
Tutti Tourney

*BARCEA Matéo , DEROUBAIX Renaud , ROLLIN Valentin ,
ROUQUAIROL Lucas*



Table des matières

1	Introduction	1
1.1	Présentation	1
1.2	Cahier des charges	1
2	Technologie utilisée	3
2.1	Angular	3
2.2	Mongodb	3
2.3	Autre	4
2.4	Intérêt des technologies utilisées	4
2.4.1	Pourquoi Angular?	4
2.4.2	Pourquoi Mongodb?	4
3	Développement logiciel	5
3.1	Modélisation	5
3.2	Fonctionnalités	6
3.2.1	Connexion / Inscription	6
3.2.2	Accueil	7
3.2.3	Barre de navigation	7
3.2.4	Création d'un évènement	8
3.2.5	Gestion du tournoi	11
3.2.6	Visualisation	12
3.3	Traitement des données	13
3.4	Statistiques	14

4	Algorithme	15
4.1	Méthode du serpent	15
4.2	Implémentation	16
5	Gestion du projet	17
5.1	Planification et gestion	17
5.2	Changements effectués	18
6	Conclusion	19
6.1	Conclusion	19
7	Bibliographie	20
7.1	Bibliographie	20
8	Annexe	22

1.1 Présentation

Le projet consiste à développer un site de gestion de tournoi sportif (football, volley, tennis, basketball..) permettant à des organisateurs (authentifiés) de pouvoir créer un événement (spécifiant un lieu, une date..) composé de tournois sportifs. L'organisateur doit être capable de gérer son dit tournoi (en saisissant des résultats, en clôturant des tours..) en éditant les feuilles de matchs.

1.2 Cahier des charges

Il nous a été demandé d'utiliser un certain nombre de technologies :

- Pour la programmation côté serveur, un framework PHP REST au choix et une base de donnée SQL ou nonSQL à choisir.
- La programmation côté client devait se faire exclusivement en JavaScript.
- Un dépôt git devait aussi être utilisé, nous avons choisi github.
- Utilisation d'une méthode de gestion de type SCRUM

Par ailleurs, un certain nombre de fonctionnalités sont aussi attendues :

- Un système d'authentification.
- La possibilité de créer un événement, spécifiant un lieu, une date, un type de jeu, le nombre ainsi que les noms des tournois.
- La préinscription d'équipes, les équipes possédant des identifiants, une liste de joueurs, ainsi qu'un niveau.
- La possibilité pour l'organisateur de définir la formule sportive du tournoi (nombre de poules, nombre d'équipes par poules..).
- L'automatisation de la répartition des équipes grâce à la méthode du serpent. Cependant,

l'organisateur doit tout de même pouvoir réaliser des permutations.

- Pouvoir éditer les feuilles de match des poules.
- Pouvoir saisir les résultats en temps réel.
- Pouvoir clôturer un tour et décider de la formule sportive à adopter pour la suite (tournoi consolante..).
- L'organisation du classement d'une poule doit se faire en tenant compte de l'ordre du nombre de matches gagnés, en cas d'égalité, on prend le quotient des sets gagnés sur les sets perdus. Enfin, s'il y a toujours égalité, on tiendra compte du quotient des points gagnés sur les points perdus.
- Après la constitution automatique des tours suivants, l'organisateur devra pouvoir modifier manuellement des poules (avec une interface drag and drop).
- Pouvoir clôturer des tournois après des matches de classement (les perdants des deux demi-finales font un match pour la troisième place).
- La progression du tournoi doit être visible en ligne.

2.1 Angular

Angular est un framework côté client, open source, basé sur TypeScript, et co-dirigé par l'équipe du projet « Angular » à Google et par une communauté de particuliers et de sociétés. Angular est une réécriture complète d'AngularJS. Il permet la création d'applications Web et plus particulièrement de ce qu'on appelle des « Single Page Applications » : des applications web accessibles via une page web unique qui permet de fluidifier l'expérience utilisateur et d'éviter les chargements de pages à chaque nouvelle action. Le Framework est basé sur une architecture du type MVC (Modèle-vue-contrôleur).

- Un modèle (Model) contient les données à afficher.
- Une vue (View) contient la présentation de l'interface graphique.
- Un contrôleur (Controller) contient la logique concernant les actions effectuées par l'utilisateur.

Cela permet donc de séparer les données, le visuel et les actions pour une meilleure gestion des responsabilités.

2.2 MongoDB

MongoDB est un système de gestion de bases de données orienté documents, répartissable sur un nombre quelconque d'ordinateurs et ne nécessitant pas de schéma prédéfini des données. Il fait partie de la mouvance NoSQL. Contrairement à une base de donnée relationnelle SQL traditionnelle, MongoDB ne repose pas sur des tableaux et des colonnes. Les données sont stockées sous forme de collections et de documents.

2.3 Autre

Nous avons eu recours à d'autres technologies telles que Node.js, bcrypt.js, jsonwebtoken, express, mongoose, cors ainsi que body-parser.

Node.js est un environnement bas niveau permettant l'exécution de JavaScript côté serveur, ce qui nous a été fort utile pour lancer le backend.

Bcrypt.js est une dépendance javascript permettant de crypter les chaînes de caractères, nous l'utilisons pour crypter les mots de passe utilisateurs afin de les sécuriser.

Jsonwebtoken nous sert à récupérer un accesstoken lors de la connexion en sessionstorage.

Express est utile pour les roots. Cors permet de régler certains conflits. Mongoose crée le lien entre mongodb et notre site. Body-parser nous sert à parser le JSON.

