

Documentation Technique

1. Objectif du Projet : Créer une application web en Symfony qui permet

- Récupérer des données de football via une API externe (ligues, saisons, équipes, classements)
- Stocker les données en base (MySQL)
- Pourvoir récupérer et afficher les données via la base de données

2. Environnement de Développement

- IDE : Visual Studio Code
- Langages : PHP 8.2, HTML/TWIG, SQL
- Base de données : MySQL via phpMyAdmin
- Framework : Symfony

3. Réalisation du Projet

3.1 Création et configuration de mon projet Symfony :

Création du projet avec la commande : `Symfony new mon_projet --webapp`

```
c:\wamp64\www>symfony new mon_projet --webapp
* Creating a new Symfony project with Composer
* Setting up the project under Git version control
  (running git init c:\wamp64\www\mon_projet)

[OK] Your project is now ready in c:\wamp64\www\mon_projet

c:\wamp64\www>cd mon_projet
```

Modification du fichier .env pour configurer la connexion à la base de données :

`DATABASE_URL="mysql://padawan:america@127.0.0.1:3306/foot?serverVersion=9.1.0-MariaDB&charset=utf8mb4"`

Création de la base de données foot : `Php bin/console doctrine :database`

3.2 Création des Entités

Les entités que j'ai créées : saisons, ligues, classements, équipes. Elles sont créées avec la commande suivante : `php bin/console make :entity`

Exemple de création d'une entité 'equipes' avec un champ nom :

Class name of the entity to create or update

>equipes

New property name :

>string

Field type :

>string

Field length [255] :

>255

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

>no

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

> (on appuie entrer si on veut créer une autre propriété)

3.3 Génération des Migrations :

php bin/console make:migration

php bin/console doctrine:migrations:migrate

Il faut savoir qu'après les commandes, les getters/setters correspondants aux entités sont créés automatiquement.

Voici les entités (tables) que j'ai créé :

Saisons :

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/> 1	id	int			Non	Aucun(e)		AUTO_INCREMENT	Modifier Supprimer Plus
<input type="checkbox"/> 2	ligue_id	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/> 3	annee	int			Non	Aucun(e)			Modifier Supprimer Plus

Ligues :

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/> 1	id	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/> 2	championnat	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/> 3	slug	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/> 4	abbr	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/> 5	url	longtext	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus

Equipes

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/> 1	id	int			Non	Aucun(e)		AUTO_INCREMENT	Modifier Supprimer Plus
<input type="checkbox"/> 2	id_equipe	int			Oui	NULL			Modifier Supprimer Plus
<input type="checkbox"/> 3	nom	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/> 4	location	varchar(255)	utf8mb4_unicode_ci		Non	Aucun(e)			Modifier Supprimer Plus
<input type="checkbox"/> 5	championnat	varchar(255)	utf8mb4_unicode_ci		Oui	NULL			Modifier Supprimer Plus
<input type="checkbox"/> 6	logo	varchar(255)	utf8mb4_unicode_ci		Oui	NULL			Modifier Supprimer Plus

Classement :

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
<input type="checkbox"/> 1	id	int			Non	Aucun(e)		AUTO_INCREMENT	Modifier Supprimer Plus
<input type="checkbox"/> 2	id_ligue	varchar(255)	utf8mb4_unicode_ci		Oui	NULL			Modifier Supprimer Plus
<input type="checkbox"/> 3	nom_equipe	varchar(255)	utf8mb4_unicode_ci		Oui	NULL			Modifier Supprimer Plus
<input type="checkbox"/> 4	rang	int			Oui	NULL			Modifier Supprimer Plus
<input type="checkbox"/> 5	games_played	int			Oui	NULL			Modifier Supprimer Plus
<input type="checkbox"/> 6	win	int			Oui	NULL			Modifier Supprimer Plus
<input type="checkbox"/> 7	losses	int			Oui	NULL			Modifier Supprimer Plus
<input type="checkbox"/> 8	saisons_id	int			Oui	NULL			Modifier Supprimer Plus

3.4 Création des Contrôleurs

Avant de récupérer les données ou les afficher, j'ai d'abord créé les contrôleurs nécessaires à mon application via :

`php bin/console make:controller`

3.5 Intégration des données de l'API dans la BDD Relations entre les Tables

Après la création des entités, j'ai récupéré les données via l'api externe et enregistrée dans la base.





























































