



PRAGOLF



Yahia LAMRI

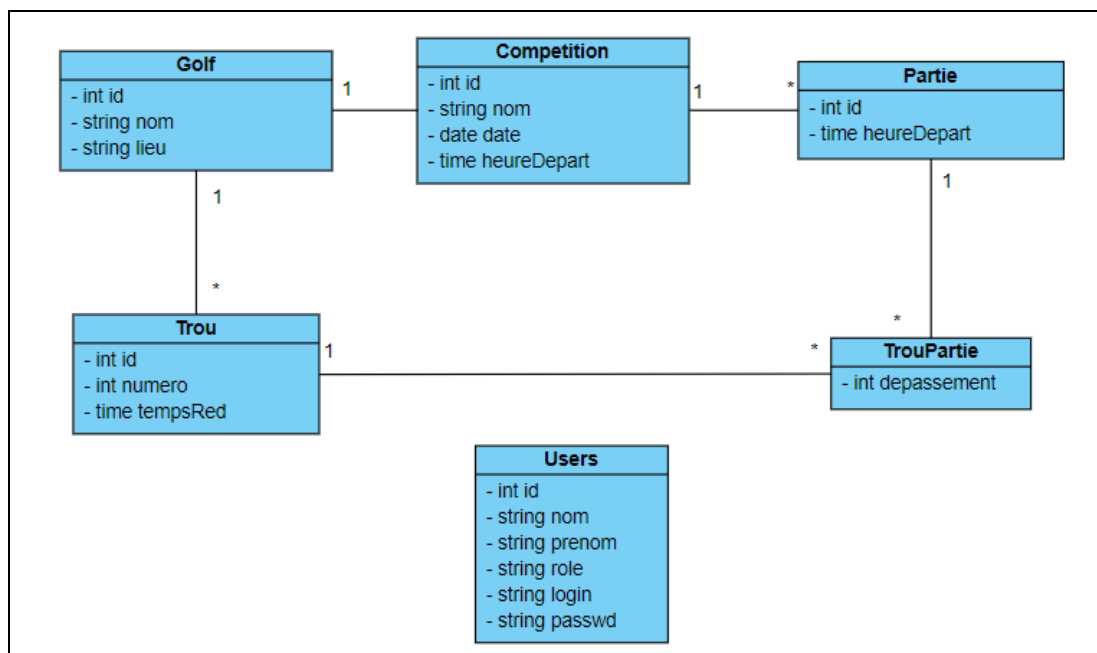
Stanislas RACZYNSKI

SIO22

PRAGolf est un application web de demande de cadences de jeu pour compétitions de golf amateur.

PARTIE 1

Dans un premier temps, nous avons défini le diagramme de classe des entités avec l'ensemble des attributs les concernant. Le diagramme est le suivant :



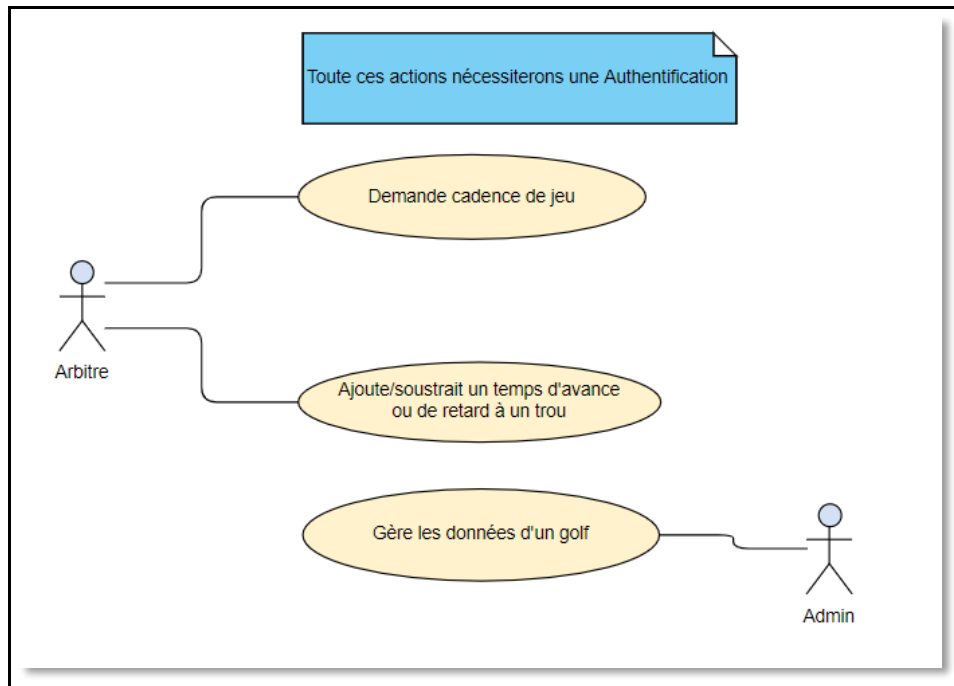
Les différentes classes utilisées représentent les différentes tables qui seront implémentées dans la base de données grâce à l'ORM Doctrine grâce à la commande :

```
> php bin/console make:entity
```

Ensuite, nous devons créer la base de données grâce aux informations se trouvant dans le fichier `.env` et à la commande `> php bin/console doctrine:database:create`.

