# README : Application Mobile DataFive :

## Présentation du projet :

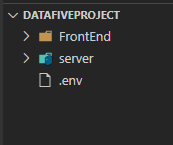
Le projet consiste en quelques lignes à créer une application Mobile DataFive, afin de pouvoir récupérer et afficher les statistiques des différents matchs auxquels le joueur va participer, mais aussi de pouvoir créer des matchs et des réservations depuis l’application mobile même.

Cependant le projet n’est pas totalement abouti, donc ses fonctionnalités ne sont disponible qu’en Local malheureusement.

1. Faire fonctionner le projet en question :

Afin de lancer le projet, il faut tout d’abord se rendre sur GitHub grâce à ce lien si vous êtres collaborateurs et de télécharger le projet en question.

L’application Mobile se définit par 2 dossiers importants :



Le premier dossier se nommant « Front-End » et le second « Server », le côté Front-End est fait en React-Native avec plusieurs bibliothèques d’importées et le côté « Serveur » est fait en Node JS (plus précisément en Express JS).

-La première étape est d’installer NodeJS sur votre ordinateur afin de ne pas rencontrer de soucis.

-La seconde étape est de cloner le projet sur votre ordinateur et de l’ouvrir avec VS Code ou avec votre IDE favori.

Sur le terminal du PC ou sur celui de VS Code, dirigiez vous vers le dossier Front-End en entrant « cd Front-End », une fois dans le répertoire du Front, vous devez exécuter la commande « npm install » afin d’installer toutes les dépendances liées au projet afin que rien ne soit oublié.

Note : Cette application mobile a été gérée par Expo, si vous voulez la tester, assurez vous de télécharger Expo sur votre téléphone et de vous connecter au même réseau que votre ordinateur.

-La troisième étape consiste alors de faire la même chose mais du côté serveur en tapant « cd server » et ensuite « npm install ».

-Une fois toutes les dépendances installées, il ne reste plus qu’à lancer le projet du coté Front-End (qui lui est géré par Expo, il faudra alors télécharger Expo sur votre smartphone afin de pouvoir scanner le QR Code) en tapant la commande « npm start ».

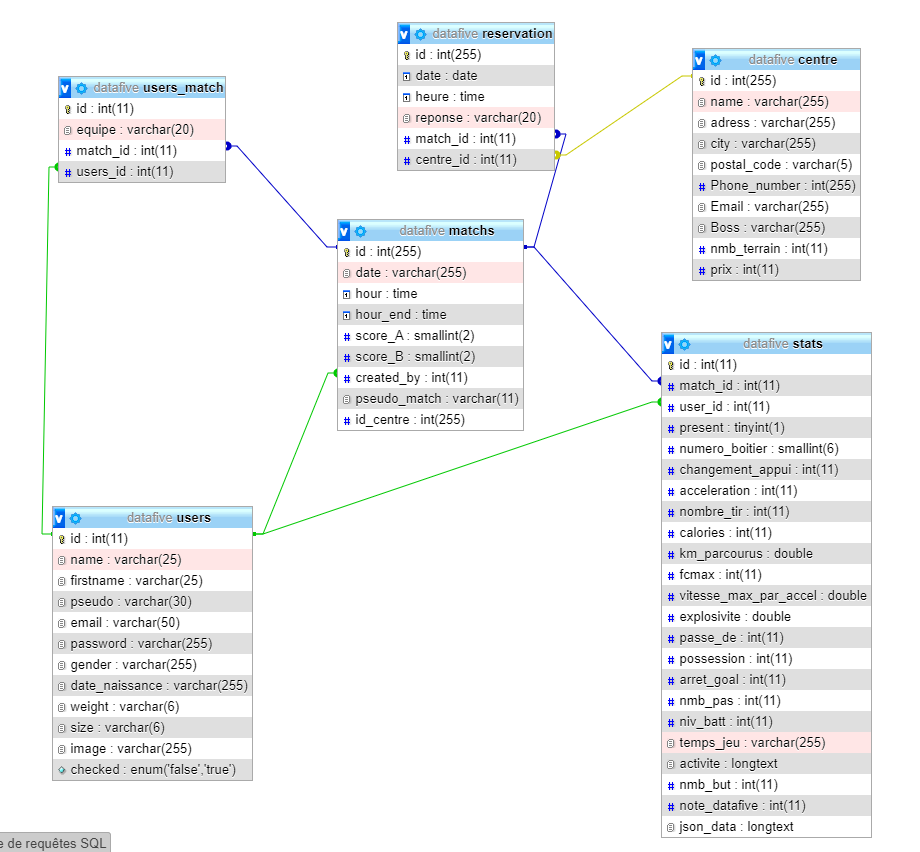
Il est possible aussi de faire tourner le projet sur Android Studio.

