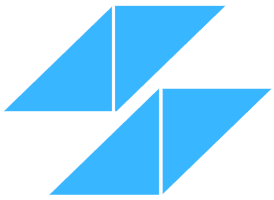

MANAGEUR DE TOURNOI

Projet de Programmation
HLIN405

BOURRET Maxime - COSSU Arnaud - HADDAD Gatien - SAID Adam
L2 Informatique

tournoi.adam-net.fr

Table des matières

1 Introduction

Le but de ce projet est de réaliser un site permettant de créer et gérer des tournois avec différents rôles selon les utilisateurs. Nous avons donc développé cette plateforme sur le site tournoi.adam-net.fr qui comprend la création, la gestion et l'affichage de tournois personnalisés avec des équipes.

Si vous voulez accéder aux fichiers du site, vous pouvez vous rendre à l'adresse tournoi.adam-net.fr/files. Pour accéder à la base de données, rendez-vous sur la page tournoi.adam-net.fr/database et connectez vous avec ***sc1samo7154_viewer*** en identifiant et ***hlin405viewer*** en mot de passe.

2 Gestion du groupe

Dans notre groupe, nous avions tous accès aux mêmes ressources, ainsi nous pouvions tous modifier les mêmes choses. Cependant, d'une manière assez naturelle, Gatien et Adam se sont plus occupés du `back-end` avec le `PHP` et le `SQL` alors qu'Arnaud et Maxime ont plus travaillé sur le `front-end` avec le `HTML` et le `CSS`. Tout le monde apportait sa pièce pour chaque partie du site et les décisions se prenaient à quatre de manière à avoir un avis unanime sur tous les défis à relever. Pour ce qui est du planning, nous avions toutes les deux semaines une visioconférence avec notre enseignant référent M. Grenet durant lesquelles nous décrivions ce que nous avions fait et planifions les tâches à réaliser pour les 14 prochains jours. Afin de communiquer entre nous, nous avons utilisé un serveur Discord pour discuter par message, s'appeler et s'envoyer les fichiers. Pour ce qui est du code, `repl.it` était utilisé comme brouillon lorsque nous modifions une partie du site avant de l'appliquer à la version originale sur `Visual Studio Code` à l'aide d'un protocole `ftp` qui permettait l'accès au fichiers du serveur. Depuis le début du projet, le site est hébergé chez un hébergeur mutualisé et possède un nom de domaine ce qui nous permet d'y accéder de partout, n'importe quand et de pouvoir le modifier en temps réel sans avoir à lancer de serveur `WAMP`. Cela nous permettait également de nous connecter en `ftp` et d'accéder au code pour l'éditer une fois que nous avions fini nos tests sur `replit`. Enfin, nous avons utilisé `Filezilla` pour effectuer des sauvegardes du site, gérer l'arborescence ainsi qu'importer des fichiers media (images).

3 Base de données

Concernant la base de données, nous avons un total de 8 tables de données interconnectées, ainsi qu'une supplémentaire pour stocker les informations de connexion des utilisateurs. Dans un premier temps, la table `User` contient les données des comptes de tous les utilisateurs inscrits : pseudo, nom, prénom, email, mot de passe chiffré et rôle sur le site. Vous pouvez en voir un schéma ci-après (Voir Figure 1).

Viens ensuite la partie concernant les équipes : la table `teams` contient le nom de l'équipe, son niveau, un moyen de contact (numéro de téléphone ou email), le nombre de joueurs ainsi que l'`id` du capitaine, ce dernier étant le compte capitaine qui l'a créée. On peut ajouter des joueurs à l'équipe, ces derniers étant stockés dans la table `players`, contenant leur nom, prénom, numéro de joueur et l'équipe à laquelle ils sont attachés. Pour gérer les tournois, la table `tournament` stocke toutes les informations nécessaires : date de création, manageur, nom du tournoi, date de début, durée, lieu, nombre d'équipes maximum, équipes inscrites et la progression du tournoi, cette dernière indiquant si le tournoi n'est pas encore ouvert, si les matchs ont déjà débuté ou s'il est terminé.

Pour inscrire une équipe à un tournoi, cette dernière sera d'abord stockée dans la table `waiting`, en attente d'acceptation du manageur du tournoi avant d'être intégrée ou non dans la table `teamtournament`, table permettant de trier plus facilement toutes les équipes affiliées à un tournoi, étant donné qu'une équipe peut être inscrite à plusieurs tournois différents.

