UNIVERSITÉ DE MONTPELLIER

FACULTÉ DES SCIENCES

Informatique

FACULTÉ DES SCIENCES

RAPPORT DE PROJET

T.E.R. HLIN601

Site web de gestion de tournoi

Proposé par :

Thomas CAPRIO LOMBARDI & Romain JAMINET & Solal GOLDSTEIN & Louis LAMALLE

Année : 2020

Encadrant :

Michel Meynard

Nous remercions notre encadrant Michel Meynard pour son accompagnement, ses conseils, son expertise aussi bien technique que sur les évènements de volley-ball, ainsi que sa disponibilité pour nous aider a mener à bien ce projet

Table des matières

Introduction

Contexte d'un évènement de volley-ball

Il arrive souvent qu'un club de volley-ball organise un à plusieurs évènements ouverts à tous, que ce soit pendant ou hors saison de championnat. Le but de ces évènements relève souvent d'une ou plusieurs raisons qui peuvent être la découverte de la discipline, la collecte de fonds, etc. Pour permettre d'augmenter la participation d'un maximum de personnes, ces évènements sont généralement constitués de nombreux tournois qui couvrent l'ensemble des catégories. La gestion de ces tournois nécessite donc la mobilisation de beaucoup de personnel (inscription des joueurs, restauration, entretien des terrains, gestion des tournois ...). La gestion des tournois s'effectuant « à la main » elle nécessite donc la présence de plusieurs personnes pour créer les poules, récupérer les scores, calculer les équipes montantes en cas d'égalités etc.

Enoncé du projet

Le projet consiste à développer un site de gestion de tournoi de volley (adaptable à d'autres disciplines) permettant à des organisateurs d'événement (authentifiés) de créer des évènements avec différents tournois auxquels les équipes de volleyeurs peuvent s'inscrire. L'organisateur pourra également influer sur la gestion du tournoi en direct par exemple en modifiant les scores ou en lançant le tour suivant grâce à une interface graphique. La gestion d'un tournoi en temps réel doit pouvoir s'effectuer du coté client pour pouvoir fonctionner sans connection internet.

Elements de motivation

Quelles ont été nos motivations pour avoir choisi ce sujet ? Ce sujet nous a interpellé dans un premier temps car il est en lien avec le domaine du sport. Etant nous quatre adeptes de sports en tout genres, nous avons donc été particulièrement intéressé par ce sujet.

Développer ce site web nous permet tout d'abord de travailler sur un projet mettant en application nos connaissances dans les technologies du web. La motivation est d'autant plus grande que c'est une application concrète dans un domaine qui nous touche. De plus, les contraintes imposées nous permettront de développer des applications web en utilisant les dernières technologies comme Angular et une toute nouvelle logique qui en découle.

Nous pensons que ce projet permet d'expérimenter plusieurs facettes de développement d'un programme collaboratif. Celui-ci satisfaisant les intérêts de chaque membre du groupe. C'est une expérience plus que positive sachant que les technologies utilisées ici sont celles qui sont le plus demandées actuellement en informatique sur le marché du travail.

Toute occasion de mise en pratique des notions que l'on étudie dans le cadre de nos études est une source de motivation et suscite d'avantage l'intérêt que nous portons pour l'informatique, et la programmation en particulier. La connaissance s'acquiert par l'expérience, tout le reste n'est que de l'information. Albert Einstein.

Chapitre 1

Organisation du projet

1.1 Mise en place d'un cahier des charges

En janvier , nous nous sommes réunis pour la première fois tous ensemble avec monsieur Meynard. Le but était de détailler l'intitulé du projet et définir un cahier des charges. Les objectifs de ce projet ont donc été clairs dès le début en sachant que le nombre de fonctionnalités potentielles étaient nombreuses. Le but était donc d'en implémenter un maximum parmis les suivantes :

- Inscription/Connexion pour un utilisateur
- Possibilité de créer un évènent et un tournoi pour un utilisateur authentifié.
- Possibilité de s'inscrire à un tournoi même sans authentification.
- Lancer l'interface graphique d'un tournoi pour l'organisateur de celui ci.
- Permettre à l'organisateur de modifier les scores, la composition des poules, passer au tour suivant.
- Avoir accès à l'historique des tournois terminés pour pouvoir consulter les résultats de ceux-ci
- Un tournoi en cours doit être visible en ligne.

Il y a également d'autres contraintes plus directement liées a l'organisation d'un tournoi de volley :

- Un tournoi doit avoir un format (3 contre 3, 4 contre 4, ...)
- La méthode de calcul du classement au sein d'une poule qui prend en compte, en ordre de priorité : le nombre de victoires, le nombre de sets gagnés et le nombre de points marqués.
- Le système de calcul du tour suivant en fonction du classement des poules, le nombres de qualifiés par poule et le nombre de poules du tour actuel.

La principale contrainte et la plus intéressante était de pouvoir faire fonctionner la gestion d'un tournoi exclusivement côté client pour pouvoir fonctionner sans avoir recours à une connexion internet. En effet, le déroulement de ces tournois nécessitant

un grand espace il n'est pas rare que ceux-ci soient organisés sur une plage ou sur des terrains de foot.

1.2 Méthode de travail

Pour mener à bien ce projet, nous nous y sommes pris en deux temps. Le premier objectif était de se renseigner sur les technologies que nous allions utiliser et de commencer une initiation au travers de documentations et autres tutoriels. Le second, quant à lui était le développement du site web en lui-même.

Nos méthodes de travail ont été profondément modifiées au cours de ce projet à cause de l'épidémie du Covid-19. On peut donc distinguer deux phases de travail.

La première période s'est étalée de mi janvier à mi mars (début du confinement). Durant cette période, nous pouvions parler quotidiennement en présentiel du projet. De plus, nous nous réunissions hebdomadairement pour faire un point sur l'avancée du projet, vérifier le respect des objectifs fixés et donner les nouveaux objectifs pour la semaine suivante. Nous avions accès aux ordinateurs de la fac pour travailler tous ensemble. Nous nous réunissions également toute les 3 semaines environ avec monsieur Meynard pour lui exposer nos avancements.

