Intégration librairie externe

Scanner un QRCode

Permissions

- Depuis la version 6.0, les permissions sont dynamiques (voir menu permissions : click long sur une application, infos sur l'appli, Autorisations).
- Vérifications à chaque utilisation.
- Différent niveaux de permissions.

https://developer.android.com/guide/topics/permissions/overview

Accéder à la caméra

- Demander la permission :
- Live Template : android_permission_single
- Choisir : CAMERA
- Suivre les indications dans les TODO
- Ignorer la partie pour les fragments
- Implémenter les actions correspondantes aux retours utilisateur
- Appeler : callAction() dans le onCreate

Scan par Application externe

- Intégrer les 2 fichiers à son projet :
  - IntentIntegrator.java
  - IntentResult.java
- Appeler : initiateScan()
- Récupérer le résultat dans le onActivityResult

URL des fichiers à intégrer :
https://github.com/zxing/zxing/tree/master/android-integration/src/main/java/com/google/zxing/integration/android

Voir détail dans le code.

Avantages/Inconvénients

- Avantages :
  - Facile à intégrer.
  - Très peu de code intégré dans l'application.
  - Mise à jour du scanner de QRCode indépendant de l'application.
- Inconvénients :
  - Dépendance à une application tierce.
  - L'utilisateur doit installer une seconde application.
  - Peu de moyens de customisation.

Scan intégré à l'application

- Intégrer la nouvelle librairie au build.gradle
- Mettre à jour le Manifest.xml pour activer l'accélération matérielle
- Lancer le scan avec l'IntentIntégrator
- Récupérer le résultat dans le onActivityResult

https://github.com/journeyapps/zxing-android-embedded

Ajouter la ligne suivante dans le fichier build.gradle (Module: app) :

```
dependencies {
 …
   implementation 'com.journeyapps:zxing-android-embedded:3.6.0‘
}
```

Dans le Manifest.xml :

```
<application android:hardwareAccelerated="true" … >
```

Dans le MainActivity.java, sur le click d'un bouton par exemple :

```
new IntentIntegrator(this).initiateScan(); // `this` is the current Activity
```

Dans la classe (en dehors d'une méthode) :

```
// Get the results:
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
   IntentResult result = IntentIntegrator.parseActivityResult(requestCode, resultCode, data);
   if(result != null) {
      if(result.getContents() == null) {
         Toast.makeText(this, "Cancelled", Toast.LENGTH_LONG).show();
      } else {
```

```
            Toast.makeText(this, "Scanned: " + result.getContents(), Toast.LENGTH_LONG).show();
        }
    } else {
        super.onActivityResult(requestCode, resultCode, data);
    }
}
```

Avantages/Inconvénients

- Avantages :
  - Assez facile à intégrer.
  - Tout est intégré à l'application : pas de dépendances externes.
- Inconvénients :
  - Taille de l'application plus important.
  - Mise à jour de l'application nécessaire s'il y a une mise à jour de la librairie.

Scan intégré à une vue

- Intégrer la vue.
- Définir la méthode de callBack.
- Lancer le décodage continu.
- Implémenter les méthodes onResume et onPause.

Documentation :
https://github.com/journeyapps/zxing-android-embedded/blob/master/EMBEDDING.md

Intégrer la vue suivante, dans votre layout.xml

```
<com.journeyapps.barcodescanner.DecoratedBarcodeView
    android:id="@+id/activity_name_decoratedBarcodeView_scanView"
    android:layout_width="@dimen/scan_size"
    android:layout_height="@dimen/scan_size"
    app:zxing_preview_scaling_strategy="centerCrop"
    app:zxing_use_texture_view="false" />
```

Définir les dimens manquantes dans le fichier correspondant.

Définir le callback (en dehors d'une méthode, dans la classe) :

```
private BarcodeCallback callback = new BarcodeCallback() {
    @Override
    public void barcodeResult(BarcodeResult result) {
    // TODO : implement code
    }

    @Override
```

```
    public void possibleResultPoints(List<ResultPoint> resultPoints) {
    }
};
```

Définit la variable de classe (juste après la déclaration de la classe, en dehors d'une méthode) :

```
private DecoratedBarcodeView barcodeScannerView;
```

Dans le OnCreate :

```
barcodeScannerView = (DecoratedBarcodeView) findViewById(R.id.
activity_name_decoratedBarcodeView_scanView);
barcodeScannerView.decodeContinuous(callback);
```

Implémenter les méthodes du cycle de vie de l'activité pour mettre en route ou arrêter le scanner :

```
@Override
public void onResume() {
    super.onResume();
    barcodeScannerView.resume();
}

@Override
public void onPause() {
    super.onPause();
    barcodeScannerView.pause();
}
```

Avantages/Inconvénients

- Avantages :
  - Forte possibilité de customisation.
  - Tout est intégré à l'application : pas de dépendances externe.
- Inconvénients
  - Code plus complexe à écrire.
  - Taille de l'application plus important.
  - Mise à jour de l'application nécessaire s'il y a une mise à jour de la librairie.