Pour migrer les entités dans la base de données on peut le faire de la grâce aux commandes suivante : `> php bin/console make:migration`, une fois cette commande effectuée, on peut utiliser la commande suivante pour migrer définitivement les entités dans la base de données : `> php bin/console doctrine:migrations:migrate`

Dans un second temps, nous avons les différents cas d'utilisations. L'acteur principal qui est l'arbitre devra être en mesure de demander une cadence de jeu au système puis il devra être en mesure de reporter les différents décalages sur chaque trou à la fin de la compétition ; ce point sera exploré de manière secondaire. Également, l'acteur secondaire qui est l'administrateur gèrera l'ensemble de l'application avec les données concernant la compétition et les golfs ainsi que la gestion utilisateurs. Bien-sûr, toutes ces actions nécessitent une authentification de la part des utilisateurs



Stratégie adoptée :

Pour extraire et exploiter les données nous allons dans un premier temps récupérer les données d'un fichier Excel grâce à une librairie PHP (choix de la librairie plus tard), les méthodes tirées de la librairie permettront de parser les données et de les transformer en JSON, sous forme de tableaux associatifs au format JSON.

Point important : les noms des joueurs ne devront pas être enregistrés dans la base de données. Ils seront simplement extraits et affichés.

Jeu de donnée :

Pour le jeu de donnée nous devons extraire les données du fichier Excel, et remplir la base de données avec celles-ci grâce à une librairie spécifique. les tables de la base de données auront la structure suivante :

- pour la table Golf :

| id | nom | lieu |
|----|-------------------------------|---------------------|
| 1 | Golf du Coudray-Monceaux | Le Coudray-Monceaux |
| 2 | Golf Club d'Ozoir-la-Ferrière | Ozoir-la-Ferrière |
| 3 | Golf ParisLongchamp | Paris |
| 4 | Golf Club de Toulouse | Vieille-Toulouse |

- pour la table Trou :

| id | golf_id | numero | temps_ref | par |
|----|---------|--------|-----------|-----|
| 1 | 1 | 1 | 07:44:00 | 3 |
| 2 | 1 | 2 | 07:59:00 | 3 |
| 3 | 2 | 1 | 07:44:00 | 3 |
| 4 | 2 | 2 | 07:59:00 | 5 |
| 5 | 3 | 1 | 07:44:00 | 4 |
| 6 | 3 | 2 | 07:59:00 | 5 |

- pour la table Compétition :

| id | golf_id | nom | date | heure_depart |
|----|---------|-------------------------|------------|--------------|
| 1 | 1 | Trophée Sénior | 2019-08-06 | 07:30:00 |
| 2 | 2 | Trophée Junior | 2019-08-09 | 07:30:00 |
| 3 | 4 | Trophée junior - Finale | 2019-10-06 | 07:30:00 |
| 4 | 3 | Trophée Sénior - Finale | 2019-10-15 | 07:30:00 |

- pour la table Partie :

| id | competition_id | heure_depart |
|----|----------------|--------------|
| 1 | 1 | 07:30:00 |
| 2 | 3 | 07:30:00 |
| 3 | 1 | 07:30:00 |
| 4 | 2 | 07:30:00 |
| 5 | 1 | 07:44:00 |
| 6 | 2 | 07:44:00 |
| 7 | 3 | 07:44:00 |
| 8 | 4 | 07:44:00 |

- Pour la table TrouParties :

| id | trou_id | partie_id | depassement |
|----|---------|-----------|-------------|
| 1 | 1 | 5 | 00:00:02 |
| 2 | 2 | 6 | 00:00:03 |
| 3 | 5 | 8 | -00:02:00 |

La structure retenue pour le fichier JSON sera la suivante après extraction des données:

```
{
  "Competition": " Trophée Junior du Coudray-Monceaux",
  "Date": "2019-08-06",
  "Parties":
    {
      "id": 1,
      "competition_id",
      "trou": 1,
      "heure_depart": "07:30:00",
      "depassement": "-1"
    }
}
```

PARTIE 2

Dans la continuité de la partie 1, nous avons commencé par créer les entités métiers qui serviront par la suite à alimenter la base de données en passant par l'ORM. Nous avons dans un premier temps cherché à comprendre comment extraire les données d'un fichier Excel. Pour ce faire, nous avons utilisé une librairie (PhpSpreadsheet) qui permet de lire et d'extraire des informations de différents types de fichier (Word, Excel, etc) et de les transformer en un tableau PHP.

