Helbet

Dans le cadre du cours de programmation mobile I

Contents

[Description 2](#_Toc131075018)

[Application de paris sportifs fictifs de football 2](#_Toc131075019)

[Liste de fonctionnalités 2](#_Toc131075020)

[Création de compte 2](#_Toc131075021)

[Système d'abonnement aux clubs 2](#_Toc131075022)

[Paris sur le résultat d'un match 2](#_Toc131075023)

[Système d'alertes 2](#_Toc131075024)

[Historique & analyse des paris 3](#_Toc131075025)

[Scénario 3](#_Toc131075026)

[Informations supplémentaires 3](#_Toc131075027)

[Application similaire 3](#_Toc131075028)

[API 4](#_Toc131075029)

[Informations 4](#_Toc131075030)

[Données requises 4](#_Toc131075031)

[Requêtes 4](#_Toc131075032)

[Récupération Championnats 4](#_Toc131075033)

# Description

## Application de paris sportifs fictifs de football

L’utilisateur devra posséder un compte. Il pourra parier grâce à des jetons virtuels fictifs sur le résultat des matchs de football des grandes ligues nationales européennes[[1]](#footnote-1). Il peut suivre des clubs. Sur la première page sont d’abord affichés les matchs à venir et ensuite, les matchs à venir des équipes suivies. Il sera notifié des matchs auxquels les équipes qu’il suit participent et pourra localiser l’emplacement d’un match sur une carte. Enfin, avec ses jetons, un utilisateur peut investir sur une équipe. Investir sur équipe signifie que le joueur envoie une quantité de jetons à l'équipe. Le dernier jour de chaque mois, l'équipe ayant amassé grâce à ses fans, le plus de jetons, est proclamée équipe du mois jusqu'au mois suivant.

## Liste de fonctionnalités

### Création de compte

A l'aide de son adresse email et d'un mot de passe, un visiteur pourra créer un compte auquel il pourra se connecter par la suite.

<https://firebase.google.com/docs/auth/android/start#java>

### Système d'abonnement aux clubs

L'utilisateur a la possibilité de suivre des clubs. Les matchs auxquels les équipes suivies participent sont mis en évidence par rapport aux autres. De plus, il recevra une notification peu de temps avant le match pour lui rappeler de placer un paris s'il ne l'a pas encore fait.

Voir fonctionnalité annexe "Système d'alertes"

### Paris sur le résultat d'un match

L'utilisateur a la possibilité sur l'écran principal de parier un montant sur le résultat du temps réglementaire d'un match (victoire d'une équipe ou de l'autre ou match nul).

### Système d'alertes

Des alertes (notifications push) sont envoyées à l'utilisateur pour l'avertir :

* Des matchs prévus du jour
* Qu'une de ses équipes suivies a un match imminent
* Du résultat d'un paris (gagnant ou perdant)

### Historique & analyse des paris

L'utilisateur pourra consulter l'historique de ses paris et aura accès à des outils d'analyse encore à définir.

## Scénario

Tartempion crée un compte sur la plateforme. Etant fan du Real Madrid, Tartempion commence à suivre le club sur la plateforme. Le jour du match Real Madrid contre Liverpool, une notification l'avertit de la rencontre. Sur la vue du match, il peut placer une mise sur la victoire de son club préféré. Si son paris est gagnant, il recevra sa mise multipliée par la cote. Sinon, il perdra simplement sa mise.

## Informations supplémentaires

* Les matchs auxquels participent les équipes non gérées par la plateforme ne seront pas listés.

## Application similaire

Omada :

* [Site Web](https://www.omadagame.com/en/home)
* [Google Play](https://play.google.com/store/apps/details?id=com.omada.social&hl=fr&gl=US&pli=1)

# API

Avant de commencer le projet, je dois m'assurer qu'il existe une API pour me fournir les informations nécessaires.

## Informations

|  |  |
| --- | --- |
| Nom | API Football |
| Site Web | https://www.api-football.com/ |
| Headers | 'X-rapidAPI-Key':'df9daf891fmsh4f0f0483c2d1a31p1321dajsn6b919996842a' 'X-RapidAPI-Host':'api-football-v1.p.rapidapi.com' |

## Données requises

* Championnats :
  + Liga Santander (Espagne)
  + Premier League (Angleterre)
  + Serie A (Italie)
  + Bundesliga (Allemagne)
  + [ Ligue 1 (France) ]
  + [ Jupiler Pro League (Belgique) ]
  + [ Eredivisie (Pays-Bas) ]
* Les équipes de ces championnats
* Les matchs prévus dans ces championnats
* Les cotes de ces matchs
* Les résultats de ces matchs
* [ Les match prévus de certaines autres compétitions européennes ]

## Requêtes

J'effectue ces tests grâce au logiciel *Postman* qui me permet de sauvegarder les requêtes.

