

Évaluation développeur Symfony 5

Étape 1 : développement du projet

Contexte :

Un club de football souhaite digitaliser la gestion de ses joueurs et de leurs statistiques. Le dirigeant du club souhaite avoir une interface lui permettant de visualiser l'ensemble des équipes de son club ainsi que les joueurs associés.

Le nom choisit pour ce projet est Club Foot car le site à pour thématique le football et il est destiné pour un club de foot.

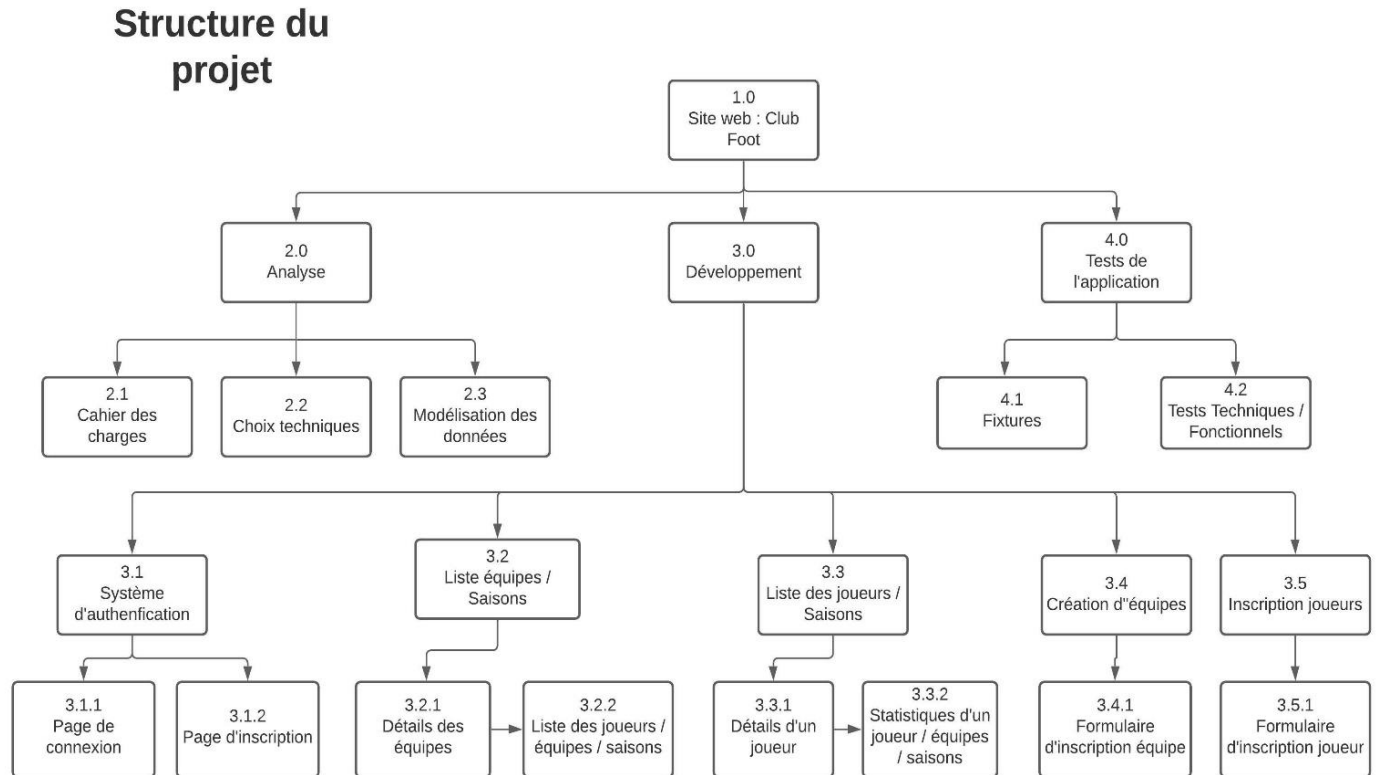


Sommaire

Contexte	1
I - Présentation détaillée du projet.....	3
1. La structure du projet	3
2. Choix techniques	3
II – Compte rendu	4
1. Analyse et conception	4
1.1 La liste des fonctions	4
1.2 Modèle conceptuel de données	5
2. Développement de la plateforme	6
2.1 Structure du site	6
2.2 Création des tables de la base de données.....	7
2.3 Mise en place des contrôleurs et des templates.....	8
2.3.1 Authentification	10
2.3.2 Liste des équipes	13
2.3.3 Liste des joueurs	15
2.3.4 Création des équipes.....	17
2.3.5 Inscription des joueurs.....	18
3. Test de l'application.....	19
3.1 Les fixtures.....	19
3.2 Tests techniques et fonctionnels.....	21

I - Présentation détaillée de la structure du projet et des choix techniques effectués

1. La structure du projet



2. Choix techniques :

- Editeur de code utilisé : Visual Studio Code
- Environnement de serveurs : WampServer (PHP, phpMyAdmin, MySQL)
- Gestionnaire de dépendances PHP : Composer
- Versioning : Git
- Symfony CLI
- Framework : Symfony, Bootstrap

