter (qui se trouvent dans les entités correspondantes) ainsi que des méthodes valides de faker (voir le lien plus haut).

12 Construire une page dynamique :

Nous avons vu dans un chapitre précédent comment personnaliser une interface (**twig**) lié à un **controller (route)**. Dans ce chapitre nous allons voir la construction d'un **controller** et d'une vue qui va représenter des **enregistrements** de la **BDD** dans des **composants Bootstrap** au sein d'une vue (**index.html.twig**).

12.1 Génération d'un controller :

Pour nous aider à la création d'un **controller**, de la **route** et de la **vue associée** nous allons utiliser l'outil **make-bundle**. Celui-ci va générer pour nous lesdits fichiers. Pour cela nous allons utiliser la commande suivante (dans le **terminal**) :

```
symfony console make:controller
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Auteur(s)	Relu, validé et visé par :		Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE :		25/09/2022	
	Resp. Secteur Numérique Occitanie			
	Florence CALMETTES :		Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Coordinatrice Filière Syst. & Réseaux			
	Sophie POULAKOS :			
	Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT			

SYMFONY

12.2 Modification de la méthode index :

Pour pouvoir injecter dans le Template twig (index.html.twig) les données issues de la base de données. Nous allons modifier la fonction comme ci-dessous :

```
<?php
namespace App\Controller;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use App\Repository\ArticleRepository;

class AccueilController extends AbstractController
{
    #[Route('/accueil', name: 'app_accueil')]
    public function index(ArticleRepository $articleRepository): Response
    {
        $articles = $articleRepository->findAll();
        return $this->render('accueil/index.html.twig',
            ['articles' => $articles]);
    }
}
```

12.2 Modification de l'interface twig :

Comme vu dans un chapitre précédent, nous avons la possibilité d'intégrer du code **Bootstrap** au sein d'un fichier **html.twig**. Pour ce faire nous devons récupérer le fichier **bootstrap.min.css** depuis le site web :

<https://bootswatch.com/>

Le fichier **CSS** devra être ajouté au projet dans le répertoire **public->asset->css**, afin qu'il soit disponible pour notre visiteur dans son navigateur web.

Pour l'intégrer aux différentes pages du site nous devons l'ajouter dans le fichier **templates\base.html.twig** au sein d'un bloc **twig stylesheets** comme ci-dessous :

```
{% block stylesheets %}
    <link rel="stylesheet" href="../asset/css/bootstrap.min.css">
{% endblock %}
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Ce fichier sert de **base** à toutes les interfaces (**extends**), nous devons y déplacer tous les éléments communs aux différents **templates**.

12.3 Bloc FOR twig :

Dans un chapitre précédent nous avons étudié plusieurs blocs twig, nous allons voir l'utilisation du bloc **For** qui va effectuer une **boucle** et répéter son contenu tant qu'il y a des données.

Il démarre par `{% for élément in tableau %}` et se termine par `{% endblock %}`

La récupération du tableau se fait par un paramètre de la fonction index méthode render (le tableau après le fichier de fichier de l'interface twig à charger) :

```
return $this->render('accueil/index.html.twig', ['articles' => $articles]);
```

Les données dynamique seront encadrées par `{{ article.attribut }}`

Exemple avec une **card Bootstrap** article :

```
{% block body %}
<div class="row">
  {% for article in articles %}
    <div class="card text-white bg-primary mb-3" style="max-width: 18rem;">
      <div class="card-header">
        <h5 class="card-title">{{ article.title }}</h5>
      </div>
      
      <div class="card-body">
        <a href="#" class="btn btn-primary">Détail</a>
      </div>
    </div>
  {% endfor %}
</div>
{% endblock %}
```

Ce Bloc va **répéter** son contenu autant de fois qu'il y a des **articles**.

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

12.4 Exercice :

Créer un contrôleur **Accueil** avec **make-bundle**, télécharger un thème **Bootstrap** et intégrer le au projet (**templates\base.html.twig**). Nettoyer le code inutile dans le fichier base, ajouter un menu **Bootstrap** dans celui-ci.

Dans le fichier **templates\accueil\index.html.twig** insérer le code du chapitre 12.3.

Éditez la méthode index comme dans le chapitre 12.1.

Vérifier que vous affichez bien avec la route <http://localhost:8000/route> la liste des articles au sein des **cards Bootstrap**.