Pour finir, la table `games` contient tous les matchs qui ont eu lieu, avec le score des deux équipes, le vainqueur, le numéro du match et le nom du tournoi auquel le match appartient.

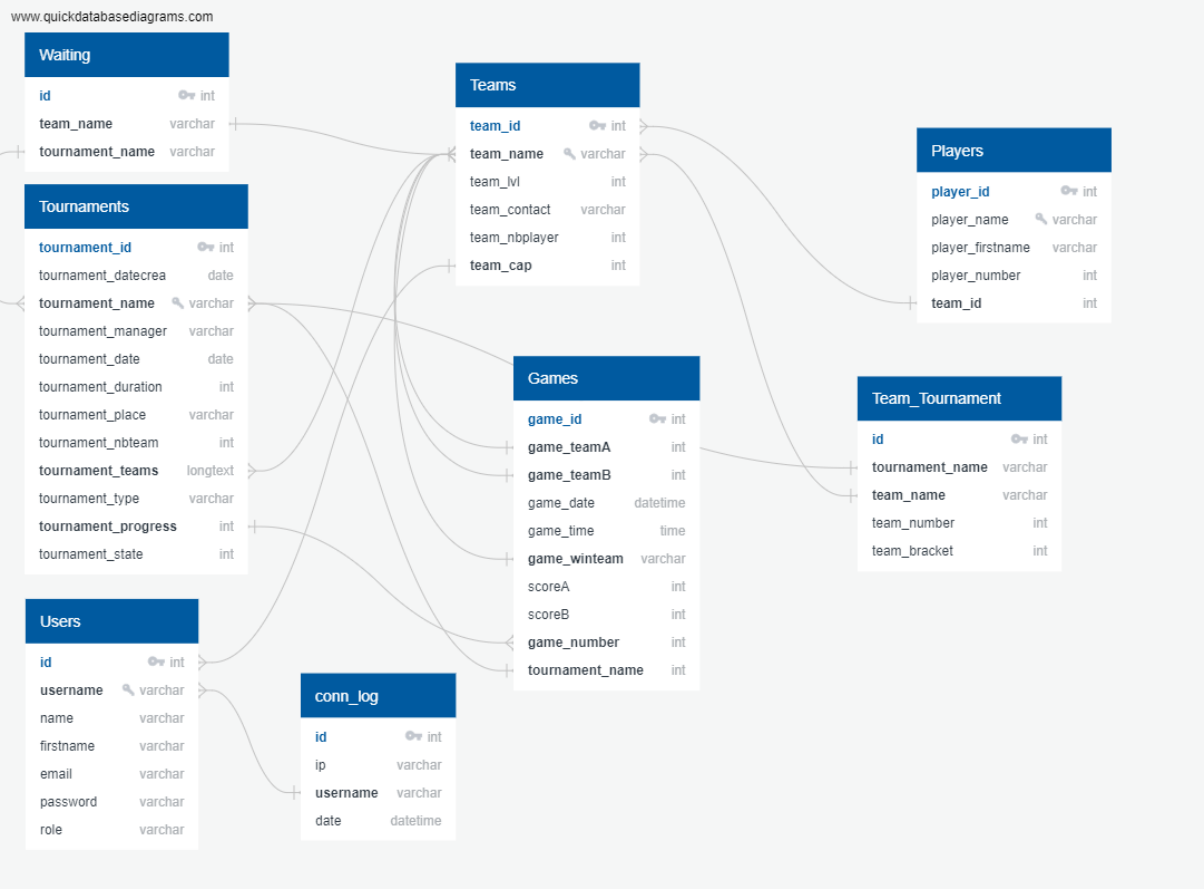

FIGURE 1 – Schéma de la base de données

4 Architecture du site

Notre site est composé de deux pages principales qui contiennent des redirections sur d'autres pages secondaires. Pour comprendre le cheminement du site vous pouvez trouver en page 6 une vue graphique du fil d'Ariane du site (Figure 2). La première, qui est la page d'accueil, contient l'affichage des tournois (on peut cliquer sur des boutons pour afficher ceux en cours, futurs ou terminés), un carrousel d'images dans un style de publicité et un menu déroulant comportant les différents boutons d'authentification. Dans l'en-tête, on peut accéder aux pages de connexion `Se connecter` et `S'inscrire` ainsi que la page `Mon compte` accessible seulement après que l'utilisateur se soit connecté. Dans le pied-de-page, deux boutons nous redirigent aux pages `Nous contacter` qui permet d'envoyer un message de demande de support au site et `Termes et Conditions` qui fournit les CGU du site et quelques règles.

