UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence 2 Informatique
INFO0303 – 2023/2024
***Resp. :* C. Rabat**

Devoir surveillé terminal

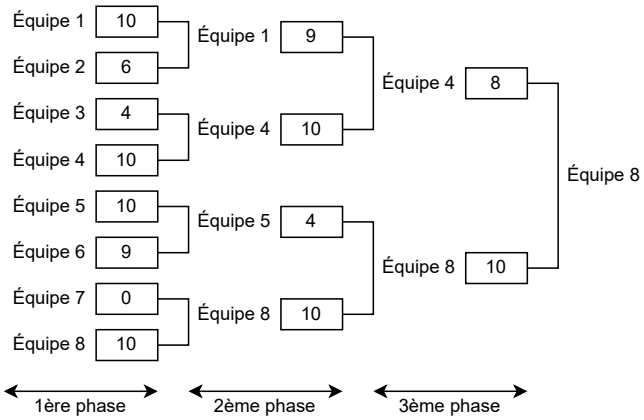
Durée 1h30
(Corrections)

Exercice 1 (Site de tournois de babyfoot - *7 pts*)
Nous souhaitons développer un site permettant de gérer des tournois de babyfoot (football de table). Ce sport se joue à deux équipes de deux joueurs. Pour gagner, il faut marquer dix buts en premier.

Tous les utilisateurs du site peuvent créer leur propre tournoi ; ils sont alors nommés des «organisateurs». Le tournoi est caractérisé par un intitulé, des dates de début et de fin pour les inscriptions, des dates de début et de fin pour le déroulement de la compétition et des frais d'inscription (nous ne gérons pas les paiements sur le site).

Les utilisateurs peuvent s'inscrire à des tournois (si la période d'inscription est ouverte) ; ils sont alors appelés des «participants». Ils doivent indiquer leur équipe (deux utilisateurs) au moment de l'inscription. L'équipe est caractérisée par un nom.

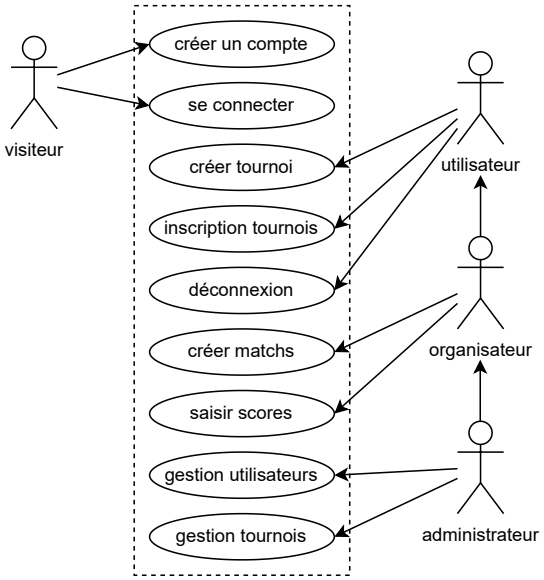
Une fois la période d'inscription terminée, l'organisateur crée les différents matchs entre les équipes, en sachant que les matchs sont à élimination directe. Lorsque le match est terminé, il renseigne le score (le nombre de buts marqués par chaque équipe). Les vainqueurs sont alors qualifiés pour la phase suivante. Le nombre de phases dépend du nombre d'équipes au départ. La figure ci-dessous illustre le déroulement complet d'un tournoi :



Le site possède également un administrateur qui a accès à toutes les informations sur le site.