12.5 Intégration d'attribut et filtrage :

Nous allons voir dans cette partie comment nous allons pouvoir filtrer un attribut. Si nous souhaitons par exemple afficher la date en l'état cela sera impossible (c'est un Objet **DateTimeImmutable** ou **DateTimeInterface**) pour se faire nous devons utiliser le symbole **|** et ajouter un typage. Nous ne pouvons pas afficher **des objets**, uniquement des **strings** au sein d'un **Template Twig**.

Nous allons ajouter à notre fichier **templates\accueil\index.html.twig** une balise **date** comme ci-dessous :

```
<div class="card text-white bg-primary mb-3" style="max-width: 18rem;">
  <div class="card-header">
    <h5 class="card-title">{{ article.title }}</h5>
    <p class="date">Ajouté le {{ article.createdAt | date("d/m/Y") }}</p>
  </div>
```

NB : Après le caractère **|** nous ajoutons la fonction **date** PHP pour afficher la date au format numéro du jour, mois et année ex : **31/05/2022**. Sans le filtre nous aurions une erreur comme ci-dessous :

Error

Object of class DateTime could not be converted to string

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

12.6 Afficher un seul enregistrement dans une interface :

12.6.1 Création de la fonction :

Pour effectuer cette action nous allons devoir ajouter une nouvelle méthode dans notre contrôleur. Pour récupérer un enregistrement par son ID, nous allons utiliser la fonction `find` (qui se trouve dans le repository) et lui passer en paramètre un ID. Nous allons écrire une fonction comme ci-dessous :

```
#[Route('/show/{id}', name: 'app_accueil_show')]
public function GetArticleById(ArticleRepository $articleRepository, $id): Response
{
    $article = $articleRepository->find($id);

    return $this->render('show/index.html.twig',
        ['article' => $article]);
}
```

NB : Nous allons passer le contenu de `$article` dans la fonction `render` (tableau associatif) et également l'interface `twig` qui va afficher le résultat. La route sera de type `localhost :8000/show/1`, on passe l'ID en second paramètre.

12.6.2 création de l'interface twig :

Nous allons ajouter un nouveau répertoire (`show`) dans `templates` avec à l'intérieur un fichier `index.html.twig` comme ci-dessous :

```
▼ show
  └─ index.html.twig
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMPHONY

Le fichier va contenir notre nouvelle vue. Nous allons intégrer l'exemple de code suivant :

```
{% extends 'base.html.twig' %}
{% block title %}Detail article{% endblock %}
{% block body %}
<h3 style="padding-left:10px">Détail de l'article :</h3>
  <div class="container" style="margin-left:0;">
    <div class="card bg-secondary mb-3" style="max-width: 20rem;">
      <h3 class="card-header">{{article.title}}</h3>
      <div class="card-body">
        <div class="d-block user-select-none" >
          
        </div>
        <p class="card-text">{{article.contenu}}</p>
      </div>
      <div class="card-footer text-muted">
        Ajouté le {{ article.createdAt | date("d/m/Y") }}
        à {{ article.createdAt | date("H/i") }}
      </div>
    </div>
  </div>
{% endblock %}
```

NB : nous allons obtenir un rendu mais le fichier CSS ne se charge, nous allons corriger le lien dans base.html.twig comme ceci :

```
{% block stylesheets %}
  <link rel="stylesheet" href="/asset/css/bootstrap.min.css">
{% endblock %}
```

12.6.3 : Correction des erreurs :

Si un utilisateur du site essaye de passer un paramètre (ID) qui n'existe pas dans la route, nous recevrons un **message d'erreur** de la sorte :

Impossible to access an attribute ("title") on a null variable.

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Symfony nous indique que l'attribut **title** (d'**article.title**) est **null**, nous allons corriger cette erreur en traitant le cas où l'article n'existe pas -> nous redirigerons vers la page qui contient toutes les **cards**. Pour ce faire nous allons utiliser la méthode **redirectToRoute** qui prend en paramètre (la propriété name de la route. Nous allons modifier la méthode **getArticleById** comme ci-dessous :