Exemple :

```
// Ligues
#[Route('/foot', name: 'foot')]
public function foot(EntityManagerInterface $entityManager): Response
{
    $json = file_get_contents('https://football-standings-api.vercel.app/leagues');
    $obj = json_decode($json);
    $tables = $obj->data;
    foreach ($tables as $table){
        $league = new Ligues();
        $league->setId($table->id);
        $league->setChampionnat($table->name);
        $league->setSlug($table->slug);
        $league->setAbbr($table->abbr);
        $league->setUrl($table->logos->light);
        $entityManager->persist($league);
    }
    $entityManager->flush();
    return new Response('Saved new product with id '.$league->getId());
}
```

Ici voilà ce qu'il se passe :

- L'utilisateur accède à l'URL /foot

- Symfony contacte l'API et récupère toutes les ligues
- Chaque ligue est transformée en objet (Ligues)
- Toutes les ligues sont persistées dans la base foot
- Une réponse s'affiche : "LIGUES enregistrées avec succès !"

				id	ligue_id	annee
<input type="checkbox"/>		Éditer		Copier		Supprimer
				1	arg.1	2025
<input type="checkbox"/>		Éditer		Copier		Supprimer
				2	arg.1	2024
<input type="checkbox"/>		Éditer		Copier		Supprimer
				3	arg.1	2023
<input type="checkbox"/>		Éditer		Copier		Supprimer
				4	arg.1	2022
<input type="checkbox"/>		Éditer		Copier		Supprimer
				5	arg.1	2021
<input type="checkbox"/>		Éditer		Copier		Supprimer
				6	arg.1	2020
<input type="checkbox"/>		Éditer		Copier		Supprimer
				7	arg.1	2019
<input type="checkbox"/>		Éditer		Copier		Supprimer
				8	arg.1	2018
<input type="checkbox"/>		Éditer		Copier		Supprimer
				9	arg.1	2017
<input type="checkbox"/>		Éditer		Copier		Supprimer
				10	arg.1	2016
<input type="checkbox"/>		Éditer		Copier		Supprimer
				11	arg.1	2015
<input type="checkbox"/>		Éditer		Copier		Supprimer
				12	arg.1	2014
<input type="checkbox"/>		Éditer		Copier		Supprimer
				13	arg.1	2013
<input type="checkbox"/>		Éditer		Copier		Supprimer
				14	arg.1	2012
<input type="checkbox"/>		Éditer		Copier		Supprimer
				15	arg.1	2011
<input type="checkbox"/>		Éditer		Copier		Supprimer
				16	arg.1	2010
<input type="checkbox"/>		Éditer		Copier		Supprimer
				17	arg.1	2009
<input type="checkbox"/>		Éditer		Copier		Supprimer
				18	arg.1	2008
<input type="checkbox"/>		Éditer		Copier		Supprimer
				19	arg.1	2007
<input type="checkbox"/>		Éditer		Copier		Supprimer
				20	arg.1	2006

Autre exemple :

```

#[Route('/saisons', name: 'saisons')]
public function saisons(EntityManagerInterface $entityManager): Response
{
    $json = file_get_contents('https://football-standings-api.vercel.app/leagues');
    $obj = json_decode($json);
    $ligues = $obj->data;

    foreach ($ligues as $ligue) {
        $ligueId = $ligue->id;
        $json = file_get_contents("https://football-standings-api.vercel.app/leagues/{$ligueId}/seasons");
        $obj = json_decode($json);
        $saisons = $obj->data->seasons;

        foreach ($saisons as $annee) {
            $saison = new Saisons();
            $saison->setLigueId($ligue->id);
            $saison->setAnnee($annee->year);
            $entityManager->persist($saison);
        }
    }
    $entityManager->flush();
    return new Response('Toutes les saisons récupérées et enregistrées.');
```

3.6 Relations entre les Tables de la BDD

Après avoir récupérer toutes les données dont j'avais besoins, j'ai fait des liens entre les différentes tables.

```
#[Route('/league', name: 'league')]
public function league(EntityManagerInterface $entityManager): Response
{
    set_time_limit(0);
    $classementRepository = $entityManager->getRepository(Classement::class);
    $equipeRepository = $entityManager->getRepository(Equipes::class);
    $ligueRepository = $entityManager->getRepository(Ligues::class);
    $allClassement = $classementRepository->findAll();

    foreach ($allClassement as $classement) {
        $nomEquipe = $classement->getNomEquipe();
        $idLigue = $classement->getIdLigue();
        $equipe = $equipeRepository->findBy(['nom' => $nomEquipe]);
        $ligue = $ligueRepository->find($idLigue);
        $equipe->addLigue($ligue);
        $entityManager->persist($equipe);
    }
    $entityManager->flush();
    return new Response('Relations ligues/équipes.');
```

Exemple : une ligue a plusieurs saisons, une saison contient plusieurs équipes.