1) Extraction

Nous commençons d'abord par instancier cette classe de la manière suivante : `$reader = new \PhpOffice\PhpSpreadsheet\Reader\Xlsx();`. Ensuite nous chargeons le fichier Excel qui a été au préalable passé en argument de la méthode `$spreadsheet = $reader->load($excelFile);`. Nous récupérerons ensuite les valeurs des cellules avec les deux méthodes suivantes : `->getCell('B12')->getValue();` de ce fait, il nous faut juste parcourir toutes les lignes d'une colonne et vérifier à quelle couleur elles appartiennent pour les stocker dans un double tableau `$joueurs[$couleur][] = $nom;` qui stocke les noms des joueurs selon leurs couleurs, nous aurons à la fin un "grand" tableau contenant deux autres "petits" tableaux qui contiendront tous les joueurs appartenant à la même couleur.

```
for ($i = 1; $i < 175; $i++) {  
    $nom = $workSheet->getCell( pCoordinate: 'B'.$i)->getValue();  
    $couleur = $workSheet->getCell( pCoordinate: 'I'.$i)->getValue();  
    if ($nom !== null && $couleur !== null && $couleur !== "Rep.") {  
        $joueurs[$couleur][] = $nom;  
    }  
}
```

Une fois que notre tableau de joueurs est rempli, nous passons à la partie qui regroupe les noms par équipes de 3 ou 2 selon leurs couleurs. Pour ce faire, nous parcourons ligne par ligne chacun des deux "sous-tableau" et nous ajoutons les noms des joueurs regroupés dans un autre tableau intermédiaire qui contiendra la couleur de la partie et le nom des joueurs faisant parti de cette partie.

Il faut cependant prendre en compte le fait que les joueurs ne peuvent pas être seul dans une partie donc nous vérifions grâce à une autre condition notre tableau pour savoir combien il reste de joueur, s'il en reste 4 nous faisons donc deux parties de 2 joueurs. Nous retournons ensuite les informations pour les envoyer vers la vue.

```
foreach ($joueurs as $couleur => $joueur) { // Tri de joueurs; créations des parties
    $joueursRestant = count($joueur);
    unset($joueursGroup3);

    foreach ($joueur as $nom) {

        $joueursGroup3[] = $nom;

        if (($joueursRestant == 4 or $joueursRestant == 2) && (count($joueursGroup3) == 2)) {
            $parties[] = array($couleur, $joueursGroup3);
            $joueursRestant -= 2;
            unset($joueursGroup3);
        } elseif ($joueursRestant == 3 && (count($joueursGroup3) == 3)) {
            $parties[] = array($couleur, $joueursGroup3);
            $joueursRestant -= 3;
            unset($joueursGroup3);
        } elseif ($joueursRestant > 4 && (count($joueursGroup3) > 2)) {
            $parties[] = array($couleur, $joueursGroup3);
            $joueursRestant -= 3;
            unset($joueursGroup3);
        }
    }
}
```

2) Conversion

Pour la partie conversion en JSON nous utilisons le composant FileSystem de Symfony qui nous permet de faire des manipulations sur des fichiers et répertoires. Nous spécifions un nouveau chemin et nous créons le nouveau fichier avec la méthode `->touch($path);`

```
public static function phpToJson($file)
{
```

```
$newFile = new FileSystem();
$path = "../public/assets/doc/partie.json";
$newFile->touch($path); //creation du fichier partie.json
```

De ce point-ci nous récupérons le fichier qui a été ouvert et nous écrivons le JSON à l'intérieur

```
$filename = '../public/assets/doc/partie.json';
$newJsonFile = new File($filename);
$newJsonFile
    ->openFile( open_mode: 'w+')
    ->fwrite(json_encode($file, options: JSON_UNESCAPED_UNICODE));
```

Le paramètre 'w+' de la méthode `openFile()` nous ouvre un fichier qui est passé en paramètre de la méthode `phpToJson($file)`, et nous supprime ce qu'il y a à l'intérieur, de plus si le fichier n'existe pas il est créé automatiquement. `$file`, est récupéré grâce au formulaire Symfony expliqué plus bas.

3) Upload

Le formulaire d'upload a été réalisé grâce au composant Forms de Symfony.

