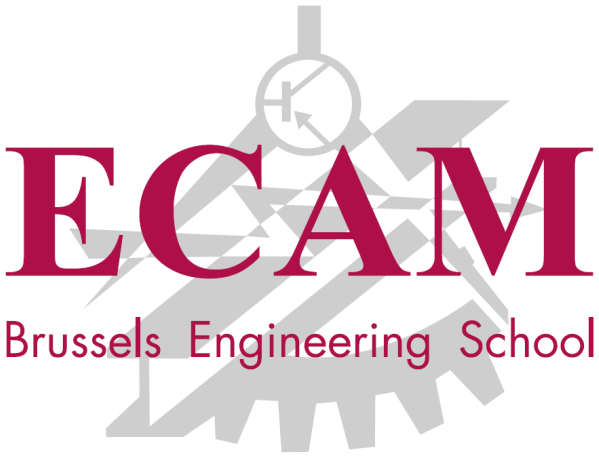

ECAM, 4EI (section génie électrique)

Projet d'application mobile

Application de récupération de score sportif

Auteurs :
M. AmauryLekens
M. BenoitWery

Encadrants :
Mr. Lurkin

Q2 2018

Table des matières

Chapitre 1

Présentation du projet

1.1 Que fait l'application ?

Le projet suivant est une application mobile qui pour but principal de rechercher et d'afficher des scores de football issus de différentes compétitions. L'utilisateur peut effectuer une recherche sur base d'une compétition, d'une équipe et d'un intervalle de temps entre deux dates. Tous les matchs correspondants à ces trois critères sont ensuite affichées. Il y a aussi la possibilité pour l'utilisateur d'enregistrer des favoris. Pour éviter de devoir sélectionner à chaque recherche une compétition et une équipe qu'il consulte fréquemment, il peut enregistrer ces deux paramètres. Ensuite, lors de la recherche suivante, en sélectionnant le favoris adéquat, il n'aura plus qu'à préciser l'intervalle de temps.
L'accent a été mis sur la simplicité d'utilisation en ne surchargeant pas l'application de fonctionnalités.

1.2 Origine des données

Pour fonctionner l'application doit évidemment avoir accès à des données. Dans l'optique d'avoir un large panel de choix dans les différents critères de recherche, il est impossible de développer une base de données personnelles. Cela demanderait un travail conséquent et inutile étant donné l'existence de solution existante et facile à mettre en oeuvre. Le choix s'est donc porté vers l'utilisation d'une API. La sélection de l'API (il en existe un paquet) s'est basée sur 3 critères : facilité d'utilisation, requête nécessaire à l'application possible, gratuité.
L'API retenue est la suivante : `https://apifootball.com/`. C'est une API REST (requête http) et elle dispose de filtre de recherche correspondant à notre application (voir documentation). Elle possède une version gratuite et une version de type premium avec 15

jours d'essais (que nous utiliserons).

1.3 Cahier des charges

1.3.1 Activité d'accueil

Cette page doit permettre d'accéder à :
— La recherche d'un score
— La gestion des favoris
— La gestion des préférence

1.3.2 Recherche d'un score

La recherche d'un score est divisée en plusieurs activités :
— Sélection compétition : Tous les compétitions disponibles sur l'API sont récupérés et présentés à l'utilisateur pour qu'il puisse effectuer un choix.
— Sélection équipe : Toutes les équipes qui font parties de la compétition sont récupérées et affichées à l'utilisateur pour qu'il puisse afficher un choix.
— Sélection date : l'utilisateur doit pouvoir sélectionner une intervalle de temps entre 2 dates.
— Affichage résultat : tous les résultats correspondants aux critères de recherche sont affichés à l'utilisateur.

1.3.3 Gestion des favoris

L'activité de gestion des favoris permet deux choses :
— D'ajouter des favoris via un bouton. L'utilisateur est alors redirigé vers l'activité de sélection de compétition et d'équipe.
— De sélectionner un des favoris et d'effectuer une recherche dessus : l'utilisateur est alors redirigé vers l'activité de choix des dates.

1.3.4 Gestion des préférences

Cette activité permet de gérer les préférences de l'application. Elle doit permettre en premier lieu de changer l'apparence de l'application (couleur)

Chapitre 2

Fonctionnalités

2.1 Recherche d'un score

2.1.1 Vérification de la connectivité

Avant de que l'utilisateur puisse effectuer une recherche le système vérifie qu'il est connecté à internet. S'il n'est pas connecté, il ne peut accéder à l'activité de recherche d'un championnat et un message s'affiche. Cette vérification est effectuée par le code ci -dessous : on vérifie qu'un réseau est disponible et que l'appareil est connecté à ce réseau.