La seconde période s'est déroulée de mi-mars à fin avril dans des conditions un peu plus originales. Les premiers temps ont été un peu confus au niveau de l'organisation mais au fur et à mesure, nous avons réussi à recréer une dynamique de travail. Nous avons donc divisé le travail entre chaque membre du projet. L'un de nous à développé une API pour l'utilisation du LocalStorage. Un autre a travaillé sur le developpement du site en lui même (partie html et typescript). Le troisième à contribué à faire le lien entre ces deux parties. Le dernier à assuré la gestion du serveur et de la base de données. Nous avons dû exploiter au maximum les outils que nous avions à notre disposition.

1.3 Outils de travail

Les outils informatiques ont été divers et très utiles durant toute la durée de notre projet. L'outil que nous avons le plus utilisé est Discord. Nous pouvions communiquer à l'écrit et partager les ressources dont nous avions besoin. Discord à été particulièrement utile pendant la période de confinement car en effet c'est grâce à cet outil que nous avons pu reconduire le principe des réunions qui se faisaient donc à distance en audio conférence. Le système de partage d'écran était également très précieux pour s'entraider à débugger à distance.

Pour le développement du projet nous avons utilisé l'outil de gestion de logiciels Github. Il nous a permis de faciliter la gestion des différentes versions de notre projet et la gestion de la répartition du travail.

Pour le développement, nous avons utilisé l'éditeur de code Visual Studio Code. Il existe des extensions à installer pour les projets Angular qui permettent l'automatisation de l'importation de fichier ou la création de nouveaux composants.

Pour noter les objectifs nous avons utilisés l'application trello. Nous pouvions également à y noter les dates de réunion.

Pour la rédaction de ce rapport, nous avons utilisé Overleaf qui permet d'éditer du latex en ligne pour les projets collaboratifs.

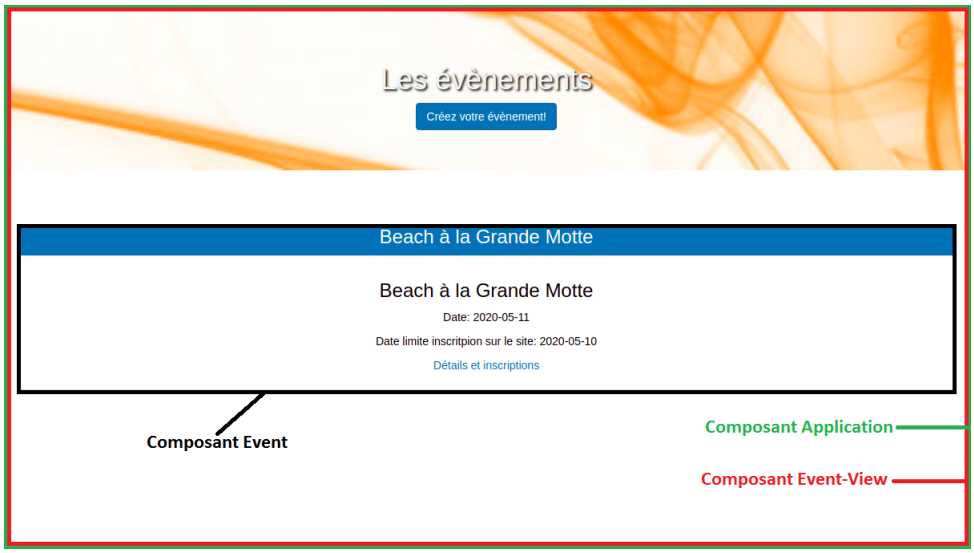
Chapitre 2

Technologies utlisées

2.1 Un site web développé grâce à Angular

Angular est un framework côté client basé sur Typescript. Il a l'avantage d'être multi-plateforme tout en conservant la réalisation de modèles et d'interfaces avec une syntaxe relativement simple. Nous avons utilisé beaucoup d'éléments d'Angular mais nous avons principalement utilisé :

Les composants (on utilise souvent le mot anglais components) qui sont les composantes de base d'une application Angular. Ils sont composés d'un template (fichier .html), d'un fichier TypeScript et de leur feuille de style scss. A leur création ceux-ci se voient attribués d'un nom (sélecteur) que l'on utilisera comme balise HTML pour les afficher. Une application angular est donc une arborescence de plusieurs composants.



Les services, qui permettent de centraliser du code et des données utilisés par plusieurs composants. Ils possèdent différents niveaux d'injection qui déterminent leur accessibilité par les différentes parties de l'application. Ici nous avons utilisé une injection simple (dans AppModule) qui permet d'utiliser la même instance d'un service par tous les composants de l'application.

La puissance d'Angular réside beaucoup dans l'interaction entre le code TypeScript et le code HTML. On l'appelle la liaison de données et elle peut s'effectuer dans les deux sens.

L'interpolation représente une des façons de lier des données du code Typescript vers le template, réalisable facilement à l'aide d'une double accolade autours d'un attributs de l'objet typescript.

```
<div class="card-header">
    <h3> {{ name }} </h3>
</div>
```

La liaison par propriété est une autre façon de dynamiser la communication en modifiant les propriétés d'un éléments du DOM en fonction du TypeScript. Il suffit d'utiliser les crochets autours d'une propriété, de la forme `[disabled]= «propriété»`

```
<button class="btn btn-primary" type="submit" [disabled]="eventForm.invalid">
```

La liaison par évènements pour communiquer du template HTML vers le code TypeScript.

```
<button class="btn btn-success" type="button" (click)="onAddTournament()">
```

Ici entre parenthèses on retrouve l'évènement click de la souris, qui déclenchera l'appelle à la méthode présente dans le TypeScript. Enfin, on peut utiliser la liaison à double sens (dite two-way-binding), qui utilise les deux liaisons précédentes en même temps. Pour cela il suffit d'utiliser les deux syntaxes des liaisons précédentes, [()] et la directive `ngModel` que nous verrons ensuite.

```
<input placeholder="Nom de l'évènement" type="text" class="form-control"
formControlName="name" [(ngModel)]="name">
```

Si on combine cet exemple avec la liaison de donnée par interpolation vu précédemment, l'utilisateur peut observer le texte changer au fur et à mesure qu'il écrit.

Les directives, qui sont divisées en deux catégories. Dans un premier temps les directives dites structurelles car elles modifient la structure du document (DOM). On retrouve dans cette catégorie les directives `*ngIf` ou encore `*ngFor`, qui sont directement écrites dans le template d'un composant (Voir partie 3.1.2 Code pour afficher l'ensemble des poules d'un round, qui illustre l'utilisation de `*ngFor`).
Ensuite, nous avons les directives par attributs, qui modifient le comportement d'un objet déjà existant. Par exemple nous avons ngStyle, qui modifiera le style d'un objet ou ngModele que nous utilisons pour le two-way-binding.