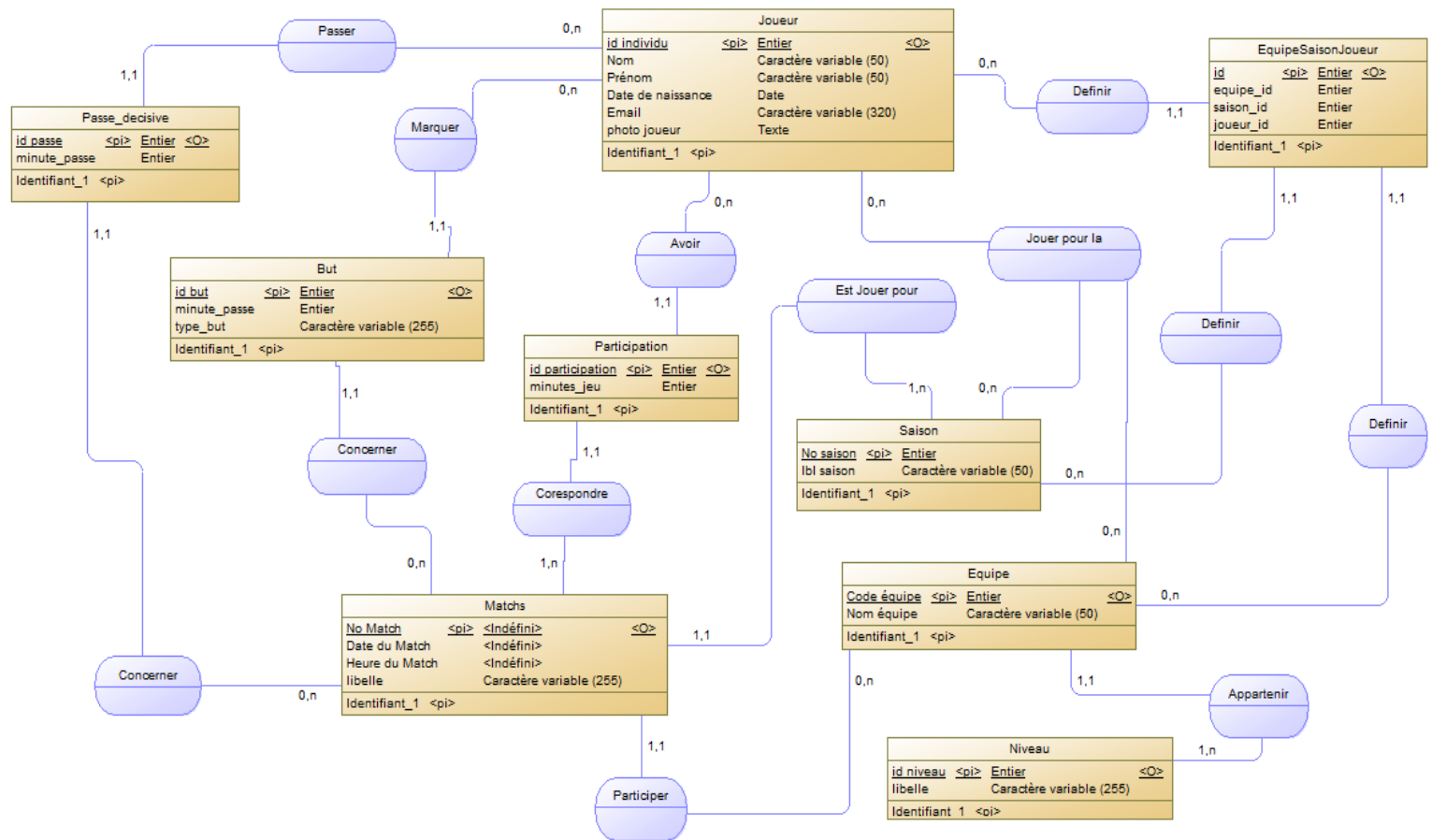
II – Compte rendu

1. Analyse et conception

1.1 La liste des fonctions

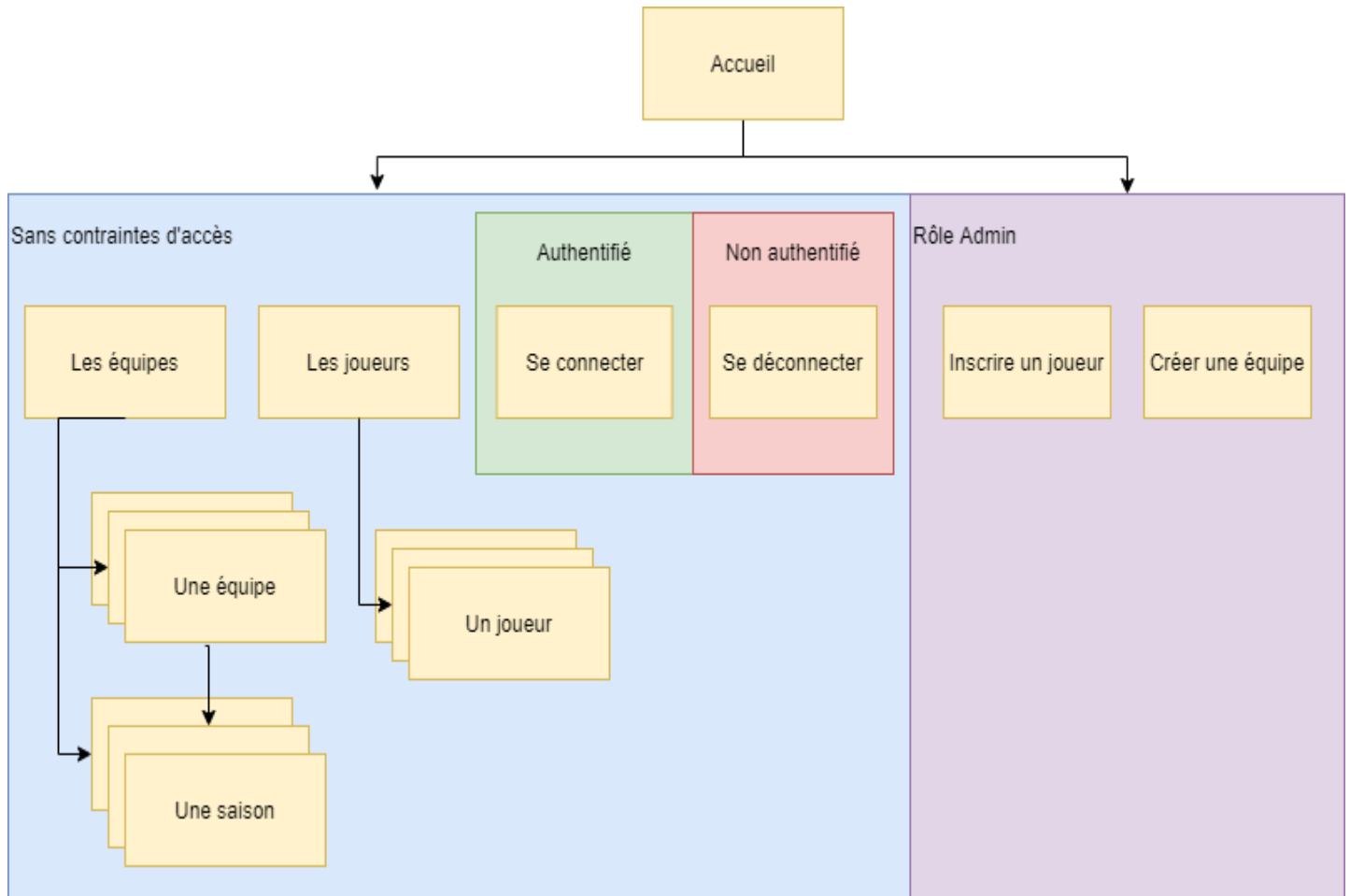
- Page de connexion permettant à l'administrateur de se connecter
- Liste des équipes
- Détails d'une équipe : la liste de ses joueurs pour une saison donnée
 - Possibilité de changer la saison courante
- Détails d'un joueur
 - Statistiques du joueur pour son équipe pour une saison donnée
 - Nombre de matchs joués
 - Nombre de minutes jouées
 - Nombre de passes décisives
 - Nombre de buts marqués
- Possibilité de créer une équipe
 - Nom de l'équipe
 - Niveau de l'équipe
- Possibilité de créer un joueur et de l'affecter à son équipe
 - Email
 - Nom
 - Prénom
 - Date de naissance

1.2 Modèle conceptuel de données



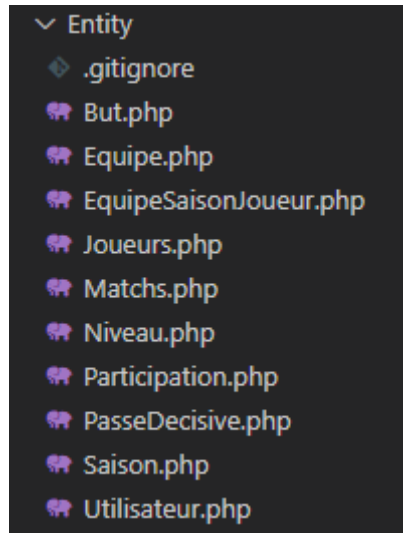
2. Développement de la plateforme

2.1 Structure du site



2.2 Création des tables de la base de données

La commande utilisée pour créer une table dans la base de données depuis la console de VScode : **php bin/console make:entity**