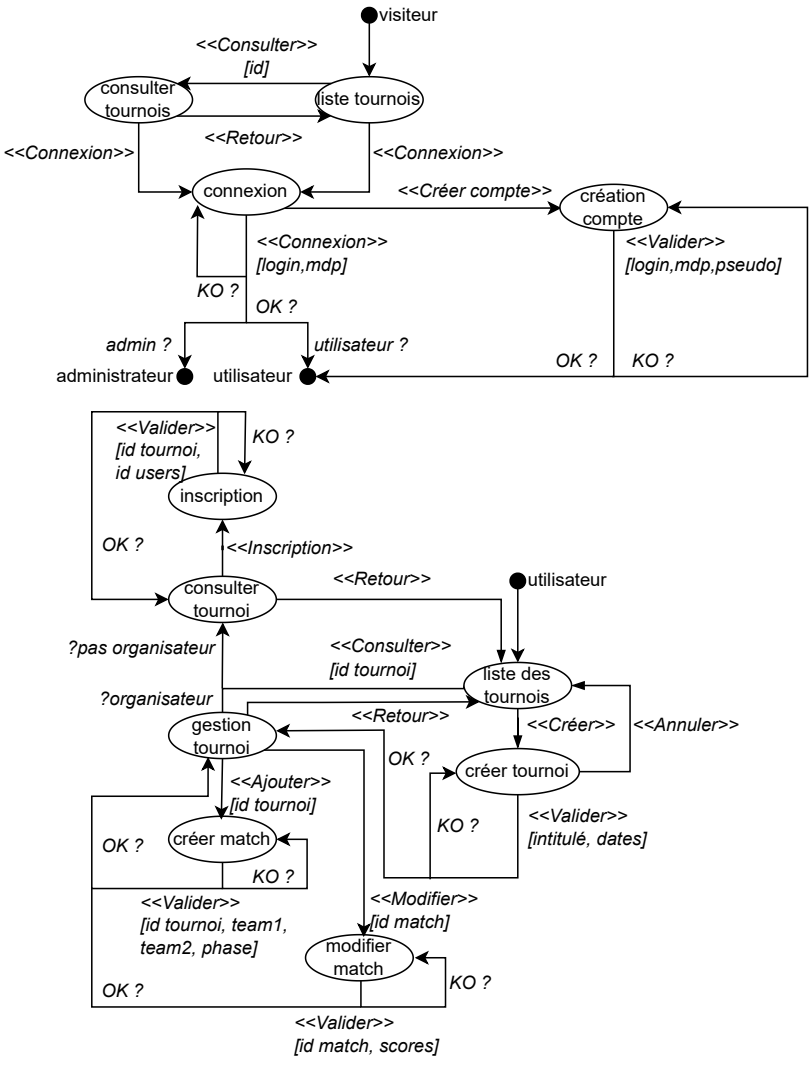
1°) Identifiez les acteurs du site et donnez les cas d'utilisation de l'application. *(1 point)*

Solution : *Il y a 4 acteurs : le visiteur, l'utilisateur (un visiteur enregistré), l'organisateur (un utilisateur qui a créé un tournoi) et l'administrateur. Le participant n'est pas forcément un acteur en tant que tel car il ne possède pas de rôle supplémentaire.*



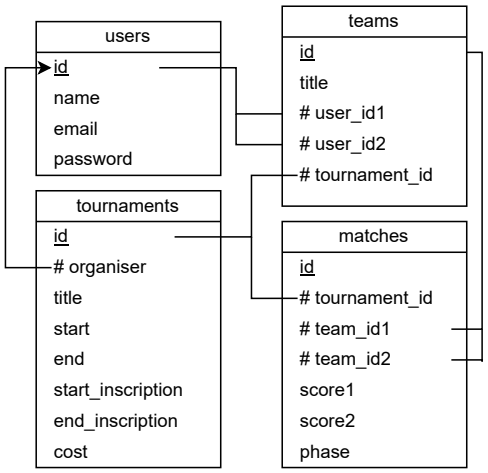
2°) Donnez les diagrammes de navigation de l'application (ne faites pas la partie administrateur, ni la partie déconnexion). *(4 points)*

Solution :



3°) Proposez un MLD et expliquez vos choix. *(2 points)*

Solution : *Les utilisateurs sont caractérisés par leur adresse email, leur nom et leur mot de passe. Un tournoi est caractérisé par un intitulé, l'organisateur, une date de début et de fin, une date d'inscription de début et de fin, un coût. Une équipe est caractérisée par un nom, par les utilisateurs qui la composent et le tournoi associé. Les matchs sont associés à un tournoi, aux deux équipes, par les scores des deux équipes et la phase. On pourrait également ajouter un ordre pour que les matchs apparaissent toujours dans le même ordre.*