2.4 Intérêt des technologies utilisées

2.4.1 Pourquoi Angular?

Au début du projet, il nous a été demandé de choisir un framework à utiliser. Nous avons le choix entre Angular et Symfony, après quelques petits projets tests, nous avons décidé de partir sur Angular. Angular possède de nombreux avantages :

- Angular profite de la rigueur et flexibilité du langage TypeScript.
- La structure du code est facile à lire et permet de bien se retrouver dans le projet.
- Angular impose une approche mieux structurée à base de composants et une façon plus claire d'échanger les données entre les composants.

2.4.2 Pourquoi Mongodb?

Nous voulions utiliser une base de données NoSQL, c'est alors que notre regard s'est posé sur Mongodb. Contrairement aux bases de données SQL, MongoDB n'implique aucune contrainte en termes de structure des documents. Les données n'ont pas de schéma préconçu, cette flexibilité rend MongoDB puissant et performant.

3.1 Modélisation

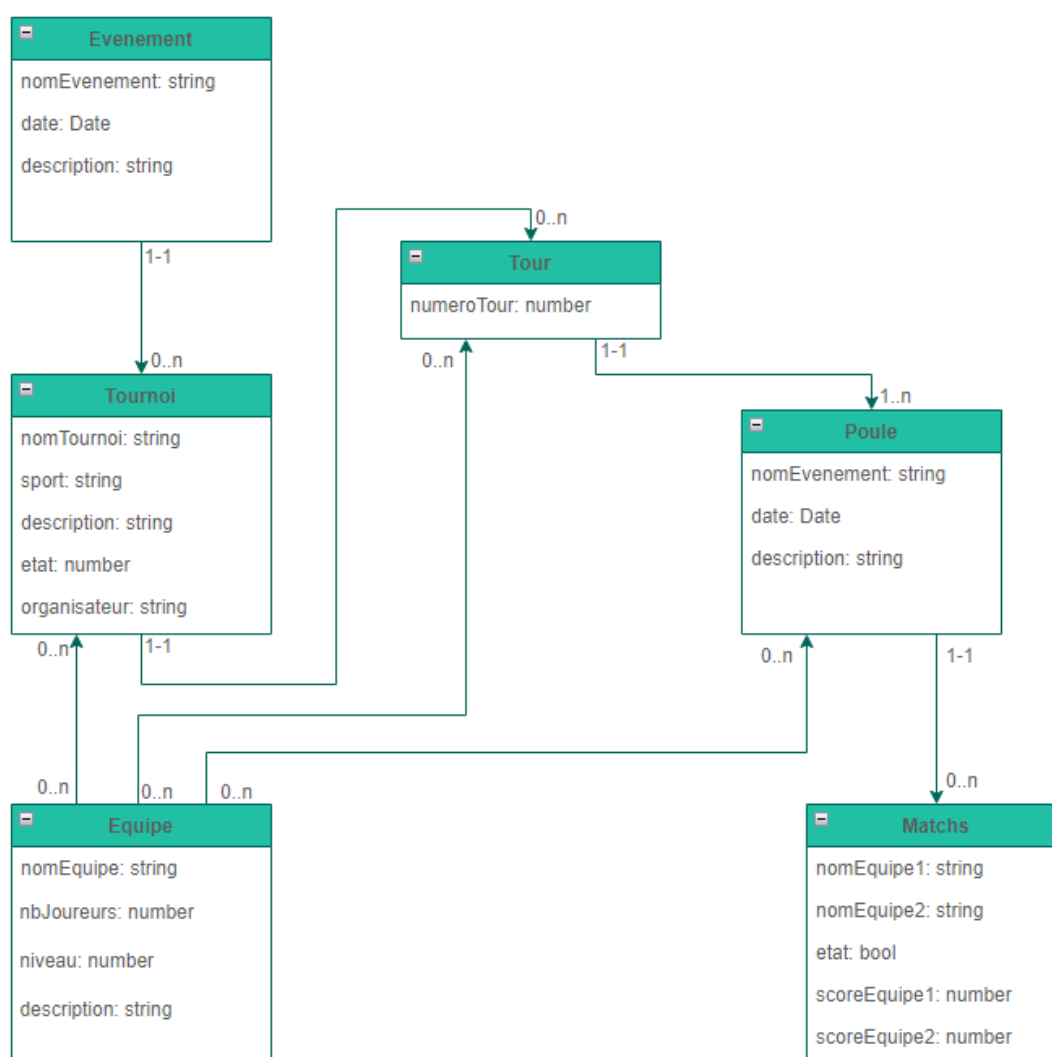


FIGURE 3.1: *Modélisation de notre base de donnée avec mongoDB*

Notre base de données est une collection 'Evenement'. Cette collection contient un objet 'Evenement'. Dans cet objet, on peut y mettre plusieurs 'Tournoi'. On inscrit des équipes à ce dit Tournoi. Notre 'Tournoi' est composé de 'Tours' qui sont eux même composés de 'Poules'. Un 'Tour' récupère les 'Equipes' qualifiées à ce tour. Les 'Poules' prennent ces 'Equipes' selon des règles établies sur le site puis fait affronter les 'Equipes' de la 'Poule' entre elles dans des Matches.

3.2 Fonctionnalités

3.2.1 Connexion / Inscription

Notre site offre la possibilité de s'inscrire ainsi que de se connecter. Ces fonctionnalités sont importantes, car afin de s'occuper d'un événement ou d'un tournoi, il est nécessaire d'en être son créateur. Pour cela, il faut un compte.