Le routing, qui nous permet de créer une "single page application" (qui est une page qui donne la sensation d'utiliser une application native). On l'utilise grâce à l'URL qui nous permet d'afficher les composants désirés. Après l'import de différents modules, il suffit de créer ce que l'on appelle des routes, qui sont composées d'un URL, du composant à afficher et parfois d'un attribut canActivate pour protéger l'accès à certaines pages – ici utilisé pour empêcher l'utilisateur non connecté de s'inscrire à un tournoi – et donc complexifier la navigation.

```
{ path: 'profile', canActivate:[ConnectionGuard], component: UserProfileComponent }
```

Un exemple de route suivit d'une des manières de naviguer à travers avec l'utilisation de *routerLink*.

```
<li routerLinkActive="active"><a routerLink='home'>Accueil</a></li>
<li routerLinkActive="active"><a routerLink='events'>Evénements</a></li>
<li routerLinkActive="active"><a routerLink='profile'>Profil</a></li>
```

Les formulaires réactifs, qui sont l'une des deux façon de faire des formulaires avec Angular est la seul que nous détaillerons ici. Cette manière de faire permet de complexifier les formulaires et de par son nom, les rendre réactif aux interactions de l'utilisateur. On utilise le type FormGroup et des classes associées comme FormBuilder pour mettre en place le formulaire.

```
initForm(){
  this.eventForm = this.formBuilder.group({
    name: ['',[Validators.required]],
    dateEv: ['',[Validators.required]],
    dateLimite: ['',[Validators.required]],
    tournaments: this.formBuilder.array([]),
    description: ['',[Validators.required, Validators.maxLength(500)]]
  });
}
```

Ici on peut voir l'utilisation d'une méthode *group()* de la classe formBuilder qui nous permet de définir un formulaire et les différents inputs qui seront présents.
Dans le template, on remarque la propriété *formControleName* qui nous permet de lier l'input à la propriété du formulaire vu dans la méthode précédente.

```
<div class="form-group">
  <label style="margin-top: 2rem" for="name">Nom de l'événement
    <input placeholder="Nom de l'évènement" type="text" class="form-control" formControlName="name"
      [(ngModel)]="name">
  </label>
</div>
```

Il va de soit que nous avons utilisé d'autres éléments d'Angular mais nous ne résumons ici que l'essentiel.

2.2 Pour gérer la partie hors-ligne : le Local Storage

La gestion d'un tournoi quand celui-ci est lancé doit pouvoir se faire lorsque l'équipement sur lequel on l'effectue est hors-ligne : lorsqu'un événement se déroule sur une plage ou des terrains extérieurs il n'est pas forcément évident qu'il y est une connexion disponible. Javascript permet d'utiliser très simplement une base de données qui stock celles-ci directement chez le client : le local storage. La base de données est assez simple puisqu'elle est de la forme clef/valeur.

Au niveau de l'utilisation, rien de plus simple, pour stocker une valeur on utilise :
window.localStorage.setItem('clef',valeur);
pour récupérer la valeur :
window.localStorage.getItem('clef');

Pour observer le Local Storage, il suffit d'ouvrir la console javascript/developpeur du navigateur, puis d'aller dans storage et de sélectionner Local Storage (cela peut dépendre des navigateurs).

2.3 Une base de données en ligne grâce au SQL

Pour un enregistrement des données et une réutilisation de celles-ci à distance, nous avons dû utiliser une base de données en ligne. Plusieurs choix s'offraient à nous (Oracle, MySQL, MongoDB...) mais cependant nous avons opté pour MySQL car l'utilisation de celle-ci nous était beaucoup plus familière que les autres et nous a permis de gagner beaucoup de temps.

MySQL est un système de gestion de bases de données relationnelles qui utilise le langage SQL. MySQL peut donc s'utiliser seul mais est la plupart du temps combiné un autre langage de programmation (pour le projet nous avons choisi de le combiner au PHP).

2.4 Une synchronisation des données grâce au PHP

Pour la récupération des données, nous avons décidé d'utiliser le PHP car il s'agit d'une technologie que nous maitrisons aisément pour effectuer des requêtes SQL notamment grâce au pilote PDO_MYSQL.

PHP est un langage de programmation interprété par le serveur et est principalement utilisé pour produire des pages Web dynamique via un serveur HTTP.

Chapitre 3

Implémentation du site de gestion de tournois

Pour nous donner une première idée des objectifs à atteindre, nous avons réalisé un diagramme de cas d'utilisation pour un utilisateur classique ainsi que pour le cas spécifique d'un organisateur d'évènement.