Les entités qui ont été créées sont situées dans le dossier 'Entity'. Les informations de chacune des tables sont récupérées et gérées grâce à l'ORM Doctrine.

Lorsque toutes les tables sont créées, les différentes relations entre ces tables sont mises en place grâce à la même commande 'php bin/console make :entity'.

Dans un premier temps, on indique le nom de l'entité où l'on veut ajouter une relation. Ensuite, on indique le nom d'une autre entité associée à la relation en tant que propriété, de type relation.

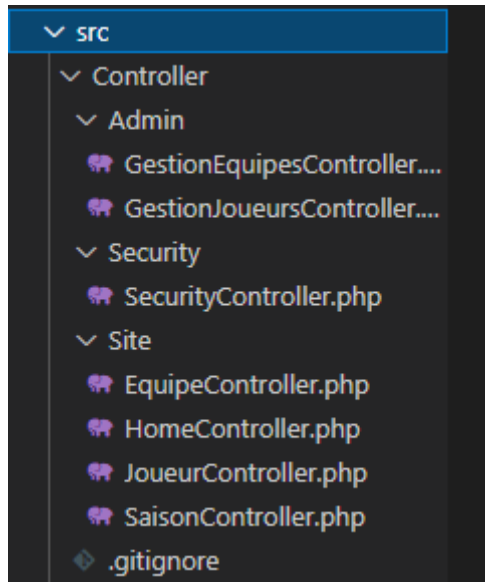
Les différentes relations qu'on peut indiquer lors de leurs créations sont :

- One-To-One : 1 entité est liée à 1 entité ;
- Many-To-One : plusieurs entités liées à 1 entité (liée à une OneTo-Many) ;
- One-To-Many : une entité liée à plusieurs ;
- Many-To-Many : plusieurs entités liées à plusieurs.

Voir le modèle conceptuel de données (1.2) pour plus d'informations sur les relations entre les tables.