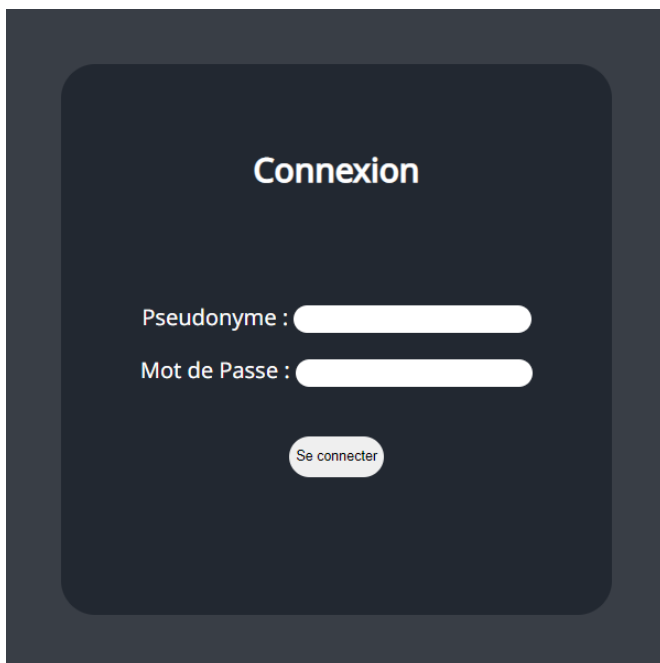
The image shows a login interface with a dark background. At the top, the word 'Connexion' is centered in white. Below it, there are two input fields: 'Pseudonyme : ' followed by a white rounded rectangle, and 'Mot de Passe : ' followed by another white rounded rectangle. At the bottom, there is a white rounded button with the text 'Se connecter'.

FIGURE 3.2: Interface de connexion

The image shows a registration interface with a dark background. At the top, the word 'Inscription' is centered in white. Below it, there are three input fields: 'Pseudonyme : ' followed by a white rounded rectangle, 'Mot de Passe : ' followed by another white rounded rectangle, and 'Email : ' followed by a third white rounded rectangle. At the bottom, there is a white rounded button with the text 'S'inscrire'.

FIGURE 3.3: Interface d'inscription

3.2.2 Accueil

Voici la page d'accueil de notre site après connexion :



FIGURE 3.4: *La page d'accueil*

3.2.3 Barre de navigation

Pour naviguer sur notre site, nous avons opté pour une simple barre de navigation. L'ensemble du CSS a été fait sans l'aide de bootstrap ou autre front-end framework par choix.



FIGURE 3.5: *Barre de navigation (utilisateur pas connecté)*

Celle-ci change en fonction de l'état de l'utilisateur, si l'utilisateur est connecté, il a accès à plus de fonctionnalités dans sa barre de navigation.



FIGURE 3.6: *Barre de navigation (utilisateur connecté)*

3.2.4 Création d'un événement

Notre site sert à gérer des tournois. Pour cela, il faut d'abord créer un événement dans lequel par la suite, on y mettra nos tournois.

The image shows a dark-themed web form titled "Création de votre événement". It contains three input fields: "Nom de l'événement" with the value "Coupe du Monde", "Date de l'événement" with the value "21/05/2022" and a calendar icon, and "Description (optionnel)". Below these fields is a button labeled "Créer événement".

FIGURE 3.7: *Interface de création d'évènement*

Quand l'événement est initialisé, on est redirigé vers la composante permettant de créer un tournoi. Celle-ci contient une liste déroulante contenant les événements existants.

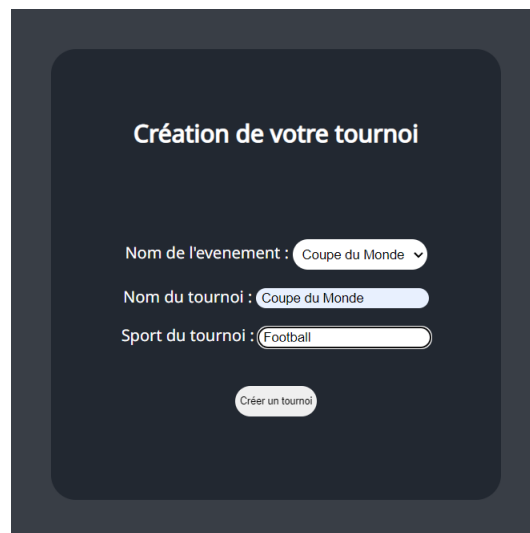
The image shows a dark-themed web form titled "Création de votre tournoi". It contains three input fields: "Nom de l'événement" with a dropdown menu showing "Coupe du Monde", "Nom du tournoi" with the value "Coupe du Monde", and "Sport du tournoi" with the value "Football". Below these fields is a button labeled "Créer un tournoi".

FIGURE 3.8: *Interface de création d'un tournoi*

Une fois le tournoi créé, il faut aller inscrire des équipes. Pour cela, on passe par notre page Home. On sélectionne notre événement, on arrive sur la page suivante, cette page affiche tous les tournois de notre événement (ici, il n'y a qu'un tournoi dans notre événement.) il faut alors inscrire

son équipe dans le tournoi qui nous intéresse.



FIGURE 3.9: *Notre événement "Coupe du monde"*

Lors de l'inscription, il nous est demandé un nom d'équipe, un nombre de joueurs ainsi qu'un niveau. Ce niveau est important pour pouvoir équilibrer les phases de groupes du tournoi.

The screenshot shows a dark grey rounded rectangle titled 'Ajouter votre équipe' in white. It contains four input fields with white labels and text: 'Nom de l'équipe :', 'Nombre de joueur dans l'équipe :', 'Niveau (de 1 (Pro) à 10 (Débutant)) :', and 'Description : (optionnel)'. Each input field has a white border and a small white button at the end. At the bottom of the form is a white button labeled 'Créer une équipe'.