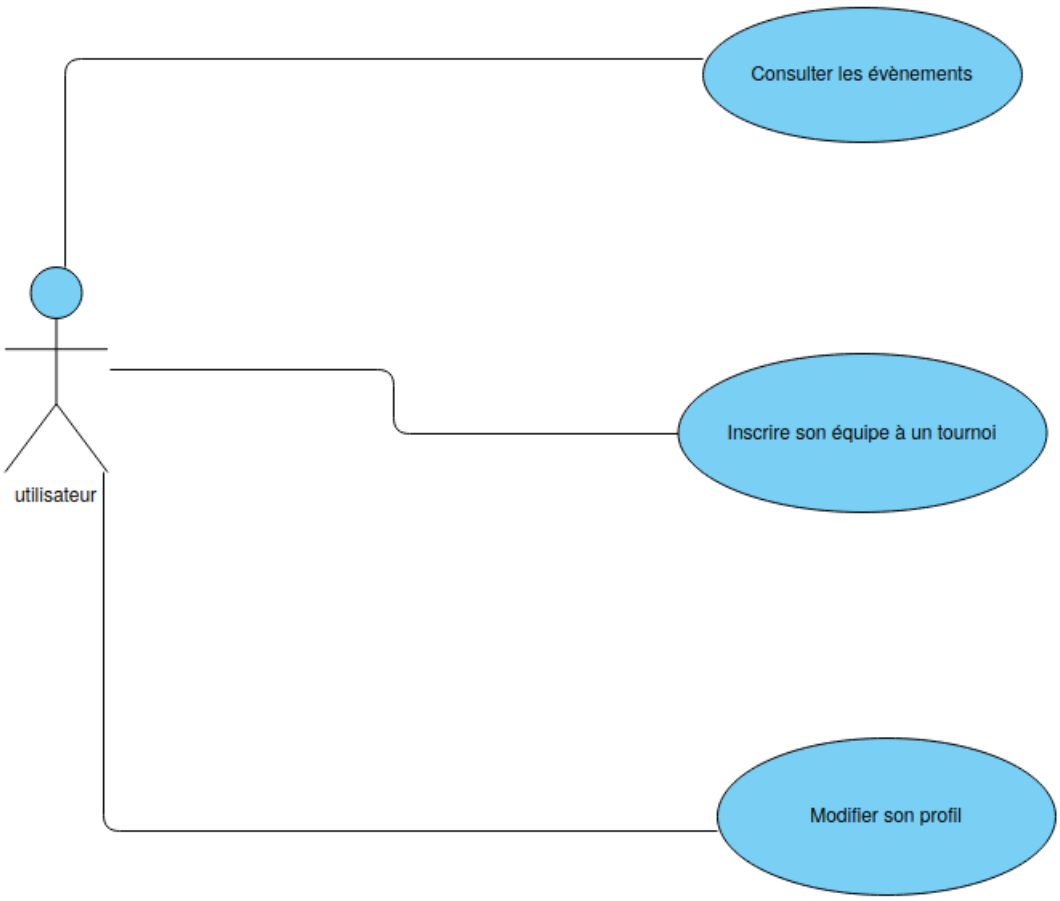


Diagramme pour un utilisateur

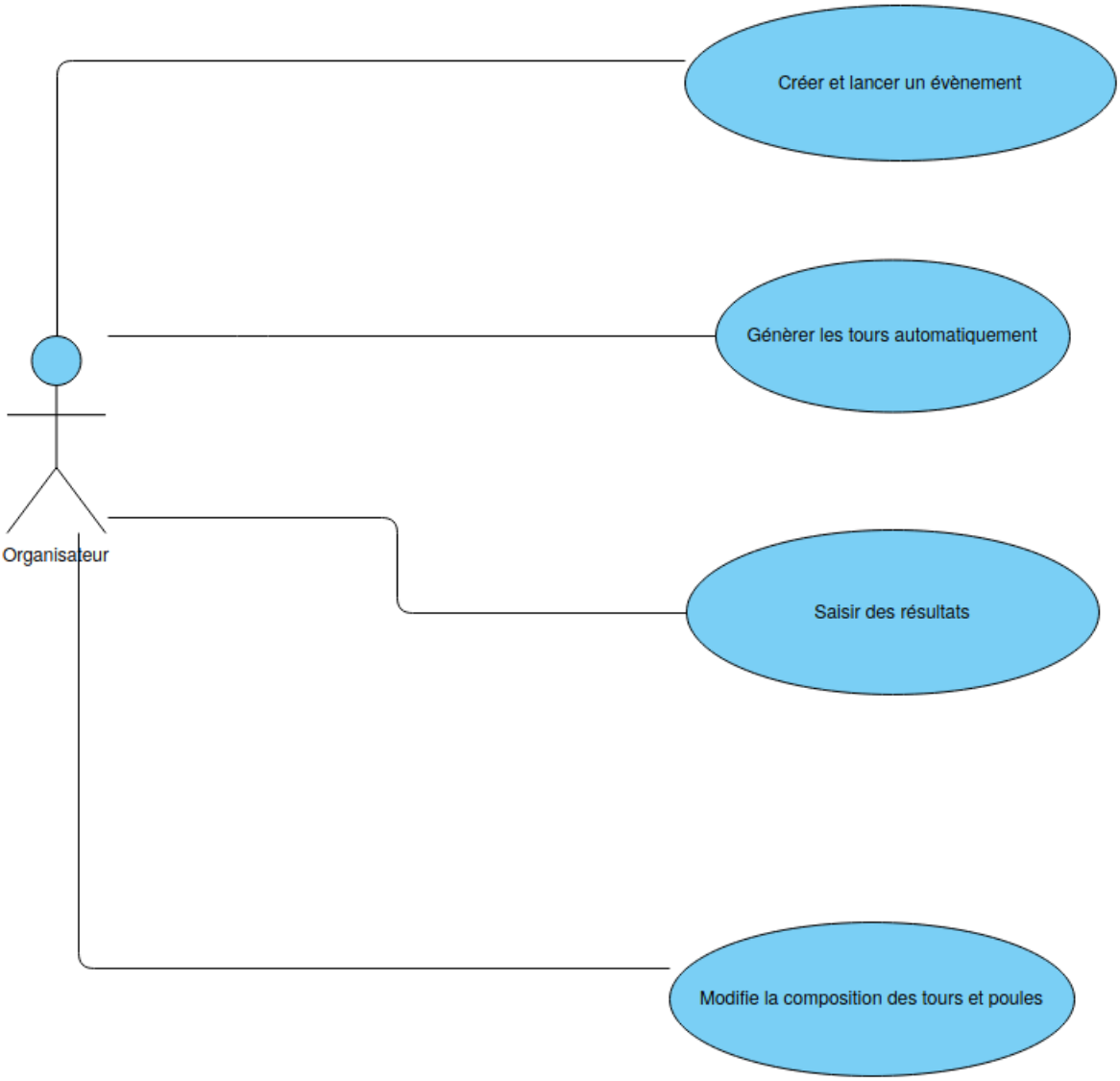


Diagramme pour un organisateur

3.1 Utilisation du Local Storage dans le projet

3.1.1 Utilisation générale

Mais concrètement dans notre cas, comment l'utiliser ? Ici nous détaillerons le fonctionnement primaire du service localStorage (du fichier localstorage.service.ts). Nous avions besoin de pouvoir stocker une liste de tournois (représentant tous les tournois d'un événement) dans une seule valeur. Pour cela nous avons décidé d'utiliser des tableaux, car avec eux on peut accéder à chaque élément sans avoir besoin de connaitre la clef qui contient la valeur (comme pour un objet par exemple).