```
#[Route('/show/{id}', name: 'app_accueil_show')]
public function getArticleById(ArticleRepository $articleRepository,
$id): Response
{
    $article = $articleRepository->find($id);
    //test si $article est null
    if(!$article){
        //redirection vers la route /home
        return $this->redirectToRoute('app_accueil');
    }
    return $this->render('show/index.html.twig',
    ['article' => $article]);
}
```

12.7 Liaison bouton vers une route :

Nous allons modifier notre interface **accueil/index.html.twig** pour faire en sorte de rediriger le clic du bouton vers la route en utilisant un lien, le fichier sera modifié comme dans l'exemple ci-dessous :

```
<div class="card-body">
    <a href="{{ path('app_accueil_show', {'id':article.id}) }}"
    class="btn btn-primary">Détail</a>
</div>
```

NB : Nous allons utiliser la balise **path** avec comme **valeur** le contenu de l'attribut **name** de la **route** et le paramètre **id** (cela va rediriger vers la fonction **getArticleById**).

12.8 Exercice :

Modifier le contrôleur **AccueilController.php** en intégrant la fonction **getArticleById** qui va afficher un article par son ID,

Créer la vue **templates\show\index.html.twig**.

Editer la vue **accueil\index.html.twig** et ajouter le bouton avec le lien qui mène vers la page de détail.

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

13 Création d'un panel Administrateur :

Dans cette partie, nous allons voir comment intégrer un panel administrateur (Dashboard) à notre projet, afin de pouvoir gérer le CRUD de nos entités (la création, la modification et la suppression).

13.1 installation et création du Dashboard :

Nous allons devoir installer le package Dashboard en utilisant la ligne de commande suivante (dans un **terminal**) :

```
composer require --dev admin
```

Nous allons ensuite le construire en utilisant make-bundle en saisissant la commande suivante (dans un **terminal**) :

```
symfony console make:admin:dashboard
```

```
Which class name do you prefer for your Dashboard controller? [DashboardController]:
>

In which directory of your project do you want to generate "DashboardController"? [src/Controller/Admin/]:
>

[OK] Your dashboard class has been successfully generated.

Next steps:
* Configure your Dashboard at "src/Controller/Admin/DashboardController.php"
* Run "make:admin:crud" to generate CRUD controllers and link them from the Dashboard.
```

L'outil nous génère un nouveau **controller** qui se trouve dans le répertoire :

src\Controller\Admin\DashboardController.php.

Pour vérifier que l'installation c'est bien effectué, nous pouvons saisir la route suivante :

localhost:8000/admin.





Nous allons avoir une page comme ci-dessous :

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Welcome to EasyAdmin 4

You have successfully installed EasyAdmin 4 in your application.

-  **Read EasyAdmin Docs**
Learn how to customize EasyAdmin to fit your needs
-  **Watch EasyAdmin Course on SymfonyCasts**
More than 30 videos to learn how to build a powerful admin area
-  **Sponsor EasyAdmin**
Fund the development of new features. One-time or regular donations
-  [Star EasyAdmin on GitHub](#)
Help us promote EasyAdmin to reach new developers

13.2 Configuration intégration des entités (CRUD) :

Nous allons devoir intégrer les différentes **entités** au **Dashboard**, pour ce faire nous allons utiliser la commande suivante (dans un **terminal**) :

`symfony console make:admin:crud`

L'outil va nous demander de sélectionner l'entité dont le **CRUD** va être généré (saisir le **nombre** puis **3 fois entrée**), nous pouvons voir le résultat de l'ajout de la première entité (**Article**) ci-dessous :

```
mathieumithridate@FORM21-06 MINGW64 /c/symfony/blog
$ symfony console make:admin:crud

Which Doctrine entity are you going to manage with this CRUD controller?:
[0] App\Entity\Article
[1] App\Entity\Category
[2] App\Entity\User
> 0

Which directory do you want to generate the CRUD controller in? [src/Controller/Admin/]
:
>

Namespace of the generated CRUD controller [App\Controller\Admin]:
>

[OK] Your CRUD controller class has been successfully generated.

Next steps:
* Configure your controller at "src/Controller/Admin/ArticleCrudController.php"
* Read EasyAdmin docs: https://symfony.com/doc/master/bundles/EasyAdminBundle/index.html
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

13.3 Exercice :

Générer les CRUD des 2 entités restantes (Category et User).

