



# Meerketball

VISER PLUS HAUT

**HLIN405: Projet de programmation**

*Site de gestion de tournois de basketball*

Cédric Cahuzac, Charlotte Fabre, Robin L'Huillier, Puccio Lorenzo, Nicolas Ruiz

Ce rapport décrit le déroulement de notre projet de programmation web, ainsi que son fonctionnement et ses spécificités. Ce projet avait pour objectif de nous permettre d'utiliser les connaissances et aptitudes que nous avons acquises depuis le début de nos études d'informatique et plus particulièrement dans le cadre des cours d'HLIN304, HLIN405 et HLIN408.

Pour rappeler brièvement le sujet, il s'agissait de développer un site de gestion de tournoi et sa base de donnée associée. Nous avons donc dû, dans un premier temps, choisir sur quoi porteraient les tournois gérés par notre site. Nous nous sommes mis d'accord sur le basketball pour plusieurs raisons. D'abord, un tournoi sportif (plutôt que festif) semblait plus approprié au travail que nous voulions produire : des règles plus strictes nous permettaient de commencer avec un cadre clair. Parmi les nombreux sports, nous avons choisi le basketball car c'est un sport d'une part assez populaire pour donner un intérêt à notre site, d'autre part assez intéressant pour nous permettre de laisser libre cours à notre volonté de rajouter des fonctionnalités. Ainsi, le basketball peut être joué dans plusieurs types de tournois, et a un nombre de joueur par équipe assez limité pour ne pas nous freiner dans notre programmation. Un autre choix important que nous avons fait est celui de proposer un site accessible et simple d'utilisation pour permettre aux supporters de conserver le contact avec les tournois, particulièrement au cours d'une crise sanitaire où il est difficile d'accéder aux rencontres en personne.

À partir de ces choix déterminants et de notre nom d'équipe, les Suricates (Meerkat en anglais), nous avons pu donner naissance à Meerketball, un site web pratique qui permet de gérer les tournois de basketball, leurs équipes et leurs rencontres. Pour en savoir plus sur la création et la programmation de ce projet, ce rapport regroupe cinq parties.

La première partie décrit la **gestion de notre groupe**. Nous y décrivons la façon dont nous nous sommes organisés à cinq pour fournir un travail le plus complet possible. Nous y parlons aussi des outils dont nous nous sommes servis pour rendre notre collaboration efficace au sein des différentes parties du projet.

La seconde partie est celle de la description de notre **base de données** (ou BDD). En tant que base fondatrice du site, la BDD a été la première étape de notre travail de groupe. Elle est aussi la phase sur laquelle nous avons passé le plus de temps en groupe, car il était essentiel que nous l'ayons tous bien en tête.

La troisième partie concerne l'**architecture** de Meerketball, c'est-à-dire la façon dont les différentes parties isolées ont été agencées et regroupées afin de fournir un site final utilisable, centré autour d'une page d'accueil simple.

La quatrième et dernière partie porte sur les **particularités techniques** que nous avons ajouté à notre projet. Ce sont ces fonctionnalités qui en font un site unique et un produit fini.

## **Gestion de groupe**

---

Le début du projet pour notre groupe était en Décembre 2020, au moment du choix des différents membres. L'important était surtout de savoir identifier les forces et les faiblesses de chacun, afin de pouvoir facilement se compléter et s'aider sur les différentes tâches à venir. C'est notamment ce diagnostic qui a permis de diriger chacun des membres du groupe vers une position définie dans le projet. Par exemple, Robin L'Huillier a été à l'origine de beaucoup d'initiatives de communication vers notre tuteur et de prise de décisions internes à notre groupe, prenant facilement la place de chef de projet. Mais bien qu'il soit chef de projet, chacune de ses propositions devait être approuvée par la grande majorité avant d'être conduite, permettant ainsi de faire valoir nos forces et nos faiblesses et planifier nos objectifs sur le court et long terme.

Pour communiquer facilement, nous avons utilisé la plateforme de discussion Discord : en plus de sa facilité d'utilisation et son système de serveurs séparés permettant une discussion totalement privée, elle permet aussi d'inclure du code d'un langage à spécifier dans un message. Cela s'est avéré très pratique lors de nos premiers essais en PHP et SQL car les différents termes dans le code étaient colorés comme dans la plupart des éditeurs de texte inclus dans une API, rendant la compréhension du travail de chacun plus simple.

Afin de pouvoir travailler efficacement, nous avons utilisé le domaine GitLab fourni par la Faculté des Sciences et nous avons créé notre répertoire de projet à partir d'ici. Nous étions peu à savoir comment utiliser GitHub Bash avant le début du projet, mais nous avons rapidement pu mettre notre savoir en commun afin de préparer l'équipe avant le début du semestre pair. De plus, l'unité d'enseignement HLIN408 – Techniques de communication et conduite de projets comprend un cours assez complet sur GitHub et son utilisation.

Enfin, la rédaction de ce rapport nous a permis d'utiliser la plateforme en ligne Overleaf, permettant d'éditer un fichier LaTeX en ligne. LaTeX est un langage de programmation spécialisé dans le traitement de texte. Plusieurs logiciels peuvent prendre en charge le langage LaTeX, dont Overleaf. La complexité du logiciel est équilibrée par la diversité du contenu qu'il comprend : une grande variété d'extensions permet d'enrichir un document d'une manière plus flexible encore que ce qu'offrent les logiciels de traitement de texte classique. N'étant pas un logiciel comme Microsoft Word ou LibreOffice Writer, il nous a été difficile de l'élaborer comme on l'aurait souhaité.

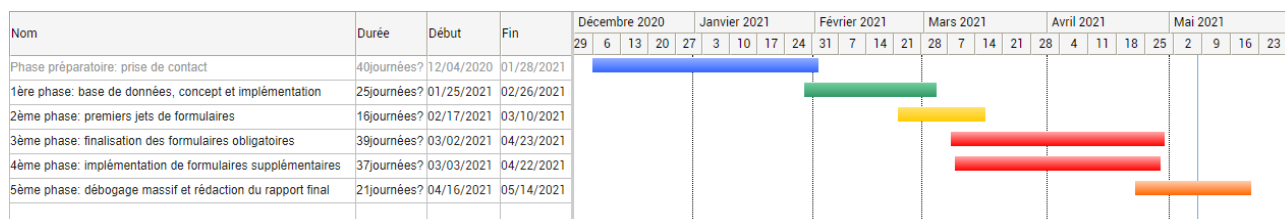
On distingue plusieurs phases du projet. La phase préparatoire est une phase précédant le semestre pair afin de pouvoir s'organiser sur les toutes premières tâches du projet et notamment leur répartition. La vraie première phase du projet permet de conceptualiser et implémenter la base de données, puis l'améliorer si besoin. La deuxième phase du projet concerne la répartition de chacun des formulaires obligatoires pour chacun des membres du groupe, ainsi que la création d'une première version. La troisième phase concerne la finalisation de ces formulaires, à savoir l'ajout de contenu supplémentaire. La quatrième phase débute en même temps que la troisième et concerne principalement la réalisation de formulaires supplémentaires. La cinquième et dernière phase concerne le débogage de tous les formulaires et la rédaction du rapport final.

Le mode de travail dans le groupe s'est organisé ainsi : nous travaillons chacun de notre côté jusqu'à une réunion le mercredi de chaque semaine. Cette réunion est organisée sur le créneau de travail en groupe commun à tous les groupes de L2 Informatique de 15h à 18h15. Quand un membre du groupe a fini de travailler sur un ou plusieurs fichiers, nous consultons le reste du

groupe afin de vérifier que l'on ne perd pas de progression sur ces fichiers. Si le fichier n'a été édité par personne d'autre dans la même période de temps, nous utilisons la fonction « push » de GitHub et envoyons les modifications sur le dossier en ligne du projet. Comme dit plus haut, nous prenons toutes les décisions ensemble et essayons toujours de trouver un compromis ou de s'aider pour faire avancer le groupe dans toutes les circonstances (difficultés techniques, membre malade, etc.). Le travail fourni a été exclusivement en distanciel depuis nos propres machines.

Pendant les différentes phases du projet, nous avons fixé plusieurs objectifs sur le court terme et le long terme. Parmi ces différents objectifs, beaucoup ont été fixés sur le créneau horaire de 15h à 18h15 détaillé dans le paragraphe précédent. Notamment, le premier jet des 5 formulaires principaux du site avait été fixé au mercredi 3 mars. D'autres objectifs comme la première esquisse de la base de données ont été fixés en dehors de ce créneau, même sans date de fin explicite. Enfin, une dernière catégorie d'objectifs à atteindre a été fixée selon les réunions de mise au point avec notre tuteur de projet ; pour les mois de mars et d'avril où la réalisation du projet était dans ses troisième et quatrième phases de projet, nous avons pris contact avec notre tuteur afin de montrer notre progression dans le projet. Cela a été très utile d'avoir son point de vue puisque d'habitude, chacun ne consulte personne d'autre que les membres du groupe lors d'un avis à donner sur les fonctionnalités du site. Ainsi, il fallait qu'avant chacune des trois réunions avec notre tuteur (17 mars, 7 avril et 22 avril), tous nos objectifs de court terme et de long terme soient atteints.

Nous avons créé un diagramme de Gantt avec le logiciel Ganttter pour Google Drive, qui rend compte de l'organisation globale du projet et du temps passé sur les différentes phases du projet expliquées plus haut. Une phase nécessite plus de travail si elle est représentée en rouge dans le diagramme, et moins si elle est représentée en bleu.



## Mise en place de la base de données

Le premier schéma de relations d'entités de notre site de tournoi ne contient pas les entités arbre et championnat, dont nous parlerons plus tard. Elles seront explicitées dans les second et troisième schémas, et correspondent à la mise en œuvre d'un type particulier de tournoi.

