Année Univ.

2020-21

Groupe:

Les Createurs

Rapport Final

Projet de Programmation
Rapport Final

BEHINAN Ophélia

DOGAN Ozgur

KULAN Onur Tan

PUJADE Joffrey

TISSERANT Tom

Fac de Science & Informatique L2

Université De Montpellier

Year 2020/21

Introduction

Dans le cadre du projet de S4, nous avions à faire un site de tournois sportif. Nous avons opté pour un site de tournois globalisé, qui n'est pas désigné pour un sport précis.

Le site compte, dans son fonctionnement, 5 rôle :

- Les administrateurs, qui ont à peu près tous les droits et, en particulier, donne le titre de gestionnaire aux utilisateurs de leur choix.
- Les gestionnaires, qui créent, organisent et gèrent les tournois.
- Les capitaines, qui créent les équipes et les pré-inscrivent aux tournois.
- Les joueurs, qui peuvent entrer dans une équipe.
- Les utilisateurs, qui n'ont que la possibilité de regarder l'avancée des tournois.

Les tournois sont visibles sur le site sous la forme d'une liste, où sont renseignés ; le nom de ce dernier, son lieu, sa date de début, sa durée en jours, le nombre d'équipes pouvant s'y inscrire et son statut temporel soit : en cours, à venir ou passé.
Lorsqu'une équipe veut participer à un tournoi, son capitaine doit la pré-inscrire, pré-inscription qui doit être validée par le gestionnaire du tournoi choisi.
Lors du déroulement du tournoi, il apparaît sous forme d'un arbre montrant les équipe et leur score pour les matches joués, scores qui peuvent être remplis à la main ou aléatoirement par le site.

Pour programmer ce site nous avons utilisé 5 languages, à savoir:

- SQL pour créer et communiquer avec la base de données.
- PHP principalement pour faire le lien entre le SQL et le HTML.
- HTML pour structurer les pages.
- CSS pour la présentation des pages.
- JavaScript pour essentiellement rendre dynamique les pages côté client.

Gestion de groupe

La gestion du groupe n'as pas été optimale dès le début. En effet, nous avons peu anticipé le volume d'un tel projet. A son amorce les réunions que nous faisions entre les membres du groupe étaient peu fréquentes. Plus nous avancions sur le projet et plus nous prenions conscience de ce qu'il nous restait à faire. Ainsi, nous avons naturellement pris la décision de réunir un peu plus souvent.

Compte-tenu du contexte actuel, nous n'étions pas dans la capacité de nous réunir physiquement. C'est pour cela que nous avons opté pour un travail a distance.

On a d'abord commencé par créer un serveur sur le service de discussion Discord pour communiquer plus facilement au sein du groupe. Salson, dans lequel, il nous était possible de communiquer de vives voix, par écrit et de diffuser notre écran au besoin.

Lors de nos réunions, nous avons pris l'habitude de rappeler ce qui avait été fait précédemment, nous anticipions ce qu'il restait à faire pour conclure le projet et nous choisissions ce sur quoi nous allions travailler durant cette réunion. Le but étant de se fixer des objetctifs à atteindre sur de courtes durées pour mieux structurer notre travail. Durant ces réunions tout le monde pouvait apporter ses idées, sa vision sur le comment les tâches allaient être réalisées et optimisées. Nous analysions tous les points de vue et choisissions tous ensemble comment procéder au mieux.

A la fin de chaque réunion, nous faisions des bilans qui prenaient en compte les avancées réalisées au cours de ces mêmes réunions et ce qu'il restait à faire. Ainsi nous pouvons nous répartir de petites tâches à faire individuellement pour les prochaines réunions. Le but étant de ne jamais laisser stagner le projet et qu'il soit en perpétuelle évolution.

