UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

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

Devoir surveillé terminal

Durée 1h30

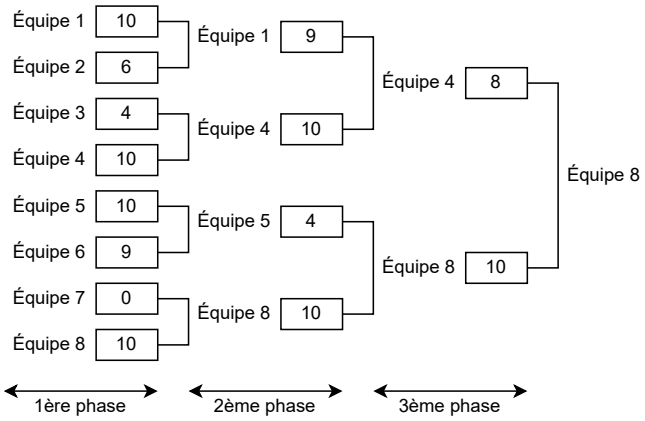
Aucun document autorisé. Toute réponse doit être justifiée. Vous ne devez pas préciser le style (CSS) dans vos scripts HTML. Vous utiliserez le typage dans vos scripts PHP.

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)*

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

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

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)*

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

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)*

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

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

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)*

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)*

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)*

```
public function up(): void {
  Schema::create('...', function (Blueprint $table) {
    ...
  }
}
public function down(): void {
  ...
}
```

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

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)*

Annexes

La liste simplifiée des méthodes de la classe `Illuminate\Database\Schema\Blueprint` :

```
Fluent drop();
Fluent dropIfExists();
Fluent dropColumn(array|mixed $columns);
Fluent dropPrimary(string|array|null $index = null)
Fluent dropUnique(string|array $index)
Fluent dropIndex(string|array $index)
Fluent dropForeign(string|array $index)
Fluent dropConstrainedForeignId(string $column)
Fluent dropForeignIdFor(Model|string $model, string|null $column = null)
Fluent dropConstrainedForeignIdFor(Model|string $model, string|null $column = null)
void dropTimestamps()
void dropSoftDeletes(string $column = 'deleted_at')
IndexDefinition primary(string|array $columns, string|null $name = null)
IndexDefinition unique(string|array $columns, string|null $name = null)
IndexDefinition index(string|array $columns, string|null $name = null)
ForeignKeyDefinition foreign(string|array $columns, string|null $name = null)
ColumnDefinition id(string $column = 'id')
ColumnDefinition increments(string $column)
ColumnDefinition char(string $column, int|null $length = null)
ColumnDefinition string(string $column, int|null $length = null)
ColumnDefinition text(string $column)
ColumnDefinition integer(string $column, bool $autoIncrement = false)
ColumnDefinition tinyInteger(string $column, bool $autoIncrement = false)
ColumnDefinition bigInteger(string $column, bool $autoIncrement = false)
ColumnDefinition unsignedInteger(string $column, bool $autoIncrement = false)
ColumnDefinition unsignedTinyInteger(string $column, bool $autoIncrement = false)
ColumnDefinition unsignedBigInteger(string $column, bool $autoIncrement = false)
ColumnDefinition float(string $column, int $total = 8, int $places = 2)
ColumnDefinition double(string $column, int|null $total = null, int|null $places = null)
ColumnDefinition unsignedFloat(string $column, int $total = 8, int $places = 2)
ColumnDefinition unsignedDouble(string $column, int $total = null, int $places = null)
ColumnDefinition boolean(string $column)
ColumnDefinition enum(string $column, array $allowed)
ColumnDefinition date(string $column)
ColumnDefinition dateTime(string $column, int|null $precision = 0)
ColumnDefinition time(string $column, int|null $precision = 0)
ColumnDefinition timestamp(string $column, int|null $precision = 0)
ColumnDefinition softDeletes(string $column = 'deleted_at', int|null $precision = 0)
ColumnDefinition addColumn(string $type, string $name, array $parameters = [])
$this removeColumn(string $name)
```

La liste simplifiée des méthodes de la classe `Illuminate\Support\Fluent` :

```
get($key, $default = null)
getAttributes()
toArray()
array jsonSerialize()
string toJson(int $options = 0)
bool offsetExists(TKey $offset)
mixed offsetGet(TKey $offset)
void offsetSet(TKey $offset, TValue $value)
void offsetUnset(TKey $offset)
$this __call(TKey $method, $parameters)
TValue|null __get(TKey $key)
void __set(TKey $key, TValue $value)
bool __isset(TKey $key)
void __unset(TKey $key)
```

La liste simplifiée des méthodes de la classe `Illuminate\Support\IndexDefinition` (hérite de la classe `Fluent`) :

```
$this algorithm(string $algorithm)
```

```
$this language(string $language)
$this deferrable(bool $value = true)
$this initiallyImmediate(bool $value = true)
```

La liste simplifiée des méthodes de la classe `Illuminate\Database\Schema\ColumnDefinition` (hérite de la classe `Fluent`) :

```
$this after(string $column)
$this always(bool $value = true)
$this autoIncrement()
$this change()
$this charset(string $charset)
$this collation(string $collation)
$this comment(string $comment)
$this default(mixed $value)
$this first()
$this from(int $startingValue)
$this generatedAs(Expression $expression = null)
$this index(string $indexName = null)
$this invisible()
$this nullable(bool $value = true)
$this persisted()
$this primary()
$this fulltext(string $indexName = null)
$this spatialIndex(string $indexName = null)
$this startingValue(int $startingValue)
$this storedAs(string $expression)
$this type(string $type)
$this unique(string $indexName = null)
$this unsigned()
$this useCurrent()
$this useCurrentOnUpdate()
$this virtualAs(string $expression)
```

La liste simplifiée des méthodes de la classe `Illuminate\Database\Schema\ForeignKeyDefinition` (hérite de la classe `Fluent`) :

```
$this cascadeOnUpdate()
$this restrictOnUpdate()
$this cascadeOnDelete()
$this restrictOnDelete()
$this nullOnDelete()
$this noActionOnDelete()
ForeignKeyDefinition deferrable(bool $value = true)
ForeignKeyDefinition initiallyImmediate(bool $value = true)
ForeignKeyDefinition on(string $table)
ForeignKeyDefinition onDelete(string $action)
ForeignKeyDefinition onUpdate(string $action)
ForeignKeyDefinition references(string|array $columns)
```