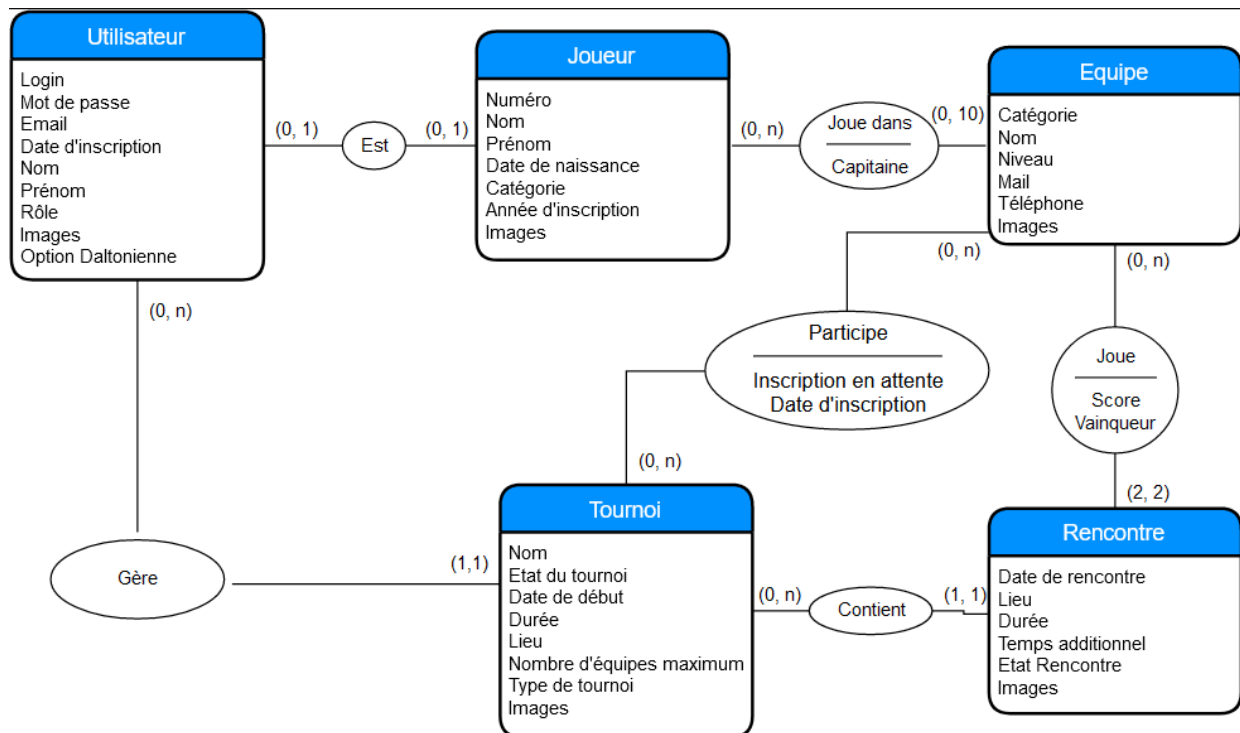


Figure 1: Schéma de relation entité-association

La première entité est l'utilisateur, ainsi que toutes ses informations. Il peut être joueur, mais aussi avoir un rôle qui définit sa position au cœur du site: "utilisateur", "gestionnaire" ou "administrateur". L'administrateur a accès à toutes les fonctionnalités, l'utilisateur peut s'inscrire et voir le déroulement des tournois, tandis que le gestionnaire peut gérer un ou plusieurs tournois (tandis qu'un tournoi n'est géré que par un gestionnaire).

Le joueur peut donc être utilisateur ainsi qu'être membre d'une ou plusieurs équipes. Les équipes comprennent de 0 à 10 joueurs, dont l'un est capitaine.

L'entité tournoi contient entre 0 et n rencontres, et une liste d'équipes. La liste d'équipe contient pour chaque équipe un booléen : chaque équipe demande à s'inscrire à un tournoi, et le gestionnaire les accepte ou rejette. Le lien entre les rencontres et le tournoi sera fait ultérieurement par une entité dépendante du type de tournoi.

L'entité Rencontre est liée aux équipes par une cardinalité (2, 2) : une rencontre est jouée par deux équipes, qui obtiennent chacune un score, et est gagnante ou perdante. Nous pouvons observer que dans la mise en œuvre de la base de données, puisque ce lien est constant, les références aux équipes, scores et vainqueur ont intégré la table rencontre.

Nous avons hésité quant au comment représenter les listes d'équipes et de joueurs dans une base de données. Initialement, nous avons tenté de créer des tables contenant autant d'attributs "équipeX"

que le maximum d'équipes que nos fonctionnalités pourraient accepter, mais cela s'est avéré peu pratique à manipuler. Nous avons alors choisi de représenter les listes d'équipes et de joueurs au travers de deux tables liées par des références aux tables concernées. Pour les listes d'équipes, chaque tuple référence une équipe et un tournoi, permettant ainsi de trier dans un sens et dans l'autre aisément tous les tournois auxquels participe une équipe, ou toutes les équipes participant à un tournoi.

Dans un premier temps, nous avons choisi de nous concentrer sur le type de tournoi "Coupe", le type correspondant aux critères attendus, à savoir une représentation en arbre. Plus tardivement, nous avons ajouté un second type de tournoi, le "Championnat", dont l'information est contenue dans une seconde table construite de manière similaire à la table "Arbre" et que nous allons détailler dans ce second schéma.

Championnat	
<b>IdChampionnat</b>	<b>INT AUTO INCREMENT</b>
Rencontre	INT REFERENCES Rencontre (IdRencontre)
Tournoi	INT REFERENCES Tournoi (IdTournoi)
Journee	INT
Resultat	INT

Dans un tournoi "Championnat", l'interface entre les entités tournoi et rencontres est représentée par la table ci-contre. Chaque entrée référence donc les deux tables, et contient deux informations supplémentaires : Journée et Résultat. Les championnats sont en effet organisés par

journée et, à la différence des coupes, acceptent des matchs se soldant par une égalité entre les deux équipes. Le résultat symbolise cela en codant l'information en fonction de l'équipe jouant à domicile (représentée par l'équipe 1 dans la table rencontre).

Ci-dessous, le schéma qui nous a servi de référence durant la phase de programmation. Lors de sa conception n'y figurait alors pas encore la table "Championnat", dont nous ne savions pas encore si nous l'insérerions. Afin de manipuler la base de données, nous avons d'abord utilisé mysql, puis sommes passés à PDO. Lors de l'implémentation de la base de données, nous avons décidé que toutes les clés primaires seraient insérées en auto-increment, c'est-à-dire que nous ne les préciserions pas lors de l'ajout de nouvelles valeurs; que les booléens seraient enregistrés sous forme de BIT(1), de valeur 0 ou 1.

La modélisation des rencontres d'un tournoi "Coupe" se fait au travers de la table arbre, qui référence le tournoi, les équipes participant à une rencontre, et une rencontre, mais aussi elle-même. Chaque nœud de l'arbre binaire correspond à une rencontre. Ses parents sont les noeuds/rencontres du tour précédent, et son fils est le noeud/rencontre du tour suivant qu'il engendre. Un attribut Hauteur permet d'enregistrer la hauteur du noeud, 0 correspondant à la finale, 1 à un noeud de demie-finale, etc.

Nous pouvons remarquer a priori que plusieurs attributs pourraient ne pas être contenus dans la table arbre: Equipe1/Equipe2 ainsi que Hauteur. Concernant Hauteur, elle pourrait être obtenue après plusieurs requêtes ou un calcul sur tous les tuples d'Arbre pour un Tournoi donné. Ce serait coûteux à mettre en place, plus que de retenir l'information dans la table. Quant à Equipe1/Equipe2, les deux attributs paraissent redondants vis à vis de la table Rencontre, qui les contient aussi. C'est une question de conception : dans un tournoi de type "Coupe", la table Arbre est construite avant que les rencontres soient créées, ce qui demande de retenir l'emplacement des équipes. Et si nous mettions en place d'autres types de tournoi, ou des rencontres amicales, la table rencontre devrait

contenir l'information. Nous avons donc fait sciemment le choix d'une apparente redondance sur ces deux attributs.

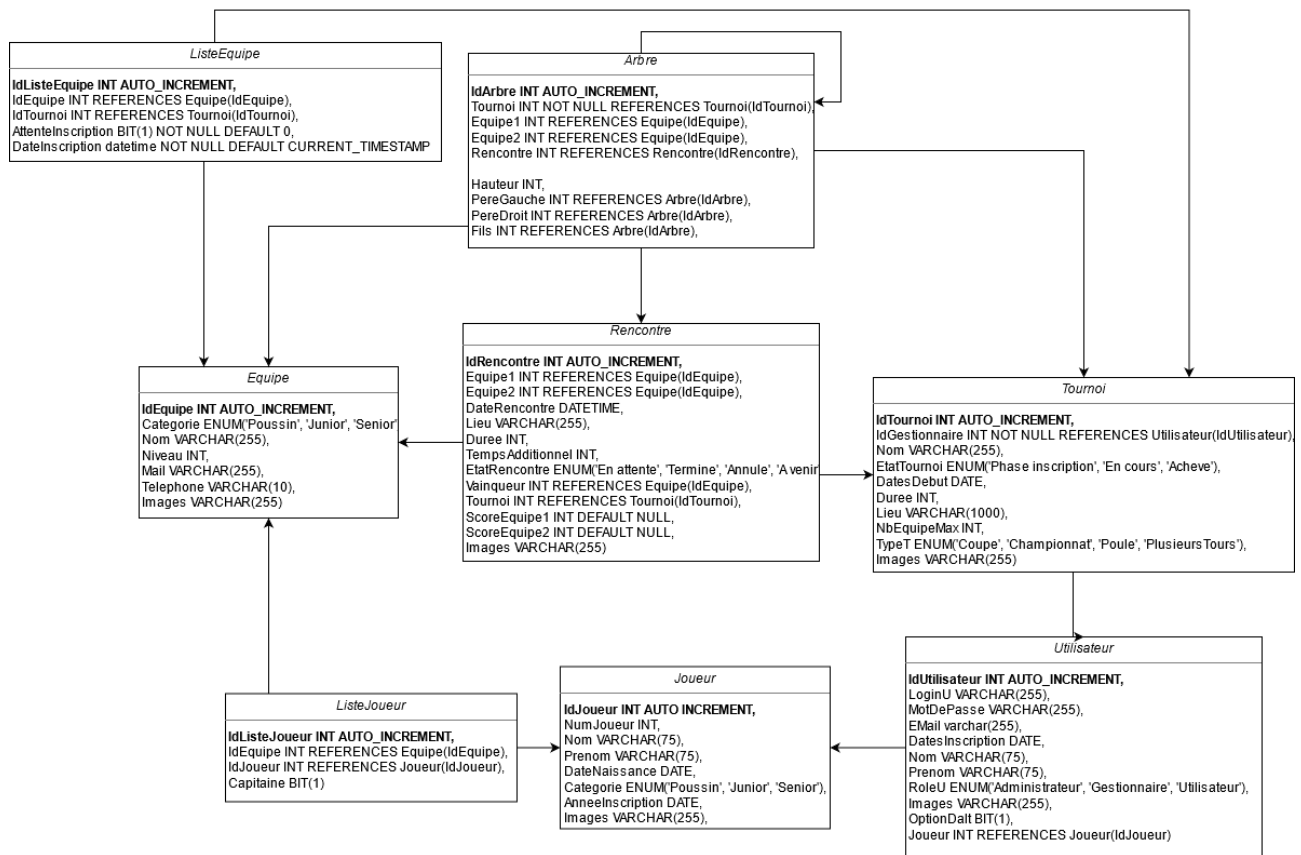


Figure 2: Table de référence pour la base de donnée

Un squelette commun est intégré dans toutes les pages par le truchement de la fonction PHP include. Il est constitué de :