2.3 Mise en place des contrôleurs et des templates

La commande utilisée pour créer un contrôleur depuis la console de VScode : **php bin/console make:controller**



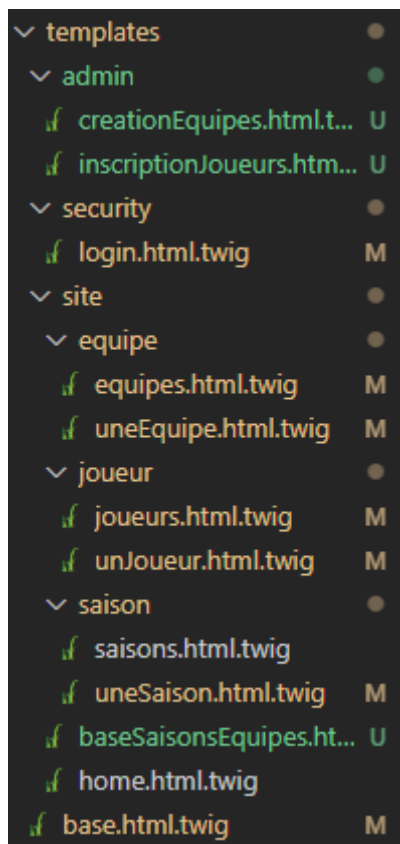
Les contrôleurs sont situés dans le dossier 'Controller' du dossier 'src'.

Les contrôleurs placés dans le dossier 'Admin' gèrent des pages avec un accès restreint à un utilisateur authentifié qui a le rôle Admin.

Le dossier 'Security' comporte le contrôleur 'SecurityController' qui permet de gérer la page d'authentification.

Le dossier 'Site' comporte des contrôleurs qui gèrent des pages sans aucune restriction d'accès.

Chaque contrôleur est associé à un fichier de template qui lui correspond. C'est un fichier twig qui permet de mettre en place la vue de la page sur le site.



Les fichiers templates se trouvent dans le dossier templates et la structure de ces fichiers est arrangée de la même manière que pour la structure des fichiers contrôleurs. Cela permet de mieux se repérer pour la suite de fichiers twig.

Chaque fichier twig correspondant à une page spécifique sur le site à l'exception des fichiers 'base.html.twig' et 'baseSaisonsEquipes.html'. L'accès à la page se fait grâce à une route définie sur le contrôleur associé.

Le fichier d'accueil est [home.html.twig](#) et l'adresse de route pour y accéder est '<http://clubfoot/>'.

- Le fichier base.html.twig contient le menu principal de navigation du site qui sera utilisé pour toutes les pages.

Club Foot Les équipes Les joueurs Se connecter

- Ce fichier contient également la présentation de style pour le site entier. On utilise le Framework Bootstrap pour la mise en page avec une bibliothèque CSS et une bibliothèque Javascript qu'on récupère grâce à lien, les codes sont déjà définis avec un style particulier.

```

templates > / base.html.twig
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta
5              charset="UTF-8">
6          {# Run `composer require symfony/webpack-encore-bundle`
7              {% block stylesheets %}<link rel="stylesheet" href="https://bootswatch.com/5/superhero/bootstrap.css">
8              {% endblock %}
9              {% block javascripts %}
10                 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/js/bootstrap.bundle.min.js"
11                     integrity="sha384-b/QdsTh/d6pkI1MST/rWKFJjaCP5gBSY4sEBT38Q/9RBh9AH40zEOg7H1q2THRZ" crossorigin="anonymous"></script>
12                 {% endblock %}
13      </head>
14      <body>

```

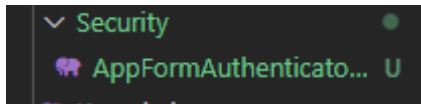
- Le fichier baseSaisonsEquipes contient un menu de saison qui sera affiché sur la page de la liste des équipes mais également sur la page d'une équipe particulière. Ce menu permet d'accéder à une page spécifique avec la liste complète des équipes selon un saison particulière.

Les équipes par saisons : Saison n°1 Saison n°2 Saison n°3 2021-2022

2.3.1 Authentification

Pour l'authentification du site, un formulaire de login est généré grâce à la commande : **php bin/console make:auth**

Dans un premier temps on choisit le type d'authentificateur qui est 'Login form authenticator'. Le nom de la classe saisi qui va servir d'authentificateur est AppFormAuthenticator. Le nom du contrôleur est SecurityController et une route de déconnexion est ajouté sur le contrôleur lors de la création du formulaire.



Ainsi, 3 fichiers sont générés pour gérer le formulaire d'authentification :

- AppFormAuthenticator.php dans src/Security/. Ce fichier permet de récupérer l'utilisateur dans la base de données, par rapport à l'email qu'il a renseigné et le mot de passe est vérifié par rapport à celui qui est enregistré. (Voir le code source du site)
- SecurityController.php dans src/Controller/Security/. C'est le contrôleur qui va gérer la connexion 'login()' et la déconnexion 'logout()' défini sur le fichier.

```
src > Controller > Security > SecurityController.php > ...
namespace App\Controller\Security;

use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Security\Http\Authentication\AuthenticationUtils;

class SecurityController extends AbstractController
{
    /**
     * Le Controller pour gérer la page d'authentification
     *
     * @Route("/login", name="app_login")
     */
    public function login(AuthenticationUtils $authenticationUtils): Response
    {
        // if ($this->getUser()) {
        //     return $this->redirectToRoute('target_path');
        // }

        // get the login error if there is one
        $error = $authenticationUtils->getLastAuthenticationError();
        // last username entered by the user
        $lastUsername = $authenticationUtils->getLastUsername();

        return $this->render('security/login.html.twig', ['last_username' => $lastUsername, 'error' => $error]);
    }

    /**
     * @Route("/logout", name="app_logout")
     * @codeCoverageIgnore
     */
    public function logout()
    {
        throw new \LogicException('This method can be blank - it will be intercepted by the logout key on your firewall.');
```