La page de connexion est très simple et contient uniquement le formulaire de connexion au site. La page d'inscription contient un formulaire plus conséquent pour se créer un compte, c'est ici que l'utilisateur va choisir son rôle sur le site (les rôles seront détaillés par la suite).

Une fois connecté, l'utilisateur a accès à la page `Mon compte` qui va lui permettre d'être redirigé sur les différentes pages de gestion de tournoi. En fonction du rôle connecté, les boutons visibles varient. Il existe quatre rôles différents dont trois sélectionnables lors de la création d'un compte :

— Capitaine : Il peut créer des équipes et demander l'inscription de ces dernières à un tournoi. Il peut également créer des joueurs et les affecter à une équipe. Il a accès aux boutons `Créer une équipe` et `S'inscrire à un tournoi`. Pour la page de création d'équipe, le capitaine renseigne toutes les informations nécessaires dans le formulaire en `HTML` qui vont être envoyées dans le fichier `PHP` qui va inscrire ces informations dans la base de données en précisant le capitaine qui a créé l'équipe. Sur la même page, il peut créer des joueurs et les affecter à l'une de ses équipes. Pour l'inscription d'une équipe à un tournoi, il suffit au capitaine de sélectionner le tournoi ainsi qu'une de ses équipes dans les listes. Ici, il a fallu faire des vérifications afin que seul le capitaine de l'équipe puisse l'inscrire. Une fois fait, si le tournoi n'est pas encore plein cela envoie une demande d'inscription au manageur du tournoi associé en inscrivant dans une table le nom de l'équipe et le nom du tournoi.

— Manager : Il s'agit du rôle avec le plus de fonctions, il a accès au bouton `Gérer un tournoi` qui lui permet de retrouver tous les tournois dont il est le manageur et seulement ceux là. Il a fallu ici aussi faire des vérifications pour n'afficher que les événements correspondants. Pour chacun d'eux, il peut : accepter ou refuser les demandes d'inscription faites par les capitaines, démarrer le tournoi, modifier les équipes inscrites et définir l'ordre des matchs en ajoutant à chaque équipe un numéro, ainsi les premiers matchs opposeront les équipes 1 et 2, 3 et 4... Il peut aussi rentrer les scores des matchs et terminer un tournoi. Pour chacune de ces actions, il peut choisir des informations aléatoires s'il ne veut pas attendre : équipes aléatoires, adversaires aléatoires, scores aléatoires.

— Administrateur : Son rôle n'est utile qu'au début car c'est lui qui s'occupe de créer des tournois. Il a donc accès au bouton `Créer un événement` où il précise toutes les informations relatives à l'évènement (nom, date, lieu, nombre d'équipes...) auquel il affectera un manageur référent.

— Modérateur : Seul rôle qui doit être assigné via la base de données, il a accès à tous les onglets et peut modifier ce qu'il veut, il combine les trois rôles ci-dessus avec l'avantage de pouvoir accéder à toutes les informations.

Tous les rôles ont aussi accès, sur la page `Mon Compte`, au bouton `Voir les tournois` qui les redirige sur une page qui affiche les tournois en cours, à venir et terminés sous forme de tableau comme sur la page d'accueil mais de manière plus détaillée et qui permet également l'affichage d'un tournoi sous forme graphique avec des crochets. Enfin, il y a la page d'`erreur 404` qui est affichée en cas de mauvaise adresse `URL` fournie.

On peut également retrouver les pages `database` et `files` accessibles uniquement en mettant l'`URL` exacte. En arrivant sur ces pages, on retrouve respectivement, un formulaire de connexion pour ensuite accéder à l'interface de gestion `PhpMyAdmin` de la base de données du site ainsi que les fichiers du site listés en consultables.