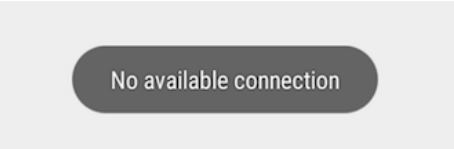

FIGURE 2.1 – Toast d'avertissement

```
ConnectivityManager connectivityManager = (ConnectivityManager)
    getSystemService(CONNECTIVITY_SERVICE);
NetworkInfo networkInfo =
    connectivityManager.getActiveNetworkInfo();

if(networkInfo != null && networkInfo.isAvailable() &&
    networkInfo.isConnected()) {
     // go to competition choice
}
```

2.1.2 Choix du championnat

En lançant cette activité l'utilisateur provoque une requête http qui va récupérer les données des championnats via l'API. Pour ne pas bloquer l'interface graphique, la récupération des données est effectuée de manière asynchrone. Concrètement on va créer dans notre activité une classe interne qui impémente la classe abstraite *AsynchTask* fournie par android.

```
public class QueryTask extends AsyncTask<String, Void, String> { }
```

Cette classe va implémenter 3 méthodes :

- *protected void onPreExecute()* : effectue une tâche avant l'exécution asynchrone.
- *doInBackground(String... params)* : effectue l'exécution asynchrone. c'est dans cette méthode que se trouvera la requête http.
- *protected void onPostExecute(String queryResults)* : effectue une tâche après l'exécution asynchrone. Cette méthode reçoit en paramètre la valeur retourné par *doInBackground*.

La requête http en tant que telle s'effectue via un *inputStream* qui est récupéré sur une connexion http. Le code suivant effectue ces deux tâches.

```
HttpURLConnection urlConnection = (HttpURLConnection)
    urlObject.openConnection();
InputStream in = urlConnection.getInputStream();
```

Une fois les données récupérées, il faut les afficher. Pour cela on va utiliser un *Adaptater* et un *AdaptaterView*.

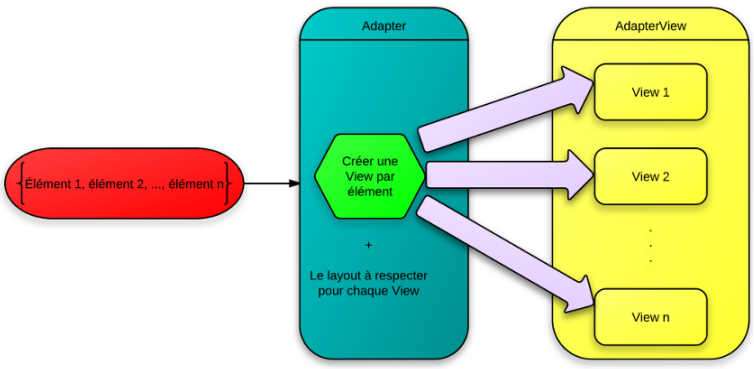

FIGURE 2.2 – Schéma de l'adaptateur (source : openclassroom)

1. On va créer une liste avec les données réceptionnées.
2. On passe cette liste à un *Adaptater* ainsi qu'un layout à respecter pour chaque élément de la liste.

3. On passe l'*Adaptater* à un *AdaptaterView* qui se charge du layout de l'ensemble.

Au niveau du code ces trois étapes sont effectuées comme ceci :

```
listView = (ListView) findViewById(R.id.listView);
ArrayAdapter<String> adapter = new
    ArrayAdapter<String>(ChampChoiceActivity.this,
    android.R.layout.simple_list_item_single_choice,
    competitionsList);
listView.setAdapter(adapter);
```

Une fois les données affichées l'utilisateur peut sélectionner une compétition et passer à l'étape suivante via le bouton *Submit*.
En interne les données suivantes sont passées à l'activité suivante via l'*Intent* : l'id de la compétition, le mode : qui permet de différencier les cas où l'utilisateur effectue une recherche de score (*search*) et les cas où il veut ajouter un favori (*addFavorite*).