13.4 Configuration intégration des entités (CRUD) (suite) :

Maintenant que les CRUD (**3 entités**) sont générés nous allons devoir éditer le fichier `src\Controller\Admin\DashboardController.php`. Dans un premier temps nous allons ajouter la dépendance suivante au controller :

```
use EasyCorp\Bundle\EasyAdminBundle\Router\AdminUrlGenerator;
```

13.4.1 Ajout du constructeur :

Nous allons ensuite ajouter un constructeur qui va s'occuper de générer, à l'aide d'**AdminUrlGenerator**, les différentes **routes** vers nos **CRUD**. Nous allons donc intégrer le **constructeur** à l'intérieur de la classe en utilisant le code suivant :

```
public function __construct(private AdminUrlGenerator $adminUrlGenerator){}
```

NB : attention aux 2 **Underscores** devant **construct**.

13.4.2 Modification de la méthode index :

Nous allons également modifier la méthode **index** de la classe **DashBoardController** afin qu'elle exploite **AdminUrlGenerator**, en remplaçant le code par celui-ci :

```
#[Route('/admin', name: 'admin')]  
public function index(): Response  
{  
    $url = $this->adminUrlGenerator  
    ->setController(ArticleCrudController::class)  
    ->generateUrl();  
    return $this->redirect($url);  
}
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

13.4.3 Modification de la méthode `configureMenuItems` :

Nous allons éditer la fonction `configureMenuItems` en les différents liens vers les sections du **Dashboard** avec le code ci-dessous :

```
public function configureMenuItems(): iterable
{
    yield MenuItem::linkToRoute('Retour à l\'accueil', 'fa fa-home', 'app_accueil');
    yield MenuItem::linkToCrud('Articles', 'fas fa-list', Article::class);
    yield MenuItem::linkToCrud('Categories', 'fas fa-newspaper', Category::class);
    yield MenuItem::linkToCrud('Users', 'fas fa-users', User::class);
}
```

Il nous reste à vérifier le bon fonctionnement du **Dashboard** en exécutant la route :

localhost:8000/admin

NB : si vous avez l'erreur suivante :

When using date/time fields in EasyAdmin backends, you must install and enable the PHP Intl extension, which is used to format date/time values.

Il faut éditer le fichier `php.ini` et décommenter la ligne **`extension=intl`**, relancer le serveur Apache .

NB : Si vous avez une erreur **`manifest.json`** :

Pour la corriger nous allons copier tout le contenu de **`/public/bundles/easyAdmin`** à la racine de **public** (sauf le fichier **`manifest.json`**).

Créer un dossier **`build`** à la racine de **public** et y déposer le fichier **`manifest.json`** à l'intérieur.

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

14 Créer un formulaire avec Symfony :

Nous avons vu avec PHP comment créer des formulaires pour interagir avec une **base de données**. Nous utilisons les super globales **GET** et **POST**.

Le fonctionnement et la mise en œuvre avec Symfony va être différente. Nous allons utiliser les **form_type** (avec **make bundle**).

Les formulaires symfony pourront fonctionner en GET, POST, PUT, DELETE. Nos formulaires Symfony seront connectés à une entité et pourront ajouter ou modifier des données.

14.1 Générer un formulaire :

Pour générer un formulaire nous allons utiliser la commande suivante (dans un **formulaire**) :

`symfony console make:form`

Les formulaires par convention vont se nommer **EntitéForm** :

Dans l'exemple nous allons créer un formulaire **mappé** sur l'entité User avec comme nom **UserType** :

```
mathieumithridate@FORM21-06-MINGW64 /c/symfony/blog2 (main)
$ symfony console make:form

The name of the form class (e.g. BravePizzaType):
> UserType

The name of Entity or fully qualified model class name that the new form
is bound to (empty for none):
> User

created: src/Form/UserType.php

Success!
```

Une nouvelle classe **UserType** va être ajoutée au projet dans le dossier **src/form/** :

`<?php`

`namespace App\Form;`

`use App\Entity\User;`

`use Symfony\Component\Form\AbstractType;`

`use Symfony\Component\Form\FormBuilderInterface;`

`use Symfony\Component\OptionsResolver\OptionsResolver;`

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
class UserType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add('name')
            ->add('firstname')
            ->add('mail')
            ->add('password')
            ->add('createdAt')
    }
}

public function configureOptions(OptionsResolver $resolver): void
{
    $resolver->setDefaults([
        'data_class' => User::class,
    ]);
}
```

Dans la méthode **configureOptions** nous allons pouvoir configurer les options de notre formulaire et rattacher la classe (dans notre cas **User::class**).