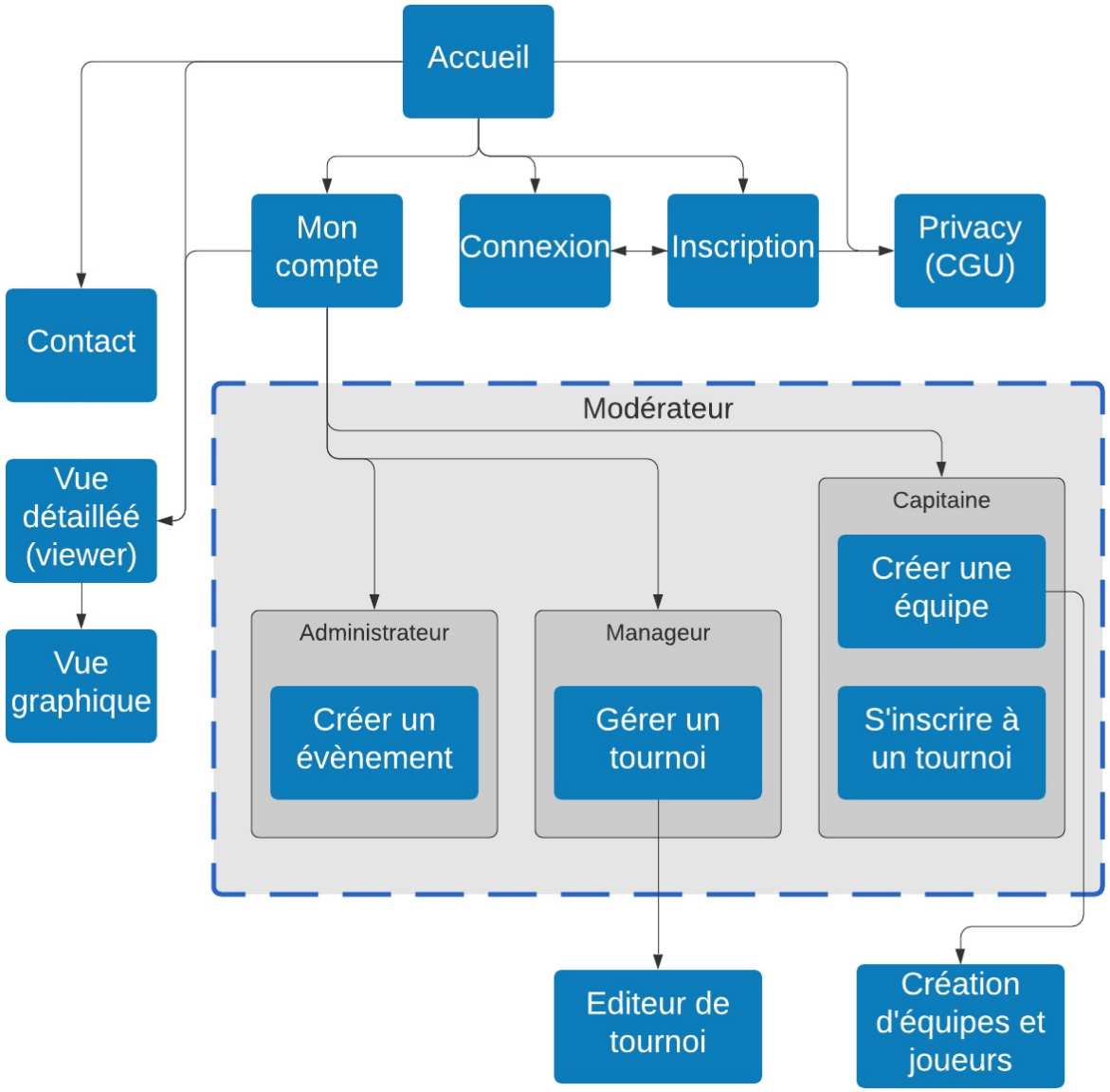

FIGURE 2 – Vue graphique du fil d'Ariane du site

5 Particularités techniques

5.1 Arborescence

Pour ce qui est de l'arborescence du code, à la racine on retrouve les fichiers pour la page d'accueil (`index.php`, `index.css`) et la page d'`erreur 404` (`404.php`, `404.css`, `script.js`). Ensuite, les différentes pages du site se trouvent dans des dossiers, chaque dossier est composé d'un fichier `index` en `HTML` ou `PHP` pour le contenu du site, un `CSS` pour la mise en forme et un fichier `server.php` pour l'envoi et la réception des données avec la base de données (Vous pouvez trouver un aperçu de l'arborescence en fin de page (Figure 3) :