Voici comment nous avons organisé ce tableau :

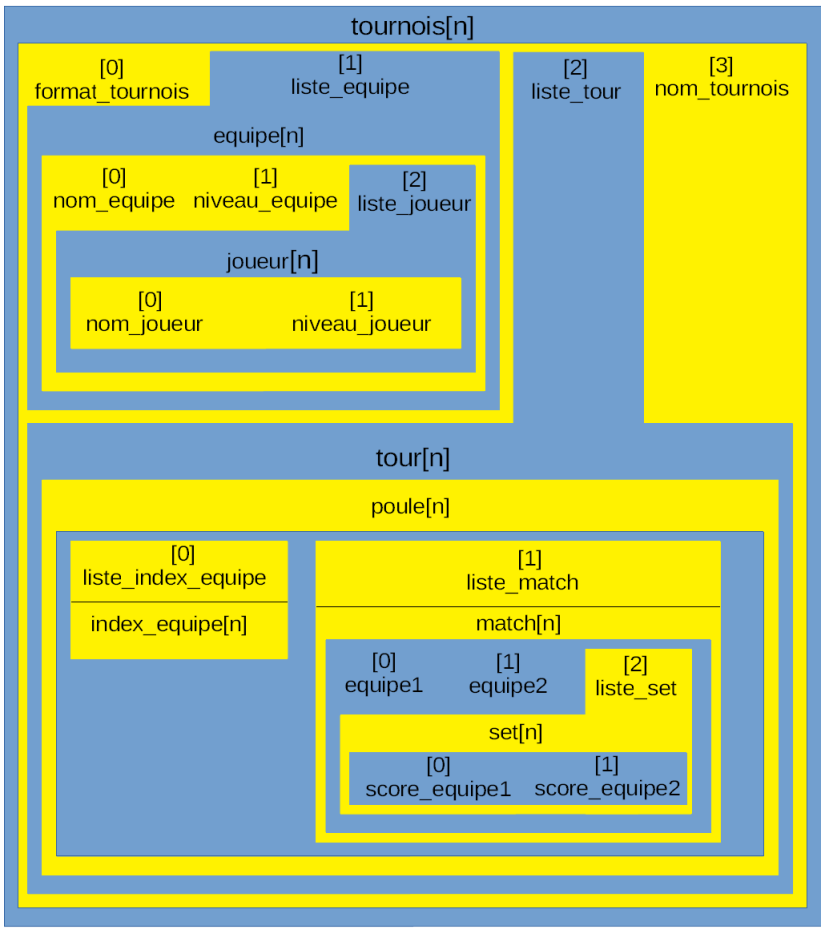


FIGURE 3.1 – Organisation du tableau en localstorage

Pour mieux comprendre voici un exemple (en imaginant que la variable contenant le tableau se nomme event) : si nous voulons le nom du premier joueur de la seconde équipe du 3eme tournois, nous y accédons avec event[2][1][1][2][0][0].

Pour accéder aux différentes parties du tableau, nous utilisons des méthodes toutes nommées avec le préfixe get. Comme *getRounds(String nomTournois)*, qui renvoie tous les rounds d'un tournoi passé en paramètre, ou *getPoolFromRound(String nomTournois, int numeroRound, int numeroPool)*, qui renvoie la poule demandée en paramètre.

Nous avons donc notre tournoi, pour ce qui est de son déroulement (et donc du remplissage du tableau) nous le verrons dans la partie suivante. Avant d'insérer le tableau dans le Local Storage nous utilisons une méthode de JSON qui permet de changer un tableau/objet en une chaine de caractères (et inversement). Ainsi avant d'insérer le tableau on utilise JSON.stringify(). Pour ensuite utiliser le tableau extrait du Local Storage on utilise la méthode JSON.parse()

3.1.2 Exemple d'une méthode

Ici nous allons détailler le fonctionnement de la méthode addRoundAutomate() qui permet d'ajouter automatiquement un round constitué de poules de deux équipes lorsque le round précédent est considéré comme terminé (toutes les méthodes du service localstorage.service.ts sont détaillées dans le fichier /documents/patchnotes-localstorageservice/README). Voici son code :

```
292   addRoundAutomate(tournamentName){
293     if(this.getRounds(tournamentName).length == 0){
294       this.addRound(tournamentName);
295       if(this.getTeams(tournamentName).length%2 == 0){
296         for(let i=0;i<this.getTeams(tournamentName).length/2;i++){
297           this.addPoolToRound(tournamentName,0);
298           this.addTeamToPool(tournamentName,2*i,0,i);
299           this.addTeamToPool(tournamentName,2*i+1,0,i);
300         }
301       }
302       else{
303         for(let i=0;i<(this.getTeams(tournamentName).length-1)/2;i++){
304           this.addPoolToRound(tournamentName,0);
305           this.addTeamToPool(tournamentName,2*i,0,i);
306           this.addTeamToPool(tournamentName,2*i+1,0,i);
307         }
308         this.addTeamToPool(tournamentName,this.getTeams(tournamentName).length-1,0,0);
309       }
310       return true;
311     }
312     else{
313       if(this.roundEnded(tournamentName,this.getRounds(tournamentName).length-1)){
314       let qualifiedTeam = this.calcQualifiedTeamRoundAutomate(tournamentName,this.getRounds(tournamentName).length-1);
315
316       this.addRound(tournamentName);
317       if(qualifiedTeam.length%2 == 0){
318         for(let i=0;i<qualifiedTeam.length/2;i++){
319           this.addPoolToRound(tournamentName,this.getRounds(tournamentName).length-1);
320           this.addTeamToPool(tournamentName,qualifiedTeam[2*i],this.getRounds(tournamentName).length-1,i);
321           this.addTeamToPool(tournamentName,qualifiedTeam[2*i+1],this.getRounds(tournamentName).length-1,i);
322         }
323       }
324       else{
325         for(let i=0;i<(qualifiedTeam.length-1)/2;i++){
326           this.addPoolToRound(tournamentName,this.getRounds(tournamentName).length-1);
327           this.addTeamToPool(tournamentName,qualifiedTeam[2*i],this.getRounds(tournamentName).length-1,i);
328           this.addTeamToPool(tournamentName,qualifiedTeam[2*i+1],this.getRounds(tournamentName).length-1,i);
329         }
330         this.addTeamToPool(tournamentName,qualifiedTeam[qualifiedTeam.length-1],this.getRounds(tournamentName).length-1,0);
331       }
332       return true;
333     }else{
334       return false;
335     }
336     }
337   }
```

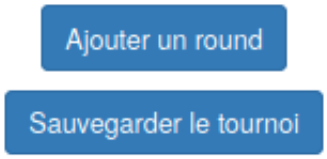
La première condition permet de savoir si nous sommes au premier tour (round) du tournoi, pour cela on regarde la taille du tableau contenant les tours, elle doit valoir zéro. Si c'est le cas on ajoute un round. Ensuite, on regarde si le nombre d'équipes est pair (puisqu'on veut former des poules de deux). Si c'est le cas on parcourt les équipes du tournoi deux par deux, on crée une poule à chaque fois et on y ajoute les deux équipes. Lorsque le nombre d'équipe est impair, on fait la même chose en ajoutant la dernière équipe dans la première poule. On renvoie true.