- un bandeau coulissant (via un bouton amovible),
- un bandeau haut contenant les boutons permettant de se déplacer sur le site,
- un bandeau bas contenant l'assistance, un bouton retour vers le bandeau haut,
- une page « connexion.php » permettant la connexion vers la base de données.

L'interface utilisée entre les pages et la base de données est PDO car elle est indépendante du système de gestion de base de données.

La page d'accueil du site « index1.php » affiche, grâce à la fonction include(), les pages « carousel2.php », « carousel2Resp.php » et « ListeDesTournois.php ».

Le carrousel situé sous le bandeau haut dans la page d'accueil est construit à partir des pages « carousel2.php » et « carousel2Resp.php », le fichier style.css se charge d'en afficher un des deux selon la taille de l'écran.

La page « listeDesTournois.php » visible depuis l'accueil conduit vers les pages « AffichageTournoiChoisit.php » et « Championnat.php » qui offrent l'accès aux arbres des tournois ainsi qu'aux classements des championnats.

La page de création des tournois « CreationTournoi.php » communique avec la base de données en utilisant les fonctions php prepare(), query(), execute() qui prennent pour paramètres les requêtes SQL Update, Select et Insert Into pour y insérer des valeurs nécessaires au bon fonctionnement du site (changement éventuel du statut d'un utilisateur, nom du tournoi, type du tournoi, durée du tournoi, etc.) . Cette page est donc accessible uniquement aux administrateurs. La vérification est faite grâce à la variable super globale « \$\_SESSION[] » , initialisée lors de la connexion sur la page « LoginUtilisateur.php ».

La page « LoginUtilisateur.php » récupère les informations de la base de données par les fonctions prepare(), execute() prenant en paramètre des requêtes SQL Select afin de vérifier l'adéquation entre l'identifiant fourni et le mot de passe saisi. Ces informations inscrites dans la base de données y ont été insérées lors de l'inscription du compte issue de la page « Inscription.php ». Une ancre HTML dans le bandeau haut lui permet d'afficher les hyperliens vers les pages « Inscription.php » et « LoginUtilisateur.php ».

Un utilisateur connecté voit affiché l'hyperlien menant à la page utilisateur « CompteUtilisateur.php » contenant toutes ses informations personnelles. Ces informations viennent de la base de données ; elles sont obtenues par l'intermédiaire d'une requête SQL Select dans une fonction php prepare() et peuvent être modifiées par la transmission des informations de changement assurée par la variable super globale \$\_POST et le formulaire HTML action qui mène vers la page « ModifInfosPerso.php ». Les modifications fonctionnent de manière analogue. Les informations de la variable super globale \$\_POST sont vérifiées par des requêtes SQL Select dans la fonction php query(), puis modifiées par des requêtes SQL Update dans la fonction php exec().



La page utilisateur permet aussi de se désinscrire du site en supprimant la ligne de la base de données contenant les informations propres à l'utilisateur via la fonction php `exec()` et la requête SQL Delete. Cette requête utilise le login de l'utilisateur qui est un attribut unique récupéré grâce à la variable super globale citée supra.

La page « Equipe.php » est disponible pour l'ensemble des utilisateurs connectés. Cette information est donnée par la variable super globale `$_SESSION[[]]`. Elle inscrit dans la base de données, par la fonction php `exec()` dont le paramètre est une requête SQL Update, l'ensemble des joueurs d'une équipe, ainsi que son capitaine. Son affichage dépend des informations qu'elle récupère via la variable super globale `$_POST[[]]`.

La page « Preinscription.php », accessible elle aussi à l'ensemble des utilisateurs connectés, vérifie par le même mécanisme pré-inscrit les équipes dans un tournoi grâce aux informations stockées dans les variables super globales `$_SESSION[[]]` et `$_POST`.

La page « GestionTournoi.php » affiche par la fonction `include()` la page « GestionTournoiSynthese.php », laquelle par un formulaire HTML lui répond en l'appelant.

L'affichage de l'onglet synthèse se fait par la récupération des informations la concernant dans la variable super globale `$_SESSION[[]]`. Une fois affichées, les informations que la synthèse procure dépendent de celles qui sont récupérées par une requête SQL Select dans une fonction php `query()`. La synthèse permet entre autres de modifier des données de la base de données par une requête SQL Update, paramètre de la fonction php `exec()`. Elle permet aussi de faire par les mêmes mécanismes les actions décrites ci-dessous. Ces mécanismes concernent les tournois à élimination directe.

La validation des équipes possède son propre onglet. Si une équipe est validée, les informations de la base de données sont modifiées au moyen d'une requête SQL Update; à l'inverse, si l'équipe est rejetée, la demande d'inscription est supprimée de la base de données au moyen d'une requête SQL Delete. Ces 2 requêtes sont dans une fonction php `exec()`.

Les scores sont gérés par une requête SQL Update, paramètre de la fonction php `exec()`, ainsi que des variables super globales `$_POST[[]]`.

Enfin, le passage au tour suivant modifie les informations de la base de données via une requête SQL Update elle aussi dans la fonction `exec()`, et insère les nouveaux nœuds d'un tournoi avec une requête SQL Insert Into dans la fonction php `query()`.

L'insertion des rencontres est exécutée par une fonction php `query()` qui prend une requête SQL Insert Into. Les informations insérées proviennent de la variable super globale `$_POST[[]]`. Le fonctionnement de la page « GestionTournoi.php » dépend d'une variable super globale `$_SESSION[[]]` qui renseigne le type de tournois. En effet, un championnat n'est pas géré de la même manière qu'un tournoi à élimination directe. L'ensemble des nœuds d'un championnat sont déjà générés automatiquement par une requête SQL Insert Into à l'intérieur de la fonction php `exec()` qui, elle-même, se situe dans la page « CreationTournoi.php » ; les championnats n'ont donc pas besoin de passer au tour suivant. L'ensemble des rencontres est fixé par le truchement d'une requête SQL Update d'une fonction php `exec()` qui se déclenche dès que le nombre d'équipes requis du championnat est atteint via la même requête Update que celle des tournois à élimination directe citée plus haut.

La gestion d'un tournoi n'est disponible que pour un administrateur qui a accès à l'ensemble des tournois ou un gestionnaire qui ne verra que ses tournois grâce à la requête SQL Select de la fonction php query(), qui utilise la variable super globale \$\_SESSION[][] fournissant le login.

La déconnexion d'un utilisateur détruit la session en cours grâce à l'appel à la fonction php session\_destroy(). La page « Deconnexion.php » communique donc indirectement avec l'ensemble des autres pages en effaçant les informations contenues dans la variable super globale \$\_SESSION[][].

Enfin, le bandeau coulissant latéral appelle par la fonction php include() les 5 carrousels disponibles sur les pages «carousel3.php », «carousel4.php », «carousel5.php », «carousel6.php », «carousel7.php ».

Les images de ces carrousels proviennent de la base de données par l'intermédiaire de requêtes SQL Select effectuées par la fonction php query(). Elles ont pour finalité de servir de lien vers la page qui affichera l'avancée du tournoi correspondant à l'image.

### HTML / CSS

Pour accéder au site côté client, il faut ouvrir le fichier `index.html` à la racine du projet. Ce fichier ne contient qu'une redirection vers un fichier `index1.php` dans le dossier nommé `CadreStatique`, structurant l'ensemble de l'application web. Il a été fait ainsi en raison de WAMP et autres outils qui sont configurés pour ouvrir par défaut tout fichier intitulé `index` et ne donnant de ce fait plus accès au reste des fichiers. Par la suite, des redirections internes aux formulaires, et autres liens vers l'accueil ont été créés, et il est devenu risqué de changer le nom et le chemin de `index1.php`, donc cette astuce de redirection garantit l'absence d'un oubli d'édition des liens/redirections.

Continuons avec `CadreStatique`, les fichiers `skel1.php` et `skel2.php` sont la condensation de dizaines d'autres fichiers originellement introduits par la fonctionnalité `include` de php et qui nous ont amenés à revoir la conception html du header (l'imbrication des `div...`), de la navigation et du bandeau latéral. Bien plus complexe de base, il était urgent de factoriser et simplifier le code pour que chacun puisse participer à l'unification des différents formulaires en une unique application web. Dorénavant, chaque nouvelle page est créée sur le modèle du `squelette.php` (voir [annexe](#)), fichier modèle dans lequel le contenu est introduit entre les éléments essentiels et/ou factorisés comme le header etc.