- `contact` contient la page `Nous contacter` ;
- `database` stocke l'interface phpmyadmin pour la gestion de la base de données ;
- `display` contient la page qui affiche un tournoi sous forme graphique ;
- `editor` contient la page où le manageur édite un tournoi ;
- `files` contient la page qui permet de visualiser les fichiers du site ;
- `media` stocke toutes les images et fichiers complémentaires nécessaires ;
- `mytournaments` contient la page où le manageur voit ses tournois et peut les gérer ;
- `privacy` contient la page `Termes et Conditions` ;
- `registration` contient les pages `Login`, `Register` et `Mon compte` ;
- `teams` contient la page qui permet de créer une équipe et des joueurs ;
- `teamsignup` contient la page qui permet d'inscrire une équipe à un tournoi ;
- `tournoi` contient la page qui permet de créer un tournoi ;
- `viewer` contient la page qui affiche en vue détaillée tous les tournois ;
- enfin, les fichiers et dossiers commençant par un point servent à configurer le serveur, le protocole `ftp` et les redirections. Par exemple, `.htaccess` gère les redirections `https` et `404`.

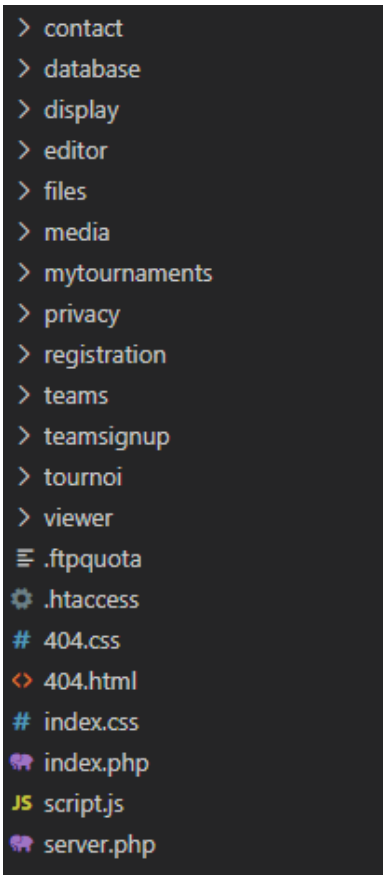

Figure 3 – Fichiers du site dans Visual Studio Code

5.2 Front-End

5.2.1 HTML

Pour ce qui est du `HTML`, nous nous en sommes servi de squelette pour notre site en utilisant les bases du langage. La plupart des pages contiennent des fonctions `PHP`, c'est pourquoi il y a très peu de `.html` dans les fichiers du site. Nous l'avons utilisé surtout pour l'architecture d'une page avec les métadonnées, le `header`, le `footer`, toute la gestion des boutons et des images.

5.2.2 CSS

Pour la partie `CSS` du site, nous avons récemment décidé de changer la palette de couleurs afin d'améliorer le rendu du site et de faciliter la consultation de certaines informations. Nous avons réalisé cette palette sur le site coolors.co. Nous avons donc obtenu la palette suivante :

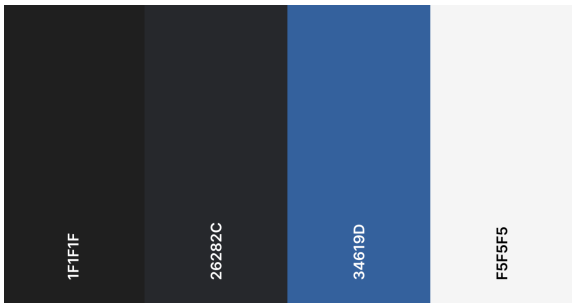

FIGURE 4 – Palette des couleurs utilisées

Une fois la nouvelle palette trouvée, nous avons commencé à changer le `CSS` de chaque page et nous avons aussi appliqué de nouveaux choix concernant le style de certains éléments. Nous avons changé la couleur des boutons du `header`, celui-ci étant devenu plus foncé avec les nouvelles couleurs, nous les avons mis en blanc, avec `hover` sur fond blanc et écriture noire, donc un inversement des couleurs. Nous avons fait de même dans le `footer` qui est lui présent uniquement sur la page d'accueil du site.

La structure est la suivante :

— boutons en blancs avec hover en blanc ;
— body en noir (teinte plus claire) ;
— header, footer et divs en bleu foncé ;
— autres boutons, liens, hovers en bleu plus clair.

Les boutons ressortent donc du header et sont beaucoup plus visibles, de même pour les hovers et les liens des autres boutons qui sont bleu clair.

