

Sujets Divers

ENSEIGNANTE: MIRNA AWAD

RESPONSABLE DE COURS: A. Toudef



Utilisation de Google maps

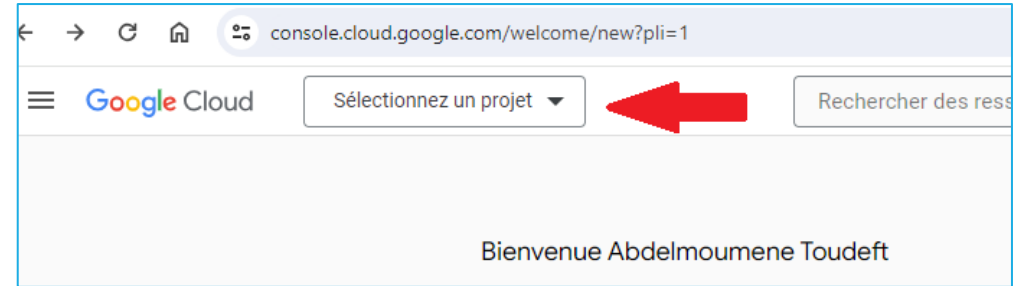
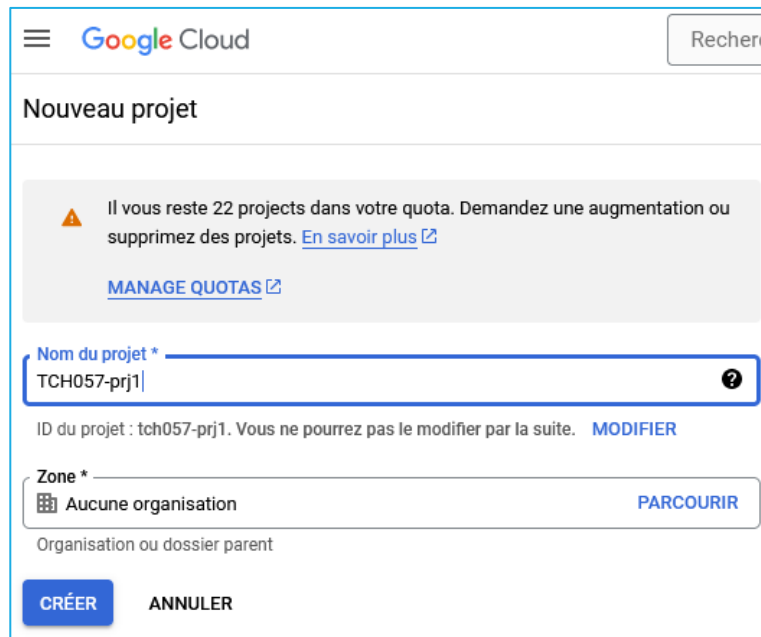
PLAN

- Obtenir une clé d'API
- Intégrer la clé
- Affichage
- Utilisation d'un appareil physique.



OBTENIR UNE CLÉ D'API

- Faut avoir un compte Google (gmail);
- Accéder à la console : <https://console.cloud.google.com/>
- Créer un projet ou sélectionner un projet existant;

Google Cloud Recherche

Nouveau projet

⚠ Il vous reste 22 projets dans votre quota. Demandez une augmentation ou supprimez des projets. [En savoir plus](#)

[MANAGE QUOTAS](#)