Le logo et le nom du domaine (`Meerketball`) redirigent vers l'accueil, et le bouton `Top` du footer est une ancre vers le haut comme son nom l'indique. Le titre principal de la page est quant à lui un lien vers lui-même pour actualiser la page. Le bouton bandeau est déplaçable n'importe où sur la page, en dehors même de sa `div`, à la façon d'un `drag and drop`. Il possède aussi la fonctionnalité de faire disparaître le bandeau latéral comme son nom le suggère, tout en conservant ce choix d'une page à l'autre. Le contenu du bandeau est celui des prochains événements (rencontres sportives), le bouton principal (événements) déplie les autres pour tout afficher en un instant, de plus chacune des sous-parties possède son propre bouton de déploiement.

L'affichage des éléments des liens et menus varie selon l'état de connexion et le rôle de la personne inscrite. Une fois l'inscription ou la connexion et leur redirection automatique effectuées, prennent place les liens de déconnexion et le login de l'utilisateur, qui est en fait un lien vers les options du compte où ce dernier peut modifier les informations saisies lors de l'inscription ; comme l'aspect visuel du site, ou supprimer son compte. Cette dernière action, supprime de la BDD l'utilisateur, le redirige vers déconnexion, détruit la session PHP et redirige à son tour vers l'accueil.

Le menu hamburger (trois traits horizontaux parallèles), permet de regrouper les différentes pages, les accès aux formulaires, la FAQ, la page d'assistance et la page d'administration pour avoir accès aux fonctionnalités de développement telles la réinitialisation automatique de la BDD, ou bien le formulaire pour hacher les mots de passe.

Certains formulaires très imposants comportent un système d'onglet pour le confort de navigation. Ils utilisent les fonctionnalités basiques `select` avec des boucles affichant les données préalablement extraites depuis la BDD (voir [figure select](#)); mais aussi des `input` de type `text` avec des sécurités sur les regex, du traitement de symboles, des options `required`, `placeholder` ou `disabled`, la sauvegarde des entrées en cas d'erreur pour ne pas avoir à tout réécrire ou rappeler les

précédents choix (voir **figure** select ligne 195), des roulettes pour les numéros, du formatage de date avec date actuelle de base, date minimale en cas d'absence etc. (voir **figure** input)

```

193 <select name="Tournois" id="Tournois"><label for="Tournois"></label></br>
194 <option label="Tournoi" DISABLED SELECTED> </option>
195 <?php if (isset($_POST['Tournois']) or isset($_POST['tournoisD'])) {
196     >
197     <option label="<?=$_SESSION['historique']['idTourni'] >" DISABLED SELECTED></option>
198 <?php
199 }
200 foreach ($tourn as $TE) : >
201     <option required value="<?=$TE->IdTournoi >" id="<?=$TE->IdTournoi >"><label for="<?=$TE->IdTournoi >"><?=$TE->Nom . " - [" . $TE->Categorie . "]" ></label></option>
202 <?php endforeach >
203 </select>

```

Figure 3: Select

```

<label for="nonTournID"> Non du tourn : </label>
<input type="text" id="nonTournID" size="25" name="nonTournIDNAME" pattern="[A-Za-z0-9À-à-ç,à-À()]*" value="<" isset($SESSION['historique'][$nonTournIDNAME]) $SESSION['historique'][$nonTournIDNAME] : "" ;>
<label for="dateTournID"> Date de début : </label>
<input type="date" id="dateTournID" name="dateTournIDNAME" value="<" isset($SESSION['historique'][$dateTournIDNAME]) $SESSION['historique'][$dateTournIDNAME] : date("Y-m-d");>?> required: <br>
<label for="dureeTournID"> Durée (jours) : </label>
<input type="number" id="dureeTournID" name="dureeTournIDNAME" min="1" value="<" isset($SESSION['historique'][$dureeTournIDNAME]) $SESSION['historique'][$dureeTournIDNAME] : 1;>?> required: <br>
<label for="lieuTournID"> Lieu : </label>
<input type="text" id="lieuTournID" size="50" name="lieuTournIDNAME" pattern="[A-Za-z0-9À-à-ç,à-À()]*" value="<" isset($SESSION['historique'][$lieuTournIDNAME]) $SESSION['historique'][$lieuTournIDNAME] : " " ;>
<label for="nbEquipesMaxTournID"> Nombre maximum d'équipes : </label>
<input type="number" id="nbEquipesMaxTournID" name="nbEquipesMaxTournIDNAME" min="2" max="128" step="2" value="<" isset($SESSION['historique'][$nbEquipesMaxTournIDNAME]) $SESSION['historique'][$nbEquipesMaxTournIDNAME] : 2;>
<label for="typeTournID"> Type : </label>

```

Figure 4: Input

Pour faire le lien avec le CSS, abordons Bootstrap. Ce framework a été introduit, tout comme d'autres éléments externes tels que les librairies JS, le plus souvent pour une fonctionnalité particulière et le responsive des classes colonnes. Son utilisation a été dans l'ensemble un frein à la conception graphique du fait de la non maîtrise de base des langages HTML et CSS. En effet, des propriétés de bases propres aux marges, aux blocs ont rendu difficile l'alignement des éléments, couplées à l'imbrication des éléments et des propriétés d'héritages en cascade. Avec la factorisation du code, ce sont les principales raisons qui nous ont amenés à diverger du mockup ([annexe](#)).

Seul le strict minimum au niveau librairie a été conservé, et ce sont relativement les plus volumineux de nos fichiers ; de plus il s'agit de la version 5 de Bootstrap, une version beta. De ce fait, nous avons rencontré des difficultés à mettre en place toutes les fonctionnalités des carrousels, il fallait en fait reprendre des librairies de la version antérieure pour que les flèches latérales (les contrôleurs) ou même que les indications caudales de positionnement apparaissent. Un deuxième type de carrousel a ensuite été introduit, celui à 5 images visibles, plus esthétique mais presque entièrement fourni par une librairie, contrairement au premier, qui lui est codé entièrement par nos soins. Le premier carrousel (à une seule image) s'est donc vu relayé à l'affichage responsive des petits écrans (remplace celui à 5 images) ainsi que pour le contenu du bandeau latéral où les multiples carrousels ont en plus été épurés des contrôleurs/indicateurs.

```

129  <div>@media (min-width:720px){
130    #blocP{order:7; padding: 0 30px 0 50px;}
131    <div>#bandeau{
132      border-color: var(--gris3);
133      background-color: #1a202c;
134      order:6;
135      border: 5px;
136      border-left: 0;
137      border-right: 10px solid;
138      border-color: #2d3748;
139      min-height: 100%;
140      padding: 0 55px 0 65px;
141      font-size: 16px;
142    }
143  }
144  }
145  }
146  }
147  }
148  }
149  }
150  }
151  }
152  }
153  }
154  }
155  }
156  }
157  }
158  }
159  }
160  }
161  }
162  }
163  }
164  }
165  }
166  }
167  }
168  }
169  }
170  }
171  }
172  }
173  }
174  }
175  }
176  }
177  }
178  }
179  }
180  }
181  }
182  }
183  }
184  }
185  }
186  }
187  }
188  }
189  }
190  }
191  }
192  }
193  }
194  }
195  }
196  }
197  }
198  }
199  }
200  }
201  }
202  }
203  }
204  }
205  }
206  }
207  }
208  }
209  }
210  }
211  }
212  }
213  }
214  }
215  }
216  }
217  }
218  }
219  }
220  }
221  }
222  }
223  }
224  }
225  }
226  }
227  }
228  }
229  }
230  }
231  }
232  }
233  }
234  }
235  }
236  }
237  }
238  }
239  }
240  }
241  }
242  }
243  }
244  }
245  }
246  }
247  }
248  }
249  }
250  }
251  }
252  }
253  }
254  }
255  }
256  }
257  }
258  }
259  }
260  }
261  }
262  }
263  }
264  }
265  }
266  }
267  }
268  }
269  }
270  }
271  }
272  }
273  }
274  }
275  }
276  }
277  }
278  }
279  }
280  }
281  }
282  }
283  }
284  }
285  }
286  }
287  }
288  }
289  }
290  }
291  }
292  }
293  }
294  }
295  }
296  }
297  }
298  }
299  }
300  }
301  }
302  }
303  }
304  }
305  }
306  }
307  }
308  }
309  }
310  }
311  }
312  }
313  }
314  }
315  }
316  }
317  }
318  }
319  }
320  }
321  }
322  }
323  }
324  }
325  }
326  }
327  }
328  }
329  }
330  }
331  }
332  }
333  }
334  }
335  }
336  }
337  }
338  }
339  }
340  }
341  }
342  }
343  }
344  }
345  }
346  }
347  }
348  }
349  }
350  }
351  }
352  }
353  }
354  }
355  }
356  }
357  }
358  }
359  }
360  }
361  }
362  }
363  }
364  }
365  }
366  }
367  }
368  }
369  }
370  }
371  }
372  }
373  }
374  }
375  }
376  }
377  }
378  }
379  }
380  }
381  }
382  }
383  }
384  }
385  }
386  }
387  }
388  }
389  }
390  }
391  }
392  }
393  }
394  }
395  }
396  }
397  }
398  }
399  }
400  }
401  }
402  }
403  }
404  }
405  }
406  }
407  }
408  }
409  }
410  }
411  }
412  }
413  }
414  }
415  }
416  }
417  }
418  }
419  }
420  }
421  }
422  }
423  }
424  }
425  }
426  }
427  }
428  }
429  }
430  }
431  }
432  }
433  }
434  }
435  }
436  }
437  }
438  }
439  }
440  }
441  }
442  }
443  }
444  }
445  }
446  }
447  }
448  }
449  }
450  }
451  }
452  }
453  }
454  }
455  }
456  }
457  }
458  }
459  }
460  }
461  }
462  }
463  }
464  }
465  }
466  }
467  }
468  }
469  }
470  }
471  }
472  }
473  }
474  }
475  }
476  }
477  }
478  }
479  }
480  }
481  }
482  }
483  }
484  }
485  }
486  }
487  }
488  }
489  }
490  }
491  }
492  }
493  }
494  }
495  }
496  }
497  }
498  }
499  }
500  }
501  }
502  }
503  }
504  }
505  }
506  }
507  }
508  }
509  }
510  }
511  }
512  }
513  }
514  }
515  }
516  }
517  }
518  }
519  }
520  }
521  }
522  }
523  }
524  }
525  }
526  }
527  }
528  }
529  }
530  }
531  }
532  }
533  }
534  }
535  }
536  }
537  }
538  }
539  }
540  }
541  }
542  }
543  }
544  }
545  }
546  }
547  }
548  }
549  }
550  }
551  }
552  }
553  }
554  }
555  }
556  }
557  }
558  }
559  }
560  }
561  }
562  }
563  }
564  }
565  }
566  }
567  }
568  }
569  }
570  }
571  }
572  }
573  }
574  }
575  }
576  }
577  }
578  }
579  }
580  }
581  }
582  }
583  }
584  }
585  }
586  }
587  }
588  }
589  }
590  }
591  }
592  }
593  }
594  }
595  }
596  }
597  }
598  }
599  }
600  }
601  }
602  }
603  }
604  }
605  }
606  }
607  }
608  }
609  }
610  }
611  }
612  }
613  }
614  }
615  }
616  }
617  }
618  }
619  }
620  }
621  }
622  }
623  }
624  }
625  }
626  }
627  }
628  }
629  }
630  }
631  }
632  }
633  }
634  }
635  }
636  }
637  }
638  }
639  }
640  }
641  }
642  }
643  }
644  }
645  }
646  }
647  }
648  }
649  }
650  }
651  }
652  }
653  }
654  }
655  }
656  }
657  }
658  }
659  }
660  }
661  }
662  }
663  }
664  }
665  }
666  }
667  }
668  }
669  }
670  }
671  }
672  }
673  }
674  }
675  }
676  }
677  }
678  }
679  }
680  }
681  }
682  }
683  }
684  }
685  }
686  }
687  }
688  }
689  }
690  }
691  }
692  }
693  }
694  }
695  }
696  }
697  }
698  }
699  }
700  }
701  }
702  }
703  }
704  }
705  }
706  }
707  }
708  }
709  }
710  }
711  }
712  }
713  }
714  }
715  }
716  }
717  }
718  }
719  }
720  }
721  }
722  }
723  }
724  }
725  }
726  }
727  }
728  }
729  }
730  }
731  }
732  }
733  }
734  }
735  }
736  }
737  }
738  }
739  }
740  }
741  }
742  }
743  }
744  }
745  }
746  }
747  }
748  }
749  }
750  }
751  }
752  }
753  }
754  }
755  }
756  }
757  }
758  }
759  }
760  }
761  }
762  }
763  }
764  }
765  }
766  }
767  }
768  }
769  }
770  }
771  }
772  }
773  }
774  }
775  }
776  }
777  }
778  }
779  }
780  }
781  }
782  }
783  }
784  }
785  }
786  }
787  }
788  }
789  }
790 
```