La première étape est de créer un FormType qui contiendra le builder de notre formulaire (localisé dans src/Form), via la commande `> php bin/console make:form`. Ce builder permet de créer tous les champs de notre formulaire et de leur donner un type :

```
$builder
    ->add('heureDepart', TimeType::class)
    ->add('cadence', TimeType::class)
    ->add('fichier', FileType::class)
    ->add('save', SubmitType::class)
;
```


Dans le cas présent, le champ `heureDepart` permet à l'arbitre de préciser l'heure de départ qu'il souhaite, le champ `cadence` donnera l'écart de temps entre les parties (*ajouté manuellement au début puis récupéré automatiquement depuis la base de données*), puis enfin le champ `fichier` lui permet d'envoyer la liste des joueurs qui lui est fourni par la FFGolf.

```
public function upload(Request $request)
{
    //instanciation d'un objet competition
    $competition = new Competition();
    //creation du formulaire et liasion avec l'entité competition
    $form = $this->createForm(UploadFormType::class, $competition);
    $form->handleRequest($request);
    //si le formulaire est soumis et valide alors:
    if ($form->isSubmitted() && $form->isValid()) {
        //récupération du fichier qui a été uploadé
        $file = $competition->getFichier();
        //stockage du futur nom du fichier que notre application connaît
        $filename = "Fichier"."."."xlsx";
        //déplacement du fichier dans l'upload directory dont le chemin
        //est spécifié dans services.yaml
        //chemin: '%kernel.project_dir%/public/assets/doc'
        $file->move($this->getParameter('upload_directory'), $filename);
        //renommage du fichier
        $competition->setFichier($filename);
        //redirection vers la vue "view"
        return $this->redirectToRoute("view");
    }
}
```

Le code en image montre la séquence d'envoi du fichier Excel soumis par l'arbitre. Le `getter` permet de récupérer les données soumises dans le formulaire (notre fichier) qui est ensuite renommé en la valeur suivante `$filename = "Fichier" . "." . "xlsx";`.

Une fois renommé il est déplacé dans le chemin sauvegardé dans le paramètre `'upload_directory'` qui déplace le fichier dans notre dossier `/public/assets/doc` du projet pour pouvoir être récupéré pour son extraction. La méthode retourne enfin vers la vue "view" qui affiche le tableau.

4) Affichage du tableau

```
public function view(ExcelExtract $excelExtract)
{
    $fichierJoueurs = $excelExtract::ExceltoPhp( excelFile: '../public/assets/doc/Fichier.xlsx');
    $tableauJoueurs = $excelExtract::phpToJson($fichierJoueurs);
    $tempsTrous = [14, 15, 13, 17, 17, 16, 14, 19, 12, 15, 14, 18, 16, 13, 14, 17, 13, 15];
    $heureDepart = 7;
    $minDepart = 30;
```

D'un point de vu Front-End, nous devons d'abord récupérer le fichier que nous convertissons, ensuite nous définissons manuellement le temps de marche et le l'heure de départ de la compétition.

Nous envoyons ensuite ces informations au moteur de templates Twig à travers duquel nous pourrons afficher nos informations.

```
return $this->render( view: 'index/view', [
    'joueurs' => $tableauJoueurs,
    'trous' => $tempsTrous,
    'h' => $heureDepart,
    'm' => $minDepart,
]);
```

Du côté Twig nous parcourons le tableau `'joueurs'`, de la manière suivante et nous affichons grâce à une boucle les temps de départ est les différents temps de jeu des parties

```
{% for partie in parties %}
    <tr>
        <td>
            #
        </td>
        <td>
            {{ h }}:{{ m }}
        </td>

        <td>
            {% for joueur in partie[1] %}
                {{ joueur }}<br>
            {% endfor %}
        </td>
    </tr>
```

Le résultat de la vue twig est le suivant : 4 équipes sont affichées sur 4 trous pour des raisons de lisibilité.

| TROPHEE SENIORS DU COUDRAY , le mardi 6 août 2019 - 117 joueurs inscrits | | | | | | | |
|--|--------|--|---------|------|------|------|------|
| n° | Départ | Joueurs | Couleur | 1 | 2 | 3 | 4 |
| # | 7:30 | BONDY Fabrice AUSTIN Florus CASGRAIN Orva | Jaunes | 7:44 | 7:59 | 8:12 | 8:29 |
| # | 7:41 | ROUX Didiâne LAPRESSE Laurent LUSSIER Joy | Jaunes | 7:55 | 8:10 | 8:23 | 8:40 |
| # | 7:52 | RICHER Mathilde CAILOT Mirabelle PINETTE Dominic | Jaunes | 8:06 | 8:21 | 8:34 | 8:51 |
| # | 8:03 | ARCHAMBAULT Gaetan METIVIER Clément BELLEFEUILLE Léa | Jaunes | 8:17 | 8:32 | 8:45 | 9:02 |

Fin de la liste avec découpage en groupes “égaux”.

| | | | | | |
|---|--------|-------|-------|-------|-------|
| ROUTHIER Éloïse RANCOURT Damiane CAMUS Pascal | Rouges | 14:42 | 14:57 | 15:10 | 15:27 |
| LUSSIER Mathieu BOUTOT Médé | Rouges | 14:53 | 15:08 | 15:21 | 15:38 |