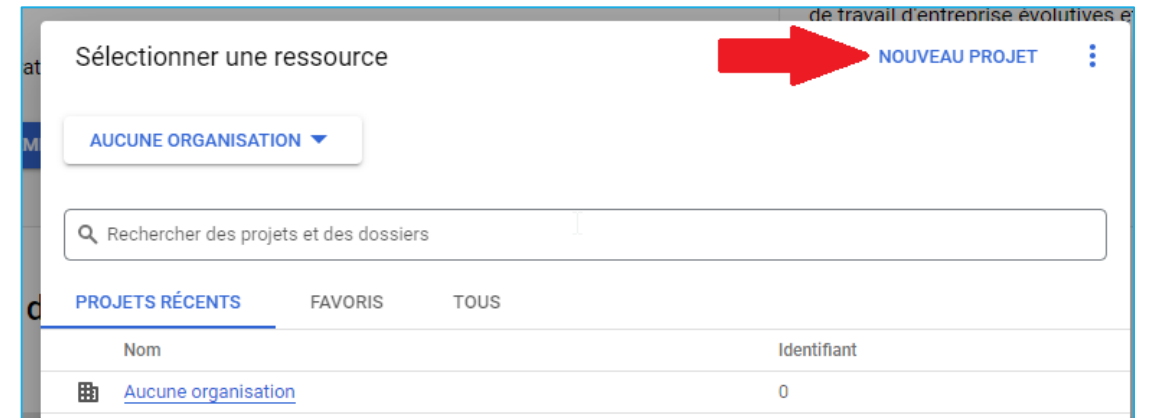
Nom du projet *
TCH057-prj1

ID du projet : tch057-prj1. Vous ne pourrez pas le modifier par la suite. [MODIFIER](#)

Zone *
Aucune organisation [PARCOURIR](#)

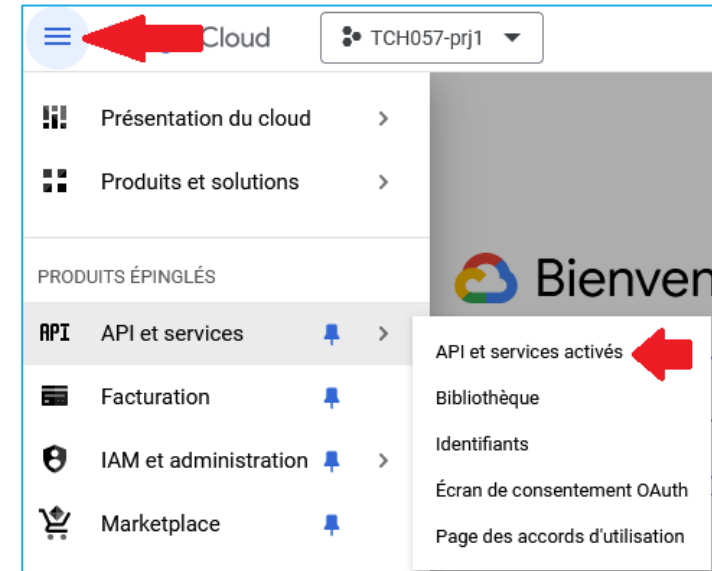
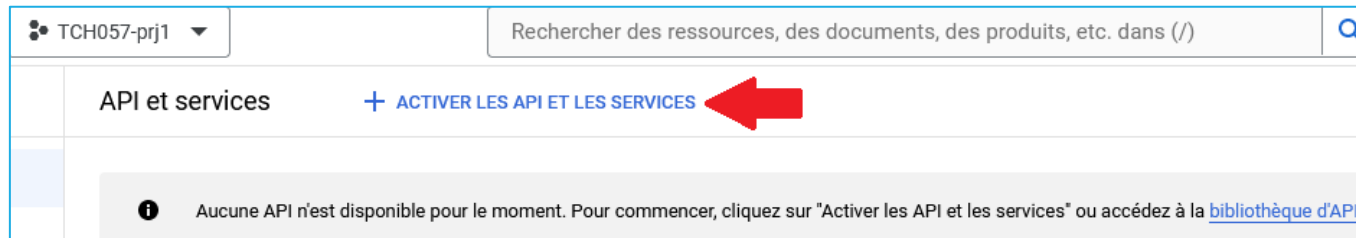
Organisation ou dossier parent

[CRÉER](#) [ANNULER](#)