J'ai fait que des relations ManyToMany.

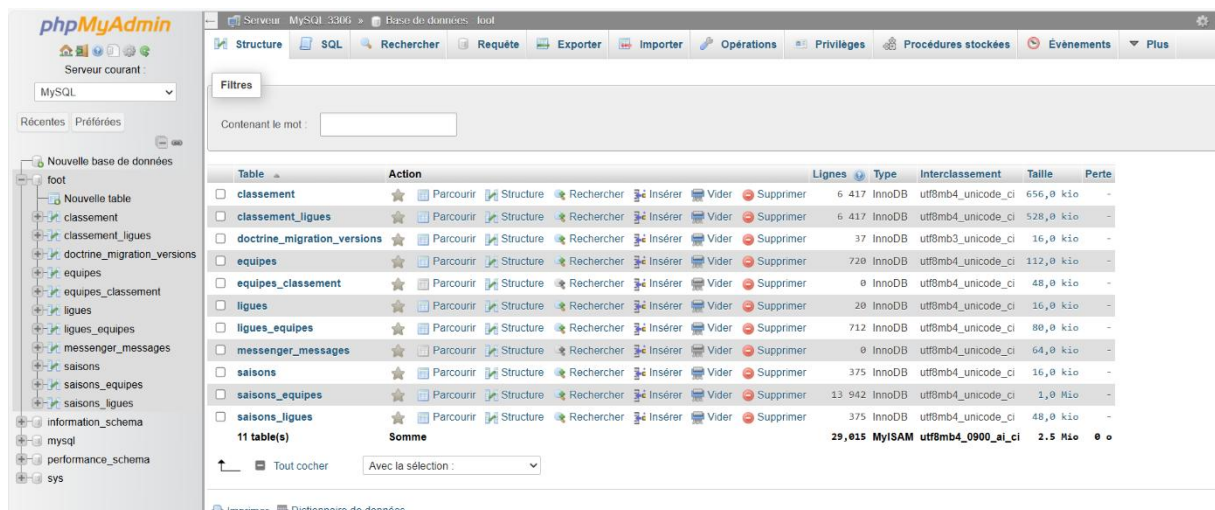


Table	Action	Lignes	Type	Interclassement	Taille	Perte
classement	Parcourir Structure Rechercher Insérer Vider Supprimer	6 417	InnoDB	utf8mb4_unicode_ci	656,0 kio	-
classement_ligues	Parcourir Structure Rechercher Insérer Vider Supprimer	6 417	InnoDB	utf8mb4_unicode_ci	528,0 kio	-
doctrine_migration_versions	Parcourir Structure Rechercher Insérer Vider Supprimer	37	InnoDB	utf8mb3_unicode_ci	16,0 kio	-
equipes	Parcourir Structure Rechercher Insérer Vider Supprimer	720	InnoDB	utf8mb4_unicode_ci	112,0 kio	-
equipes_classement	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	48,0 kio	-
ligues	Parcourir Structure Rechercher Insérer Vider Supprimer	20	InnoDB	utf8mb4_unicode_ci	16,0 kio	-
ligues_equipes	Parcourir Structure Rechercher Insérer Vider Supprimer	712	InnoDB	utf8mb4_unicode_ci	80,0 kio	-
messenger_messages	Parcourir Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_unicode_ci	64,0 kio	-
saisons	Parcourir Structure Rechercher Insérer Vider Supprimer	375	InnoDB	utf8mb4_unicode_ci	16,0 kio	-
saisons_equipes	Parcourir Structure Rechercher Insérer Vider Supprimer	13 942	InnoDB	utf8mb4_unicode_ci	1,0 Mio	-
saisons_ligues	Parcourir Structure Rechercher Insérer Vider Supprimer	375	InnoDB	utf8mb4_unicode_ci	48,0 kio	-
11 table(s)	Somme	29,015	MyISAM	utf8mb4_0900_ai_ci	2.5 Mio	0 o