Il a fallu rendre par la suite tout le site responsive, et en particulier les éléments factorisés. Les classes `img-fluid` ont été particulièrement utiles pour les images des carrousels, chapotées par des propriétés de `position`, de `max height`, de `width auto`, etc. L'autre particulier de

*Figure 5: Media*

notre code CSS en matière de responsive est l'utilisation de @media (voir [figure media](#)) qui complète le maillage (grid) de bootstrap (col-sm-12...). Différentes tailles d'écran ont été étudiées et l'ordre des éléments html s'est vu modifié grâce au CSS (propriété order). La taille des images, des polices sont ajustées à chaque définition ainsi que le positionnement du bandeau latéral à gauche pour les ordinateurs/tablettes et à droite de l'article principal pour les plus petits écrans (en raison de la prévalence de droitiers dans la population et de l'utilisation du pouce sur les téléphones).

Enfin, a été mise en place une version dite « daltonienne » du CSS, qui consiste à griser de façon réfléchie les couleurs pour obtenir des nuances de gris contrôlées et empêcher qu'un effet/élément disparaisse du spectre de vision des personnes atteintes de dyschromatopsie (voir charte graphique [annexe](#)). Cet effet est disponible dès l'inscription, modifiable en option de compte, et repose sur le principe que la dernière inclusion CSS prime, donc son inclusion est faite postérieurement au style.css principal en fonction d'une condition issue d'une requête PHP.

## PHP / SQL

Les différentes requêtes sont effectuées en passant par PDO, au moyen de query, exec, prepare suivi de execute puis l'extraction est faite en utilisant fetch/fetchall et FETCH\_ASSOC ou OBJ puis while ou foreach. Il a été très utile de passer les requêtes dans des variables avant leur exécution, le tout entre simples quotes ' et non pas entre doubles quotes, pour permettre l'interprétation d'autres variables, avec comme inconvénient que les variables comportant des quotes comme peuvent l'être les \$\_POST['var'] ne sont pas déspecialisables, il faut donc les affecter dans des variables aux noms plus standards. (Voir [figure requête](#))

```
374 // test de comptabilité des catégories
375 $IdTourn = $_POST['Tournois'];
376 $IdEquip = $_POST['Equipes'];
377
378 $sql42 = "SELECT Catégorie
379         FROM Equipe
380         WHERE IdEquip = $IdEquip";
381 $sql43 = "SELECT Catégorie
382         FROM Tournoi
383         WHERE IdTournoi = $IdTourn";
384 $stmt42 = $bdd->query($sql42);
385 $stmt43 = $bdd->query($sql43);
386
387 $Id42 = $stmt42->fetch(PDO::FETCH_ASSOC);
388 $Id43 = $stmt43->fetch(PDO::FETCH_ASSOC);
389
390 if ($Id42['Catégorie'] == $Id43['Catégorie']) {
391     $preinscription = $bdd->prepare('INSERT INTO ListeEquip (IdEquip, IdTournoi, AttenteInscription, DateInscription) VALUES (:IdEquip, :IdTournoi, 1, :datedemande)');
392     $preinscription->execute([
393         'IdTournoi' => $IdTourn,
394         'IdEquip' => $IdEquip,
395         'datedemande' => date('Y-m-d H:i:s')
396     ]);
397 }
```

Figure 6: Requête

En ce qui concerne les formulaires, pour faire un select à n valeurs venant d'un array/objet servant à stocker les données de la requêtes SQL précédente, il faut en principe écrire en pseudo code : value = « \$nème valeur ». Or, le form est en html, et pour passer une telle variable il fallait repasser (switcher) en PHP et empêcher la déspecialisation du \$, des balises et du ?. Cependant, il est impossible de passer <?php après de simples/doubles quotes du HTML. Il fallait donc contourner le problème en utilisant une balise ouvrante raccourcie <?= qui elle n'est pas refusée et permet de switcher PHP/HTML à l'intérieur des doubles quotes du value. La balise fermante ?> étant la même dans les deux cas. (voir [figure select ligne 201](#)). Quand bien même cette astuce marche in vitro sur WAMP, mais ces balises appartiennent à d'autres langages tels que XML, et les serveurs disposent de protections contre les injections et pour se prémunir, retirent les options de balises raccourcies que nous propose la config local de PHP. En conséquence, il est probable que certains serveurs ne reconnaissent plus notre code, l'interprétant dans le mauvais langage. Séparer

tous les codes HTML du PHP, comme le proposent les architectures MVC pourrait être une solution mais la configuration actuelle est limitante : il est difficile d'adopter cela en groupe, après plusieurs semaines de travail, et jusqu'alors non étudiée.

```

22
23 if ((isset($_POST['Toursnoi'])) {
24     $_SESSION['historique'][$_idLogin];
25 } else if ((isset($_POST['depresinscription'])) {
26     $_SESSION['historique'][$_idLogin];
27 } else {
28     if (isset($_SESSION['historique'][$_idLogin])) {
29         $_SESSION['historique'][$_idLogin];
30     } else {
31         $_SESSION['historique'][$_idLogin] = null;
32     }
33 }
34
35
36 $idshortcut = $_SESSION['historique'][$_idLogin];
37
317
318
319
320
321 if (!empty($_POST['Toursnoi']) and !empty($_POST['Equipes'])) {
322     //print_r($_POST);
323     $success = null;
324     if (
325         $role == 'Administrateur' or $role == 'Gestionnaire' or $scapitan
326     ) {
327         $stmt4 = $bdd->prepare('SELECT AttenteInscription FROM ListeEquipe WHERE IdToursnoi = :IdToursnoi AND IdEquipe = :IdEquipe');
328         $stmt4->execute([
329             'IdToursnoi' => $_POST['Toursnoi'],
330             'IdEquipe' => $_POST['Equipes']
331         ]);
332         $preinsOK = $stmt4->fetch(PDO::FETCH_OBJ);

```

Figure 7: Isset, empty

Les fonctions `isset` et `empty` ont été particulièrement utiles (voir [figure IssetEmpty](#)), du fait de la redirection sur l'url courante de bon nombre de formulaires. Il fallait prévoir les affichages différents selon la présence de certaines variables, en plus de gérer les autorisations d'accès, en fonction des rôles, qui ont été introduites avant la création de l'espace membre et de la facilitation qu'offre le stockage des ID et rôles dans la super globale SESSION. Il en découle des requêtes et des vérifications qui ne sont plus optimales mais qui sont intimement liées les unes les autres, du fait d'une conception précoce des formulaires.

```

4 $sqlBand7 = " SELECT DISTINCT IdEquipe, IdToursnoi, DateInscription
5 FROM ListeEquipe
6 ORDER BY DateInscription desc
7 LIMIT 5";
8
9
10 $resultBand7 = $bddBand->query($sqlBand7);
11 $entreesBand7 = $resultBand7->fetchAll(PDO::FETCH_OBJ);
12 $n = 0;$k=0;
13 $toNom;
14 $toType;
15 $toLieu;
16 $equ;
17 $ima1;$ima2;$ima3;$ima4;$ima5;
18 foreach ($entreesBand7 as $insctou) {
19     $sqlBand7tour = " SELECT Nom, TypeT, Lieu
20 FROM Toursnoi
21 WHERE IdToursnoi = $insctou->IdToursnoi";
22 $resultBand7tour = $bddBand->query($sqlBand7tour);
23 $entreesBand7tour = $resultBand7tour->fetchAll(PDO::FETCH_OBJ);
24 foreach ($entreesBand7tour as $toursno) {
25     $toNom = $toursno->Nom;
26     $toType = $toursno->TypeT;
27     $toLieu = $toursno->Lieu;
28 }
29 $sqlBand7equ = " SELECT Nom, Images
30 FROM Equipe
31 WHERE IdEquipe = $insctou->IdEquipe";
32 $resultBand7equ = $bddBand->query($sqlBand7equ);
33 $entreesBand7equ = $resultBand7equ->fetchAll(PDO::FETCH_OBJ);
34 foreach ($entreesBand7equ as $equip) {
35     $equ = $equip->Nom;
36     $k++;
37     switch ($k) {
38         case "1":
39             $ima1=$equip->Images;
40             break;
41         case "2":
42             $ima2=$equip->Images;
43             break;
44         case "3":
45             $ima3=$equip->Images;
46             break;