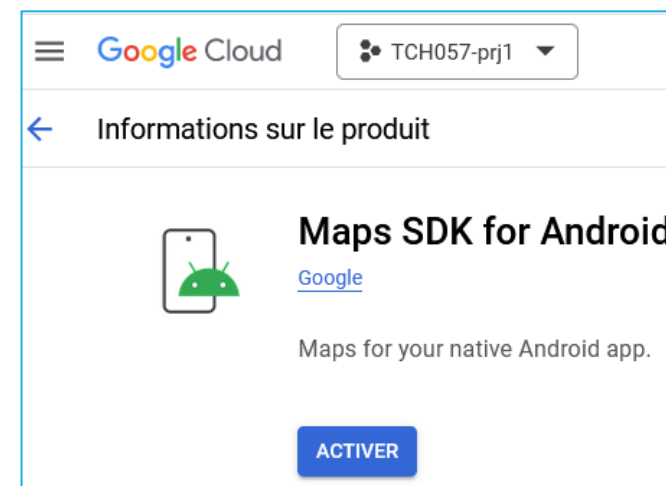
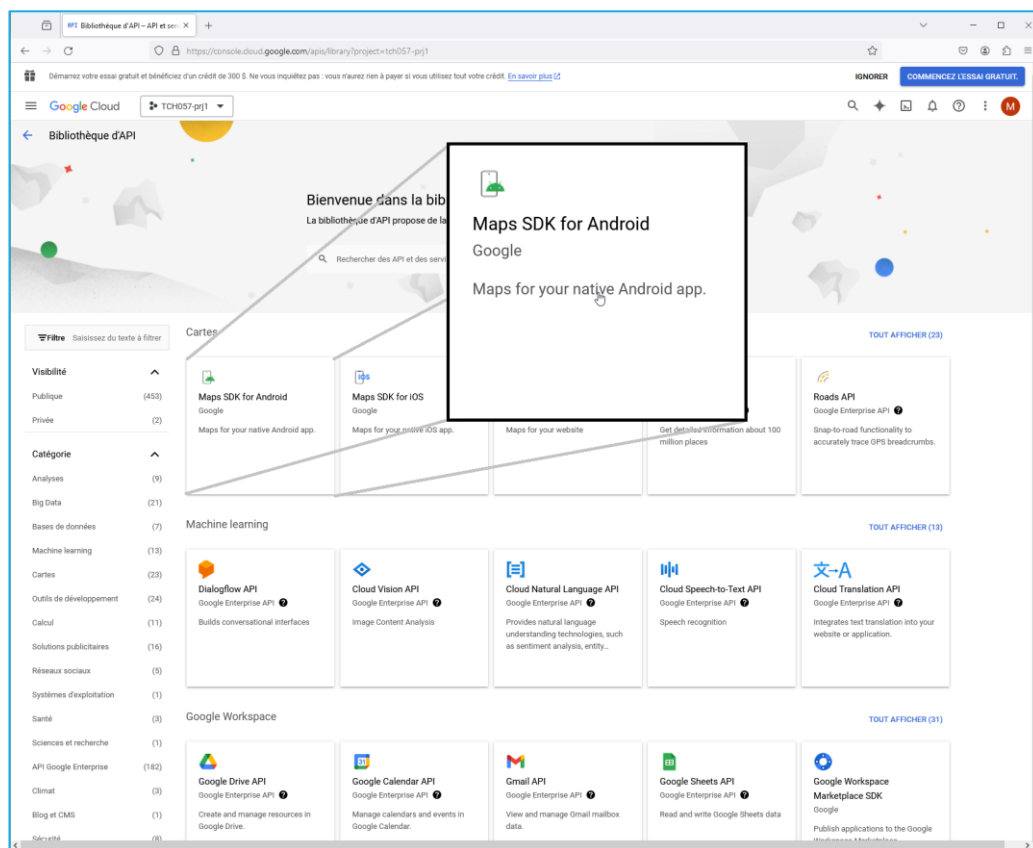
OBTENIR UNE CLÉ D'API

- Menu API et services | API et services activés;
- Activer les API et les services.



OBTENIR UNE CLÉ D'API

- Activer Maps SDK pour Android.



OBTENIR UNE CLÉ D'API

- Accepter les conditions;
- Fournir une carte pour les paiements (la carte n'est pas utilisée dans l'essai gratuit);
- Répondre aux questions sur le profil et les besoins d'utilisation;
- La clé d'API est générée :

Premiers pas avec Google Maps Platform

Vous pouvez maintenant commencer à développer votre application. Voici la clé API dont vous avez besoin pour la mise en œuvre. La clé API peut être référencée dans la section "Identifiants".