Menu de sélection Administrateur (Back-end / Maintenance) et Arbitre (Client)



PARTIE 3

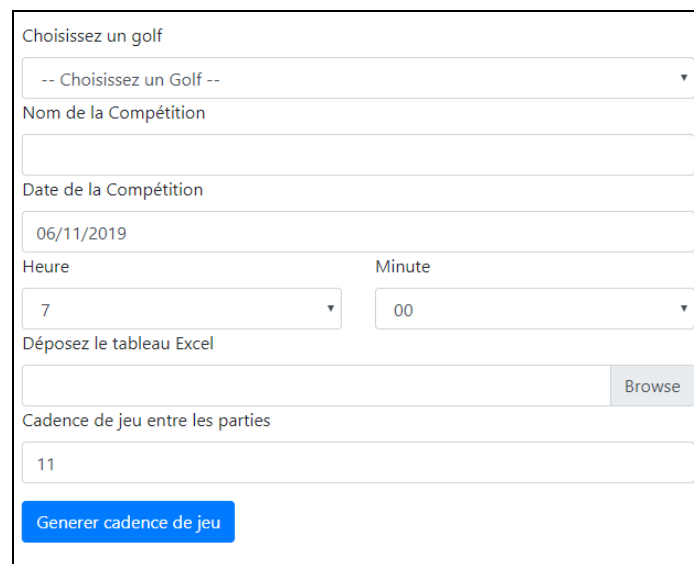
Lors des phases précédentes nous avons implémentés les entités métiers, l'extraction et le formatage des données lues depuis le fichier Excel, pour cette phase finale, nous avons implémenté le système de connexion avec la différenciation des rôles (USER, ADMIN) avec la génération du fichier PDF, la mise en place d'un formulaire servant à alimenter la base de données avec les informations concernant la compétition (date, lieu, golf, ...).

Différentes correction et ajustement ont été fait pour répondre au mieux aux attentes de notre projet.

Nous avons également mis en place un système de back office pour la gestion des golfs et des données utilisateurs.

1) Demande de cadence de jeu

Se présentant de la manière suivante, le formulaire servira avant tout à l'arbitre afin de renseigner les informations concernant la compétition dans laquelle il va arbitrer, pour ainsi générer sa cadence de jeu.



The form is titled "Choisissez un golf" and contains the following fields and controls:

- A dropdown menu with the placeholder text "-- Choisissez un Golf --".
- A text input field labeled "Nom de la Compétition".
- A date input field labeled "Date de la Compétition" with the value "06/11/2019".
- Two dropdown menus for "Heure" (set to "7") and "Minute" (set to "00").
- A text input field labeled "Déposez le tableau Excel" with a "Browse" button next to it.
- A text input field labeled "Cadence de jeu entre les parties" with the value "11".
- A blue button at the bottom labeled "Generer cadence de jeu".

L'implémentation se fait grâce au `formbuilder` qu'intègre Symfony. Il nous est plus simple de créer et aussi de relier nos champs formulaire aux attributs des Entités.

Les points importants de ce formulaire sont le fait que certaines valeurs sont non-modifiable par l'arbitre, il ne peut donc pas modifier leurs valeurs textuellement, mais il peut cependant les choisir dans une liste déroulante (nom du golf ou temps de départ par exemple).

```

$builder
  →add( child: "golf", type: EntityType::class, [
    'class' ⇒ Golf::class,
    'choice_label' ⇒ 'nom',
    'query_builder' ⇒ function (EntityRepository $e){
      return $e→createQueryBuilder( alias: 'g')→orderBy( sort: 'g.nom', order: 'ASC');
    },
    'choice_value' ⇒ 'nom',
    'placeholder' ⇒ '-- Choisissez un Golf --'
  ])
)

```

Pour sélectionner le golf nous faisons une requête qui récupère toutes les valeurs dans la table golf et nous les affiches sous forme d'une liste.

D'autres attributs comme l'heure de départ de la compétition sont aussi sélectionnables parmi une liste de valeurs.

```

→add( child: 'heureDepart', type: ChoiceType::class, [
  'choices' ⇒ [
    '7' ⇒ '7',
  ],
  'label' ⇒ 'Heure de départ'
])
→add( child: 'minuteDepart', type: ChoiceType::class, [
  'choices' ⇒ [
    '00' ⇒ '00',
    '30' ⇒ '30'
  ]
])

```

(tables compétition & golf)

| id | golf_id | date | nom_golf | nom_compet |
|----|---------|------------|-------------------|---------------------|
| 1 | 1 | 28/10/2019 | Golf du Coudray | Trophée Seniors |
| 2 | 1 | 29/10/2019 | Golf du Coudray | Trophée Juniors |
| 3 | 3 | 28/10/2019 | Golf du Montereau | Finale Junior |
| 4 | 1 | 31/10/2019 | Golf du Coudray | Trophée FFGolf |
| 5 | 1 | 06/11/2019 | Golf du Coudray | Entraînement Junior |

| id | nom | lieu |
|----|-------------------|-----------|
| 1 | Golf du Coudray | Coudray |
| 2 | Golf de Melun | Melun |
| 3 | Golf de Montereau | Montereau |

Les valeurs sont directement transmises depuis le formulaire jusqu'à la base de données puis récupérées dans le contrôleur pour ensuite être envoyées à la vue.