- Login.html.twig dans templates/security/ c'est la vue qui permet d'afficher le formulaire de connexion.

```

templates > security > if login.html.twig
1  {% extends 'base.html.twig' %}
2
3  {% block title %}Connectez vous !
4  {% endblock %}
5
6  {% block body %}
7      <form method="post">
8          {% if error %}
9              <div class="alert alert-danger">{{ error.messageKey|trans(error.messageData, 'security') }}</div>
10             {% endif %}
11
12             {% if app.user %}
13                 <div class="mb-3">
14                     Vous êtes déjà connecté avec l'adresse mail :
15                     {{ app.user.email }},
16                 </div>
17             {% endif %}
18
19             <h1 class="h3 mb-3 font-weight-normal">Connectez vous</h1>
20
21             <label for="inputEmail">Email</label>
22             <div class="input-field col s12 m6 l6 offset-m3 offset-l3">
23                 <input type="email" value="{{ last_username }}" name="email" id="inputEmail" class="form-control" autocomplete="email" required autofocus>
24             </div>
25             <label for="inputPassword">Mot de passe</label>
26             <div class="input-field col s12 m6 l6 offset-m3 offset-l3">
27                 <input type="password" name="password" id="inputPassword" class="form-control" autocomplete="current-password" required>
28             </div>
29
30             <input type="hidden" name="_csrf_token" value="{{ csrf_token('authenticate') }}">
31
32             <button class="btn btn-lg btn-primary" type="submit">
33                 Se connecter
34             </button>
35         </form>
36     {% endblock %}
37
38

```

La vue sur le site. l'adresse de route pour y accéder est 'http://clubfoot/login'.

Connectez vous

Email

Mot de passe

Se connecter

Le fichier security.yaml qui se trouve dans config/packages/ est également modifié avec l'ajout de l'authenticator et l'entrée pour le logout.

```
providers:
  users_in_memory: { memory: null }
  users_in_database:
    entity:
      class: App\Entity\Utilisateur
      property: email
firewalls:
  dev:
    pattern: ^/(_(profiler|wdt)|css|images|js)/
    security: false
  main:
    lazy: true
    provider: users_in_database

    custom_authenticator: App\Security\AppFormAuthenticator

    logout:
      path: app_logout
      # activate different ways to authenticate
      # https://symfony.com/doc/current/security.html#firewalls-authentication
      # https://symfony.com/doc/current/security/impersonating\_user.html
      # switch_user: true
```

2.3.2 Liste des équipes

Le contrôleur qui gère la page de la liste des équipes et des informations d'une équipe particulière est le suivant : EquipeController, il se trouve dans Controller/Site/.

Ce contrôleur comporte deux fonctions :

- équipes

```
/**
 * Le controller pour gérer la page qui affiche la liste complète des équipes
 */
* @param \App\Repository\SaisonRepository $repoSaison
* @param \App\Repository\EquipeRepository $repo
* @return \Symfony\Component\HttpFoundation\Response
* @route("/les-equipes", name="les_equipes")
*/
public function equipes(SaisonRepository $repoSaison, EquipeRepository $repo)
{
    /**
     * @var \App\Entity\Saison[] $saisons
     */
    $saisons = $repoSaison->findAll();

    /**
     * @var \App\Entity\Equipe[] $equipes
     */
    $equipes = $repo->findAll();

    return $this->render('site/equipe/equipes.html.twig', [
        'controller_name' => 'EquipeController',
        'saisons' => $saisons,
        'equipes' => $equipes
    ]);
}
```

La vue sur le site grâce au fichier twig 'equipes.html.twig'. (Voir code source).
L'adresse de route pour y accéder est 'http://clubfoot/les-equipes'.

Les équipes par saisons : Saison n°1 Saison n°2 Saison n°3 2021-2022

Liste des équipes

Equipe n°1

Lesotho

Niveau de l'équipe : Niveau n°1

voir les détails

Equipe n°2

Hongrie

Niveau de l'équipe : Niveau n°1

voir les détails

Equipe n°3

Guadeloupe

Niveau de l'équipe : Niveau n°2

voir les détails

Equipe n°4

Arménie

Niveau de l'équipe : Niveau n°1

voir les détails

- uneEquipe

```
/**
 * Le controller pour gérer la page d'une équipe particulière avec la liste de ses joueurs par saison
 *
 * @param \App\Repository\SaisonRepository $repoSaison
 * @param \App\Repository\EquipeSaisonJoueurRepository $repoLicence
 * @param \App\Entity\Equipe $equipe
 * @return \Symfony\Component\HttpFoundation\Response
 * @Route("/les-equipes/{id}", name="une_equipe")
 */
public function uneEquipe(SaisonRepository $repoSaison, EquipeSaisonJoueurRepository $repoLicence, Equipe $equipe)
{
    /**
     * @var \App\Entity\Saison[] $saisons
     */
    $saisons = $repoSaison->findAll();

    /**
     * @var \Doctrine\Common\Collections\Collection $saisonsEquipe
     */
    $saisonsEquipe = $equipe->getSaison();

    /**
     * @var \App\Entity\EquipeSaisonJoueur[] $licences
     */
    $licences = $repoLicence->findAll();

    return $this->render('site/equipe/uneEquipe.html.twig', [
        'equipe' => $equipe,
        'saisons' => $saisons,
        'saisonsEquipe' => $saisonsEquipe,
        'licences' => $licences
    ]);
}
```

La vue du site grâce au fichier twig 'uneEquipe.html.twig'. (Voir code source).
L'adresse de route pour y accéder est 'http://clubfoot/les-equipes/**id_equipe**'.