Exercice 2 (Saisie d'un match de babyfoot - *7 pts*)
Nous souhaitons développer le site de tournois de babyfoot de l'exercice précédent à l'aide de *Laravel*. Nous nous intéressons à la création d'un match. L'organisateur doit sélectionner la phase puis renseigner les deux équipes du match. Nous supposons la présence d'un contrôleur `MatchController`.

1°) Rappelez ce qu'est un *template* en *Laravel* et expliquez les intérêts. *(1 point)*

Solution : *Un template permet de créer un patron pour un ensemble de pages, ce qui permet de factoriser le code. Lorsqu'une vue hérite d'un template, elle indique simplement le contenu des sections.*

2°) Quelle(s) section(s) est/sont nécessaire(s) dans le cas général dans un template ? Donnez son format général. *(1 point)*

Solution : *Il suffit d'une section `content` qui contiendra le contenu de la page. On peut ajouter des champs `title` pour le titre, `head` pour l'inclusion de scripts CSS/JavaScript, `script` pour des scripts JavaScript. Le format général est le suivant :*

```
<!doctype html>
<html lang="fr">
  <head>
    <title>@yield{'title'}</title>
    <!-- inclusion de CSS, de JavaScript, champs meta -->
    <title>@yield{'head'}</title>
  </head>
  <body>
    <!-- ici le menu, par exemple -->
    @yield('contenu')
    @yield('script')
  </body>
</html>
```

3°) Quelle(s) méthode(s) est/sont nécessaire(s) dans le contrôleur pour la création d'un match ? Indiquez ce qu'elle(s) fait/font. *(1 point)*

Solution : *Il faut deux méthodes : la méthode `create` et la méthode `store`. La première est appelée lorsque l'utilisateur crée le match et la seconde est appelée lorsque le formulaire est validé. Pour créer le match, on récupère la liste des équipes et on appelle ensuite la vue en lui passant les données en paramètres. La méthode `store` vérifie que les données saisies sont correctes (par exemple via une classe héritant de*

la classe `Request`) puis les données sont stockées dans la base de données. Les deux méthodes doivent vérifier si l'utilisateur courant possède les droits de réaliser cette action (on peut également utiliser une classe `Policy`).

4°) Quelles données sont nécessaires pour la vue de création d'un match ? Comment les obtenir (indiquez les éléments *Laravel* nécessaires) ? *(1 point)*

Solution : *La création du match nécessite la phase, le tournoi et la liste des équipes présentes. En supposant qu'un modèle `Team` existe, il suffit d'appeler la méthode `get` pour récupérer toutes les équipes.*

5°) Écrivez la vue permettant de créer un match (ne faites pas de gestion d'erreur et de mise en forme). *(2 points)*

Solution : *Voici une solution :*

```
@extends('template')
@section('content')
<form action="{{url('match')}}" method="post">
@csrf
  <label for="team1">Equipe 1</label>
  <select name="team1" id="team1">
@foreach($teams as $team)
    <option value="{{$team->id}}">{{$team->name}}</option>
@endforeach
  </select>
  <label for="team2">Equipe 2</label>
  <select name="team2" id="team2">
@foreach($teams as $team)
    <option value="{{$team->id}}">{{$team->name}}</option>
@endforeach
  </select>
  <button type="submit">Ajouter</button>
  <a href="{{route('match')}}">Annuler</a>
</form>
@endsection
```

6°) À partir d'un match donné, comment est-il possible de récupérer les équipes correspondantes ? Quel(s) élément(s) *Laravel* est/sont nécessaire(s) ? *(1 point)*

Solution : *Les équipes sont des clés étrangères vers la table `teams`. Dans le modèle `Match`, on ajoute les deux méthodes ci-dessous. En supposant que l'on ait un match instancié, il suffit alors d'appeler par exemple la méthode `$match->team1()` pour obtenir la première équipe.*