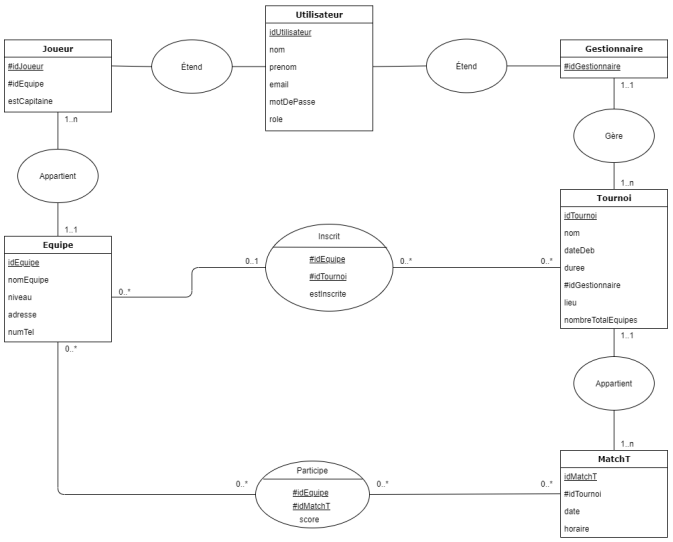
Environ au premier tiers du projet, nous avons créé un dépôt git afin de permettre une mise en commun des codes. Au départ nous déposions nos codes individuellement sur le git. Car il était possible de créer différentes branches où chacun pouvait mettre ses propres ce qu'il avait fait, sans écraser le travail des autres. Mais très vite nous nous sommes rendus compte que tout le monde travaillait que sur sa branche et donc sur des codes sources trop différents les uns des autres. Par conséquent il était compliqué d'avoir une complémentarité des codes les unes avec les autres. Il nous arrivait de réaliser des tâches en doublons ou d'avoir des difficultés

sur certaines parties, sans savoir qu'elles étaient déjà terminées par quelqu'un d'autre. Alors nous avons pris la décision de faire une réunion pour rassembler tout ce qui avait était fait séparément jusqu'à lors (et procéder à un énorme déboguage du à la mise en commun). Par la suite, la version principale et sans bugs du projet a été déposée sur la branche main (principale). Ainsi, lorsque nous avancions, nous prenions l'habitude de vérifier que les nouvelles modifacations apportées étaient sans bugs avant de les déposer sur la branche principale. De ce fait nous avions une base commune mieux exploitable.

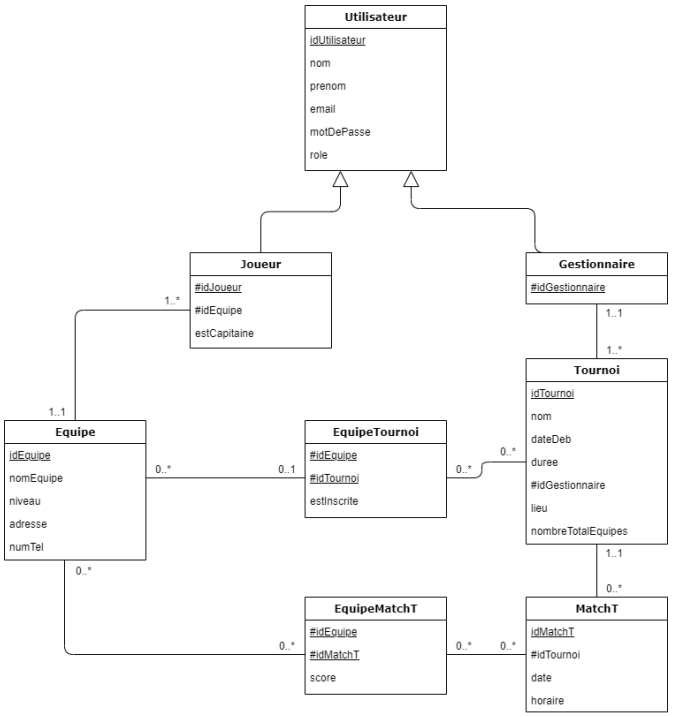
Finalement la façon dont nous avons travaillé a permis d'utiliser les compétences de chacun de telle sorte à pouvoir se compléter les uns les autres et tirer profit de tous nos points forts.

Base de données

Voici le schéma entités-relations de notre base de données :



Voici le schéma relationnel de notre base de données :



Le script de création de tables de la base de données a été écrit manuellement. On y retrouve donc des contraintes SQL spécifiques, permettant d'assurer l'intégrité des données. Par exemple, il y a des contraintes sur les clefs étrangères, qui doivent faire référence à des clefs primaires existantes d'autres pages. Il y a également, avec ces contraintes, la mention "ON UPDATE CASCADE" (contrainte supplémentaire modifiant une ligne d'une table lors de la modification de la ligne d'une autre table), ainsi que la mention "ON DELETE CASCADE" (contrainte supplémentaire supprimant les lignes dont les clefs étrangères font référence à une ligne en cours de suppression).