2.1.3 Choix de l'équipe

L'organisation de cette activité est très semblable à l'activité précédente (requête http asynchrone, affichage des données, choix utilisateur) mais varie en quelque points :

- La requête http et le traitement des données réceptionnées sont différents.
- La gestion de l'événement *Submit* : la méthode qui gère l'événement (*onClick()*) va vérifier le mode (qui a été transmis depuis l'activité précédente). Dans le cas du mode "*search*", il lance l'activité du choix de la date en lui passant l'identifiant de la compétition et le nom de l'équipe. Dans le cas du mode "*addFavorite*", il écrit le nom de l'équipe et l'identifiant du championnat en dur dans un fichier et lance l'activité de gestion des favoris.

2.1.4 Choix de la date

Dans cette activité, il n'y pas de requête http ou de gestion d'affichage de liste. Il y a simplement deux widgets de type *DatePicker* qui permettent à l'utilisateur de sélectionner deux dates.
Lors de l'événement *Submit*, l'activité de l'affichage du résultat est lancée en lui passant l'identifiant du championnat, le nom de l'équipe et les deux dates.

2.1.5 Affichage du résultat

Cette activité va effectuer une requête http avec tous les paramètres reçus de l'activité précédente et gérer l'affichage des données réceptionnées.

2.2 Gestion des favoris

Cette application propose deux fonctions :
— D'effectuer une recherche sur un favori
— D'ajouter un favori

2.2.1 Recherche sur un favori

A son lancement l'activité de gestion des favoris va lire dans un fichier les équipes précédemment enregistrées. Cette opération est effectuée comme une simple lecture dans un fichier en Java.

```
FileInputStream input = openFileInput(originFile);
StringBuffer readed = new StringBuffer();

while((value = input.read()) != -1) {
    readed.append((char)value);
}
```

Une fois les données récupérées les données sont affichées comme précédemment avec un *Adaptater* et un *AdaptaterView*. L'utilisateur peut sélectionner une équipe et effectuer une recherche : il sera directement renvoyé vers l'activité du choix de la date.

2.2.2 Ajouter un favori

pour ajouter un favori l'utilisateur clique sur le bouton *Add favorite* qui va lancer successivement l'activité de choix de championnat et d'équipe mais cette fois avec le mode "*addFavorite*". Une fois le choix de l'équipe validé avec le bouton *Submit*, la méthode *onCreate()* reconnait le mode "*addfavorite*". L'identifiant de la compétition et le nom de l'équipe sélectionnés sont alors enregistrés en dur sur un fichier. Cette opération est réalisée par une simple écriture en Java.

```
FileOutputStream output = openFileOutput(destinationFile,
   MODE_APPEND);
String toWrite = "toWrite";
output.write(toWrite.getBytes());
```

2.3 Préférences

Cette fonctionnalités permet à l'utilisateur de configurer l'application. Pour l'instant elle ne permet que la configuration de la couleur. Android permet une gestion aisée de la

configuration avec ses composants : *PreferenceScreen*, *PreferenceFragment* et *SharedPreferences*

PreferenceScreen C'est un composant XML, à l'intérieur duquel on va définir nos différentes options de configuration. A chaque option, on va attribuer une clé afin de pouvoir récupérer les valeurs dans les différentes activités :

```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
<ListPreference
    android:defaultValue="red"
    android:entries="@array/color_labels"
    android:entryValues="@array/color_values"
    android:key="color"
    android:title="Color" />
</PreferenceScreen>
```

PreferenceFragment Il permet d'intégrer le composant *PreferenceScreen* dans le layout de configuration de l'application

SharedPreferences Il permet de récupérer les valeurs des préférences dans la logique de l'application. Dans notre cas, on récupère la valeur de la couleur sélectionnée pour appliquer le thème adéquat.

```
SharedPreferences sharedPreferences =
    PreferenceManager.getDefaultSharedPreferences(this);
String theme = sharedPreferences.getString("color", "");
```

Chapitre 3

Améliorations

3.1 Refractoring

Le projet avait pour but principal de découvrir l'environnement de développement d'android. Le code est loin d'être optimal : il semble possible de l'améliorer tout en conservant ses fonctionnalités.

3.2 Mode offline

Un mode offline où les données de l'API seraient téléchargées et stocker selon un format défini en local peut encore être développer

3.3 Configurations supplémentaires

L'application est pout l'instant pauvre en configuration (seulement le couleur). D'autres configurations comme par exemple l'autorisation d'utiliser les réseaux 3G, 4G peuvent être ajoutées.

Annexe A

Screenshots des activités