Nous devions afficher plusieurs `divs`, dans un `container` intitulé `Vos tournois` dans l'espace `Gérer les tournois` de `Mon compte`. Ainsi nous avons décidé de partir sur deux gros conteneurs, en largeur, qui font 80% de la page et avec des bords arrondis pour respecter le style du site. Le système de `grid` précédemment utilisé posait quelques problèmes et avait du mal à s'adapter aux différents conteneurs (sûrement dû à des conflits de `CSS`). Nous avons donc décidé de revenir sur un système de `divs` normales, en `inline-block` et aligné au centre. Nous pouvons avoir 6 petites `divs` contenues dans ce même gros conteneur et grâce à la propriété `display flex`, les `divs` supplémentaires s'ajouteraient en restant dans le même conteneur simplement en l'agrandissant. Nous avons réitéré le même procédé pour le deuxième conteneur qui contient les demandes de tournois.

Pour le carrousel, nous avons choisi de réaliser ce dernier seulement en `CSS` et `HTML` pur. Nous avons commencé par réaliser 3 fausses publicités, images faisant 728 pixels de large par 90 pixels de haut. Nous les avons ensuite affichées sur le site en créant 3 `divs`, un conteneur contenant une autre `div`, contenant elle-même les `slides` (pubs). Pour la création, nous avons simplement fait une `div` de la longueur d'une image (le conteneur), soit 728 pixels de large, pour ensuite ajouter l'attribut `hidden overflow`, qui a permis de cacher les deux autres images. Ainsi, pour l'instant, nous avons les 3 images collées côtes à côtes sur le site dans une `div` appelée `slides` et qui fait en largeur trois fois 728 pixels, avec les deux plus à droite cachées. Nous avons aussi ajouté l'attribut `float left` dans la `div` la plus imbriquée qui représente chaque pub pour que les images s'alignent sur la même ligne.

Dans la `div` slides qui contient donc les 3 pubs, nous avons mis une animation nommée "glisse" de 20 secondes, qui suffira pour 3 images avec un attribut `infinite` pour qu'elle tourne en continu. Nous avons ensuite ajouté un `keyframes` nommé "glisse" qui servira d'animation et qui fera des translations sur l'axe X. Nous lui mettons les valeurs suivantes : pour 0% il ne bouge pas, nous avons notre première image. Pour 33% soit un tiers, il passe à la deuxième image donc il fait -728 pixels. Pour 66%, il passe à la troisième en faisant -1456 pixels. Pour finir à 100%, il n'a aucune valeur et ainsi retourne à sa position initiale donc à la première image.

Pour le `header`, nous avons décidé de faire une `div` sans bords arrondis, comportant le nom du site, ainsi qu'un bouton qui permet de nous connecter ou de nous inscrire. Pour le nom, nous avons simplement mis un `hover` avec effet et une bordure avec coins arrondis. Pour le bouton, nous avons fait un hover dépliant qui laisse donc place aux deux boutons, se connecter, et s'inscrire. Nous avons choisi un `header` simple, qui ne comporte pas trop d'options afin que la navigation reste claire pour l'utilisateur. Pour le `footer`, là aussi nous avons opté pour la simplicité, avec deux boutons et leurs `hovers`, permettant de se diriger soit vers une page de contact, soit vers la page contenant les conditions d'utilisation.

Pour le `sliders cards` (boites noires au-dessous des fausses publicités), nous avons simplement fait un conteneur qui contient quatre `divs` qui comprennent un titre, ainsi qu'une `div` contenant elle-même 3 `divs` et une image. Les trois `divs` sont une barre de progression vide, une barre de progression superposée qui fera effet de remplissage en bleu et une `div` contenant l'image. Nous avons fixé la taille du conteneur à 300 pixels de hauteur et 600 pixels de largeur.

Pour les quatre "cartes", elles font 280 pixels de hauteur pour 200 pixels de largeur. Nous avons appliqué à ces cartes un effet d'ombre et des coins arrondis, ainsi qu'une transition de 0.4 secondes. Si la souris passe au-dessus de la carte, cette dernière fait une translation sur l'axe de -20 pixels en 0.4 secondes, ce qui donne l'effet de carte qui sort du groupe. Nous avons ensuite placé les éléments dans leurs `divs`, notamment les icônes et les barres de progression. Nous avons mis une transition de remplissage sur les barres, ainsi les animations se coordonnent entre elles. Pour les icônes qui étaient de base en noir, nous avons utilisé `filter invert` qui a inversé le noir et le blanc.