Votre clé API

[Clé API masquée]

- ☒ Activer toutes les API Google Maps pour ce projet ?
- ☒ Créer des alertes budgétaires pour m'aider à gérer mes dépenses et m'avertir lorsque je suis sur le point de dépasser les 200 \$ de crédit Google Maps mensuels ?

ACCÉDER À GOOGLE MAPS PLATFORM

Étape 1 sur 2 Informations de compte



Moumene Toudeft
[Email masqué]@gmail.com

CHANGER DE COMPTE

Pays

Canada

En utilisant cette application, vous acceptez les conditions d'utilisation de [Google Cloud Platform](#), les [conditions complémentaires liées à l'essai gratuit](#), ainsi que les conditions d'utilisation des [services et API concernés](#).

ACCEPTER ET CONTINUER



- ```
xml x MainActivity.java MF AndroidManifest.xml x
<application
 android:allowBackup="true"
 android:dataExtractionRules="@xml/data_extraction_rules"
 android:fullBackupContent="@xml/backup_rules"
 android:icon="@mipmap/ic_launcher"
 android:label="@string/app_name"
 android:roundIcon="@mipmap/ic_launcher_round"
 android:supportsRtl="true"
 android:theme="@style/Theme.Exemple_Google_Maps"
 tools:targetApi="31">

 <!-- ... -->
 <meta-data
 android:name="com.google.android.geo.API_KEY"
 android:value=" " />

 <activity
```



# PERMISSIONS

- Dans le manifeste :

```
<uses-permission android:name="android.permission.INTERNET" />
```

Et Si l'application doit accéder à une localisation précise ou approximative de l'appareil:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

# AFFICHER LA CARTE

- Ajouter un fragment à l'écran de l'activité :

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context=".MainActivity">

 <fragment
 android:id="@+id/mapFragment"
 android:name="com.google.android.gms.maps.SupportMapFragment"
 android:layout_width="match_parent"
 android:layout_height="match_parent"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

# AFFICHER LA CARTE

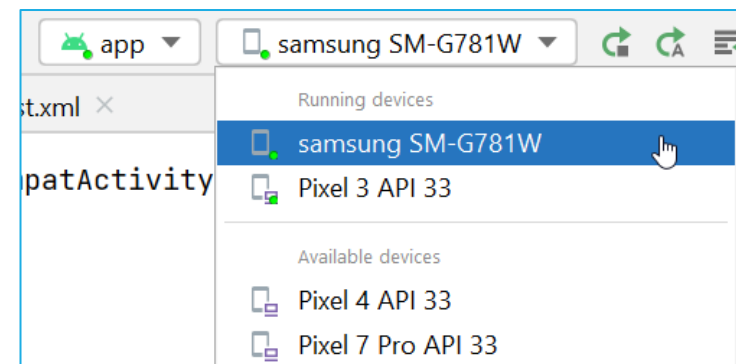
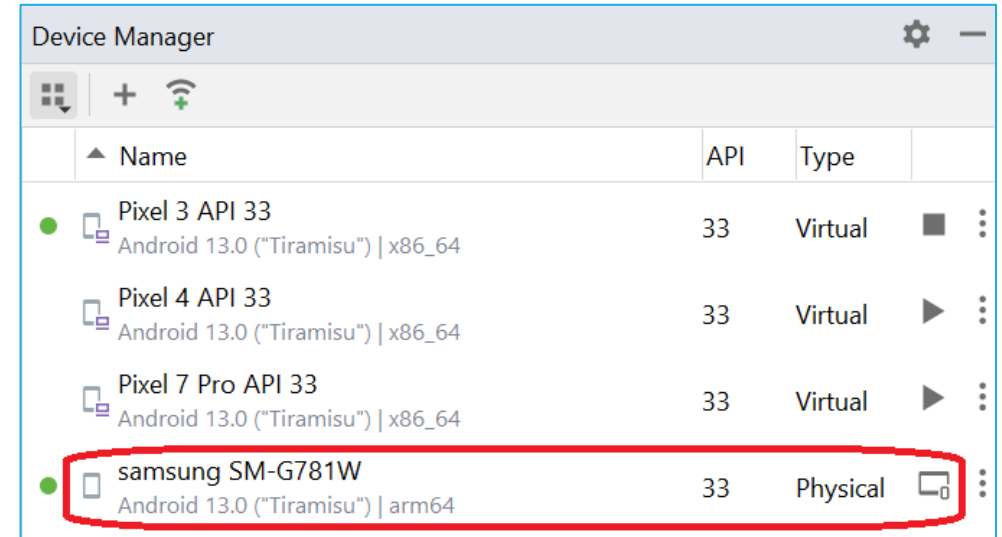
- Ajouter un fragment à l'écran de l'activité :

```
private GoogleMap googleMap;
@Override
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);

 SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
 .findFragmentById(R.id.mapFragment);
 mapFragment.getMapAsync(new OnMapReadyCallback() {
 @Override
 public void onMapReady(@NonNull GoogleMap gMap) {
 googleMap = gMap;
 LatLng montreal = new LatLng(45.50, -73.56);
 gMap.addMarker(new MarkerOptions().position(montreal).title("Ici Montréal"));
 gMap.moveCamera(CameraUpdateFactory.newLatLng(montreal));
 }
 });
}
```

