Travail final (partie 1)

Spotify, Spotify, dis-moi qui est le plus … ?

Dans ce travail, vous devrez mettre en œuvre les compétences acquises durant le cours afin de réaliser une application de type Quiz se servant des différentes données ouvertes de l'application Spotify.

Plus particulièrement, vous devrez prouver votre compétence à :

- Utiliser la librairie Volley afin de faire des requêtes http
- Utiliser la librairie Gson afin d'extraire des données par le processus de réflexion Java
- Réaliser un quiz intéressant et se renouvelant à chaque utilisation
- Utiliser des animations tel que vu en classe
- Utiliser un modèle objet et le modèle de conception Singleton

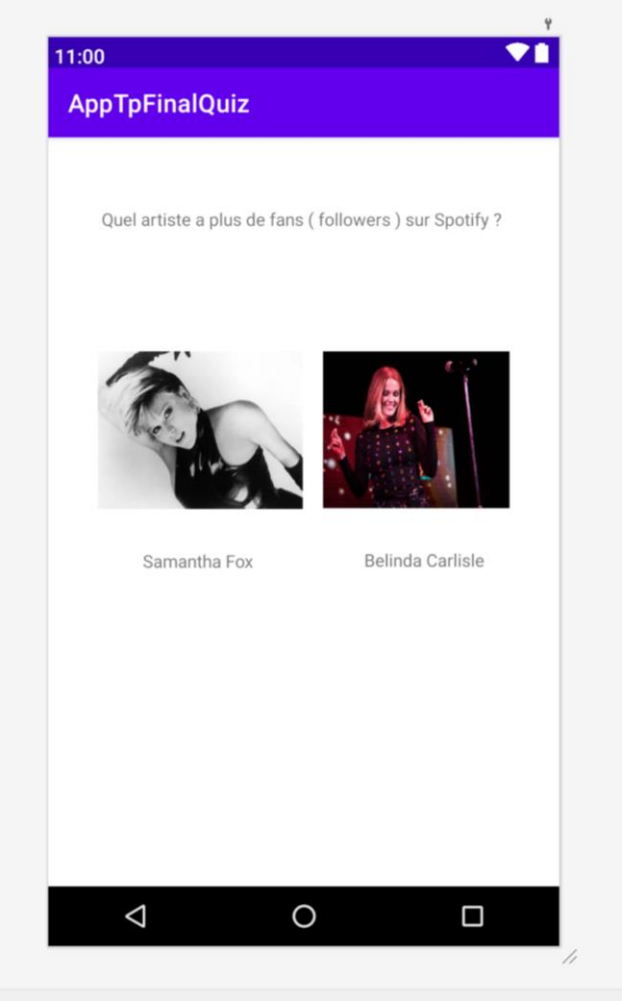
A) Plan / recherche

https://developer.spotify.com/documentation/web-api/reference/#/operations/get-an-artist # notre lien pour le request

Le quiz devra se composer de 3 questions à choix de réponses où les choix et les valeurs à comparer se trouvent dans les données ouvertes de Spotify :

https://developer.spotify.com/documentation/web-api/reference

- Établissez des questions portant sur des données différentes
- Établissez des questions intéressantes
- Le choix de réponses peut être de 2 ou 3 choix; une image chargée de Spotify doit permettre d'identifier les choix de même que le nom de l'artiste / de l'album, etc
- Les choix et les réponses doivent être générées à partir de requêtes GET de Volley



Un exemple de questions

B) Démarrage du projet

- De la façon du tp1, branchez-vous au Dashboard de Spotify for Developers afin de créer une nouvelle app dont les spécifications correspondront à celles de votre projet AndroidStudio

  *** Il ne doit pas y avoir de soulignement (_) dans le nom de votre package

- Générez un code SHA1 de la même façon que le tp1
- Ajoutez le fichier .aar pour avoir accès au sdk de Spotify ainsi que les liens vers les librairies Volley et Gson
- Donnez la permission INTERNET à votre application

C) S'authentifier comme usager Spotify

Pour utiliser l'API Web de Spotify et avoir accès aux données ouvertes, on doit utiliser le processus d'autorisation de Spotify, utilisant le framework OAuth2.0

Pour les infos :
https://developer.spotify.com/documentation/general/guides/authorization/

Faites vos recherches …

- Qu'est-ce qu'un Scope ?
- Quel Scope serait nécessaire pour connaître la chanson jouée actuellement par l'usager ?
- Quels sont les scopes qui auraient été nécessaires si on avait fait le tp1 en utilisant le Web API plutôt que le Android SDK ?

Les classes constituant le processus d'autorisation sont fournies dans LÉA (classes User, UserService, l'interface VolleyCallback et l'activité IdentActivity

- Intégrez-les dans votre projet en changeant le nom du package
- L'activité IdentActivity doit être incluse dans le fichier manifest et être celle qui sera appelée au démarrage
- Le xml de positionnement associé à IdentActivity doit être vide
- L'intent associé doit se rendre à votre Activité qui présentera le quiz

IMPORTANT

- Il est possible qu'il y ait des dédoublements dans les classes fournies par Gradle, pour pouvoir compiler, ajoutez dans le fichier gradle.properties :

```
android.enableJetifier=true
```

à la suite de ce qui est déjà écrit

D) Élaboration d'un Singleton responsable des requêtes Volley

Lire sur les avantages de placer l'objet RequestQueue dans un Singleton sur le site de Volley :

https://google.github.io/volley/requestqueue.html

Intégrez un singleton similaire à votre projet

Faites vos recherches …

- Que signifie le type de retour / le générique <T> ?
- À quoi sert un ImageLoader ?

E) Amorce

Afin de vous familiariser avec les différentes données du serveur de Spotify, récupérez le nom d'artistes dont vous ne connaissez que l'URL correspondant à l'artiste à partir d'une Activité

- *Utiliser* un URL choisi
- Faites la requête Volley
- Ajouter un header à la requête
- Ajoutez à la queue de votre Singleton
- Récupérer le nom dans la réponse

Ajouter un header à la requête

Le header d'une requête GET :

- On peut ajouter un header à une requête http afin de :
    - D'indiquer les formats permis de la réponse

- Fournir l'autorisation nécessaire
- Contrôler la cache
- Obtenir l'information à propos du User Agent
- Etc…

Exemples d'headers :

```
GET /home.html HTTP/1.1
Host: developer.mozilla.org
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:50.0) Gecko/20100101 Firefox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://developer.mozilla.org/testpage.html
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Mon, 18 Jul 2016 02:36:04 GMT
If-None-Match: "c561c68d0ba92bbeb8b0fff2a9199f722e3a621a"
Cache-Control: max-age=0
```

Liste des headers possible : https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

Nous avons besoin particulièrement du header authorization qui utilise le token généré lors du processus d'autorisation qui nous identifie en tant qu'usager authentifié pour faire des recherches.

Comment ajouter un header à nos requêtes :

1. Le token est sauvegardé dans nos SharedPreferences (la boucle est bouclée)

```
sharedPreferences =
getApplicationContext().getSharedPreferences("SPOTIFY", 0);
```

2. On redéfinit la méthode getHeaders à la fin de notre requête :

```
}) // fin de l'appel du constructeur de JSONArrayRequest ou
JSONObjectRequest
{
    @Override
    public Map<String, String> getHeaders() throws AuthFailureError {
        Map<String, String> headers = new HashMap<>();
        String token = sharedPreferences.getString("token", "");
        String auth = "Bearer " + token;
        headers.put("Authorization", auth);
        headers.put("Content-Type", "application/json; charset=utf-8");
        return headers;
    }
};
```