3.7 Récupération des données via la base de données MySQL et les classes

Après avoir insérer les données, on peut donc les récupérer via des requêtes :

Par exemple :

```

public function findLiguesByAnnee($annee)
{
    return $this->createQueryBuilder('s')
        ->innerJoin('select ligue_id')
        ->andWhere('s.annee = :annee')
        ->setParameter('annee', $annee)
        ->getQuery()
        ->getResult();
}

```

Ici c'est une requête qui nous permet de trouver une ligue selon l'année choisi

Les requêtes peuvent être fait sur les Controller mais il est préférable de les faire dans les repository pour ne pas mélanger la logique métier.

Ensuite on récupère la fonction qu'on veut dans le Controller. Exemple :

```

#[Route('/', name: 'accueil')]
public function accueil(Request $request, EntityManagerInterface $entityManager): Response
{
    $equipeRepository = $entityManager->getRepository(Equipes::class);
    $saisonRepository = $entityManager->getRepository(Saisons::class);
    $ligueRepository = $entityManager->getRepository(Ligues::class);
    $classementRepository = $entityManager->getRepository(Classement::class);
    $nbEquipes = $equipeRepository->compterEquipes();
    $nbSaisons = $saisonRepository->compterSaisons();
    $nbLigues = $ligueRepository->compterLigues();
    $ligues = $ligueRepository->findAll();
    $top = $request->get('top');
    $equipes = [];

    if ($top === 'plusvictoires') {
        $equipes = $classementRepository->topVictoires();
    } elseif ($top === 'moinsvictoires') {
        $equipes = $classementRepository->MoinsVictoires();
    } elseif ($top === 'plusdefaites') {
        $equipes = $classementRepository->TopDefaites();
    } elseif ($top === 'moinsdefaites') {
        $equipes = $classementRepository->MoinsDefaites();
    } elseif ($top === 'plusligues') {
        $equipes = $classementRepository->mEquipes();
    }

    return $this->render('projet/accueil.html.twig', [
        'nbEquipes' => $nbEquipes,
        'nbsaisons' => $nbSaisons,
        'nbligues' => $nbLigues,
        'ligues' => $ligues,
        'equipes' => $equipes,
        'top' => $top
    ]);
}

```

3.8 Affichage des données

Après avoir récupérer les fonctions qu'on veut, on peut les afficher dans une page web à l'aide de Twig.

Exemple :

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

{% block title %}Projet Foot{% endblock %}

{% block body %}
<section style="font-family: sans-serif;">
  <p>
    <a href="{{ path('accueil') }}" style="text-decoration: none; background: steelblue; color: white; padding: 5px 10px; border-radius: 8px; display: inline-block;">🔥 Accueil</a>
  </p>
  <h1>🏆 RECHERCHER DES INFOS SUR VOS LIGUES PRÉFÉRÉES ⚽</h1>
  <p>Notre application a pour but de faciliter vos recherches sur le foot durant ces 25 dernières années.</p>

  <hr style="margin-top: 30px; border: 1px solid #ccc;">

  <!-- recherche par saisons -->
  <form method="get" action="projet">
    <h2>📅 Recherche par Saison :</h2>
    <label for="saison">Saison :</label>
    <input list="saisons" name="saison" placeholder="Entrez une saison" value="{{ annee }}" required>
    <datalist id="saisons">
      {% for saison in saisons %}
        <option value="{{ saison.annee }}"></option>
      {% endfor %}
    </datalist>
    <button type="submit">Rechercher</button>
  </form>

  {% if annee %}
  <form method="get" action="projet">
    <h3><u>Retrouvez Les Différentes Top Classement selon la saison {{ annee }}:</u></h3>
    <label for="topsaisons">Sélectionnez :</label>
```

Résultat :

Accueil

🔍 RECHERCHER DES INFOS SUR VOS LIGUES PRÉFÉRÉES ⚽

Notre application a pour but de faciliter vos recherches sur le foot durant ces 25 dernières années.

📅 Recherche par Saison :

Saison :

Retrouvez Les Différentes Top Classement selon la saison 2025:

Sélectionnez :

🏆 Top du classement rechercher plusdefaites au total :

Nom équipe
Yokohama FC
Yokohama F. Marinos
Changchun Yatai

📅 Recherche par Ligue :

Ligue :