La table "Utilisateur" est une table essentielle pour la base de données. Elle définit un utilisateur du site, ainsi que le rôle en tant qu'utilisateur simple ou qu'administrateur. De cette table dépendent les deux tables "Joueur" et "Gestionnaire".

Pour ce qui est de "Gestionnaire", seul l'identifiant est stocké, faisant référence à l'identifiant d'un utilisateur.

Dans "Joueur", il y a l'identifiant du joueur référençant l'identifiant d'un utilisateur, ainsi que l'identifiant de l'équipe à laquelle il appartient. Le champ "estCapitaine" est un booléen désignant si le joueur est capitaine d'équipe.

Architecture

Nous n'avons pas utilisé d'architecture particulière pour notre projet, mais l'organisation de celui-ci peut rappeler l'architecture MVC.

En effet, l'architecture MVC se décompose comme suit :
- des **M**odèles : des fichiers dans lesquels le code interagit avec une table de la base de données ; ils font le pont entre la BDD et les contrôleurs ;
- des **V**ues : des fichiers n'ayant pour unique but que la présentation de données et l'affichage du site ; c'est l'interface graphique (ou IHM) et n'interagissent qu'avec les contrôleurs ;
- des **C**ontrôleurs : ce sont des fichiers reliant les modèles et les vues ; ils s'occupent typiquement de tous les traitements algorithmiques.

Notre projet est organisé de manière similaire. Sauf que la plupart des traitements algorithmiques sont réalisés en en-tête et uniquement en en-tête des fichiers qui font office de vues.

Voici, en quelques lignes, quelle est l'organisation de notre projet :
- un répertoire **BDD**, contenant le script de création de tables et le jeu de données de tests de notre site, ainsi que différents fichiers PHP interagissant chacun avec une table de la base de données ;
- un répertoire **css**, contenant toutes les feuilles de style du site ;
- un répertoire **img**, contenant toutes les images du site ;
- un répertoire **js**, contenant les fichiers contenant le code JavaScript nécessaire au site ;
- un répertoire **module**, contenant essentiellement des classes représentant des données de BDD, ainsi qu'un fichier de fonctions générales et une classe représentant un Tas Max nous aidant à gérer les tournois ;
- un répertoire **php**, contenant tous les fichiers PHP assimilés à des vues, qui possèdent, pour la plupart, des traitements algorithmiques dans les en-têtes des fichiers (i.e. placés avant le début du code HTML).

Particularités techniques

Le code PHP régit le fonctionnement du site. Nous l'avons utilisé pour écrire toutes nos fonctions, toutes les classes, manipuler des variables. Ce langage nous a permis de mettre en place une fonctionnalité particulière, à savoir, afficher l'arbre de tournoi de manière dynamique. Dans un premier temps, nous avons essayé de nous documenter afin de trouver des codes en JavaScript ou JQuery qui auraient pu remplir cette tâche. Mais nous n'avons trouvé aucune solution qui nous auraient permis de générer un arbre quelque soit le nombre d'équipes.Puisque c'est le nombre d'équipes participantes qui détermine le nombre de noeuds. Donc, pour afficher l'arbre, on a allié l'utilisation du PHP, HTML et CSS. L'arbre s'affiche toujours dans son entièreté, cependant, seuls les noeuds contenant des matchs programmés ou terminés sont affichés. Pour le savoir on doit se baser sur l'avancement du TasMax (classe qui gère le déroulement d'un tournoi). On doit aussi avoir connaissance du nombre de tours et du nombre de matchs. Ces derniers se calculent en fonction du nombre d'équipes.
Pour ce qui est de l'affichage, on a défini des tailles fixes pour les noeuds. Cela pose certains problèmes si on a trop d'équipes qui participent au tournoi. Au delà de 32 équipes ce n'est plus optimal car l'arbre sort de la zone d'affichage.
Nous avons créé différentes variables associées aux marges et espacements qui évoluent dans des boucles "for". Étant donné que l'arbre est affiché "horizontalement" avec la racine à droite, à chaque niveau de noeuds correspondants aux tours du tournoi, la mise en page doit changer.

