# DOCUMENTATION TECHNIQUE DE L'IMPLÉMENTATION

## 1. Frontend - Tournament WebApp

Le frontend a été développé avec Vue.js version 3.2, en utilisant une architecture classique basée sur des composants modulaires.

### Structure des fichiers (src/) :

- App.vue : composant principal contenant la structure globale (header + navigation dynamique + <router-view>).
- main.js : fichier d'entrée qui configure Vue Router et importe le store d'état global.
- Components/ :
  - Login.vue : formulaire d'authentification utilisateur.
  - Register.vue : formulaire d'inscription.
  - AdminDashboard.vue : interface complète d'administration (gestion des équipes, joueurs, matchs, standings).
  - Matches.vue : affichage public des matchs.
  - Standings.vue : affichage du classement.
  - Teams.vue : gestion des équipes pour les admins et les coachs.
  - Player.vue : gestion des joueurs pour les coachs.
- router/ :

  index.js : configuration des routes principales et sécurisation par rôle utilisateur.

- store.js :

  Fichier centralisé pour stocker les informations d'authentification (isLoggedIn, username, role, etc.).

### Communication avec le backend :

- Utilisation d'Axios pour envoyer les requêtes HTTP (GET, POST, PUT, DELETE) vers le serveur Express.js.
- Toutes les requêtes sont asynchrones (async/await) pour garantir une navigation fluide sans blocage.

### Sécurité côté client :

- Token d'authentification JWT stocké dans localStorage.
- Navigation sécurisée selon le rôle de l'utilisateur (admin, coach).

# 2. Backend - API Node.js Express

Le backend a été construit avec Node.js et Express.js, en suivant une structure modulaire.

### Structure des fichiers (backend/) :

- server.js : point d'entrée du serveur, avec configuration de CORS, parsing JSON et routes API.
- db/ :
    - db.js : connexion à la base de données MySQL.
- routes/ :
    - users.js : gestion de l'authentification, de l'inscription et des utilisateurs.
    - teams.js : gestion CRUD des équipes.
    - players.js : gestion CRUD des joueurs.
    - matches.js : création, mise à jour et suppression des matchs.
    - standings.js : gestion et mise à jour automatique du classement.

### Sécurité backend :

- Utilisation de bcrypt pour hasher les mots de passe avant enregistrement dans la base.
- Middleware pour vérifier et valider les entrées utilisateurs avant traitement.

# 3. Base de données - MySQL

La base de données suit un modèle relationnel simple pour assurer efficacité et évolutivité.

### Tables principales :

- Users (userID, username, password, role)
- Teams (teamID, name, coachID)
- Players (playerID, name, teamID)
- Matches (matchID, team1ID, team2ID, result, matchDate)
- Standings (teamID, tournamentID, points, ranks)

Les relations sont cohérentes :

- Chaque joueur appartient à une équipe.
- Chaque équipe est liée à un coach.
- Les matchs lient deux équipes.
- Le classement est mis à jour automatiquement selon les résultats des matchs.

# 4. Hébergement local

Frontend lancé avec :

**npm run serve**

Backend lancé avec :

**node server.js**

Le frontend écoute sur le port 8081 et envoie les requêtes API au backend écoutant sur le port 3000 (avec une gestion CORS configurée pour autoriser la communication).