FIGURE 3.10: *La page d'ajout d'équipe*

Quand l'organisateur du tournoi le décide, il peut alors arrêter les inscriptions et commencer le tournoi. Un tournoi est composé de tours, on doit d'abord choisir la formule sportive du premier tour du tournoi : c'est-à-dire le nombre de poule, si le nombre de poule correspond à la moitié des inscrits, il s'agit alors d'un arbre binaire appelé bracket.

Tournoi : Coupe du Monde

Sport : Football

Le tournoi n'a pas commencé

Il y a 32 équipes inscrites au tournoi.

Indiquez le nombre de poules souhaités :

Nombre de poules conseillées:

- 16 poule(s) de maximum 2 équipes.
- 8 poule(s) de maximum 4 équipes.
- 4 poule(s) de maximum 8 équipes.

Envoyez

FIGURE 3.11: Notre interface de création des poules

Les poules initialisées, notre site les remplit automatiquement avec les équipes inscrites. Les poules sont alors équilibrées en fonction du niveau des équipes. Si les poules nous conviennent, on appuie alors sur le bouton en haut à gauche et le tour commence.

TUTTI TOURNEY | Tournois terminés | Tournois en cours | Mes tournois | Créer un événement | Créer un tournoi | Bonjour, test ! | Déconnexion

Les poules me conviennent

Poule A	Poule B	Poule C	Poule D	Poule E	Poule F	Poule G	Poule H
Equipes :	Equipes :	Equipes :	Equipes :	Equipes :	Equipes :	Equipes :	Equipes :
Qatar niveau : 1	Angleterre niveau : 1	Argentine niveau : 1	France niveau : 1	Espagne niveau : 1	Belgique niveau : 1	Bresil niveau : 1	Portugal niveau : 1
Uruguay niveau : 2	Croatie niveau : 2	Suisse niveau : 2	Etats-Unis niveau : 2	Mexique niveau : 2	Allemagne niveau : 2	Pays-Bas niveau : 2	Danemark niveau : 2
Senegal niveau : 3	Iran niveau : 3	Japon niveau : 3	Maroc niveau : 3	Serbie niveau : 3	Pologne niveau : 3	Corée du Sud niveau : 3	Canada niveau : 3
Tunisie niveau : 4	Cameroun niveau : 4	Ghana niveau : 4	Equateur niveau : 4	Arabie Saoudite niveau : 4	Nouvelle-Zélande niveau : 4	Pays de Galles niveau : 4	Perou niveau : 4

FIGURE 3.12: La liste des poules

3.2.5 Gestion du tournoi

Chaque poule d'un tour contient des matchs entre équipes de cette même poule. L'organisateur doit alors remplir les résultats des matchs et finir le match. Lorsqu'un utilisateur visualise le tournoi, il a alors accès au résultat en direct, il peut voir quel match a fini, lesquelles sont toujours en cours et leur résultat actuel.

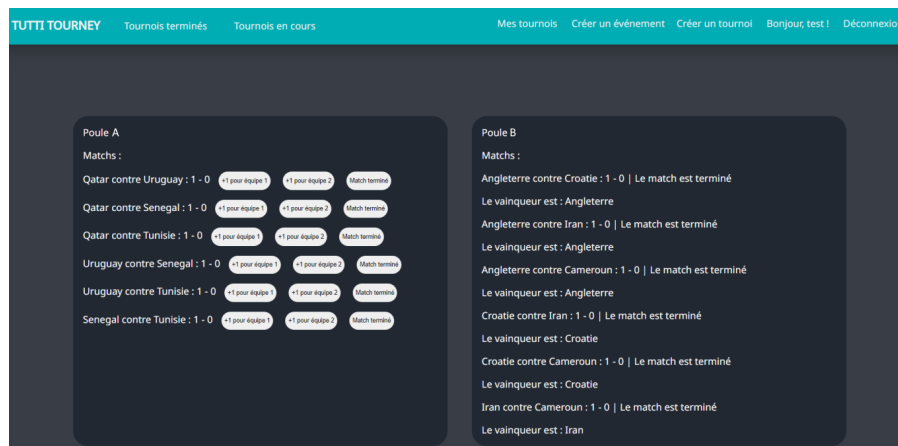


FIGURE 3.13: *La gestion du tournoi*

Une fois tous les matchs terminés, il faut passer au tour suivant. L'organisateur du tournoi doit alors choisir la formule du prochain tour ainsi que le nombre d'équipes de chaque poule passant au tour suivant. Si le tour finissant n'est composé que de deux équipes, alors il s'agit de la finale, le site propose donc de mettre fin au tournoi.