# UTILISATION D'UN APPAREIL PHYSIQUE

- Si la carte ne s'affiche pas clairement sur l'émulateur, utilisez un appareil physique.
- Utilisation d'un appareil physique :
  - Brancher l'appareil au port USB;
  - Activer les options de débogage - suivre les étapes décrites dans les pages :
    - <https://developer.android.com/studio/run/device?hl=fr>
    - <https://developer.android.com/studio/debug/dev-options?hl=fr>
  - L'appareil apparaît dans le *Device Manager*;
  - Le sélectionner et exécuter l'application.



## Utilisation de la Caméra

# PLAN

- Utilisation de la caméra
- Sauvegarde sur disque local
- Préférences.



# UTILISATION DE LA CAMÉRA

## Permission

### ■ Manifeste :

```
<uses-feature android:name="android.hardware.camera" android:required="true" />
<uses-permission android:name="android.permission.CAMERA" />
```

### ■ Initialiser un lanceur pour gérer la permission requise :

```
ActivityResultLauncher <String> permissionLauncher = registerForActivityResult(
 new ActivityResultContracts.RequestPermission(),
 new ActivityResultCallback <Boolean> () {
 @Override
 public void onActivityResult(Boolean isGranted) {
 if (isGranted) {
 prendrePhoto();
 } else {
 Toast.makeText(MainActivity.this, "Permission caméra refusée",
 Toast.LENGTH_SHORT).show();
 }
 }
 });
```



# UTILISATION DE LA CAMÉRA

## Permission

- Demande d'accorder la permission :

```
int resultat = ContextCompat.checkSelfPermission(PrendrePhotoActivity.this,
 Manifest.permission.CAMERA);

if (resultat == PackageManager.PERMISSION_GRANTED) {
 prendrePhoto();
}
else {
 permissionLauncher.launch(Manifest.permission.CAMERA);
}
```



# UTILISATION DE LA CAMÉRA

## Lancement de la caméra

```
ActivityResultLauncher <Intent> cameraLauncher = registerForActivityResult(
 new ActivityResultContracts.StartActivityForResult(),
 new ActivityResultCallback<ActivityResult>() {
 @Override
 public void onActivityResult(ActivityResult result) {
 }
 }
);

private void prendrePhoto() {

 Intent iPrendrePhoto = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
 cameraLauncher.launch(iPrendrePhoto);

}
```



# UTILISATION DE LA CAMÉRA

## Récupération et affichage de la photo dans la méthode onActivityResult() du lanceur

```
protected void onActivityResult(ActivityResult result) {

 if (result.getResultCode() == RESULT_OK && result.getData() != null) {

 Toast.makeText(MainActivity.this, "Photo prise", Toast.LENGTH_SHORT).show();
 Bundle extras = result.getData().getExtras();

 if (extras != null) {
 Bitmap imageBitmap = (Bitmap) extras.get("data"); //Récupérer l'image en bitmap
 ivPhoto.setImageBitmap(imageBitmap); //Afficher l'image dans l'ImageView
 }

 }else
 Toast.makeText(MainActivity.this, "Photo annulée", Toast.LENGTH_SHORT).show();
}
```

# SAUVEGARDE SUR DISQUE LOCAL

- **Problème** : La photo prise et affichée (exemple précédent) est perdue lorsque l'application se termine;
- **Solution** : Sauvegarder la photo sur disque et la recharger au démarrage de l'application;
- L'utilisation d'une base de données (SQLite) est inacceptable : BD **volumineuse** et problème de fragmentation du disque;
- Meilleure approche : **utilisation de fichiers**.