Les équipes par saisons : Saison n°1 Saison n°2 Saison n°3 2021-2022

L'équipe Lesotho

Liste des saisons de l'équipe :

Saisons : Saison n°1 Saison n°2 Saison n°3

Liste des joueurs saison Saison n°3:

	Nom	Prénom	Adresse email	Date de naissance	Informations
1	Dupuis	Antoine	victor.arnaud@ifrance.com	19/04/2008	Voir les détails
2	Bertrand	André	marc.guillet@voila.fr	17/02/1989	Voir les détails
3	Blin	Matthieu	guy86@wanadoo.fr	23/08/1984	Voir les détails
4	Ribeiro	Michel	bernadette57@wanadoo.fr	27/06/2011	Voir les détails
5	Perez	Olivier	jules55@marchal.fr	01/07/1976	Voir les détails

2.3.3 Liste des joueurs

Le contrôleur qui gère la page de la liste des joueurs et des informations d'un joueur particulier est le suivant : `JoueurController`, il se trouve dans `Controller/Site/`.

Ce contrôleur comporte deux fonctions :

- `joueurs`

```
/**
 * Le controller pour gérer la page de la liste complète des joueurs
 *
 * @Route("/les-joueurs", name="les_joueurs")
 */
public function joueurs(JoueursRepository $repo)
{
    /**
     * @var \App\Entity\Joueurs[] $joueurs
     */
    $joueurs = $repo->findAll();

    return $this->render('site/joueur/joueurs.html.twig', [
        'controller_name' => 'JoueurController',
        'joueurs' => $joueurs,
    ]);
}
```


La vue du site grâce au fichier twig 'joueurs.html.twig'. (Voir code source).
L'adresse de route pour y accéder est 'http://clubfoot/les-joueurs'.

Liste des joueurs

Joueur n°1

Dupuis Antoine

Né le 19/04/2008




voir les détails

Joueur n°2

Bertrand André

Né le 17/02/1989




voir les détails

Joueur n°3

Blin Matthieu

Né le 23/08/1984



voir les détails

- unJoueur

```
$arraySaisonsJoueur = array();

/**
 * On parcourt toutes les saisons
 */
foreach ($saisons as $saison) {

    /**
     * On récupère les joueurs de chaque saison
     * @var mixed[] $joueurs
     */
    $joueurs = $saison->getJoueurs();


    /**
     * On parcourt la liste des joueurs pour récupérer la liste des saisons d'un joueur particulier
     */
    foreach ($joueurs as $valeur) {
        if ($valeur->getId() == $id) {
            array_push($arraySaisonsJoueur, $saison);
        }
    }
}

return $this->render('site/joueur/unJoueur.html.twig', [
    'joueur' => $joueur,
    'equipesJoueur' => $equipesJoueur,
    'saisonsJoueur' => $arraySaisonsJoueur,
    'licences' => $licences,
    'matches' => $matches,
    'participations' => $participations,
    'buts' => $buts,
    'passes' => $passes
]);
```

La vue du site grâce au fichier twig 'unJoueur.html.twig'. (Voir code source).
L'adresse de route pour y accéder est 'http://clubfoot/les-joueurs/**id_joueur**'.

Dupuis Antoine

Né le 19/04/2008



Dupuis Antoine à jouer pour les équipes suivantes :

Equipes :

Lesotho

Saisons :

Saison n°1

Saison n°2

Saison n°3

Les statistiques du joueur dans l'équipe Lesotho pour la saison Saison n°3

Date	Match	Passes décisive	Buts	Minutes de jeu
23-08-1990	11 - match n°1	0 passes décisive	0 buts	90 min
23-02-1997	12 - match n°2	1 passes décisive	1 buts	90 min
01-10-2008	13 - match n°3	0 passes décisive	0 buts	90 min
15-09-1982	14 - match n°4	0 passes décisive	0 buts	90 min
22-09-1970	15 - match n°5	0 passes décisive	0 buts	90 min
Total	Nombre de matchs : 5	Passes décisive : 1	Buts : 1	Temps total : 450 min

2.3.4 Création des équipes

Le contrôleur qui gère la page de la création d'une équipe est le suivant : GestionEquipesController, il se trouve dans Controller/Admin/.

Ce contrôleur comporte la fonction creationEquipe et gère un formulaire pour la création d'une équipe

```
/**
 * Formulaire de création d'équipe avec un champ pour le nom de l'équipe et un champ de sélection pour son niveau
 *
 * @var \Symfony\Component\Form\FormInterface $form
 */
$form = $this->createFormBuilder($equipe)
    ->add('nom')
    ->add('niveau', EntityType::class, [
        'class' => Niveau::class,
        'choice_label' => 'libelle',
    ])

    ->getForm()
    ->handleRequest($request);

if ($form->isSubmitted() && $form->isValid()) {

    $manager->persist($equipe);

    // On ajoute dans l'équipe la dernière saison en cours
    $equipe->addSaison($lastSaison);
    $manager->flush();

    return $this->redirectToRoute('une_equipe', ['id' => $equipe->getId()]);
}

return $this->render('admin/creationEquipes.html.twig', [
    'saisons' => $saisons,
    'formEquipe' => $form->createView(),
]);
```

(Voir le code source)

La vue du site grâce au fichier twig 'creationEquipes.html.twig'. (Voir code source).
L'adresse de route pour y accéder est 'http://clubfoot/admin/creation-equipes'.

Si l'utilisateur n'est pas authentifié et s'il n'est pas un administrateur il sera automatiquement redirigé vers la page de connexion.