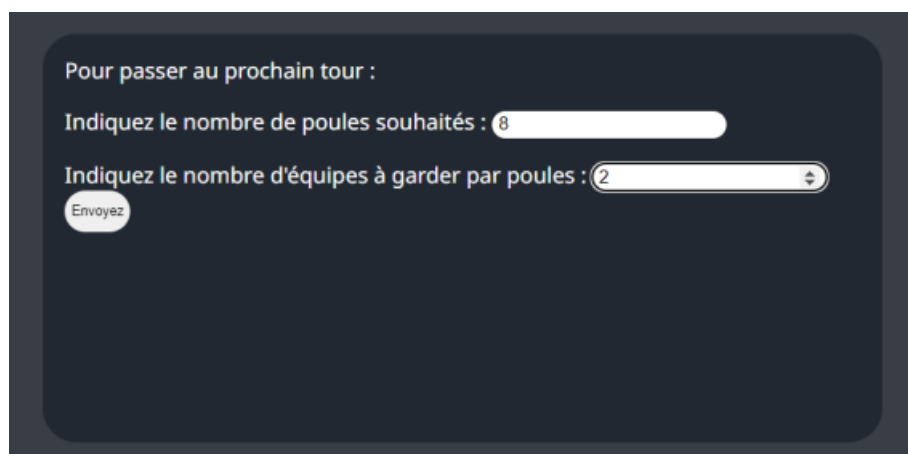
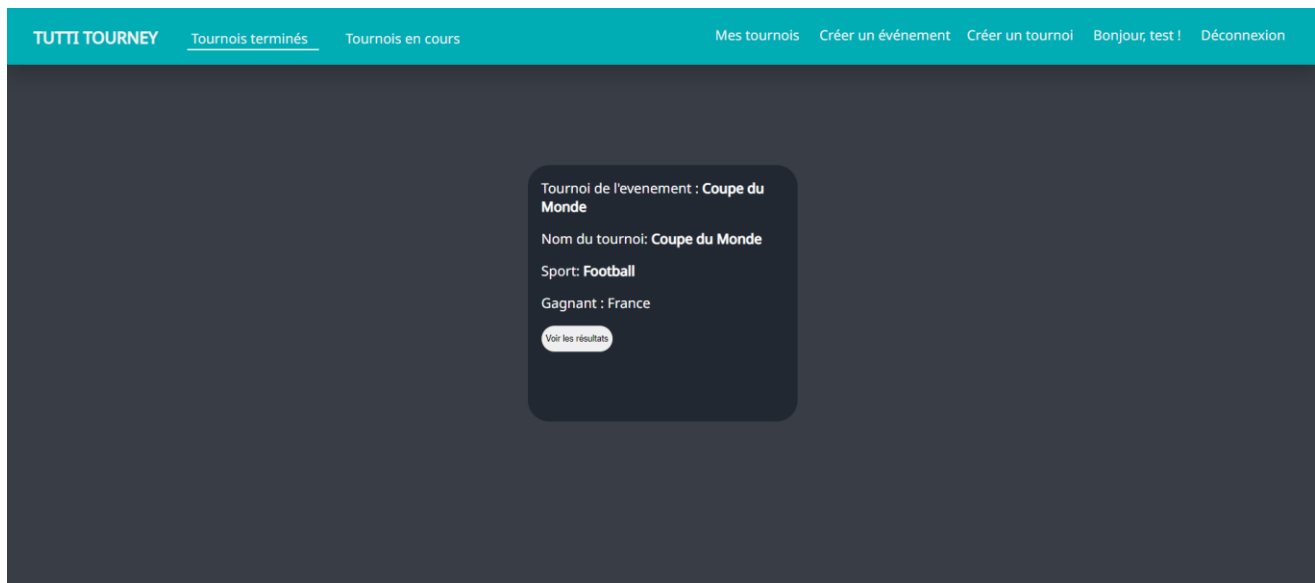


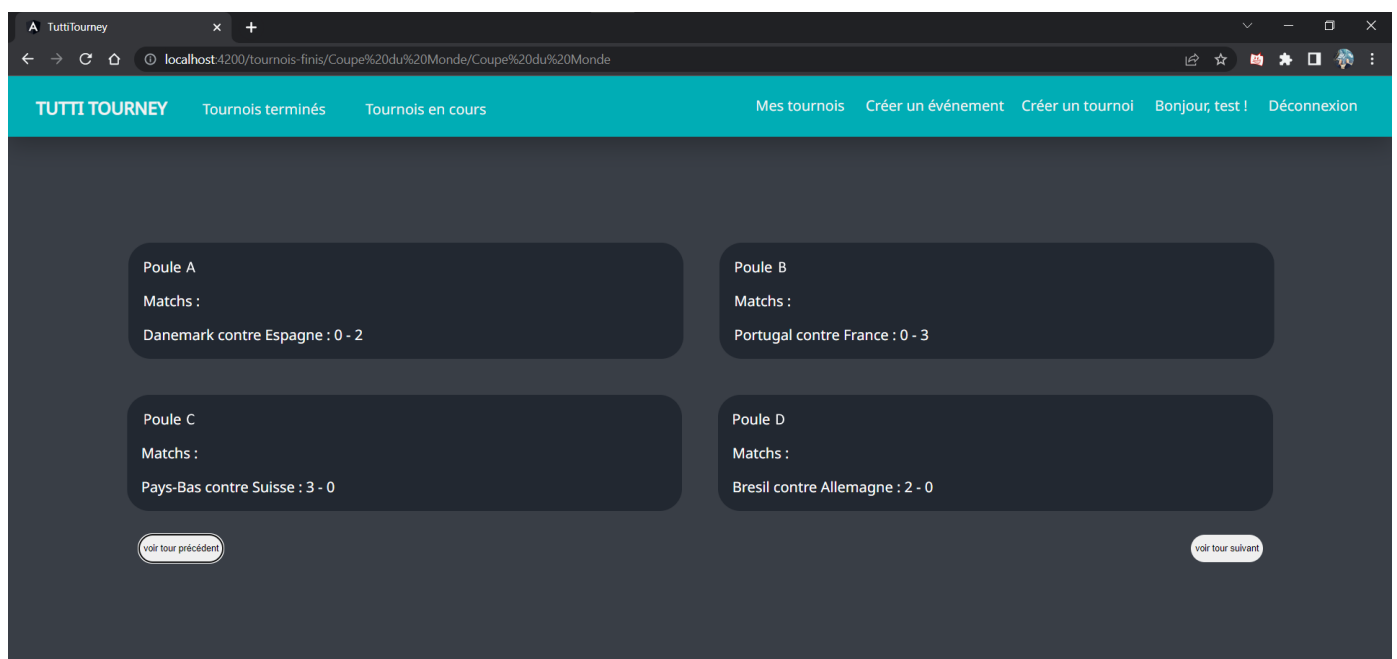
FIGURE 3.14: *La gestion du tour suivant*

Une fois le tournoi terminé, celui-ci passe dans l'onglet 'Tournoi fini' de notre barre de navigation, depuis cet onglet, on peut voir tous les tournois étant terminés.