Dans l'éventualité où ce n'est pas le premier round, on regarde d'abord si le round actuel est terminé avec la méthode *roundEnded()*. Si c'est le cas on fait la liste des équipes qualifiées avec *calcQualifiedTeamRoundAutomate()* (qui prend les équipes présentes dans la moitié supérieure du classement de chaque poule). Ensuite on ajoute un round puis on déroule de la même façon que lors du premier round mais avec la liste des équipes qualifiées. Pour finir on renvoie true. Sauf si le tour n'est pas terminé, on renvoie false.

3.2 Déroulement d'un tournoi

3.2.1 Explications

Etape 1 : Au départ, la page du tournoi est vide. Il faut lancer le premier round en utilisant le bouton "ajoutez un round" qui créé le round automatiquement en fonction du nombre d'équipes inscrites. La répartition standard est de faire des poules de 2 ou de 3. Ceci est fait grâce à l'appel de la méthode addRoundAutomate du service localStorage.

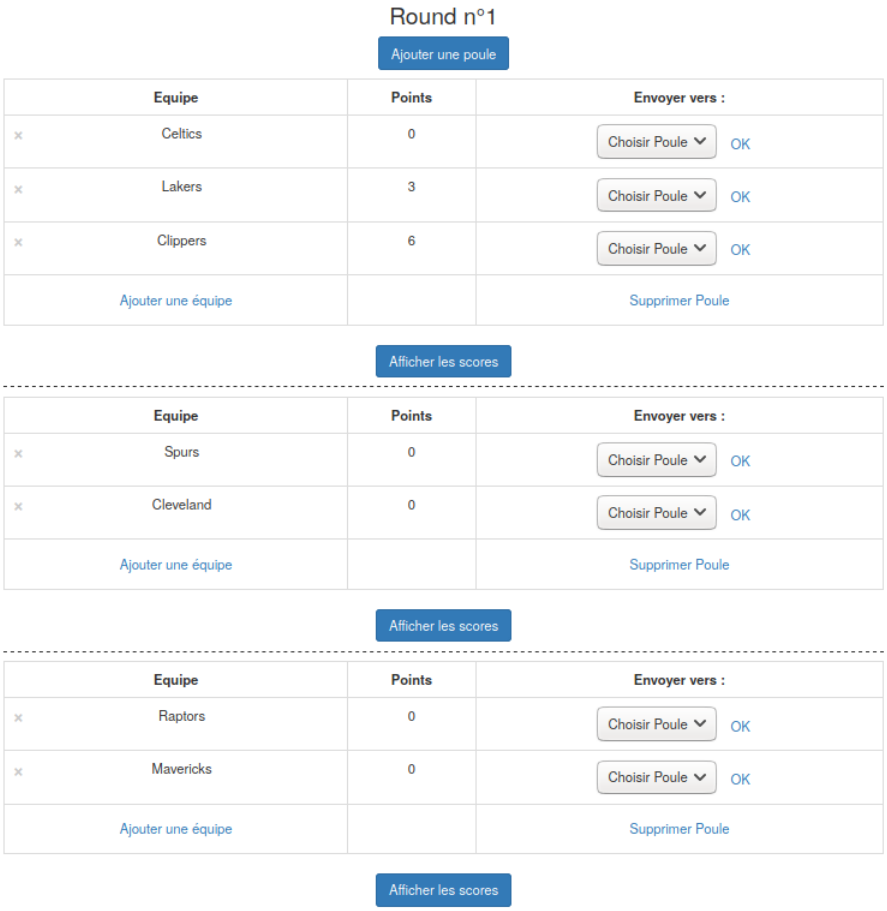


Etape 2 : L'organisateur peut amménager l'organisation du round. Il peut :

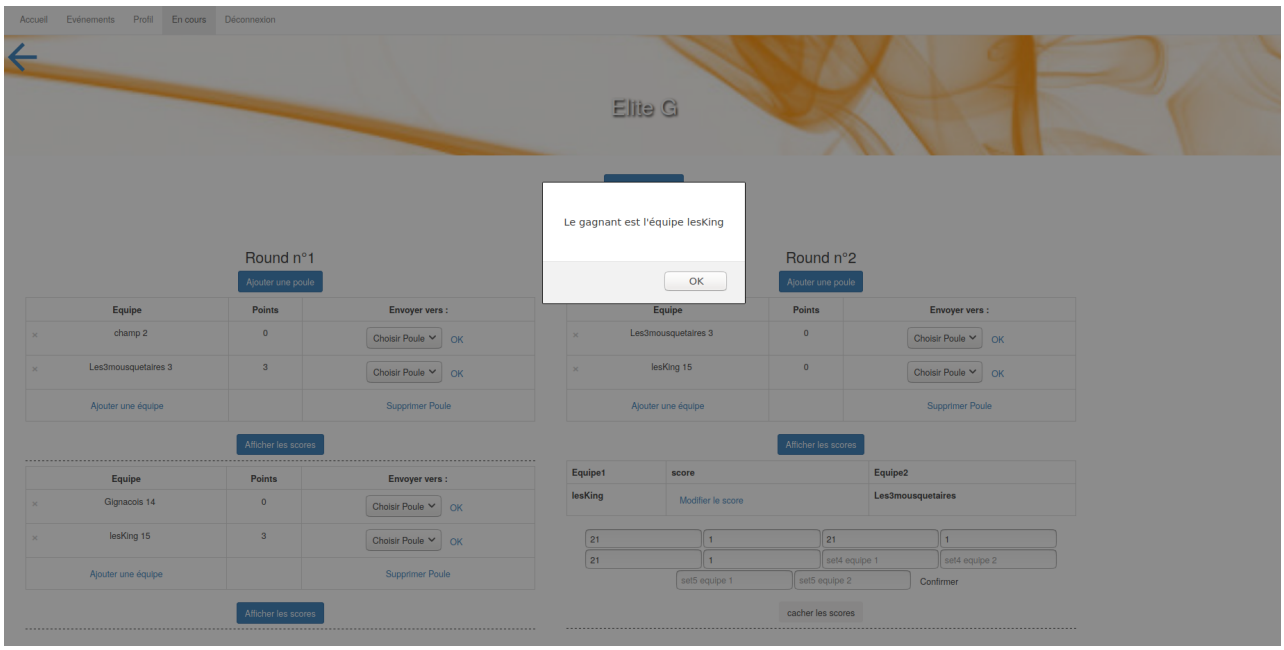
— Ajouter une poule avec le bouton "Ajouter une poule" (appel de la méthode addPoolToRound).
— Supprimer une poule avec le bouton "supprimer poule" (appel de la méthode delPool)
— Ajouter une équipe à une poule avec le bouton "Ajouter une équipe" (appel de la méthode addTeamToPool)
— Supprimer une équipe avec la croix à gauche de chaque équipe (appel de la méthode delTeamFromPool)