2.3.5 Inscription des joueurs

Le contrôleur qui gère la page de l'inscription d'un joueur est le suivant : GestionJoueursController, il se trouve dans Controller/Admin/.

Ce contrôleur comporte la fonction inscriptionJoueurs et gère un formulaire pour l'inscription d'un joueur. (voir code source)

La vue du site grâce au fichier twig 'inscriptionJoueurs.html.twig'. (Voir code source). L'adresse de route pour y accéder est 'http://clubfoot/admin/inscription-joueurs'.

Si l'utilisateur n'est pas authentifié et s'il n'est pas un administrateur il sera automatiquement redirigé vers la page de connexion.

Les inscriptions de joueurs

Nom

Nom du joueur

Prenom

Prénom du joueur

Email

Email du joueur

Date naissance

1

▼

janv.

▼

2015

▼

Equipe

Arménie

▲

▼

Inscrire le joueur

3. Test de l'application

3.1 Les fixtures

En première étape, une base de données pour l'environnement de test a été créée avec la commande :

```
php bin/console doctrine:database:create --env=test
```



La commande pour appliquer les migrations de données dans l'environnement de test est la suivante :

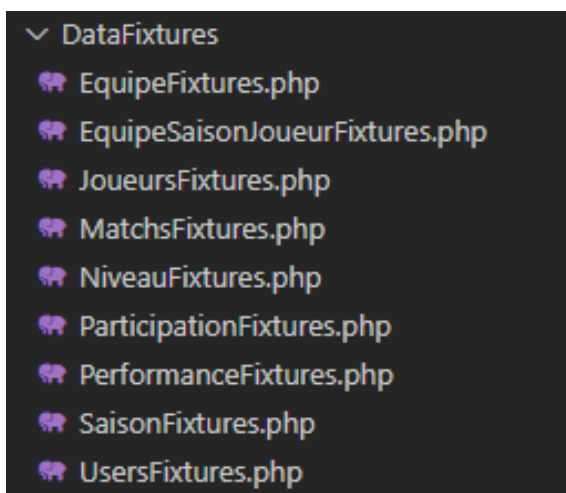
```
php bin/console doctrine:migrations:migrate --env=test
```

Ensuite, on obtient la même structure de base de données que la principale.

Avant de créer des fixtures on installe son composant avec la commande suivante :

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

Les fichiers de fixtures se trouvent dans 'src/DataFixtures'



Les fixtures permettent de remplir la base de données avec des fausses données. Pour tester le site directement les fixtures ont été lancées dans la base de données principale grâce à la commande suivante :

```
php bin/console doctrine:fixtures:load
```

Pour lancer les fixtures dans l'environnement de test on utilise la commande :

```
php bin/console doctrine:fixtures:load --env=test
```

Lorsque les fixtures sont lancées, l'ordre de chargement des fichiers est obtenu grâce aux dépendances qu'on a défini sur ces fichiers. L'ordre de chargement est la suivante :

```
PROBLÈMES  SORTIE  TERMINAL  CONSOLE DE DÉBOGAGE

PS C:\wamp64\www\ClubFoot> php bin/console doctrine:fixtures:load --env=test

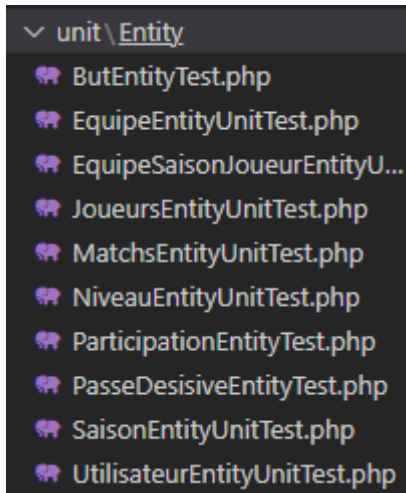
Careful, database "clubfoot_test" will be purged. Do you want to continue? (yes/no) [no]:
> y

> purging database
> loading App\DataFixtures\UsersFixtures
> loading App\DataFixtures\NiveauFixtures
> loading App\DataFixtures\EquipeFixtures
> loading App\DataFixtures\SaisonFixtures
> loading App\DataFixtures\JoueursFixtures
> loading App\DataFixtures\EquipeSaisonJoueurFixtures
```

(Voir le code source pour plus de détails avec des explications)

3.2 Tests techniques et fonctionnels

- Tests Unitaires pour la base de données

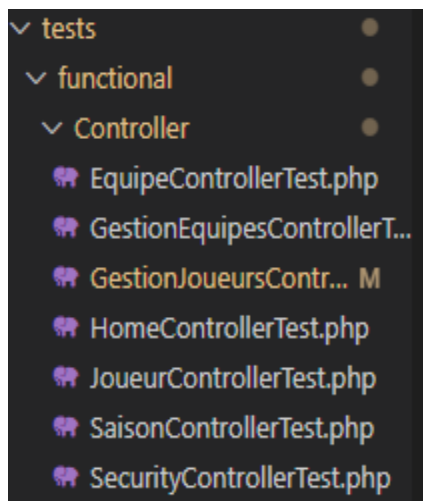


Dans chaque fichier on test une entité de la base de données. On vérifie leurs fonctions avec des tests unitaires. Pour effectuer des tests unitaires on utilise le framework de PHPUnit et on fait appel à la classe 'TestCase'.

Des assertions sont effectuées pour chaque fonction d'une entité. Ainsi, différentes conditions sont vérifiées avec les méthodes suivantes : 'assertTrue()', 'assertFalse()', 'assertEmpty()' ou 'assertContains'.