Si le formulaire est soumis et est valide alors nous récupérons les valeurs de ce dernier et, grâce aux accesseurs de l'Entité compétition nous pouvons agir sur les attributs pour en modifier leurs valeurs.

| | |
|--|--|
| <pre>if (\$form->isSubmitted() && \$form->isValid()) { \$heureDepart = \$competition->getHeureDepart(); \$minuteDepart = \$competition->getMinuteDepart(); \$date = \$competition->getDate(); \$nomCompet = \$competition->getNomCompet(); }</pre> | <pre>\$competition ->setHeureDepart(\$heureDepart) ->setMinuteDepart(\$minuteDepart) ->setDate(\$date) ->setNomCompet(\$nomCompet)</pre> |
|--|--|

Nous validons les requêtes effectuées et les envoyons dans la base de données, grâce au méthode `->persist($competition);` et `->flush();`. Nous retournons ensuite à la méthode `view` du contrôleur pour récupérer ces informations et les envoyer à la vue `return $this->redirectToRoute("view");`.

Pour avoir ces données nous récupérons le dernier élément qui vient d'être ajouté à la base de données (avec le formulaire) grâce à la méthode `->findOneBy([], ['id' => 'DESC']);` qui récupère toutes les informations du dernier élément.

D'un point de vue du rendu en twig nous envoyons les informations concernant la compétition, le tableau des ordonné des joueurs et les temps des trous qui n'ont pas encore été implémenté de manière automatique, nous les définissons donc comme attribut privé du contrôleur : `private $tempsTrous = [14, 15, 13, ... ,15];`

Les valeurs des trous sont calculées depuis le fichier pdf fournis en exemple pour pouvoir générer les autres cadences. *(ces temps peuvent être ajoutés par la suite dans la base de données et modifiés par les arbitres grâce à un formulaire lors de la demande de cadence).*

```
$tempsTrous = $this->tempsTrous;
$info_compet = $this->getDoctrine()
    ->getManager()
    ->getRepository( className: Competition::class)->findOneBy([], ['id' => 'DESC']);

return $this->render( view: 'index/view', [
    'info_compet' => $info_compet,
    'parties' => $tableauJoueurs,
    'trous' => $tempsTrous,
]);
```

2) Génération PDF

Après avoir générer le tableau du côté de la vue nous pouvons donc nous pencher sur la génération du fichier PDF qui servira d'appui au corp arbitral durant tout le déroulement de la compétition.

Pour générer le PDF nous avons créé un "doublon" de la vue de base nommé `pdf.html.twig` qui sert à la création du fichier PDF et contenant les mêmes informations que la vue, nous utilisons ensuite une librairie `Html2Pdf` (sans oublier l'injection de dépendance dans le fichier `services.yaml`).

```
$template = $this->renderView( view: 'index/pdf', [
    'info_compet' => $info_compet,
    'parties' => $tableauJoueurs,
    'trous' => $tempsTrous,
]);

$pdfFile = new Html2Pdf( orientation: 'L', format: 'A4', lang: 'fr');
$pdfFile->writeHTML($template);
$pdfFile->output( name: "compet.pdf");
```

Finalement le PDF est généré avec les bonnes informations remontées depuis la base de données et des données saisies par l'arbitre dans le formulaire.