5.2.3 JavaScript

Nous avons également utilisé un peu de `javascript` dans le développement de notre site. Tout d'abord nous avons dû en inclure lors de l'inscription des utilisateurs en créant une fonction qui permettait de vérifier que les deux mots de passe entrés correspondaient en récupérant les deux champs de texte et en les comparant. Une grosse fonctionnalité du site a été faite en `javascript`, il s'agit des différents boutons aléatoires lors de l'édition des tournois qui permettent de remplir le tournoi et de rentrer les scores de manière aléatoire. Cela a été réalisé grâce à des boucles `for` et `while` qui vont vérifier le nombre d'équipes déjà inscrites et générer des noms aléatoires qui vont être rajoutés dans la table des équipes inscrites.

Des petites fonctions ont également été rajoutées dans la page de création des tournois, afin de bloquer la possibilité de créer des tournois (date de début postérieure à la date du jour, bloquer le nombre de places pour les équipes à des puissances de 2). D'autres scripts on été rajoutés afin d'utiliser des `plugins javascript`. On en retrouve un sur la page de création de tournoi pour afficher la carte avec `Leaflet` et un autre sur la page `gérer un évènement` qui permet d'afficher les tournois sous forme graphique avec le plugin `Jquery-Bracket`.

5.3 Back-End

5.3.1 Php

Dans un premier temps, il est tout de même important de préciser que nous avons fait les requêtes `SQL` et les connexions à la base de données en `PHP`, notamment grâce à une interface en `PHP Data Objects` (`PDO`), ce qui constitue une partie conséquente du trafic sur le serveur. De plus, le langage `PHP` nous a permis de créer plusieurs fonctionnalités importantes du site, notamment les fonctions aléatoires, associées à la création d'équipes (pour remplir un tournoi en générant des équipes), ainsi que pour le déroulement complet du tournoi. Ces deux fonctionnalités ayant pour but premier la présentation, elles se sont avérées être également utiles pour deboguer le site ou pour tester des fonctionnalités sans avoir à faire un grand nombre de manipulations. Le `PHP` a également permis de faire certaines vérifications côté serveur, par exemple pour vérifier quand un tournoi va débuter ou bien pour éviter que deux équipes portent le même nom.

5.3.2 SQL

L'accès à la base de données étant primordial pour stocker les informations nécessaires au bon fonctionnement du site, les requêtes `SQL` sont présentes dans la quasi totalité des pages :

- lors de l'inscription, pour stocker les informations du compte,
- lors de la création d'une équipe, ainsi que pour tous les joueurs qui y sont assignés,
- pour l'affichage des tournois, une requête permet d'aller chercher l'intégralité des tournois selon leur stade d'avancement (passés/en cours/futurs),
- pendant l'inscription à un tournoi pour enregistrer cette dernière,
- et pendant le déroulement de ce dernier, pour stocker les matchs, les scores, l'avancement du tournois, etc...

5.4 Hébergement

Comme nous l'avons dit au début du rapport, nous avons décidé d'héberger notre site chez un hébergeur mutualisé chez qui l'un de nous avait déjà une offre d'hébergement et un domaine. Cela ne nous a donc pas rajouté de frais supplémentaires. Cette configuration avait de nombreux avantages comparé à d'autres solutions telles que les serveurs `LAMP` avec un `git`. Tout d'abord, cela nous permettait de pouvoir travailler directement sur les fichiers et que tout le monde puisse voir le résultat final instantanément en allant sur le site et ce depuis n'importe quel appareil. Cela nous évitait donc de devoir faire des `git commit` et de démarrer des serveurs locaux à chaque fois. Cette solution nous a également permis un accès à plusieurs outils que ne pourrait pas offrir un serveur local notamment la génération de certificat `SSL` afin d'avoir un accès sécurisé en `HTTPS`, la création et gestion de boîtes mails pour pouvoir proposer un système d'envoi de mails sur notre site, une installation rapide de l'interface de `PhpMyAdmin` et de la base de données ; ainsi que la création d'accès personnalisés, les accès `FTP` aux fichiers personnalisés et sécurisés, la possibilité de créer des dossiers de pré-production et des sauvegardes du sites, une gestion des redirections simplifiées et un suivi de l'audience de notre site. Tout ceci grâce à l'interface d'administration `cPanel` du domaine.

6 Conclusion

6.1 Synthèse

Durant le développement du site nous avons fait face à différents imprévus et complications sur lesquels nous avons passé plus de temps mais avons réussi à les surmonter.