14.2 Configurer un formulaire ajouter des attributs non mappés :

Nous avons la possibilité d'ajouter des **attributs** supplémentaires à un formulaire qui ne sont pas **mappé** (qui n'existe pas dans l'entité) :

Symfony permet 3 paramètres à la fonction **add** :

Le nom de l'attribut, le type et un tableau associatif :

Nous allons ajouter un nouveau champ comme ci-dessous :

//ajout d'un attribut non mappé

```
->add('age', NumberType::class, [
    'mapped' => false
])
```

Vous retrouverez les différents types de formulaire à l'adresse suivante :

<https://symfony.com/doc/2.8/reference/forms/types.html#supported-field-types>

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordnatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordnatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

14.3 intégrer le formulaire à un controller :

Pour intégrer le formulaire à un controller nous allons utiliser la commande suivante (dans un terminal) :

`symfony console make:controller`

```
$ symfony console make:controller

Choose a name for your controller class (e.g. FiercePizzaController):
> UserForm

created: src/Controller/UserFormController.php
created: templates/user_form/index.html.twig

Success!

Next: Open your new controller class and add some pages!
```

Nous allons ensuite éditer la méthode index pour intégrer le formulaire :

```
#[Route('/user/form', name: 'app_user_form')]
public function index(): Response
{
    $form = $this->createForm(UserType::class);
    return $this->render('user_form/index.html.twig', [
        'form' => $form->createView()
    ]);
}
```

Il faut bien penser à ajouter l'import suivant :

`use App\Form\UserType;`

14.4 Ajouter le formulaire dans la vue :

Pour ajouter le formulaire nous allons ajouter dans l'interface twig le code ci-dessous :

```
{% block body %}

    {{ form(form) }}

{% endblock %}
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Si on souhaite ajouter le formulaire champ par champ nous utiliserons le code ci-dessous :

```
{% block body %}
    {{ form_start(form) }}
    {{ form_label(form.name) }}
    {{ form_widget(form.name) }}
    {{ form_rest(form) }}
    <button type="submit">Enregistrer</button>
    {{ form_end(form) }}
{% endblock %}
```

14.5 Ajouter le champ submit :

Pour ajouter le champs submit à un formulaire nous avons 2 méthodes :

Dans la classe form :

```
->add('save', SubmitType::class)
```

Dans l'interface twig :

```
<button type="submit">Enregistrer</button>
```

14.5 Récupérer les données :

Pour récupérer les données de notre formulaire, nous allons éditer la méthode dans notre controller.

Nous allons ajouter une nouvelle instance de notre classe User dans la méthode **index** et la passer au formulaire :

```
public function index(): Response
{
    $user = new User();

    $form = $this->createForm(UserType::class, $user);

    return $this->render('default/index.html.twig', [
        'form' => $form->createView()
    ]);
}
```

Pour récupérer les données et vérifier que le bouton **submit** nous allons ajouter le code suivant :

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
public function index(Request $request): Response
{
    $user = new User();

    $form = $this->createForm(UserType::class, $user);

    $form->handleRequest($request);
    if ($form->isSubmitted() && $form->isValid()){
        dump($user);die;
    }

    return $this->render('defaultuser_form/index.html.twig', [
        'form' => $form->createView()
    ]);
}
```

14.6 Sauvegarder les données en BDD :

Nous allons maintenant ajouter notre utilisateur en BDD et pour cela nous allons persister les **données** dans notre **entityManager** en utilisant le code ci-dessous :

```
namespace App\Controller;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Routing\Annotation\Route;
use App\Form\UserType;
use App\Entity\User;
use App\Repository\UserRepository;
use Doctrine\ORM\EntityManagerInterface;
public function index(Request $request, EntityManagerInterface
$entityManager): Response
{
    $user = new User();

    $form = $this->createForm(UserType::class, $user);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()){
        $entityManager->persist($user);
    }
}
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```

        $entityManager->flush();
        //refresh la page :
        return $this->redirectToRoute('app_user_form');
    }

    return $this->render('user_form/index.html.twig', [
        'form' => $form->createView()
    ]);
}

```

NB : pensez à ajouter **entityManager** à votre controller.