| Finale Coupe du Monde 2019, le 07/12/2019 -- 117 joueurs. Heure Départ : 7h00 | | | | | | | | | | | | | | | | | | |
|---|--------|--|---------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| # | Départ | Joueurs | Couleur | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 7.0 | | BONDY Fabrice AUSTIN Floris CASGRAIN Onis | Jaunes | 7:14 | 7:29 | 7:42 | 7:59 | 8:16 | 8:32 | 8:46 | 9:05 | 9:17 | 9:32 | 9:46 | 10:04 | 10:20 | 10:33 | 10:47 |
| 7.13 | | ROUX Odane LAPRESSE Laurent LUSSIER Joy | Jaunes | 7:27 | 7:42 | 7:55 | 8:12 | 8:29 | 8:45 | 8:59 | 9:18 | 9:30 | 9:45 | 9:59 | 10:17 | 10:33 | 10:46 | 11:00 |
| 7.26 | | RICHER Mathilde CALOT Mirabelle FINETTE Dominic | Jaunes | 7:40 | 7:55 | 8:08 | 8:25 | 8:42 | 8:58 | 9:12 | 9:31 | 9:43 | 9:58 | 10:12 | 10:30 | 10:46 | 10:59 | 11:13 |
| 7.39 | | ARCHAMBAULT Gaetan METTIER Clement BELLEFEUILLE Léa | Jaunes | 7:53 | 8:08 | 8:21 | 8:38 | 8:55 | 9:11 | 9:25 | 9:44 | 9:56 | 10:11 | 10:25 | 10:43 | 10:59 | 11:12 | 11:26 |
| 7.52 | | ROBAILLE Pascal CARONN Hugh BLONDLOT Alain | Jaunes | 8:06 | 8:21 | 8:34 | 8:51 | 9:08 | 9:24 | 9:38 | 9:57 | 10:09 | 10:24 | 10:38 | 10:56 | 11:12 | 11:25 | 11:39 |
| 8.05 | | BREBETTE Jeannine BEAUSOLEIL Damien AVARE Hales | Jaunes | 8:19 | 8:34 | 8:47 | 9:04 | 9:21 | 9:37 | 9:51 | 10:10 | 10:22 | 10:37 | 10:51 | 11:09 | 11:25 | 11:38 | 11:52 |
| 8.18 | | LAMARE Anais AUCLAIR Sylvie L'ANGLAIS Nathalie | Jaunes | 8:32 | 8:47 | 9:00 | 9:17 | 9:34 | 9:50 | 10:04 | 10:23 | 10:35 | 10:50 | 11:04 | 11:22 | 11:38 | 11:51 | 12:05 |
| 8.31 | | BONNET Rémy MAILLOUX Roger CING-MARS Pamy | Jaunes | 8:45 | 9:00 | 9:13 | 9:30 | 9:47 | 10:03 | 10:17 | 10:36 | 10:48 | 11:03 | 11:17 | 11:35 | 11:51 | 12:04 | 12:18 |
| 8.44 | | CUILLERER Franck DAVID Corderella BEGIN Maurella | Jaunes | 8:58 | 9:13 | 9:26 | 9:43 | 10:00 | 10:16 | 10:30 | 10:49 | 11:01 | 11:16 | 11:30 | 11:48 | 12:04 | 12:17 | 12:31 |
| 8.57 | | DUBEAU Gef FONTAINE Sophie CHICONE Valentine | Jaunes | 9:11 | 9:26 | 9:39 | 9:56 | 10:13 | 10:29 | 10:43 | 11:02 | 11:14 | 11:29 | 11:43 | 12:01 | 12:17 | 12:30 | 12:44 |
| 9.10 | | DESCOTEAUX Fleurine EPICIER Adeline MAHEU Patricia | Jaunes | 9:24 | 9:39 | 9:52 | 10:09 | 10:26 | 10:42 | 10:56 | 11:15 | 11:27 | 11:42 | 11:56 | 12:14 | 12:30 | 12:43 | 12:57 |
| 9.23 | | BIRLAND Lucile BIZRE Sébastien LABERGE Florant | Jaunes | 9:37 | 9:52 | 10:05 | 10:22 | 10:39 | 10:55 | 11:09 | 11:28 | 11:40 | 11:55 | 12:09 | 12:27 | 12:43 | 12:56 | 13:10 |
| 9.36 | | BONNEVILLE Christian JOSSEALME Aurene BOVIN Yves | Jaunes | 9:50 | 10:05 | 10:18 | 10:35 | 10:52 | 11:08 | 11:22 | 11:41 | 11:53 | 12:08 | 12:22 | 12:40 | 12:56 | 13:09 | 13:23 |
| 9.49 | | BEAUCHEUNE Salber CAYA Gaston GIGUE Apolline | Jaunes | 10:03 | 10:18 | 10:31 | 10:48 | 11:05 | 11:21 | 11:35 | 11:54 | 12:06 | 12:21 | 12:35 | 12:53 | 13:09 | 13:22 | 13:36 |
| 10.02 | | MAILLANCOUR Dominique LANDRY Marshall LACAILLE Jolie | Jaunes | 10:16 | 10:31 | 10:44 | 11:01 | 11:18 | 11:34 | 11:48 | 12:07 | 12:19 | 12:34 | 12:48 | 13:06 | 13:22 | 13:35 | 13:49 |
| 10.15 | | GOUDREAU Noemie CHAUVET Fanchon MOREAU Cecidion | Jaunes | 10:29 | 10:44 | 10:57 | 11:14 | 11:31 | 11:47 | 12:01 | 12:20 | 12:32 | 12:47 | 13:01 | 13:19 | 13:35 | 13:48 | 14:02 |
| 10.28 | | THIVIERGE Thierry SICARD Caro EDOUARD Ingrid | Jaunes | 10:42 | 10:57 | 11:10 | 11:27 | 11:44 | 12:00 | 12:14 | 12:33 | 12:45 | 13:00 | 13:14 | 13:32 | 13:48 | 14:01 | 14:15 |
| 10.41 | | GILBERT Frelty GAMBLIN Nathalie MARLEAU Rio | Jaunes | 10:55 | 11:10 | 11:23 | 11:40 | 11:57 | 12:13 | 12:27 | 12:46 | 12:58 | 13:13 | 13:27 | 13:45 | 14:01 | 14:14 | 14:28 |
| 10.54 | | BOURDETTE Christian LAUX Brady PATENAUD Claire | Jaunes | 11:08 | 11:23 | 11:36 | 11:53 | 12:10 | 12:26 | 12:40 | 12:59 | 13:11 | 13:26 | 13:40 | 13:58 | 14:14 | 14:27 | 14:41 |