(Voir le code source pour plus de détails)

- Tests Fonctionnels sur les Controller



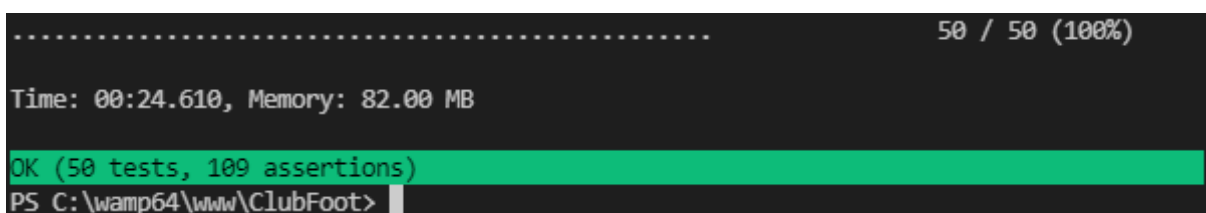
Les tests des contrôleurs ne se font pas avec des tests unitaires mais des tests fonctionnels. Les contrôleurs sont exécutés avec une requête HTTP.

On utilise la classe `Symfony\Bundle\FrameworkBundle\Test\WebTestCase` à la place de `PHPUnit\Framework\TestCase`.

Avec la variable `$client` on simule un navigateur. A la place de faire des appels HTTP au serveur, on a recours directement à l'application Symfony. Avec ces tests nous pouvons analyser l'état des services après chaque requête

HTTP. Des assertions sont également effectuées avec diverses méthodes avec PHPUnit. (Voir le code source pour plus de détails)

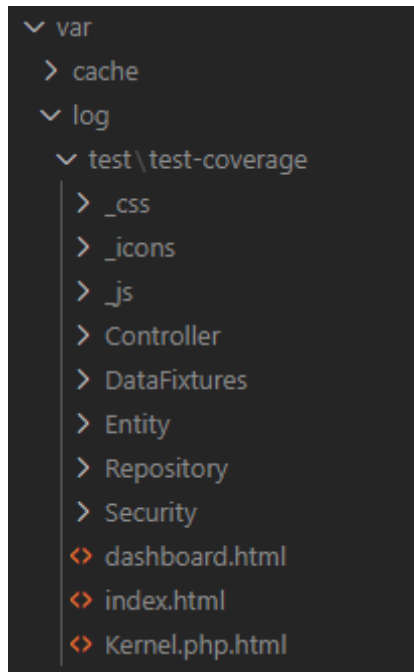
- Commande pour lancer les tests avec PHPUnit : **php bin/phpunit**



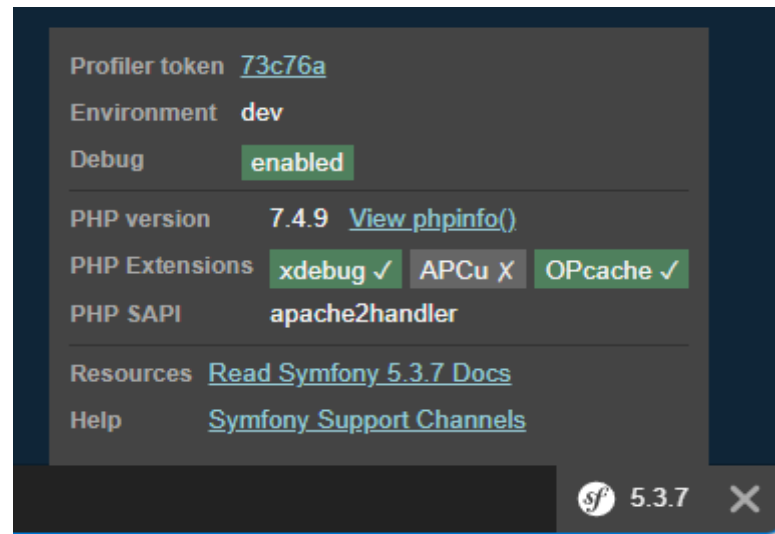
On a 50 tests qui sont effectués avec 109 assertions. Tous les tests effectués sont validés.

- Commande pour lancer les tests et avoir la couverture de l'ensemble des classes et des fonctions :

php bin/phpunit --coverage-html var/log/test/test-coverage







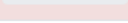












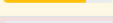

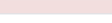
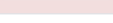
Pour avoir cette partie-là il faut prendre le soin de vérifier que l'extension Xdebug pour PHP est bien activé sur Wamp.



Lorsque on exécute la commande avec la couverture de code en paramètre un fichier en format HTML est généré et il rapporte le résultat qu'on demande.

Ainsi, en ouvrant le fichier index.html dans un navigateur la couverture globale des classes et des fonctions est affichée.

C:\wamp64\www\ClubFoot\src / (Dashboard)					
		Lines			
Total		97.57%		602 / 617	
Controller		100.00%		120 / 120	
DataFixtures		100.00%		301 / 301	
Entity		99.34%		150 / 151	
Repository		100.00%		20 / 20	
Security		91.67%		11 / 12	
Kernel.php		0.00%		0 / 13	

Code Coverage					
Functions and Methods			Classes and Traits		
	97.10%	134 / 138		92.11%	35 / 38
	100.00%	10 / 10		100.00%	7 / 7
	100.00%	17 / 17		100.00%	9 / 9
	98.95%	94 / 95		90.00%	9 / 10
	100.00%	10 / 10		100.00%	10 / 10
	75.00%	3 / 4		0.00%	0 / 1
	0.00%	0 / 2		0.00%	0 / 1