— Changer une équipe de poule avec le petit formulaire à droite de chaque équipe (appel de la méthode moveTeamFromPool).
— Afficher les scores avec le bouton "afficher les scores" (apparition d'un nouveau composant set-results pour modifiez les scores).
— Modifiez les scores en remplissant le formulaire du composant set-results.

Lorsque tout les scores des matchs du round ont été entré, on peut de nouveaux cliquer sur le bouton "Ajouter un round". Le nouveau round est automatiquement créé en fonction des résultats du round précédent. On répète cette étape jusqu'a avoir un dernier round de 2 équipes. C'est la finale



Etape finale : On rentre les scores de la finale toujours de la même manière. Lorsqu'on valide la modification des scores, un message est affiché pour annoncer l'équipe qui remporte le tournoi. Pour cela, nous avons un emetteur d'évènement qui émet un message lorsque le résultat de la final est saisi. A la réception de ce message, le composant parent récupère le vainqueur du tournoi et procède à l'affichage.



3.2.2 Code pour afficher l'ensemble des poules d'un round

Pour afficher dynamiquement un nombre de composant qui sera variable dans le temps, Angular met à disposition la directive *ngFor. Grâce à cette directive, nous pouvons boucler sur le tableau contenant les poules d'un round et afficher autant de composant "poule" qu'il y en à dans ce round.

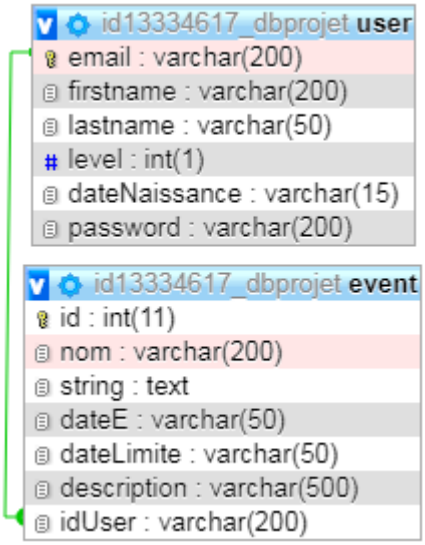
```
<!--------------------------------tableau des poules---------------------------------------->
<div style="margin-top:1%; border-bottom: dashed 1px" *ngFor="let item of pouleArray;index as i">
    <app-poule (refreshPool)=(refreshRound())  [numRound]=this.numRound [numPoule]=i></app-poule>
</div>
```

Les attributs entre crochets sont les propriétés créées spécialement pour le composant poule. Ici, nous transmettons l'information du numéro de round et du numéro de la poule au composant poule que nous venons de créer. Le (refreshPool) est un écouteur d'évènement. C'est à dire que lorsque un émetteur d'évènement refreshPool d'un composant poule émet un signal, la fonction refreshRound est appelée.

3.3 Récupération des données du serveur

3.3.1 La base de données

Pour conserver et pouvoir récupérer et modifier les données, nous avons conçu une base de données qui s'apparente le plus à nos besoins. Celle-ci est hébergée en ligne ce qui nous a permis de l'utiliser à distance sans difficulté. La base de données à été modélisé de la façon suivante :



L'attribut string de la table event contient un json correspondant à la modélisation des différents tournois.

3.3.2 L'API

Pour communiquer depuis le client à la base de données, nous avons décidé de faire une API, que nous avons programmée en PHP, elle nous permet de traiter une requête et d'afficher sa réponse dans une page. Pour utiliser l'API il faut envoyer une requête HTTP en POST ou en GET avec comme information une chaîne de caractères qui correspond à la requête SQL et le serveur renvoi la réponse.

— SELECT : Le serveur renvoi la réponse
— UPDATE-DELETE-INSERT : Le serveur renvoi le nombre de lignes modifiées
— Erreur : Le serveur renvoi un message/code d'erreur

Voici une portion du code que nous utilisons pour traiter les requêtes :

```php
if (strtoupper(substr($_REQUEST['request'],0,6)) == "SELECT") {
    $a = "";
    $a = $a."[";
    foreach ($dbh->query($_REQUEST['request']) as $row) {
        $a = $a.json_encode($row);
        $a = $a.",";
    }
    $a = substr($a, 0, -1);
    $a = $a."]";
    print($a);
} else {
    $req = $dbh->exec($_REQUEST['request']);
    echo $req.' ligne(s) modifiée(s).';
}
```

Si la requête est un "SELECT", alors la réponse renvoyée par le serveur sera formatée en json pour que le client puisse la traiter facilement. Nous avons conscience qu'une API comme celle-ci est faillible et donc est dangereuse pour une utilisation réelle, si nous avions eu plus de temps nous aurions pu songer à rendre cette API plus robuste, par

exemple avec un système d'authentification complexe, ou avec un système qui interdit la récupération et la modificaition de certaines données.