```

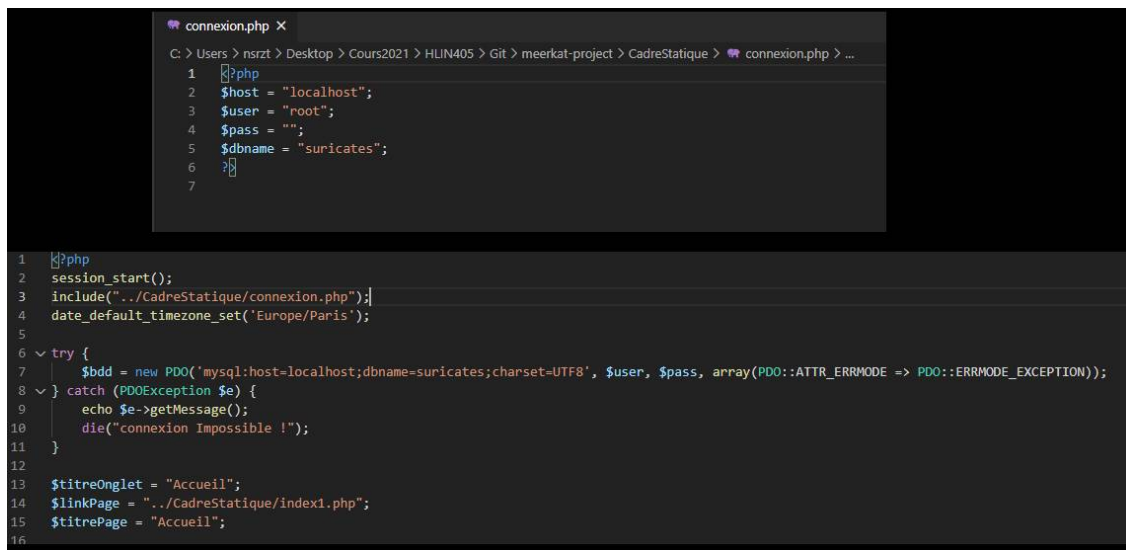
Figure 8: Carrousel 7

L'utilisation en matrice de plusieurs dimensions de la super globale SESSION permet de réduire les probabilités au sein d'un groupe de développeurs d'utiliser un même nom et d'écraser par inadvertance ou par effet de bord le contenu.

À la différence des carrousels centraux (carousel2) de l'accueil, les carrousels (carrousels 3 à 7) du bandeau latéral sont dotés d'un appel à la base de données pour afficher les images correspondantes aux informations présentes. L'implémentation est plus élégante et ne demande pas aux développeurs de modifier le code pour changer les images. Cependant la conception n'est pas moderne, avec un appel à la BDD à chaque page (le bandeau étant un élément constant, qui est caché par `display none` lorsqu'il est désactivé) du fait de la non-utilisation d'AJAX pour cibler et du MVC ainsi que l'utilisation non efficace des données stockées depuis une requête unique vers la BDD (plusieurs

requêtes dans des boucles). Le carrousel 7 illustre particulièrement ce point de vue, la requête principale se fait sur une table ne disposant pas d'image comme attribut, ce sont donc les requêtes internes aux boucles qui les fournissent et il faut donc instancier des variables globales au carrousel pour les afficher postérieurement. (voir [figure](#) carrousel 7)

Par ailleurs, l'utilisation de PDO permet l'appel dans le bandeau et dans les formulaires de la même BDD, simultanément et sans compromettre son intégrité, ce que ne permet pas mysqli, et qui nous aurait obligés à être encore plus normatifs quant au nommage de la connexion vers la BDD, pour permettre de gérer au mieux les exceptions. En effet, a été mise en place une connexion commune pour les administrateurs (voir [figure](#) ConnexionBDD), où seul le nom de la variable qui stocke la connexion est autorisé à changer ; les autres variables internes sont fournies par l'inclusion du fichier connexion.php où chaque administrateur aura renseigné ses identifiants et mot de passe utilisées pour se connecter à phpmyadmin.



```

1  <?php
2  $host = "localhost";
3  $user = "root";
4  $pass = "";
5  $dbname = "suricates";
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Figure 9: Connexion BDD

La possibilité de téléverser (upload) des images a été mise en place lors de la création des tournois/équipes. Une variable gérant les exceptions est initialisée avant le processus d'envoi. Si aucune image n'est chargée, la variable image sera celle par défaut. Si une image est chargée et le tournoi créé, le chemin est généré, sauf si l'image est vide ce qui déclenche une exception de format (getimagesize). Si l'image est déjà présente (filename et filesize : nom et tailles différentes), seul le champ correspondant dans la BDD sera initialisé/modifié, sans avoir à faire d'upload surnuméraire. Enfin si aucune exception n'est déclenchée, en plus de l'action sur la BDD, l'image est chargée dans le dossier correspondant (ImagesUtilisateur/Tournois).

La génération d'arbre est entièrement gérée en PHP et HTML et non pas à l'aide d'une librairie JS prête à l'emploi. Les cases sont des div dont le contour est affiché, de même que les traits de jonctions, dont seul le contour haut ou bas ou latéral est affiché (voir [figure](#) Arbre1).

```

echo "<div id=\"CaseArbre\" . $HauteurActuelle . $i . \"\" style=\"border: 1px solid black; position: absolute; left: \" . $left . \"px; top: \" . $top . \"px; margin: 0; padding: 0; text-align: center; width: \" . $tailleDiv . \"px; height: \" . $hauteurDiv . \"px; font-size: 80%\"
onClick='ouvrirChange(\"\" . $HauteurActuelle . $i . \"\")'> </div>";

```

Figure 10: Arbre 1



Le calcul du placement des div est fait par rapport à la hauteur en cours (selon l'avancement du tournoi) et du nombre de cases par hauteur correspondante. Et grâce à un certain nombre d'autres variables, l'arbre est relativement responsive, conservant ainsi ses proportions si la longueur ou la largeur donnée venait à être modifiée. Sur chaque case est appliquée un tooltip (infobulle) et le CSS adéquat, couleur pour les gagnants, réaction au passage de la souris (hover) etc (voir [figure Arbre2](#)).

```
$left = ($i - 1) * ($total * pow(2, ($HauteurMax - $HauteurActuelle))) + $total * pow(2, ($HauteurMax - $HauteurActuelle) - 1) - $total / 2 +
$separationDivLargeur + $leftAbsolute;
$top = $HauteurActuelle * ($hauteurDiv + $separationDivHauteur) + $tailleDiv + $topAbsolute;

$tailleDiv = 50;
$hauteurDiv = 50;
$separationDivLargeur = 10;
$separationDivHauteur = 30;
$total = $tailleDiv + $separationDivLargeur;
$topAbsolute = 50;
$leftAbsolute = 80;
```

Figure 11: Arbre 2

## Javascript

L'utilisation de JS concerne principalement le bandeau, les redirections, les onglets et l'affichage via les clicks sur des liens et/ou boutons. L'une des contraintes majeures aura été le fait de ne pas pouvoir aisément passer de PHP à JS, ou de faire appel à une requête SQL suite à une action coté utilisateur. C'est évident, l'un est serveur, l'autre client, mais c'est une frontière moins visible pour le développeur immergé qui est vite séduit par une action/réaction entre ces deux langages. Bien sûr le transtypage (cast) explicite de PHP vers JS est possible, et des fonctionnalités analogues à la super globale SESSION sont présentes comme sessionStorage.getItem avec certaines conditions par exemple cette dernière ne conserve que sur un même onglet de navigateur.

```
96
97 </script>
98
99 <body onload="cascade()">
100
101
102 <div id="parent" class="container-fluid bodyP">
103 <div id="child1" class="header HBF0">
104 <div id="rowhead" class="row">
```

Figure 12: Balise body

fonction de démarrage au chargement, actuellement.

Via JS, le CSS est modifié, en particulier le style display qui alterne entre none et inline/block pour l'affichage, de même que la classe peut être éditée comme c'est le cas des fonctions qui affectent le bandeau et l'article principal pour rééquilibrer l'affichage en col-3 et col-9 ou col-12 (voir [figure JS](#)) ou même les onglets, ce qui est l'équivalent de la fonction toggle.

La fonction cascade (skel1.php) (voir [figure Balisebody](#)) est un reliquat des précédentes implémentations, comme il n'est pas possible de placer plus d'une seule fonction au lancement d'une page avec onload dans la balise body. Cette fonction appelle toutes les fonctions, elle a été conservée par la suite, dans l'éventualité d'autres ajouts, bien que ne comportant qu'une seule

```
42 var show;
43
44 function cascade() {
45     if (sessionStorage.getItem("BandDisplay")=="1"){
46         show_hide();
47     }
48 }
49
50 function show_hide() {
51     if (show==null) {
52         document.getElementById("bandeau").style.display = "inline";
53         document.getElementById("bandeau").className = "col-3 bandeau";
54         document.getElementById("blocP").className = "col-9 blocP";
55         sessionStorage.setItem("BandDisplay", "1");
56         return show = 1;
57     } else {
58         document.getElementById("bandeau").style.display = "none";
59         document.getElementById("bandeau").className = "col-0 bandeau";
60         document.getElementById("blocP").className = "col-12 blocP";
61         sessionStorage.setItem("BandDisplay", "0");
62         return show = null;
63     }
64 }
```

Figure 13: js



