Documentation Epicture

I) Android

L'application est constituée de quatre activités et d'un Singleton Epicture qui permet de stocker des données nécéssaires tout au long de l'application (données de l'utilisateur connectées, instance du requête manager, etc...). Les requêtes sont simples à utiliser: la class ImgurAPI est un wrapper qui permet d'appeller toutes les requêtes en renseignant la requête manager du Singleton de l'application et les paramètres nécéssaires à la requête.

1) MainActivity

MainActivity est l'activité launcher de l'application, elle contient une webview pour se connecter au service Imgur. Elle sert à connecter l'utilisateur et à récupérer son ID nécéssaire à toutes les requêtes pour l'API imgur.

2) PicturesActivity

PicturesActivity est l'activité principale de l'application elle contient le menu de navigation. Cette activité charge des fragments. L'activité peut charger quatre fragments qui correspondent aux quatre options du menu. On a le fragment AllPicturesFragment qui permet de parcourir les images posté par les utilisateur avec possibilité d'utiliser des filtres. On a ensuite SearchFragment qui permet de chercher des images dans l'application avec un tag ou par mots clés. UploadFragment est un fragment qui permet à l'utilisateur d'upload des images à partir de la gallery ou en prenant une photo. Enfin ProfileFragment est un fragment qui permet de consulter les informations du profil ainsi que les images favorites et les images postées par l'utilisateur.

3) FinishUploadActivity

FinishUploadActivity est proposé à l'uttilisateur après qu'il ai pris une photo ou choisi une image dans la gallerie pour l'upload. Cette activité permet de renseigner un nom d'image, un nom d'album et une description avant d'upload.

4) ImageViewActivity

ImageViewActivity permet de voir les détails dans une image après avoir cliquer dessus. On peut voir son nom, sa description et son nombre de vues. De plus cette activité permet de upvote, downvote, mettre l'image en favori ou encore partager l'image.

II) Instance IMGUR

Le back se compose de managers et d'instances, le principe étant que chaque requête renvoyée par IMGUR est un JSON qui une fois deserialisé devient une instance stockée dans un manager.

1) Création d'une instance.

Une instance doit être créée comme ceci avec @Data obligatoirement:

```java
package com.example.benki.epicture.ImgurAPI.Instances;

import lombok.Data;

@Data
public class MonInstance {

}
```

En fonction de ce que renvoie le JSON nous créons les champs avec des private final:

Example JSON:

```
{
        "data": {
                "ups": 6395,
                "downs": 89
        },
        "success": true,
        "status": 200
}
```

Code:

```java
package com.example.benki.epicture.ImgurAPI.Instances;

import lombok.Data;

@Data
public class MonInstance {

    private final int ups;

    private final int downs;

}
```

Pour finir cette instance doit pouvoir créer une instance d'elle même en fonction d'un JSON:

```java
package com.example.benki.epicture.ImgurAPI.Instances;

import com.example.benki.epicture.Utils.JSONObjectCustom;

import lombok.Data;
import lombok.SneakyThrows;

@Data
public class MonInstance {

    private final int ups;

    private final int downs;

    @SneakyThrows
    public static MonInstance newMonInstance(JSONObjectCustom obj) {
        return new MonInstance(obj.getInt( name: "ups"), obj.getInt( name: "downs"));
    }

}
```

Votre instance est maintenant fini, passons au manager.

2) Manager

Un manager doit contenir une ou plusieurs instances.

Un manager doit implementer l'interface ImgurInstances et donc implementer la méthode deserialize:

```java
package com.example.benki.epicture.ImgurAPI.Instances.Managers;

import com.example.benki.epicture.ImgurAPI.Instances.ImgurInstances;
import com.example.benki.epicture.ImgurAPI.Instances.MonInstance;
import com.example.benki.epicture.Utils.JSONObjectCustom;

import org.json.JSONException;
import org.json.JSONObject;

import lombok.Getter;

@Getter
public class MonInstanceManager implements ImgurInstances
{

    private MonInstance instance;

    @Override
    public void deSerialize(JSONObject object) throws JSONException {
        this.instance = MonInstance.newMonInstance(new JSONObjectCustom(object));
    }
}
```

III) Requests

1) GET

Pour faire une requête GET rien de plus simple !

Vous devez avoir une instance de RequestManager avec les bons champs.

Une fois cette instance créée, vous devez juste faire appel à la fonction newRequest prenant :

- URL
- Instance d'interface pour le header
- La classe manager
- Des params pour l'URL (Optionnelle)

Example:

```
public class TestRequest {

    // PUT IDS INSIDE
    static RequestManager manager = new RequestManager( username: "",  access: "",  clientid: "");

    public static void Test() {

        MonInstanceManager instanceManager = (MonInstanceManager) manager.newRequest( url: "https://api.imgur.com/3/gallery/@1/votes",
                new RequestManager.RequestProcessing() {
                    @Override
                    public String setAuthorisation(RequestManager.UserSettings settings) {
                        return "Client-ID " + settings.getId();
                    }
                }, MonInstanceManager.class,  ...params: "0ygD2Tj");

        MonInstance instance = instanceManager.getInstance();

        System.out.print(instance.getUps());

    }
}
```

2) POST

Une requête POST demande un peu plus de technique.

Vous devez avoir une instance de RequestManager avec les bons champs.

Une fois cette instance créée vous devez faire appel a la fonction newPostRequest prenant :

- URL
- Instance d'interface URLRequester ActionOnPost
    - Action (build du body)
    - getBody (création du body)
- Instance d'interface pour le header
- La classe manager
- Des params pour l'URL (Optionnelle)

Example:

```java
public class TestRequest {

    // PUT IDS INSIDE
    static RequestManager manager = new RequestManager( username: "",  access: "",  clientid: "");

    public static void Test() {

        MonInstanceManager instanceManager = (MonInstanceManager) manager.newPostRequest( url: "https://api.imgur.com/3/gallery/@1/votes",
                new RequestManager.RequestProcessing() {
                    @Override
                    public String setAuthorisation(RequestManager.UserSettings settings) {
                        return "Client-ID " + settings.getId();
                    }
                }, new UrlRequester.actionOnPost() {
                    @Override
                    public void action(Request.Builder build, RequestBody body) {
                        build.post(body);
                    }

                    @Override
                    public RequestBody getRequestBody() {
                        return RequestBody.create( contentType: null, new byte[]{});
                    }
                }, MonInstanceManager.class,  ...params: "0ygD2Tj");

        MonInstance instance = instanceManager.getInstance();

        System.out.print(instance.getUps());

    }

}
```