-Maintenant il ne reste plus qu’à lancer le serveur de la même façon avec la commande « npm start ».

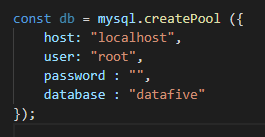
Assurez vous que vous disposez bien de Xampp ou de Wampp server afin de pouvoir communiquer avec la base de données qui est sous SQL.

Note IMPORTANTE : Il est important de rappeler que dans ce projet, s’il faut lancer l’application, il est important de mettre le smartphone ainsi que l’ordinateur dans le même réseau local (partage de connexion depuis le smartphone par exemple).

Il faudra aussi créer la base de données en Local, voici une image de la base de données en question :



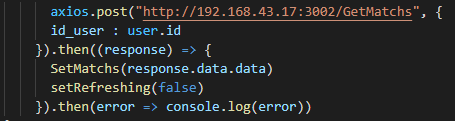
Et voici la partie du compte qui sert la connexion à cette même base de données :



Le champs Checked dans la table Users représente si l’utilisateur a validé son inscription par Email ou pas, si celle si retourne « False » alors il n’a toujours pas validé par email sinon c’est que c’est bon.

# Explication brève du code :

Afin de communiquer entre Front et Back, il est important de souligner que l’on utilise « axios » pour mener à bien cette tache, voici un exemple de requête « axios » allant vers le Back-End (serveur NodeJs) :



Nous effectuons une requête vers le Back de l’application avec la terminaison « GetMatch », ce qui représente la route pointant sur la fonction en question, cependant l’adresse : « 192.168.43.17 :3002 » représente l’adresse du serveur NodeJs (Back-End).

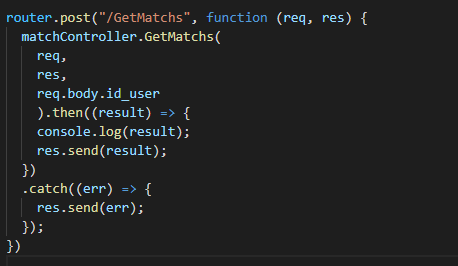
Ce dernier prend automatiquement l’adresse IP de votre ordinateur dans un réseau précis.

Pour ma part, le partage de connexion étant toujours actif, le serveur se retrouve toujours avec la même adresse IP attribuée.

Le nombre « 3002 » se trouvant à la fin de cette adresse représente le port que le serveur utilise afin de le contacter.

Nous pouvons ensuite grâce à cette requête axios envoyer des paramètres tel que « id\_user : user.id », afin que la route se trouvant au niveau du back puisse la récupérer sans soucis.

Note : Il faudrait que le nom de la variable que l’on envoie depuis le Front soit la même que l’on reçoit au niveau du Back-End, exemple :



Voici la route en question au niveau du serveur, c’est une requête POST avec comme variable récupérée « id\_user » que l’on récupère grâce à « req.body.id\_user », et cette route mènera directement au Controller de l’application avec les paramètres déjà entrés :

« MatchController.GetMatch(req, res, req.body.id\_user », voici cette fonction en question se trouvant dans le controller des Matchs.

Cette fonction comme vous pouvez le voir, permet de récupérer tous les matchs auxquels le joueur en question a participé.

Une fois les résultats obtenus, il ne reste plus qu’a les renvoyer au Front afin de pouvoir utiliser les résultats en question.

L’application fonctionne de cette façon avec des dizaine d’autres fonctions comme celles-ci, cependant le schéma reste le même.