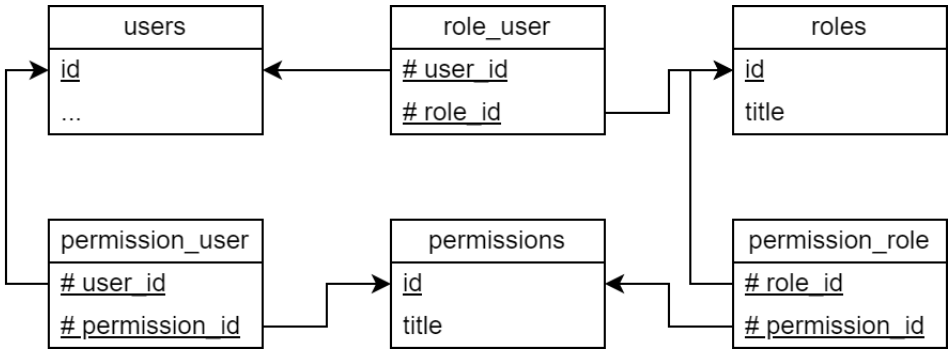
```
public function team1() : BelongsTo {
  return $this->belongsTo(Team::class, 'team1_id');
}
public function team2() : BelongsTo {
  return $this->belongsTo(Team::class, 'team2_id');
}
```

Il suffit d'appeler cette méthode pour récupérer l'équipe associée.

Exercice 3 (*Gestion des droits - 6 pts*)
Dans une application *Laravel*, nous souhaitons réaliser la gestion des droits d'accès des utilisateurs manuellement. Nous pouvons spécifier un ou plusieurs rôles à chaque utilisateur et à chaque rôle est associé un ensemble de permissions. Lorsqu'un rôle est attribué à un utilisateur, il possède de fait l'ensemble des permissions associées. Il est également possible d'accorder des permissions aux utilisateurs indépendamment de leur rôle.

1°) Proposez un MLD correspondant à cette modélisation. Vous utiliserez le schéma de nommage de *Laravel*. *(2 points)*

Solution : *Voici une solution :*



2°) Complétez les '...' dans les fonctions suivantes (vous trouverez les méthodes de la classe `Blueprint` en annexes) correspondant à la table pivot entre les rôles et les utilisateurs : *(2 points)*

Solution : *Voici une solution :*

```php
public function up(): void {
  Schema::create('role_user', function (Blueprint $table) {
    $table->unsignedBigInteger('user_id');
    $table->unsignedBigInteger('role_id');
    $table->primary(['user_id', 'role_id']);
    $table->foreign('user_id')->references('id')->on('users');
    $table->foreign('role_id')->references('id')->on('roles');
  }
}
public function down(): void {
  Schema::dropIfExists('role_user');
}
```

3°) *Laravel* propose l'utilisation du *soft deleting*. Rappelez son principe et ses intérêts. *(1 point)*

Solution : *Le soft deleting permet de supprimer les données des tables de manière non temporaire. Elles ne sont plus accessibles par défaut lors des requêtes classiques, mais elles peuvent être récupérées ou restaurées. Pour cela, une colonne `delete_at` est ajoutée sur les tables avec la valeur `NULL` par défaut.*

4°) Finalement, nous souhaitons utiliser le composant *Spatie* pour la gestion des droits. Expliquez l'intérêt d'un tel composant et comment l'utiliser dans *Laravel*. *(1 point)*

Solution : *Le composant Spatie contient des classes pour simplifier la création des rôles et des permissions (et l'association de permissions aux rôles), ainsi que l'association des rôles et permissions aux utilisateurs. Grâce à ce composant, nous pouvons utiliser le système de droits de Laravel avec les annotations Blade `@can()` ou via les fichiers `Policy` associés aux contrôleurs ou aux modèles. L'installation de Spatie se fait avec `composer`, ce qui crée les classes, dont le modèles et les migrations nécessaires.*