3) Login

Pour permettre l'authentification des utilisateurs, nous avons utilisé la commande de Symfony `> php bin/console make:auth` qui nous permet de générer un Login Form Authenticator qui va construire le formulaire que l'utilisateur utilisera pour se connecter.

Connexion

Connexion

Le controller SecurityController permet le contrôler la connexion de l'utilisateur et d'afficher les erreurs si besoin. En cas de succès il redirige vers l'index.

Les mots de passes sont chiffrés automatiquement par le Framework comme suit :

```
security:
    encoders:
        App\Entity\User:
            algorithm: auto
```

4) Back Office

Réservé à l'administrateur

[Gestion des golfs](#)[Gestion des utilisateurs](#)

Le back office que nous avons créé permet à l'administrateur d'ajouter et de supprimer des golfs, il en va de même pour les utilisateurs. Pour se faire, une simple requête à la base de données nous permet d'afficher une liste contenant tous les golfs contenus dans la base grâce à l'ORM Doctrine : `$golfs = $this->getDoctrine()->getManager()->getRepository(Golf::class)->findAll();`

```
<form action="{{ path('/backoffice/golf/add') }}" method="post" class="m-4">
  <h2>Ajouter un golf</h2>

  <div class="row">
    <div class="col">
      <label for="name">Nom</label>
      <input type="text" name="newGolfName" id="name" class="form-control">
    </div>
    <div class="col">
      <label for="location">Lieu</label>
      <input type="text" name="newGolfLocation" id="location" class="form-control">
    </div>
  </div>
  <button type="submit" class="btn btn-outline-success m-3">Ajouter golf</button>
</form>
```

Ajouter un golf

Nom

Lieu

[Ajouter golf](#)

Gestions des Golfs

| # | Titre | Lieu | Actions |
|---|-------------------|-----------|---------------------------|
| 1 | Golf du Coudray | Coudray | Supprimer |
| 2 | Golf de Melun | Melun | Supprimer |
| 3 | Golf du Montereau | Montereau | Supprimer |

L'ajout d'un golf et son envoi vers la base de données se fait grâce à la méthode suivante située dans l'AdminController.

Nous récupérons les "nouvelle données" passées en *POST* grâce à `$request`


```

public function addGolf(Request $request)
{
    $golf = new Golf();
    $em = $this->getDoctrine()->getManager();
    $newName = $request->request->get( key: 'newGoflName');
    $newLocation = $request->request->get( key: 'newGoflLocation');
    $golf->setNom($newName)->setLieu($newLocation);

    $em->persist($golf);
    $em->flush();

    return $this->redirectToRoute( route: 'backoffice_golf');
}

```

Les autres opérations de CRUD concernant les golfs et les utilisateurs se font côté administrateurs uniquement, seul ce dernier bénéficie des accès requis.

5) Sécurité

Symfony, via son fichier security.yaml, propose une solution pour protéger l'accès à certaines pages de notre application, si jamais l'utilisateur n'est pas connecté à l'application ou s'il ne possède pas le rôle adéquat. Ici, on voit que la route `/backoffice` est seulement accessible aux utilisateurs possédant le rôle Administrateur.

Symfony permet aussi de hiérarchiser les rôles, c'est à dire que l'Administrateur par exemple possède aussi le rôle d'arbitre et le rôle d'user.

| | |
|--|--|
| <pre> access_control: - { path: ^/backoffice, roles: ROLE_ADMIN } - { path: ^/upload, roles: ROLE_ARBITRE } - { path: ^/uploadTrou, roles: ROLE_ARBITRE } - { path: ^/profile, roles: ROLE_USER } </pre> | <pre> role_hierarchy: ROLE_ADMIN: [ROLE_ARBITRE, ROLE_USER] ROLE_ARBITRE: ROLE_USER </pre> |
|--|--|