```

65     var dragValue;
66     var enDeplacement = 0;
67
68     function move2() {
69         var element = document.getElementById("boutonBandeau");
70         element.style.position = "absolute";
71         element.style.margin = "-25px 0 0 0px";
72
73         element.onmousedown = function() {
74             dragValue = element;
75             enDeplacement = 1;
76         }
77     }
78
79     document.onmouseup = function(e) {
80         dragValue = null;
81         enDeplacement = 0;
82     }
83     document.onmousemove = function(e) {
84         var x = e.pageX;
85         var y = e.pageY;
86         if(enDeplacement == 1) {
87             dragValue.style.left = x + "px";
88             dragValue.style.top = y + "px";
89         }
90     }

```

Figure 14: bouton bandeau

La fonctionnalité de drag du bouton bandeau, à travers toute la page, (voir [figure 10](#) btnbandeau) est permise par 3 fonctions et 2 variables locales, dragValue sert à agir sur la position (pageX/Y, style.left/top) tandis que enDeplacement sert à contrôler les appels aux fonctions, le tout couplé aux fonctions JS qui interagissent avec la souris (onmouseup/down).

## Outils développement

Le hachage des mots de passe s'est fait via les fonctions password\_hash et verif. Tout se fait en interne, ni l'utilisateur ni même les développeurs ne voient leurs habitudes modifiées suite à l'introduction de cette fonctionnalité. Les mots de passe en clairs (usage développeur) sont indiqués en commentaire dans la BDD au côté de la table d'insertion d'exemple qui contient les mots de passe hachés. (voir [annexe MDP](#))

La réinitialisation de la BDD permet de détruire l'ensemble des tables, les reconstruire et de réimplanter les exemples de bases nécessaires au développement des formulaires, et que ces derniers modifient. Le code comprend le SQL nécessaire ainsi qu'une redirection en JS, grâce à la fonction de stockage activée lors du clic sur le lien de la page Administrateur, aussi présente sur les liens de connexion. Si le développeur accède à ces pages par l'url sans clic préalable sur les liens activant les fonctions de stockage et que le stockage est de ce fait vide, une redirection vers l'accueil est prévue.

## Conclusion

---

Dans ce rapport, nous avons expliqué comment nous avons géré notre travail d'équipe, construit notre bases de données, mis en place l'architecture de notre site et ajouté des particularités techniques. Pour le conclure, il nous semble pertinent d'analyser et de comprendre ce qu'un tel projet nous a apporté en tant qu'élèves et futurs professionnels. Pour cela, nous allons aborder différents problèmes que nous avons pu rencontrer et la façon dont nous les avons traités et surmontés. Tout au long du projet, il a été important pour nous de ne pas voir ces problèmes, qu'ils soient techniques ou organisationnels, comme des erreurs mais comme des tremplins de progression qui nous permettraient d'améliorer notre comportement en programmation à long terme. Cette perspective nous a permis de progresser et de ne pas rester bloqués face aux imprévus.

Nous allons aborder ici trois formes de problème auxquels nous avons fait face : la peur de l'inconnu, la difficulté de combiner cinq individualités et le défi de la normalisation. Ces trois sujet sur lesquels nous avons buté nous semblent intéressants car ils regroupent à la fois des point relationnels et techniques et sont ceux qui nous ont le plus permis d'évoluer.

Premièrement, il a été assez difficile de se lancer dans un projet dans lequel tout ou presque semblait nouveau. À la première lecture, les consignes semblaient assez longues et regorgeaient d'informations. Il semblait assez compliqué de mettre en place un calendrier de dates limites pour se rassurer puisqu'aucun de nous ne connaissait vraiment ce genre d'exercice. Il était assez difficile de s'y retrouver entre la peur de ne pas finir à temps et celle de ne pas fournir un travail du niveau attendu. En plus de ces points d'ombre organisationnels, nous avons très vite compris que les cours portant sur le projet ne seraient pas suffisants pour tout coder. Ces cours étaient très récents et donc pas encore inscrits profondément dans nos mémoires. Après un an et demi d'informatique, nous savons qu'apprendre un langage ne se fait pas en regardant des gens programmer mais en programmant nous-mêmes. Face à la multiplicité de langages relativement nouveaux pour nous, nous avons donc eu rapidement peur d'être débordés. De plus, en dehors des langages, le programme des cours qui devaient nous aider à structurer notre projet ne collait pas vraiment à notre calendrier.

Nous sommes donc tous partis avec des a priori et des peurs qui auraient pu nous freiner si nous ne les avons pas mis en commun. Une des forces de notre équipe a été de beaucoup communiquer et de s'entraider à chaque étape complexe. Chacun s'est donc formé aux langages que nous devions utiliser, à travers les cours mais également grâce à des ressources trouvées en lignes, et a partagé ses connaissances afin de permettre à tous d'avancer. Nous avons aussi beaucoup utilisé nos outils de communication pour poser des questions auxquelles ceux qui pouvaient répondaient. Enfin, les trois heures hebdomadaires de réunion ont vite été allongées pour permettre à chacun de "former" les autres. Cette façon de travailler nous a permis d'avancer efficacement, de prendre beaucoup d'initiatives et d'apprendre à travailler sans toujours suivre le chemin tracé d'un cours. Là où nous avions l'habitude d'avoir un cours magistral puis une séance de travaux dirigés associée, ce projet nous a permis d'apprendre à piocher dans les différentes banques de savoir à notre portée et de nous en servir pour avancer.

Même si nous avons habilement réussi à contourner cette épreuve, notre tendance à prendre beaucoup d'initiatives a parfois été source d'une nouvelle problématique.

Deuxièmement, combiner les idées, les initiatives et les méthodes de travail de cinq personnes a souvent été source de désaccord.

Comme nous l'avons vu plus tôt, nous avons chacun appris du cours mais également de sources indépendantes. Cette système d'apprentissage et d'initiative propre à chacun a été assez difficile à gérer. D'une part, générer son propre savoir et suivre ses propres idées empêche parfois d'être objectif et de différencier l'anecdotique du nécessaire, ce qui nous a parfois poussé à nous acharner sur des problèmes techniques. D'autre part, nos bonnes volontés respectives nous ont parfois menés au conflit. En effet, le problème de se mettre à plusieurs sur des tâches communes pour tous apprendre était de devoir accepter que d'autres personnes touchent à un morceau de code déjà écrit. Cette situation a parfois été difficile à gérer. À côté de ce problème technique, s'est aussi posé un problème organisationnel : être une équipe de cinq a autant d'avantages que d'inconvénients. En effet, plus le nombre de collaborateurs à un projet est grand, plus le nombre d'idées est important. À nous tous, une fois les premières étapes du projets passées, nous avions une liste d'idées supplémentaires complètement débordante et assez difficile à trier, puisque chacun avait du mal à accepter que ses propositions restent inexploitées. À ce problème de sélection d'idées s'est ajouté l'inconvénient de devoir faire des choix qui ne pourraient pas convenir à tout le monde. Plus il y a d'opinions différentes, plus ces opinions peuvent être en opposition avec le choix de la majorité.

La majorité est justement la méthode qui nous a permis de résoudre la plupart de nos mésententes. Nous nous sommes ainsi beaucoup servis de nos outils de communication pour chacun proposer un avis construit avant de voter et de choisir l'idée la plus fédératrice. Cette façon de faire a pu être source de frustrations voire même d'erreurs, mais elle nous a permis de rester soudés jusqu'au bout sans que l'équipe ne se découpe en plusieurs entités voulant emprunter des voies différentes. Pour calmer la frustration de chacun, nous avons décidé de permettre à chacun de travailler sur les fonctionnalités supplémentaires qui l'intéressaient. Ainsi, nous avons tous pu suivre à la fois un chemin principal commun et passer plus de temps sur les notions qui nous intéressaient.

En voyant les différents moments de conflit et les solutions que nous avons mises en place, nous avons compris qu'il était particulièrement important pour une équipe de prendre des décisions fortes et d'y rester fidèle pour ne pas alimenter des débats déjà clos. C'est cette détermination qui nous a permis de ne pas trop nous attarder sur de nombreux points qui auraient pu nous retarder. Cependant, parmi ces points, il y a en a que nous avons abordés assez tard dans le projet : la normalisation des codes.

Troisièmement, normaliser les morceaux de codes pour les rendre lisibles par tous les membres et compatibles entre eux a été une étape complexe.

Notre problème de normalisation est la suite logique des deux complications précédentes. En effet, les initiatives nombreuses de chacun et le fait de manipuler des langages nouveaux pour nous nous ont beaucoup freiné au moment de rendre les différents codes compatibles les uns avec les autres pour tous les relier. Puisque nous n'avions utilisé que très peu et très récemment les principaux langages du projet web, nous avons chacun développé nos propres habitudes et automatismes. Nous n'avions pas anticipé que ces automatismes propres à chacun pourraient être un obstacle à la réunification des codes, mais au moment d'assembler les codes et de chacun regarder les productions des autres, nous nous sommes rendus compte de notre erreur.

Hors de cet imprévu technique, normaliser nos approches organisationnelles et nos méthodes de travail été une grande partie du travail que nous n'avions pas prévue. Chacun des cinq membres a un parcours scolaire complètement différent et a donc engrangé des méthodes de travail différentes. De plus, un membre est du groupe est en double licence, deux n'ont pas suivi une première année d'informatique, certains utilisent Windows et d'autres Linux, etc. Toutes ces différences qui peuvent sembler peu importantes ont parfois été délicates à traiter puisqu'il fallait trouver une méthode commune ou convenant à tous, tout en conservant les particularités de chacun pour les exploiter au mieux. Dans un contexte dans lequel chacun d'entre nous n'a jamais rencontré ou même vu les autres, le concept de travail d'équipe semblait très abstrait. Pour pallier ce problème, nous avons beaucoup utilisé les temps de réunion pour nous mettre d'accord sur les normes que nous appliquerions pour construire la structure du site, nous avons également appris à travailler en nous adaptant les uns aux autres. Ces étapes primordiales, bien que tardivement réalisées, nous ont permis de faciliter la mise en commun et le débogage puisque chacun connaissait et comprenait le travail des autres. Par dessus tout, ce qui aurait pu sembler être une erreur rédhibitoire pour la suite du projet nous a permis d'apprendre à mieux prévoir et anticiper les différents afin de ne plus en subir les conséquences.

Les nombreuses semaines pendant lesquelles nous avons travaillé sur ce projet nous ont permis d'apprendre la gestion du temps et la pression, le travail de groupe dans un environnement et un contexte quasiment inconnu, et par dessus tout la prise d'initiative et de décisions à plusieurs. Au-delà des langages que nous avons appris à connaître plus en profondeur, nous avons surtout su retenir de ce projet qu'au sein d'une équipe, la collaboration est la clé de la réussite.

## Annexe