# SAUVEGARDE SUR DISQUE LOCAL

## Espace disque privé

- Chaque application dispose d'un espace disque privé (inaccessible aux autres applications);
- Contient les bases de données (SQLite);
- L'application peut y stocker dossiers et fichiers;
- **Exemple** : créer un dossier *images*, s'il n'existe pas

```
Context contexteDApplication = getApplicationContext();
File dossierPrive = contexteDApplication.getFilesDir();
File repertoire = new File(dossierPrive, "images");

if (!repertoire.exists())
 repertoire.mkdirs();
```

# SAUVEGARDE SUR DISQUE LOCAL

## Création de fichier et sauvegarde de photo

- Utilise des flux d'entrée/sortie (*input/output streams*) : `java.io.FileOutputStream`,....

```
File fichier = new File(repertoire, "profil.png");
Bitmap imageProfil = (Bitmap) extras.get("data");
try (FileOutputStream fos = new FileOutputStream(fichier)) {
 imageProfil.compress(Bitmap.CompressFormat.PNG, 100, fos);
 fos.flush();
 Toast.makeText(this, "Photo sauvegardée", Toast.LENGTH_SHORT).show();
} catch (IOException e) {
 Toast.makeText(this, "Problème d'E/S", Toast.LENGTH_SHORT).show();
}
```

# SAUVEGARDE SUR DISQUE LOCAL

## Chargement de la photo

- Au démarrage de l'application, on décode l'image à partir du fichier :

```
Context contexteDApplication = getApplicationContext();
File dossierPrive = contexteDApplication.getFilesDir();
File repertoire = new File(dossierPrive, "images");
File fichier = new File(repertoire, "profil.png");

if (fichier.exists()) { //Si le fichier de photo existe

 //On récupère la photo comme un Bitmap :
 imageProfil = BitmapFactory.decodeFile(fichier.getAbsolutePath());

 //On affiche la photo dans le ImageView :
 ivPhoto.setImageBitmap(imageProfil);
}
```



# PRÉFÉRENCES

- Chaque application dispose d'un espace pour stocker les préférences de l'utilisateur/application;
- Ne peut contenir que des données primitives et des chaînes de caractères;
- Accès aux préférences (en mode privé, les données stockées ne sont pas accessibles aux autres applications):

```
String nomApplication = "Exemple_Photo_Sauvegarde";
```

```
SharedPreferences preferences = getSharedPreferences(nomApplication,MODE_PRIVATE);
```

- Pour stocker des préférences, il faut un éditeur de préférences :

```
SharedPreferences.Editor editeurDePreferences = preferences.edit();
```

```
//On sauvegarde des données avec l'éditeur :
```

```
editeurDePreferences.putString("NOM_PHOTO_PROFIL", "profil.png");
```

```
//On valide les sauvegardes :
```

```
editeurDePreferences.commit();
```

# PRÉFÉRENCES

- Méthodes de sauvegarde de l'éditeur de préférences :
  - Editor `putString(String cle, String valeur)`
  - Editor `putInt(String cle, int valeur)`
  - Editor `putLong(String cle, long valeur)`
  - Editor `putFloat(String cle, long valeur)`
  - Editor `putBoolean(String cle, boolean valeur)`
  - Editor `putStringSet(String cle, Set<String> valeur)`
- Autres méthodes de l'éditeur :
  - Editor `remove(String cle)`
  - Editor `clear()`



# PRÉFÉRENCES

- Méthodes d'accès/récupération aux préférences (de l'objet *SharedPreferences*) :
  - `String getString(String cle, String default)`
  - `int getInt(String cle, int default)`
  - `long getLong(String cle, long default)`
  - `float getFloat(String cle, long default)`
  - `boolean getBoolean(String cle, boolean default)`
  - `boolean Contains(String cle)`
  - `Set<String> getStringSet()`
  - `Map<String, ?> getAll()`

ex:

```
nomFichierPhotoProfil = preferences.getString(("NOM_PHOTO_PROFIL", "INEXISTANT");
```

# Requêtes OkHttp asynchrones avec MVVM

# PLAN

- Rappel
- Intégration dans MVVM.