FIGURE 3.15: *La fin du tournoi*

3.2.6 Visualisation

Lorsque qu'on souhaite observer un tournoi en cours ou fini, en le sélectionnant, on a accès au tour en cours ainsi que les tours précédents ainsi que les résultats des matchs. Sur l'image suivante, on peut observer les quarts de finale du tournoi.

FIGURE 3.16: *Visualisation du 3eme tour*

Quand on observe un tour, si on le souhaite, on peut obtenir à tout moment le classement de chaque poule avec leur nombre de victoires par match et leur score total.

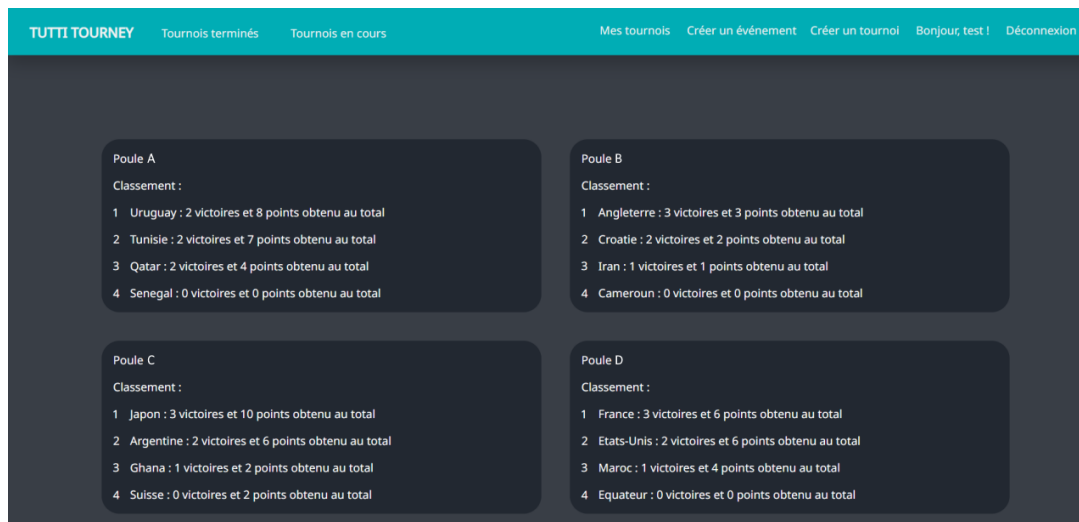


FIGURE 3.17: *Visualisation du classement des poules*

3.3 Traitement des données

Sur notre site, le traitement des données se fait à l'aide d'un formulaire. On a alors implémenté une fonction permettant de faire la liaison entre les données du formulaire et la collection. On les récupère dans un objet du type qui leur correspond (ex : `Evenement`, `Tournoi`, `Equipe`,...), ces types sont définis dans des interfaces Angular que nous avons créées, ensuite on stocke ces données en `LocalStorage` sur le navigateur.

Le `localStorage` permet de garder nos objets sous forme de chaîne de caractères en local, cela permet à nos objets de rester intacts même après changement de page ou redémarrage du navigateur. Ensuite on récupère nos valeurs grâce à des méthodes telles que `getItem()`.

```
onSubmit(): void {
  // //ajout a la liste d'evenement
  this.events.push(this.newEvenement);
  localStorage.setItem('events', JSON.stringify(this.events))

  // //post to restAPI
  this.eventService.createEvent(this.newEvenement).subscribe(
    (data) => console.log(data),
    (error: any) => console.log(error),
    () => console.log("Successfully added Event to database")
  );
}
```

FIGURE 3.18: *OnSubmit*

Ensuite, elles sont traitées et envoyées à notre collection via l'appel d'une fonction (`createEvenement`) qui transforme les données JSON passées à la fonction en String grâce à `json.stringify` et les transmet grâce à un Observable ainsi que le module `HttpClientModule` d'Angular.

```
createEvenement(evenement: Evenement): Observable<Evenement>{  
  const headers = { 'content-type': 'application/json' };  
  const body = JSON.stringify(evenement);  
  console.log(body);  
  return this.http.post<Evenement>(this.apiUrl+'/evenements', body,  
  {  
    headers: headers,  
  });  
}
```

FIGURE 3.19: *createEvenement*

3.4 Statistiques

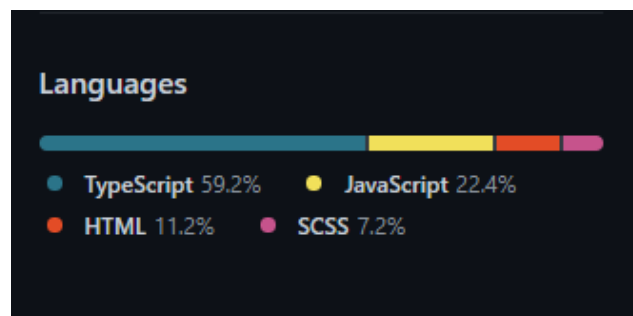
Nombres de composants angular : 16

Nombres de fonctions : 52

Nombres de classes : 25

Nombres d'interface angular : 6

Nombre de commit sur github : 43

FIGURE 3.20: *Répartition des langues utilisés*

4.1 Méthode du serpent

Il nous a été demandé dans le cahier des charges d'utiliser la méthode du serpent afin de répartir efficacement les équipes dans des poules. Le principe du serpent est de répartir les équipes en fonction de leur niveau pour obtenir des poules de niveau homogène. Pour ce faire, on va classer chaque équipe en fonction de leur niveau à l'aide d'un entier.