```
squelette.php ●
C: > Users > nsrzt > Desktop > Cours2021 > HLIN405 > Git > meerkat-project > DocDivers > squelette.php > ...
1  <?php
2  session_start();
3  include("connexion.php");
4  date_default_timezone_set('Europe/Paris');
5
6  try {
7      $bdd = new PDO('mysql:host=localhost;dbname=suricates;charset=UTF8', $user, $pass, array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION));
8  } catch (PDOException $e) {
9      echo $e->getMessage();
10     die("connexion Impossible !");
11 }
12
13 $titreOnglet="";
14 $linkPage="";
15 $titrePage="";
16
17 ?>
18 <!doctype html>
19
20
21
22
23
24
25
26 </html>
```

Figure 15: Squelette

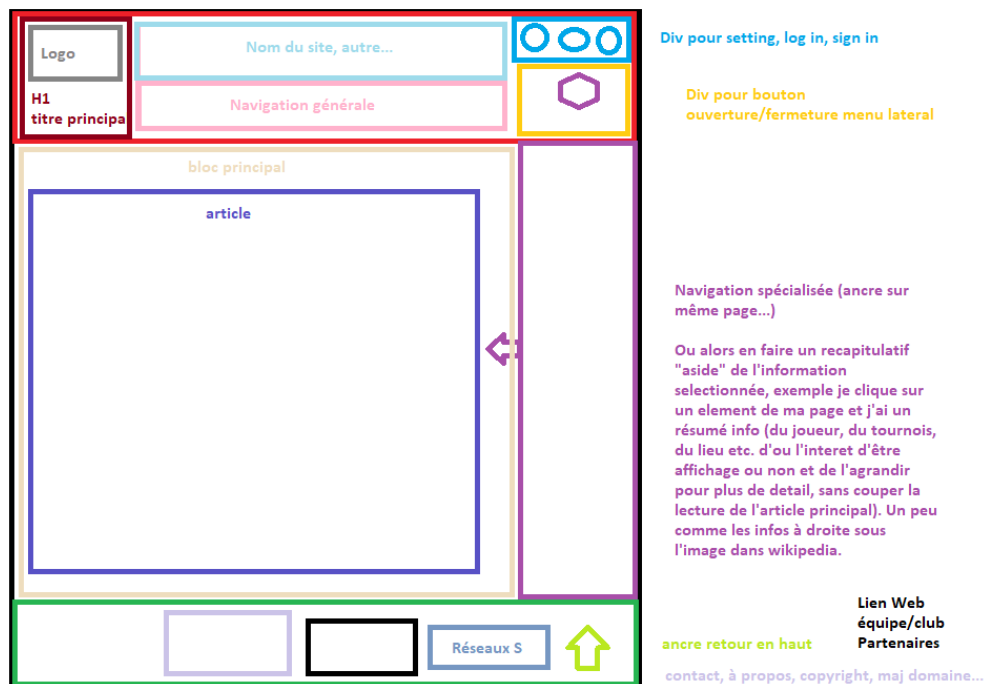


Figure 16: Mockup

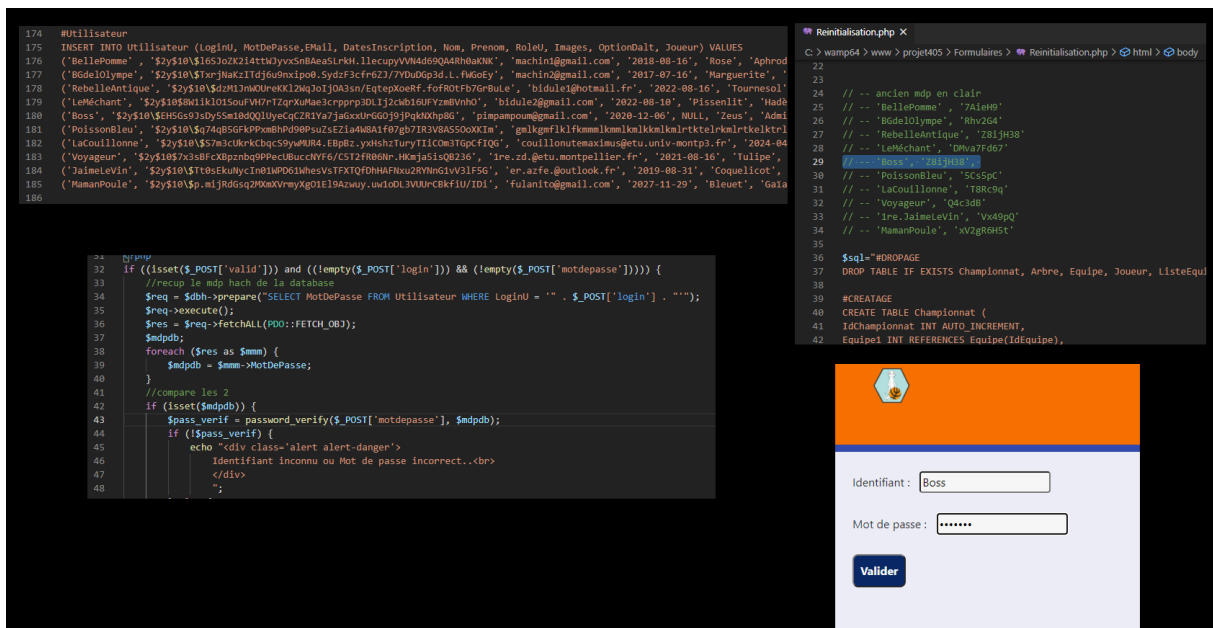
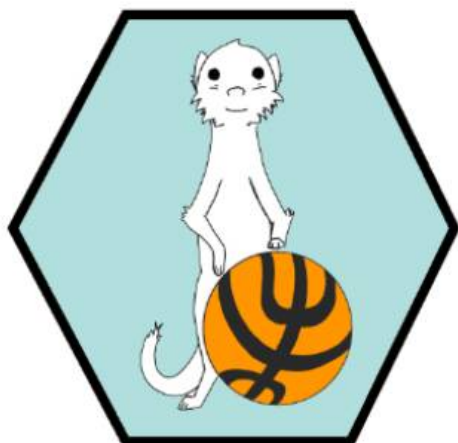


Figure 17: mot de passe

## Charte graphique



**Meerketball**  
VISER PLUS HAUT

## Introduction

Meerketball, avant d'être un site pour les gestionnaires de tournois, est un site pour les fans de baskets, pour les supporters, pour tous ceux qui aiment le sport.

Ainsi, son identité graphique simple a pour objectif de représenter sa vaste accessibilité et son utilisation simple.

## Sommaire

- Logo & Nom du site
- Choix des couleurs
- Palette
- Police
- Autres choix

## Logo & Nom du site

Comme nous l'avons déjà précisé, Meerketball est un site qui se veut tout public. Pour symboliser cela efficacement, nous avons choisi d'ériger le Suricate en mascotte de notre projet. En effet, quoi de mieux qu'un animal "mignon" pour attirer le regard d'un public moins habitué au basket ? Les familles des joueurs, les nouveaux supporters, ... Meerketball est accessible à toutes et à tous.

C'est pour cela qu'à la fois dans le nom de notre site et dans son logo, le Suricate est le premier élément visible et le basket est second, tout en restant au premier plan.

## Choix des couleurs

Nous voulions, dès le début, centrer notre palette autour d'une touche de orange, pour rappeler la couleur du ballon de basket. Après de nombreuses hésitations quant à la nuance à préférer, nous avons opté pour un orange "flashy" et vibrant, pour donner un certain dynamisme à notre palette. Pour compléter ce choix assez osé, nous avons utilisé du bleu, couleur complémentaire du orange, dans des tons pastels ou foncés pour garder une certaine harmonie. Enfin, dans un souci de lisibilité, nous avons choisi d'afficher les informations textuelles en noir ou bleu foncé sur fond clair et blanc sur fond sombre.

## Palette

### Logo



#addddb



#ff9100



#242424

### Cadre du site



#ebebfb

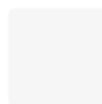


#f56a00



#2e44a7

### Autres couleurs



#f5f5f5



#b1b1e4



#06225e

## Police

Pour permettre à Meerketball d'être facile d'accès et d'utilisation, il était important pour nous de choisir une police unique et facilement lisible. La police utilisée dans tous le site est la Segoe UI, sans serif pour faciliter la lecture sur écran et accentuer l'effet net recherché.

Pour choisir les différentes tailles de police, nous avons encore une fois mis l'accent sur l'importance de la lisibilité : l'utilisateur doit pouvoir repérer en un clin d'oeil à quel endroit du site il se trouve. Les titres principaux, présents sur chaque page, sont ainsi affichés en 400%, les sous-titres en 200% et les autres paragraphes en 100%.

Les marges autour de toutes les zones de texte ont également été réglées avec précision pour accompagner au mieux la lecture.

Enfin, dans un souci de clarté, les paragraphes annonçant des erreurs ou des réussites suite à des actions d'utilisateurs sont affichés dans un code couleur évident : rouge et vert.



## Autres choix

Un des choix graphiques les plus marquant de Meerketball est celui de proposer un "mode daltonien". En effet, dans notre objectif "tout public", il semblait primordial de permettre aux supporters ne distinguant pas certaines couleurs d'accéder au site et de l'utiliser complètement.

Pour cela, une nouvelle palette a été créée. Basée sur les niveaux de gris (greyscale) et la palette colorée du site, elle a pour but de conserver chez les utilisateurs daltoniens un certain confort et un dynamisme assumé.