# RAPPEL

- On enfile la requête dans une file et on fournit un Callback qui va réagir une fois la réponse obtenue :

```
OkHttpClient okHttpClient = new OkHttpClient();

Request request = new Request.Builder()
 .url(URL_POINT_ENTREE + "/livres/"+path)
 .build();

okHttpClient.newCall(request).enqueue(new Callback() {
 @Override
 public void onResponse(Call call, Response response) throws IOException {
 final String jsonStr = response.body().string();
 //Traitement de la réponse ici
 }
 @Override
 public void onFailure(Call call, IOException e) {
 call.cancel();
 }
});
```



# INTÉGRATION DANS MVVM

- Écouteur de données : objet chargé de réagir à la réception d'une réponse :

```
public interface EcouteurDeDonnees {

 void onDataLoaded(Object data);

 void onError(String errorMessage);

}
```

# INTÉGRATION DANS MVVM

- Le service utilise l'écouteur de données pour renvoyer les données :

```
public void getLivres(..., EcouteurDeDonnees chargeurDeDonnees)
 throws IOException, JSONException {
 OkHttpClient okHttpClient = new OkHttpClient();
 Request request = new Request.Builder().url(URL_POINT_ENTREE + "/livres").build();
 okHttpClient.newCall(request).enqueue(new Callback() {
 @Override
 public void onResponse(Call call, Response response) throws IOException {
 final String jsonStr = response.body().string();
 if (jsonStr.length() > 0) {
 ObjectMapper mapper = new ObjectMapper();
 try {
 List<Livre> resultats = Arrays.asList(mapper.readValue(jsonStr, Livre[].class));
 chargeurDeDonnees.onDataLoaded(resultats);
 } catch (JsonProcessingException e) {
 chargeurDeDonnees.onError("Problème du JSON dans les livres reçus");
 }
 }
 }
 })
}
```



# INTÉGRATION DANS MVVM

- Le DAO transmet l'écouteur de données au service :

```
public class LivreDao {

 public static void getLivres(String isbn, String nomAuteur, int anneeMin, EcouteurDeDonnees
 ecouteurDeDonnees) throws JSONException, IOException {

 new HttpJsonService().getLivres(isbn, nomAuteur, anneeMin, ecouteurDeDonnees);
 }

}
```

# INTÉGRATION DANS MVVM

- Le repository fournit un objet anonyme qui implémente l'écouteur de données :

```
public void getLivres(String isbn, String nomAuteur, int anneeMin,
 MutableLiveData<List<Livre>> livresLiveData,
 MutableLiveData<String> errorLiveData) throws JSONException, IOException {

 LivreDao.getLivres(isbn, nomAuteur, anneeMin, new EcouteurDeDonnees() {
 @Override
 public void onDataLoaded(Object data) {
 livresLiveData.postValue((List<Livre>) data);
 }

 @Override
 public void onError(String errorMessage) {
 errorLiveData.postValue(errorMessage);
 }
 });
}
```



# INTÉGRATION DANS MVVM

- Le ViewModel appelle le repository pour récupérer les données et les exposer en ligne.

```
public class LivreViewModel extends ViewModel {
 private MutableLiveData <List<Livre>> livres = new MutableLiveData<>(new ArrayList<>());
 private MutableLiveData <String> error = new MutableLiveData<>();
 private LivreRepository repository = new LivreRepository ();

 public LiveData <List<Livre>> getLivres() {
 return livres;
 }
 public LiveData <String> getError() {
 return error;
 }
 public void chercherLivres(String isbn, String nomAuteur, int anneeMin) {
 try {
 repository.getLivres(isbn, nomAuteur, anneeMin, livres, error);
 } catch (IOException | JSONException e) {
 error.postValue("Erreur: " + e.getMessage());
 }
 }
}
```

## PROCHAINE SÉANCE

- Sujets Divers (suite);
- Révision.