JQuery est une bibliothèque JavaScript libre et multiplateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML des pages web. Nous avons utilisé jquery et datetimepicker pour notre page "SaisieDateTournoi.php" (page de saisie de date de tournoi). C'est une page qui permet aux gestionnaires d'entrer les dates des matchs de tournois à venir dont dont ils sont responsables. Dates que les utilisateurs peuvent consulter en amont du dit tournoi.Nous avons deux attributs dateMin et dateMax, correspondant aux dates de début et de fin de tour. Nous avons préféré le jQuery, car bien que le PHP nous aurait permis de réaliser cela de manière plus "native", jQuery permet un affichage ainsi qu'un choix dynamique. Également, cette technologie permet le choix du format de date (et d'horaire), afin de mieux

gérer l'insertion en base de données, et gérer le changement de date si on a un choix d'horaire à minuit.

En utilisant notre datetimepicker dans deux boucle "for", la première pour le nombre de tours et la deuxième pour les matchs du tour courant, on a stocké les données des dates et horaires dans le tableau datetimepicher[nombreDeTour][NombreDeMatchDansCeTour]. Le format de date et horaire choisi est de la forme "format:'Y-m-d H:i', avec un espace entre la date et l'horaire. Nous récupérons les données saisies par le gestionnaire via le tableau POST (créé avec la méthode "POST" d'un formulaire HTML). Les données de dates et horaires dans le tableau sont récupérables avec des clefs de la forme POST['datetimepicker(idMatch)(idTournoi)']. Ces dernières sont séparées via la fonction "explode", et l'insertion peut se faire plus facilement.

Par la suite, nous avons décidé de créer des classes dans le but de faciliter la manipulation des données récupérées dans la base. La plupart des classes créées sont donc des "conteneurs" de données de la BDD.
Néanmoins, nous avons une classe "TasMax", que nous avons créé pour faciliter la gestion des tournois et qui, elle, a été réalisée avec une meilleure approche de l'orienté objet.

Autre particularité utilisée pour notre site : le hachage du mot de passe.
Il y a une chose à savoir concernant le hachage ainsi que la différence avec le cryptage : le cryptage est une notion mathématique et informatique visant à "encoder" des données (afin de les cacher ou les sécuriser), afin de les décoder. En d'autre termes, tout ce qui est crypté est destiné à être décrypté pour les personnes à qui les données sont destinées.
Le hachage, quant à lui, est une notion visant à transformer un texte qui ne peut pas être déchiffré par quelque algorithme que ce soit. Le hachage est donc une transformation définitive.
Nous avons choisi de hacher les mots de passes des utilisateurs présents dans la base de

données, via l'algorithme SHA-256. Cela permet de conserver les mots de passe en base de données, mais si elle venait à être piratée, ces mêmes mots de passe ne pourraient pas être déchiffrés.

Afin d'authentifier un utilisateur qui souhaite se connecter, nous n'avons qu'à hacher le mot de passe saisi avant de le comparer avec celui enregistré en base de données.

Conclusion

Finalement, tout au long de notre projet, nous avons été amenés a rencontrer des difficultés. Certaines d'entre elles ont trouvé des solutions d'autres non. Notamment à la fin de la conception de la base de données, lorsqu'on a commencé à implémenter des fonctionnalités telles que l'inscription, certains problèmes se sont posés. En effet, certaines insertions ne fonctionnaient pas car elles nécessitaient l'existence de clés primaires qui n'existaient pas. Pour résoudre ces problèmes on a du créer des tables d'associations reliant la table initiale à la table contenant la clé primaire manquante. De ce fait, nous pouvions procéder à l'insertion sans la clé primaire manquante.
Par la suite, nous avons aussi eut quelques problèmes d'affichage avec l'ordre des valeurs renvoyées par la base des données. En effet, selon sa clé primaire, les tuples d'une table vont être ordonnées différemment. Il nous aura fallu traiter les données renvoyées en les réordonnant pour bien les exploiter.

Nous avons aussi pris du retard sur des problèmes qu'il fallait déboguer car il était nécessaire de les résoudre pour continuer à avancer sur le projet. Mais à force de persévérance nous avons réussi à surmonter ces difficultés.

Malgré la tâche complexe qu'a été la réalisation du projet, nous avons tous appris et gagné en expérience tout du long.

- Nous avons pu nous faire une idée plus précise de ce qu'était le développement Web et de l'organisation méthodique que cela implique.
- Nous avons pu mettre en application nos connaissances sur les bases de données au travers de la gestion de données de notre site.
- Nous avons su nous adapter aux uns et aux autres afin de tirer le maximum de chacun .
- Nous avons pu avoir une approche plus approfondie du logiciel de "versionning" GitHub.

Pour conclure, ce projet nous aura permis à tous de mieux cerner nos affinités informatiques.