### Récupération Championnats

<https://api-football-v1.p.rapidapi.com/v3/leagues?current=true&type=league>

|  |  |
| --- | --- |
| Liga Santander | 140 |
| Premier League | 39 |
| Serie A | 135 |
| Bundesliga | 78 |
| Ligue 1 | 61 |
| Jupiler Pro League | 144 |
| Eredivisie | 88 |

# Développement

## Login – Register & Realtime database

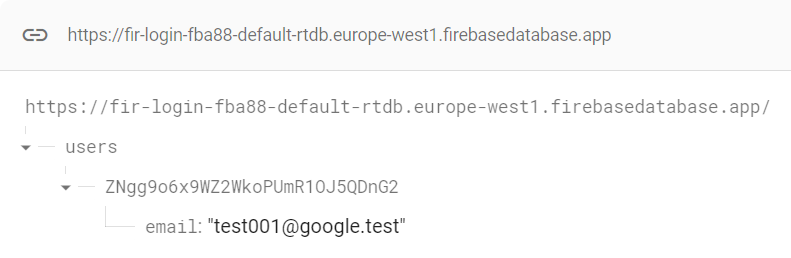
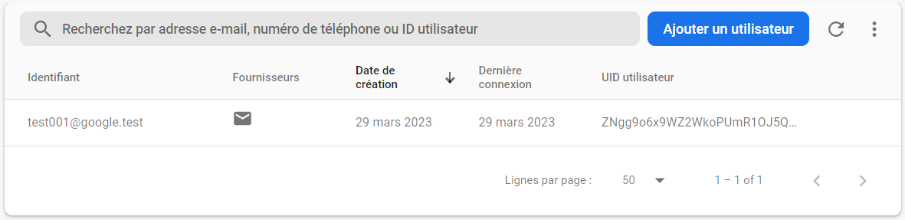
Pour l'enregistrement et la connexion à un compte, j'utilise le système d'authentification du service **Firebase**. Pour ce faire, j'exploite deux activités ; "registerActivity" et "loginActivity". J'ai suivi d'abord le tutoriel fournit par mon professeur (lien [ici](https://www.youtube.com/watch?v=WykjSz9K05A)) mais au moment d'enregistrer l'utilisateur dans la **Realtime Database**, rien ne se passe et un message d'avertissement (et non d'erreur) me signale que la localisation de ma base de donnée n'est pas correcte voir *Challenges techniques > C1 – Realtime Database* pour la résolution du problème.

### Classe User

Ce qui est sauvegardé pour la clé "users" de la Realtime Database est une liste d'utilisateurs conceptualisés par la classe User :

@IgnoreExtraProperties  
public class User {  
 private static DatabaseReference *mDatabase* = FirebaseDatabase.*getInstance*().getReference();  
  
 public String email;  
  
 public User() {  
 // Default constructor required for calls to DataSnapshot.getValue(User.class)  
 }  
  
 public User(String email) {  
 this.email = email;  
 }  
}

Pour l'instant, elle n'est composée que d'un attribut **email** qui se réfère à l'email entré lors de l'inscription. Quant au mot de passe, il est enregistré dans la partie **Authentication**. L'**UID** (identifiant unique) de l'utilisateur est utilisé comme clé ce qui veut dire qu'on peut faire directement le lien entre la partie authentification et base de données concernant les utilisateurs.



### Classe RealtimeDatabaseManager

Je décide d'implémenter une classe qui me servira d'interface avec la Realtime Database.

#### Liste des méthodes

## Requêtes de l'API

Pour faire les requêtes API nécessaires à mon application, j'utilise le module **Volley**. Avec lequel j'ai pu me familiariser lors d'une activité autonome proposée par mon professeur, monsieur Riggio.

Mon but ici est :

1. Récupérer certaines données de l'API et les stocker dans la Realtime Database
2. Afficher ces données dans un **RecyclerView** (servant à les lister)

### Classe APIConfig

Premièrement, pour pouvoir me connecter à l'API et effectuer des requêtes, je vais avoir besoin de ma clé d'identification. Cependant, mon projet est répertorié sur GitHub en publique. Je vais donc stocker cette information en tant que constante dans une autre classe que je vais appeler "Secret" dont le fichier sera référencé dans le **.gitignore** afin que la clé secrète ne se retrouve pas publiquement sur GitHub lors d'un push.

Avançant dans mon développement, je me rends compte que j'aurais besoin d'autres informations relatives à l'API (comme les IDs des ligues concernées par l'application) je renomme donc la classe (et le fichier par la même occasion) "APIConfig" pour centraliser les constantes relatives à l'API.

On y retrouve :

* L'URL de l'API sous forme de constante ***API\_URL***
* Les en-têtes nécessaires sous forme de HashMap ***HEADERS*** :
  + Clé "***X-RapidAPI-Key***"
  + Hôte "***X-RapidAPI-Host***"
* Les IDs des ligues concernées par l'application (voir [ce tableau](#_Récupération_Championnats))

### Classe APIManager

La classe APIManager me sert d'interface pour récupérer certaines données depuis l'API.

1. Les ligues nationales sont composées de clubs d'un même pays et se déroulent sur une saison. [↑](#footnote-ref-1)