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

15 Inscription et connexion sécurisé :

Pour mettre en place un système d'inscription et de connexion nous allons utiliser un module intégré au Framework symfony. Nous l'avons déjà utilisé pour générer nos entités et controllers (**symfony console make**).

Nous allons l'utiliser en 2 temps :

-Création de l'entité User (gestion des utilisateurs) (**make:user**)

-Mise en place de l'authentification (**make:auth**)

Ouvrir le dossier projet dans un terminal (**git bash here**).

Créer un nouveau projet avec la commande suivante :

```
symfony new nom_projet --webapp
```

Se déplacer dans le projet avec la commande suivante :

```
cd nom_projet
```

15.1 Création de l'entité User avec la commande suivante :

```
symfony console make:user
```

La commande va nous demander de donner un nom pour l'entité utilisateur, on va choisir **User**

```
$ symfony console make:user

The name of the security user class (e.g. User) [User]:
> User
```

On choisit ensuite si l'on veut stocker les utilisateurs en base de données. (On saisit **Yes** et **entrée**)

```
Do you want to store user data in the database (via Doctrine)? (yes/no) [yes]:
>
```

Nous allons sélectionner avec quel attribut nous allons-nous connecter.

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
Enter a property name that will be the unique "display" name for the user (e.g. email, username, uuid) [email]:
> email
```

On active le hashage du mot de passe

```
Does this app need to hash/check user passwords? (yes/no) [yes]:
>
```

Les fichiers suivants vont être générés :

```
created: src/Entity/User.php
created: src/Repository/UserRepository.php
updated: src/Entity/User.php
updated: config/packages/security.yaml
```

Success!

```
Next Steps:
- Review your new App\Entity\User class.
- Use make:entity to add more fields to your User entity and then run make:migration.
- Create a way to authenticate! See https://symfony.com/doc/current/security.html
```

Entité **User**, le repository **UserRepository**, le fichier de configuration **security.yaml**.

15.2 Mise en place de l'authentification avec la commande suivante :

```
symfony console make:auth
```

La commande nous demande si l'on souhaite générer un formulaire nous allons saisir 1.

```
what style of authentication do you want? [Empty authenticator]:
[0] Empty authenticator
[1] Login form authenticator
> 1
```

Nous allons donner un nom à la classe d'authentification :

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
The class name of the authenticator to create (e.g. AppCustomAuthenticator):  
> AppAuth
```

Nous allons choisir un nom pour notre controller d'authentification :

```
Choose a name for the controller class (e.g. SecurityController) [SecurityController]:  
> Security
```

La commande nous propose de générer la méthode de déconnexion nous allons choisir **Yes** pour l'ajouter.

Les fichiers suivants vont être créés et ou mis à jour :

```
created: src/Security/AppAuthAuthenticator.php  
updated: config/packages/security.yaml  
created: src/Controller/RegisterController.php  
created: templates/security/login.html.twig
```

Success!

Dans le fichier **security/AppAuthenticator.php** nous avons une méthode : **onAuthenticationSuccess** que l'on peut customiser afin de gérer ce qui se passe quand on se connecte. Nous allons modifier le code comme ci-dessous :

```
public function onAuthenticationSuccess(Request $request, TokenInterface $token, string $firewallName): ?Response  
{  
    if ($targetPath = $this->getTargetPath($request->getSession(), $firewallName)) {  
        return new RedirectResponse($targetPath);  
    }  
    //la méthode generate attend en paramètre la route vers une fonction (paramètre name):  
    return new RedirectResponse($this->urlGenerator->generate('app_home'));  
    //throw new \Exception('TODO: provide a valid redirect inside '.__FILE__);  
}
```

return new RedirectResponse(\$this->urlGenerator->generate('app_home'));

En paramètre nous renseignons la valeur name d'une méthode. Si l'authentification est correcte l'utilisateur va être redirigé vers la fonction liée à cette route. (**app_home**)

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordnatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordnatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Exemple :

```
#[Route('/home', name: 'app_home')]
public function index(): Response
{
    return $this->render('home/index.html.twig', [
        'controller_name' => 'HomeController',
    ]);
}
```

15.3 Configuration de l'inscription :

15.3.1 Création du formulaire

Nous allons créer une classe **formulaire** et la lier à l'entité **User** avec la commande suivante :

Symfony console make:form

La commande demande de choisir un nom pour la classe du formulaire, la convention de symfony est **NomClasseType**.