Equipes :	Equipe1	Equipe2	Equipe3	Equipe4	Equipe5	Equipe6	Equipe7	Equipe8
Niveaux :	1	2	3	4	5	6	7	8

FIGURE 4.1: On crée 8 équipes, et on leur attribue un niveau.

Ensuite, on va les répartir de manière à ce que les sommes des entiers de chaque poule soient les plus proches possible.

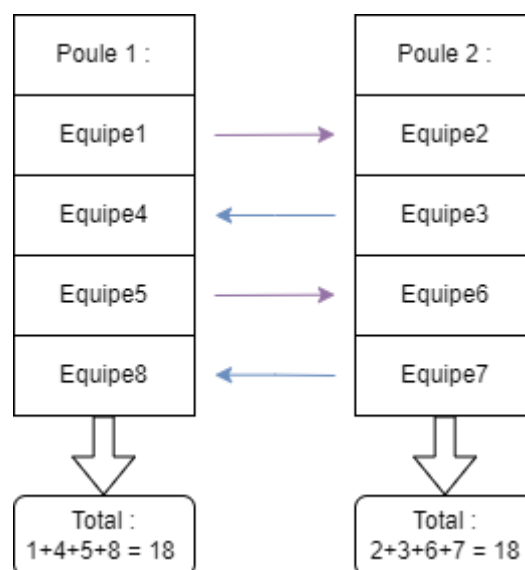


FIGURE 4.2: La répartition en 2 poules s'établit ainsi, les poules sont de niveaux équivalents.

4.2 Implémentation

Nous avons créé un algorithme prenant en entrée une liste d'équipes déjà triée ainsi qu'une liste de poules.

```

attributionPoule(equipes : Equipe[], poules: Poule[]) : void {

    let nombrePoule = poules.length;
    let i : number = 0;
    let sens = 0;
    if (nombrePoule == 0 || nombrePoule==undefined){
        return ;
    }
    while (equipes.length > 0){

        if (sens == 0 && i<nombrePoule){
            poules[i].equipes?.push(equipes[0]);
            equipes.shift();
            i++;
        }
        else if (sens !=0 && i>=0){
            poules[i].equipes?.push(equipes[0]);
            equipes.shift();
            i--;
        }
        else if (sens == 0){
            sens=1;i--;
        }
        else {
            sens=0;
            i++;
        }
    }
}

```

FIGURE 4.3: *Implémentation algorithmique de la méthode du serpent*

On fait une boucle while parcourant la liste d'équipes de la tête jusqu'à la queue. Enfin, on rentre chaque équipe dans chaque poule sous forme de serpent (comme vu dans l'exemple précédent).

5.1 Planification et gestion

Tout d'abord, nous avons rendez-vous avec monsieur Meynard, toutes les deux semaines afin de faire le point sur notre avancée. Pendant ces réunions, nous avons mis en place un google doc afin de pouvoir prendre des notes ensemble et en direct. Après chaque rencontre, nous faisons un débrief afin de discuter des différentes idées évoquées précédemment, de la manière dont on comptait effectuer les changements à apporter au site, ainsi que la répartition du travail. De notre côté, nous avons formé deux équipes pour partager les tâches et ainsi travailler plus efficacement. La première composée de Matéo et Valentin, et la seconde composée de Renaud et Lucas. Nous avons alors utilisé la méthode de programmation en binôme ("pair programming"), que nous avons maintenue tout le long du projet.

C'est une méthode dans laquelle le binôme travaille ensemble sur une même partie de code. L'un occupant le rôle de pilote en écrivant les lignes de code tout en exposant ses idées. L'autre occupant le rôle de copilote en suivant assidûment et signalant les erreurs effectuées par le pilote.

Nous avons aussi mis en place un groupe discord afin de communiquer plus efficacement entre nous. Ce groupe nous a permis de nous retrouver tout au long du développement afin de faire les liaisons entre binômes et de discuter des différentes difficultés rencontrées.

Enfin, lors de la dernière semaine, nous avons effectué un rush massif afin de finaliser le projet. Nous avons rendez-vous tous les jours à 10 heures afin de montrer ce que chacun avait fait la veille et discuter de ce que nous allions faire le jour même.

5.2 Changements effectués

Au fil du projet, nous avons pris conscience d'un certain nombre de choses que nous faisions mal ou qui ne nous étaient pas très utiles. Au début du projet, nous nous sommes lancés avec l'outil de gestion Trello. Trello est un outil collaboratif permettant d'optimiser l'organisation dans un groupe de travail par le biais de la visualisation et de l'assignation de tâches à effectuer. Nous avons très vite remplacé cet outil par les réunions que nous faisions après les différentes rencontres avec monsieur Meynard.

Pour MongoDB nous utilisions au début plusieurs collections reliés entre elles en utilisant un modèle relationnel comme une base de données SQL mais nous avons changé par la suite car ce n'était pas adapté au modèle NoSQL que nous utilisions avec MongoDB. Nous avons donc changé le modèle relationnel en modèle imbriqué, c'est-à-dire une seule collection Evenements qui contient des objets Evenement qui possèdent chacun une liste d'objets Tournoi.

Au début sur toute les pages on utilisait un appel à la base de données pour récupérer les objets Evenement, Tournois, etc puis on a changé ça en appelant seulement sur les pages où c'était nécessaire et en stockant ces données en localStorage pour pouvoir ensuite les récupérer grâce à celui-ci plutôt qu'en faisant des appels tout le temps.