Tout d'abord, nous avons été confrontés au problème "Comment coder ensemble et éditer en même temps le site internet". Pour pallier ce dernier, nous avons décidé d'utiliser 3 environnements pour communiquer et éditer le code. Nous utilisions Discord pour parler entre nous. Lorsque quelqu'un modifiait un fichier, il envoyait un message pour prévenir et spécifier l'endroit où les autres ne devaient pas aller avant qu'il ait fini. Si on voulait essayer de modifier quelque chose, on dupliquait la page sur un `replit` de préproduction, on testait nos modifications avant de les appliquer au site afin d'éviter les erreurs et de pouvoir y coder à plusieurs. Nous avons utilisé `Visual Studio Code` et un `plugin` permettant l'accès au site via un protocole `ftp`, car `replit` ne supporte pas le `PHP` et la gestion d'une base de données.

Ensuite, nous avons rencontré des difficultés pour afficher une page `Mon compte` différente selon le rôle de l'utilisateur connecté. Afin d'obtenir une page dynamique fonctionnelle, nous avons dû ajouter certains paramètres lors de l'authentification, notamment la récupération du rôle de l'utilisateur qui se connecte en créant une variable de session (`session_start` en `PHP`) contenant les identifiants de l'utilisateur (pseudo, rôle, email). Une fois sur la page `Mon compte` on cherche à récupérer le rôle de l'utilisateur : s'il n'y en a pas alors l'utilisateur n'est pas connecté, il est alors redirigé sur la page de connexion, sinon on retrouvera une condition qui, si vérifiée, affichera une certaine `div` contenant les fonctionnalités propre à l'utilisateur.

Enfin, des complications sont survenues lorsqu'il a fallu trouver un moyen d'implémenter un tournoi en base de données. Notre problème principal était de faire avancer le tournoi selon les résultats des matchs. Nous avons au départ pensé à déduire le match actuel en fonction des précédents, mais cette option nous obligeait à faire de nombreuses requêtes et nous avons pensé que ce n'était pas compatible avec de futurs ajouts (par exemple si on choisit un nombre d'équipes qui n'est pas une puissance de deux).
Nous avons finalement décidé de stocker l'avancée du tournoi en base de données, avec une colonne `tournamentprogress`. Cette dernière garde en mémoire l'avancée du tournoi et plus précisement le numéro du match actuel, ce qui nous permet également de récupérer les équipes qui devaient s'affronter. Par exemple, avec un tournoi contenant 4 équipes, il y aura trois matchs, dont deux en demi-finale et un en finale : le match 1 opposera l'équipe 1 et 2, le match 2 opposera l'équipe 3 et 4, et le troisième match opposera les vainqueurs des matchs précédents.

Toutes les difficultés auxquelles nous avons été confrontés ont été surmontées. Nous avons toujours trouvé une solution afin d'avancer dans notre projet. Pour ce qui est du problème de reconnaissance du rôle de l'utilisateur ou pour l'implémentation d'un tournoi, avec de la persévérance et des recherches nous avons pu arriver à nos fins. Pour ce qui est de l'édition collaborative, nous avons mis en place une communication plus claire et l'utilisation de différents outils informatiques nous a permis de réussir à articuler le tout.

En conclusion, ce projet nous a permis d'apprendre énormément de nouvelles notions, d'utiliser nos connaissances dans certains domaines et d'améliorer nos capacités à travailler en groupe. Cela nous a permis de développer nos compétences en web (par exemple la gestion d'utilisateurs et d'affichage en `PHP`) et l'application de nos connaissances du semestre 3 (par exemple le développement de la base de données en `SQL`). Nous avons appris à nous organiser sur le développement d'un site internet, à planifier le déroulement du projet en fonction des tâches à réaliser et à communiquer de manière plus claire et compréhensible afin de tous se comprendre.

6.2 Sources

Logo Faculté des Sciences : [En ligne] Grand (2392*2950).png
sciences.edu.umontpellier.fr

Logo Université Montpellier : [En ligne] LOGO-original-RVB-WEB-1.png
umontpellier.fr

Logo titre Manageur de Tournoi : logo.png - Créé sur Canva
canva.com

Figure 1 - Schéma de la base de données : bdd.png - Généré depuis QuickDBD
quickdatabasediagrams.com

Figure 2 - Vue graphique du fil d'Ariane du site : ariane.png - Créé sur Lucidchart
lucid.app

Figure 3 - Fichiers du site dans Visual Studio Code : arboresence.png - capture d'écran de l'arborescence du site dans Visual Studio Code

Figure 4 - Palette des couleurs utilisées : couleurs.png - export de la palette depuis Coolors.co
Coolors.co