Nous allons choisir comme nom de classe **User** (symfony va rajouter **Type** à la fin du nom de la classe).

```
$ symfony console make:form

The name of the form class (e.g. GrumpyPuppyType):
> User
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

Nous allons lier le formulaire à l'entité **User** comme ci-dessous :

```
The name of Entity or fully qualified model class name that the new form will be bound to (empty for none):
> User

created: src/Form/UserType.php

Success!

Next: Add fields to your form and start using it.
Find the documentation at https://symfony.com/doc/current/forms.html
```

Nous allons maintenant customiser le formulaire (**UserType.php**) comme ci-dessous :

```
//import des modules
use App\Entity\User;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;
use Symfony\Component\Form\Extension\Core\Type\EmailType;
use Symfony\Component\Form\Extension\Core\Type>PasswordType;
use Symfony\Component\Form\Extension\Core\Type\SubmitType;
//fonction qui construit le formulaire
public function buildForm(FormBuilderInterface $builder, array $options): void
{
    $builder
        ->add('email', EmailType::class, [
            'attr' => ['class' => 'formulaire'],
            'label' => 'mail',
            'required' => true,
        ])
        ->add('password', PasswordType::class, [
            'attr' => ['class' => 'formulaire'],
            'label' => 'Mot de passe',
            'required' => true,
        ])
        ->add('Ajouter', SubmitType::class);
}
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

La fonction **buildForm** va donc créer un formulaire avec 2 champs input (**email** et **password**) ainsi qu'un bouton **submit** pour envoyer les données saisies au back-end.

15.4 Création du controller et de la méthode addUser :

De base le module de création de compte ne va pas hasher notre mot de passe. Nous allons devoir utiliser le module **PasswordHasher** pour effectuer cette action.

Création d'un nouveau controller pour gérer la création des comptes avec la commande suivante :

```
symfony console make:controller
```

Nous allons éditer le controller **RegisterController.php** comme ci-dessous :

Ajout des dépendances et création de la fonction addUser :

```
<?php
namespace App\Controller;
//import des modules
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\HttpFoundation\Request;
use App\Form\UserType;
use App\Entity\User;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Component>PasswordHasher\Hasher\UserPasswordHasherInterface;

class RegisterController extends AbstractController
{
    //fonction pour ajouter un compte utilisateur
    #[Route('/register', name: 'app_register')]
    public function addUser(Request $request,
        EntityManagerInterface $manager, UserPasswordHasherInterface $hasher): Response
    {
        //instance d'un objet User
        $user = new User();
        //variable qui contient un objet UserType (formulaire)
        $form = $this->createForm(UserType::class, $user);
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
//stocker le résultat du formulaire
$form->handleRequest($request);
//condition validation du formulaire
if($form->isSubmitted() && $form->isValid()){
    //récupération du mot de passe en clair
    $pass = $_POST['user']['password'];
    //hasher le mot de passe
    $hashPassword = $hash->hashPassword($user, $pass);
    //setter les valeurs (mot de passe activation et le rôle)
    $user->setPassword($hashPassword);
    $user->setRoles(['ROLE_USER']);
    //faire persister les données
    $manager->persist($user);
    //ajout en BDD
    $manager->flush();
    //génération du formulaire et redirection
    return $this->render('register/index.html.twig', [
        'form' => $form->createView(),
        'add' => $user->getName(),
    ]);
}
//génération du formulaire
return $this->render('register/index.html.twig', [
    'form' => $form->createView(),
    'add' => "",
]);
}
```

La méthode ci-dessus va ajouter un nouveau compte utilisateur en BDD, en utilisant **L'ORM** de **Symfony**. Le module **PasswordHasher** va lui hasher le mot de passe avec l'algorithme par défaut (**Bcrypt**). Si l'on veut utiliser un autre algorithme il faut éditer le fichier **config/package/security.yaml** et remplacer la valeur **auto** par le nom de l'algorithme de votre choix. (Voir la documentation de symfony)

<https://symfony.com/doc/current/security.html>

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

```
security:
  password_hashers:
    # By default, password hashers are resource intensive and take time. This is
    # important to generate secure password hashes. In tests however, secure hashes
    # are not important, waste resources and increase test times. The following
    # reduces the work factor to the lowest possible values.
    Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface:
      algorithm: auto
      cost: 4 # Lowest possible value for bcrypt
      time_cost: 3 # Lowest possible value for argon
      memory_cost: 10 # Lowest possible value for argon
```