6.1 Conclusion

Du point de vue de ce que nous voulions faire, nous sommes plutôt satisfaits. Le site dispose d'un système d'authentification fonctionnel. Il est stable et nous pouvons créer des événements, des tournois et y inscrire des équipes. La méthode du serpentín a bien été implémentée. Les feuilles de match sont éditables et les résultats peuvent être saisis en temps réel.

Cependant, il reste encore beaucoup de choses que nous aurions pu améliorer. Nous aurions bien voulu ajouter des fonctionnalités comme la possibilité de faire payer des droits d'inscriptions à certains événements ou tournois. Nous aurions aussi pu travailler sur le fait de mettre en ligne le site. Il aurait aussi été logique de continuer à développer l'interface graphique afin d'avoir quelque chose de plus ergonomique et plus attrayant. Nous aurions pu notamment avoir un meilleur affichage concernant les tournois en faisant des arbres.

Ce projet nous a permis tout d'abord d'apprendre à travailler en équipe sur un même projet à partir d'un cahier des charges dont le référentiel a été respecté, et à respecter le délai imparti.

De plus, notre compréhension du monde professionnel et de l'informatique s'est améliorée, nous avons maintenant de meilleures bases et connaissance sur le développement web, le framework Angular et le TypeScript ainsi que MongoDB. Nous avons aussi pu appliquer ce que nous avons vu et appris dans le cours de génie logiciel HAI501I du semestre précédent en utilisant git dans un environnement plus adapté.

7.1 Bibliographie

Liens youtube utiles :

<https://www.youtube.com/watch?v=k8qDGBFPfXk>

<https://www.youtube.com/watch?v=U80kf5QeAF8>

Aide github :

<https://education.github.com/git-cheat-sheet-education.pdf>

Aide Angular / TypeScript :

<https://openclassrooms.com/fr/courses/7471261-debutez-avec-angular>

<https://moodle.umontpellier.fr/course/view.php?id=25893>

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/JSON

<https://guide-angular.wishtack.io/angular>

Aide Mongodb :

<https://mongoosejs.com/>

<https://www.mongodb.com/docs/manual/>

<https://betterprogramming.pub/embedded-vs-referenced-documents-in-mongodb-how-to-choose-correctly-for-increased-performance-d267769b8671>

Aide LaTeX :

<https://latex.developpez.com/faq/>

<https://www.ukonline.be/programmation/latex/index.php>

<https://fr.overleaf.com/>

Autre aide :

<https://stackoverflow.com/>

<https://fontjoy.com/>

Image du modele NOSQL 3.1 :

https://github.com/ValentinRollin/tutti_tourney/tree/main/RestAPI/schemas

Image OnSubmit3.3 :

https://github.com/ValentinRollin/tutti_tourney/blob/main/TuttiTourney/src/app/create-evenement/create-evenement.component.ts

Image createEvenement3.3 :

https://github.com/ValentinRollin/tutti_tourney/blob/main/TuttiTourney/src/app/services/evenement.service.ts

Image du serpent4.2 :

https://github.com/ValentinRollin/tutti_tourney/blob/main/TuttiTourney/src/app/show-poule/show-poule.component.ts

Image page connexion3.2.1 :

https://github.com/ValentinRollin/tutti_tourney/tree/main/TuttiTourney/src/app/connexion-user

Image page inscription3.2.1 :

https://github.com/ValentinRollin/tutti_tourney/tree/main/TuttiTourney/src/app/create-user

Image page accueil3.2.2 :

https://github.com/ValentinRollin/tutti_tourney/tree/main/TuttiTourney/src/app/show-evenements

Image composante barre de navigation 13.2.3 :

https://github.com/ValentinRollin/tutti_tourney/tree/main/TuttiTourney/src/app/navbar

Image composante barre de navigation 23.2.3 :

https://github.com/ValentinRollin/tutti_tourney/tree/main/TuttiTourney/src/app/

navbar

Image page création d'évènement3.2.4 :

https://github.com/ValentinRollin/tutti_tourney/tree/main/TuttiTourney/src/app/create-evenement

Image page création de tournoi3.2.4 :

https://github.com/ValentinRollin/tutti_tourney/tree/main/TuttiTourney/src/app/create-tournoi

Image page de tournoi3.2.4 :

https://github.com/ValentinRollin/tutti_tourney/tree/main/TuttiTourney/src/app/show-evenements

Image d'ajout d'équipe3.2.4 :

https://github.com/ValentinRollin/tutti_tourney/tree/main/TuttiTourney/src/app/create-equipe

Image page création de poule3.2.4 :

https://github.com/ValentinRollin/tutti_tourney/tree/main/TuttiTourney/src/app/organisation-tournoi

Image page listage des poules3.2.4 :

https://github.com/ValentinRollin/tutti_tourney/tree/main/TuttiTourney/src/app/show-poule

Image page gestion de tournoi3.2.5 :

https://github.com/ValentinRollin/tutti_tourney/blob/main/TuttiTourney/src/app/gerer-tournoi/gerer-tournoi.component.ts

image composante de tour suivant3.2.5 :

https://github.com/ValentinRollin/tutti_tourney/tree/main/TuttiTourney/src/app/next-tour

Image de tournoi terminés3.2.5 :

https://github.com/ValentinRollin/tutti_tourney/tree/main/TuttiTourney/src/app/tournois-finis

Image page d'un tour3.2.6 :

https://github.com/ValentinRollin/tutti_tourney/tree/main/TuttiTourney/src/app/voir-tournoi

Image page affichage classement3.2.6 :

https://github.com/ValentinRollin/tutti_tourney/blob/main/TuttiTourney/src/app/next-tour/next-tour.component.ts