3.3.3 Le service request

La récupération des données côté client se fait à l'aide du service "request.service.ts" que nous avons créé. Pour cela nous utilisons la technologie Ajax présente dans JavaScript qui nous permet d'effectuer des requêtes HTTP et de récupérer la réponse. Nous utilisons un objet `XMLHttpRequest` que nous paramétrons afin d'envoyer une requête HTTP en POST. La requête que nous envoyons est synchrone, c'est à dire que le programme attend la réponse avant de continuer son execution. Nous avons fait ce choix car elle est utilisée dans une méthode nécessitant une réponse avant d'être retournée. Cela peut éventuellement ralentir l'application du client, cependant la méthode est simple et est fonctionnelle, mais si nous avions eu plus de temps, nous aurions pu nous occuper de l'optimisation.

Chapitre 4

Bilan et difficultés rencontrées

4.1 Etat actuel du projet

Au jour du 30 avril, date de fin du projet, un certains nombre de fonctionnalités du cahier des charges ont été implémentées :

- Possibilité de s'inscrire et se connecter avec un identifiant et un mot de passe
- Possibilité de créer un évènement lorsque l'on est connecté
- Possibilité d'ajouter des tournois dans des évènements avec un certain format
- L'utilisateur ayant créé le tournoi peut le lancer. A partir de ce moment il peut modifier les scores, changer les équipes de poule (pour une gestion plus personnalisée), calculer le tour suivant en fonction du résultat du tour actuel, ajouter des équipes et ajouter des poules. L'organisateur peut aussi enregistrer l'état actuel de son tournoi dans la base de données même s'il n'est pas terminé.
- Un utilisateur connecté peut inscrire son équipe à un tournoi préalablement créé. Il peut y ajouter des joueurs en renseignant le niveau de chaque joueur. Le niveau moyen de l'équipe sera calculé en fonction des notes des joueurs. Ce niveau est à titre informatif qui peut être utilisé pour la composition des poules.
- Un utilisateur peut consulter et modifier son profil à tout moment.

De plus, grâce à la technologie d'angular, la plupart des opérations sont faites du côté client. Cela permet d'avoir une navigation plus fluide en évitant les chargements de pages.

Le projet à été décomposé en 20 composants et 8 services. Sans compter ceux automatiquement créés (entièrement ou partielement) par Angular, nous avons édités plus de 70 fichiers, le plus important étant le service Local Storage comprenant plus de 600 lignes de codes.

4.2 Perspectives d'amélioration

Nous sommes plutôt fièrs du résultat final de notre projet qui à demandé plus de travail que nous le pensions. C'est pourquoi certaines fonctionnalités que nous aurions aimé introduire ne sont pas présentes :

- La mise en ligne automatique des résultats d'un tournoi en cours dès que la connection le permet.
- Le calcul du tour suivant en fonction du tour actuel est standardisé. C'est à dire que l'on ne peut pas choisir le nombre de qualifiés par poule ainsi que le nombre de poules du prochain tour. Par défaut, le nombre de qualifiés par poule correspond à la moitié du nombre d'équipes dans une poule.
- La répartition automatique des équipes dans les poules ne prend pas en compte le niveau des équipes ce qui empêche d'avoir des poules équilibrées.
- Actuellement, lors d'un tournoi en cours, on peut changer une équipe de poule grâce à un formulaire. Avec plus de temps, nous aurions intégré un système de drag and drop pour déplacer une équipe d'une poule à une autre.
- Un affichage plus diversifié au niveau de la page d'accueil, avec les évènements passés il y a peu et les évènements avec une date proche.
- Un enregistrement plus détaillé des tournois dans la base de données pour nous permettre d'afficher les équipes auxquelles le joueur connecté est inscrit.
- Une meilleure gestion de l'intégrité des données.

4.3 Difficultés rencontrées

Ce projet à été passionnant et nous a apporté beaucoup de connaissances mais il n'a pas été réalisé sans quelques embûches aussi bien techniques qu'humaines.

4.3.1 Difficultés techniques

La première difficulté technique à été la prise en main du framework angular. Il a fallu changer notre vision de la structuration d'un site web en décomposant au maximum les pages en composants. Cette approche, qui ressemble aux langanges orientés objets, nous à permis de développer un site plus efficacement par exemple avec une réutilisation de composants et la centralisation des fonctions nécessaires au sein de services.

La seconde difficulté rencontrée concernait le local storage. Il a fallu en effet apprendre à modifier et récupérer des informations de ce stockage. Nous avons également dû beaucoup travailler sur la structure de données que nous allions stocker. Le

but était de trouver une structure manipulable assez facilement tout en étant suffisament "compacte" pour pouvoir la stocker dans la base de données.

La dernière grande difficulté à été de récupérer les données d'une page web avec angular. En effet certaines méthodes courantes en javascript classique n'étaient pas celles à utiliser ici. Il a fallu donc bien adopter les pratiques propres à angular.

4.3.2 Difficultés humaines

Au delà des difficultés techniques, nous avons surtout rencontré, comme beaucoup durant ce semestre, de vraies difficultés humaines. En effet, notre enthousiasme à pris du plomb dans l'aile lors des premières semaines de confinement. Nous étions tous un peu désorientés et affectés psychologiquement par la situation. Le travail, surtout de groupe, est beaucoup plus complexe lorsque l'on se retrouve isolé dans son appartemment et cette difficulté supplémentaire est loin d'être négligeable.

La communication à également été difficile au début de cette période - ce qui n'a pas aidé les quelques désaccords quant à l'implémentation de certaines parties du projet - mais nous avons réussi à nous adapter et retrouver de l'efficacité dans notre travail.

Conclusion

Finalement, ce projet nous a aidé à progresser sur beaucoup de points notamment sur la gestion d'un projet et la communication. Si on ajoute à cela les connaissances techniques que nous avons pu acquérir, on peut dire que ce TER aura été une expérience très enrichissante pour chacun d'entre nous. Les contraintes de langages nous on permit d'élargir nos connaissances dans le domaine du Web, celui-ci étant non négligeable en informatique. Nous sommes plutôt satisfait d'avoir atteint ce résultat malgré les différentes contraintes. Avec quelques ajouts et corrections, nous pensons que cette application pourrait être prise au sérieux par les organisateurs d'évènements sportifs.

Vous trouverez dans la bibliographie un lien menant au github de notre projet ainsi qu'une vidéo de démonstration d'utilisation de notre site.

Bibliographie

— Lien github : `https://github.com/Skual7/hlin601.git`
— OpenClassrooms : Developper des applications Web avec Angular `https://openclassrooms.com`
— Bootstrap : `https://getbootstrap.com/`
— Angular : `https://angular.io/docs`