15.5 Edition de la vue :

Nous allons éditer le template (register/index.html.twig) comme ci-dessous :

```
{% extends 'base.html.twig' %}
{% block title %}Création de compte{% endblock %}
{% block body %}
  <h3>formulaire d'inscription :</h3>
  {% création du formulaire %}
  {{ form_start(form) }}
    {{ form_rest(form) }}
  {{ form_end(form) }}
  {% test si le compte à été ajouté %}
  {% if add != "" %}
    <div class="alert alert-dismissible alert-success">
      <button type="button" class="btn-close" data-bs-
dismiss="alert"></button>
      Le compte {{ add }} a été ajouté.
    </div>
  {% endif %}
{% endblock %}
```

Pour générer le formulaire on utilise les méthodes **form_start** et **form_end**.

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

15.6 Connexion :

Pour la connexion, le module make:auth à mis en place tous ce que l'on a besoin.

Nous pouvons customiser le fichier template : **security/login.html.twig** comme dans l'exemple ci-dessous :

```
{% extends 'base.html.twig' %}
{% block title %}Connexion{% endblock %}
{% block body %}
<form method="post">
    {% if error %}
        <div class="alert alert-danger">{{
error.messageKey|trans(error.messageData, 'security') }}</div>
    {% endif %}
    {% if app.user %}
        <div class="mb-3">
            Utilisateur connecté : {{ app.user.userIdentifier }}, <a
href="{{ path('app_logout') }}">Déconnexion</a>
        </div>
    {% endif %}
    <h1 class="h3 mb-3 font-weight-normal">Se connecter</h1>
    <label for="inputEmail">Email</label>
    <input type="email" value="{{ last_username }}" name="email"
id="inputEmail" class="form-control" autocomplete="email" required
autofocus>
    <label for="inputPassword">Password</label>
    <input type="password" name="password" id="inputPassword" class="form-
control" autocomplete="current-password" required>
    <input type="hidden" name="_csrf_token"
        value="{{ csrf_token('authenticate') }}"
    >
    <button class="btn btn-lg btn-primary" type="submit">
        Connexion
    </button>
</form>
{% endblock %}
```

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
Mathieu MITHRIDATE	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		

SYMFONY

15.7 Redirection à la connexion :

Pour rediriger l'utilisateur en cas de réussite de connexion nous allons éditer la fonction `onAuthenticationSuccess` contenu dans le fichier ***Security/AppAuthAuthenticator.php*** comme ci-dessous :

```
/* Redirection quand la connexion a réussi vers la fonction index de
HomeController (valeur //de name): */
return new RedirectResponse($this->urlGenerator->generate('app_home'));
```

15.8 Protection des routes :

Nous pouvons configurer le fichier ***config/package/security.yaml*** et inclure des règles de sécurité. Elles vont nous permettre de rendre obligatoire la connexion et un rôle pour y accéder. Si on essaye d'atteindre la route sans être connecté ou avec le mauvais rôle Symfony va nous rediriger automatiquement vers le formulaire de connexion (***route /login***).

Exemple de configuration des routes :

```
access_control:
  - { path: ^/admin, roles: ROLE_ADMIN }
  - { path: ^/profile, roles: ROLE_USER }
```

Le code ci-dessus oblige à être connecté en tant que **rôle Admin** pour accéder à la route ***/admin*** et en tant que **rôle User** pour la route ***/profile***.

Pour sécuriser une ou plusieurs routes nous avons juste besoin de l'ajouter à la suite dans le fichier.

16 Création d'un service :

Quand on développe un

Auteur(s)	Relu, validé et visé par :	Date de création :	Date dernière MAJ :
<u>Mathieu MITHRIDATE</u>	Jérôme CHRETIENNE : Resp. Secteur Numérique Occitanie	25/09/2022	
	Florence CALMETTES : Coordinatrice Filière Syst. & Réseaux	Toute reproduction, représentation, diffusion ou rediffusion, totale ou partielle, de ce document ou de son contenu par quelque procédé que ce soit est interdite sans l'autorisation expresse, écrite et préalable de l'ADRAR.	
	Sophie POULAKOS : Coordinatrice Filière WEBDESIGN / DEVELOPPEMENT		