Preuves.md

Preuves

Je sais utiliser les Intent pour faire communiquer deux activités.

Dans la classe AddFragment, le fait de cliquer sur le bouton d'ajout nous renvoie à l'activité de la camera.

```
buttonCapture.setOnClickListener(new View.OnClickListener(){
    @Override
  public void onClick(View v) {
        Intent intent = new Intent(v.getContext(), CameraActivity.class);
  startActivity(intent);
  }
});
```

Je sais développer en utilisant le SDK le plus bas possible.

L'application utilise le SDK le plus bas possible avec 100% de portabilité vers les téléphones Android: l'API 15: avec la version d'Android 4.0.3 (IceCreamSandwich)

Je sais distinguer mes ressources en utilisant les qualifier.

L'utilisation de qualifier est présent dans le code. Notamment lors d'appels aux ressources Strings.

Extrait du code de la classe SuppresionDialogFragment

```
builder.setMessage(R.string.sentence_suppresion).setPositiveButton(R.string.delete_confirmation, [...]);
```

Je sais modifier le manifeste de l'application en fonction de mes besoins.

Le manifeste de l'application a été modifié afin de permettre l'ajout des autorisation et aussi de bloquer la rotation sur la partie Caméra.

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

<uses-feature android:name="android.hardware.camera"
  android:required="false"/>
<uses-feature android:name="android.hardware.camera.autofocus" />


<activity
  android:name=".MainActivity"
  android:label="@string/app_name">
 <intent-filter> <action android:name="android.intent.action.MAIN" />
 <category android:name="android.intent.category.LAUNCHER" />
 </intent-filter></activity>

<activity
  android:name=".Fragments.AddFragment"
  android:label="@string/title_activity_add" />

<activity android:name=".CameraFiles.CameraActivity"
  android:screenOrientation="portrait"
  android:parentActivityName=".Fragments.AddFragment">
 <meta-data android:name="android.support.PARENT_ACTIVITY"
  android:value=".Fragments.AddFragment"/>
```

```
</activity>
```

Je sais faire des vues xml en utilisant layouts et composants adéquats.

Dans le fragment_add.xml nous utilisons un Coordinator Layout afin de coordonner l'affichage de la ListView et du Floating Button.

Je sais coder propement mes activités, en m'assurant qu'elles ne font que relayer les évènements.

Chaque classe à une responsabilité propre, lorsque une classe a besoins d'effectuer une action qu'elle ne peut pas faire, elle relaye la demande à une classe qui en est responsable. Notamment dans la partie accession des fichiers, notre application utilise ce principe, FileAccessor et FileModifier ont réspéctivement la responsabilité de gérer l'accès à un fichier, et la responsabilité de de modifier ce qui se trouve dans un fichier donné.

Je sais coder une application en ayant un véritable metier

Le metier est séparé de la Vue, la vue ne contient pas de métier, ils sont strictement séparé.

Je sais parfaitement séparer vue et modèle

La vue et les modèles sont parfaitement séparés, en effet le modèle n'interfère jamais dans la création de la vue, la vue peut très bien fonctionné sans le modèle.

Je maîtrise le cycle de vie de mon application.

Le cycle de vie de mon application intervient plusieurs fois, lors d'une rotation nous sauvegardons la position du menu grâce au onSaveInstanceState et nous la restorons au moment voulu grâce au onRestoreInstanceState. La caméra aussi est relaché lors de l'appel du onDestroy de l'application.

```
void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putInt("positionNavigation", itemSaved);
    }

@Override
protected void onRestoreInstanceState(Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
    if (savedInstanceState.getInt("positionNavigation") != 0){
        int id = savedInstanceState.getInt("positionNavigation");
        Log.d("devNote", "onRestoreInstanceState: " + id + " " + R.id.navigation_creation);
        switch (id) {
            case R.id.navigation_home:
                selectedFragement = HomeFragment.newInstance();
                break;
            case R.id.navigation_creation:
                selectedFragement = CreationFragment.newInstance();
                break;
            case R.id.navigation_add:
                selectedFragement = AddFragment.newInstance();
                break;
        }
        itemSaved = id;
        FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();
        transaction.replace(R.id.main_fragment, selectedFragement);
        transaction.commit();
        }
    }
```

Je sais utiliser le findViewByUd à bon escient

Nous utilison le findViewById plusieurs fois dans notre application pour connecter notre layout avec notre Vue, nous l'utilisons notamment lors de l'instanciation notre Preview pour la Caméra, nous ajoutons notre preview dans le FrameLayout qui comporte l'id "camera_preview" dans la vue "activity_camera".

```
mPreview = CameraPreview(this, mCamera);
final FrameLayout preview = (FrameLayout) findViewById(R.id.camera_preview);
preview.addView(mPreview);
```

Je sais gérer les permissions dynamiques de mon application

Nous utilisons deux types de permissions dans l'application: la Caméra, et l'autorisation d'écriture dans un fichier externe de l'application, soit directement dans le téléphone. Si une des permissions est refusé l'utilisateur ce vois refuser certaines parties de l'application. Disponible dans la classe CameraActivity

Extrait du onCreate de la classe CameraActivity

```
if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA)
                == PackageManager.PERMISSION_DENIED) {
    ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA}, MY_CAMERA_REQUEST_CODE);
}
```

Extrait du onRequestPermissionResult de la classe CameraActivity

```
case MY_CAMERA_REQUEST_CODE: {
                if (grantResults.length > 0
                        && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                    Log.d("devNote", "Permission granted for camera");

                    mCamera = cameraManager.getCameraInstance();
                    Log.d("devNote", String.valueOf(Camera.getNumberOfCameras()));

                    cameraManager.setCameraDisplayOrientation(this, 0, mCamera);

                    // Create our Preview view and set it as the content of our activity.
                    mPreview = new CameraPreview(this, mCamera);
                    final FrameLayout preview = (FrameLayout) findViewById(R.id.camera_preview);
                    preview.addView(mPreview);

                } else {
                    Log.d("devNote", "onRequestPermissionsResult: permission refuse");
                    findViewById(R.id.camera_preview).setVisibility(View.GONE);
                    findViewById(R.id.textView_permissionError).setVisibility(View.VISIBLE);
                    findViewById(R.id.button_capture).setVisibility(View.GONE);
                }
                return;
            }
```

Je sais gérer la persistance légère de mon application

Nous utilisons le principe de Bundle afin de persister des données au sein de l'application,

Je sais gérer la persistance profonde de mon application

L'application utilise la Caméra, de ce fait nous avons utiliser la possibilité de capture d'image, ils nous as donc fallu enregistrer ces images dans un fichier. Les classes FileAccessor et FileModifier accède et modifie dans un fichier. Nous enregistrons les images capturées grâce à la méthode takePicture qui fait appel à un PictureCallback présent dans la classe CameraActivity. Dans cette methode nous enregistrons les données de la photo capturé.

Extrait de Camera.PictureCallback dans la classe CameraActivity

```
        File pictureFile = fi.getOutputMediaFile(FileAccessor.MEDIA_TYPE_IMAGE, ALBUM_NAME);


            if (pictureFile == null){
                Log.d("devNote", "check storage permission, error creating media file");
```

```
            return;
        }

        try {
            FileOutputStream fos = new FileOutputStream(pictureFile);
            fos.write(data);
            fos.close();
            mCamera.startPreview();
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA)
            == PackageManager.PERMISSION_DENIED) {
        ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CAMERA}, MY_CAMERA_REQUEST_CODE);
    }
```

Je sais afficher une collection de données

Nous affichons une ListView qui affichent une collections d'Images présent dans un dossier du téléphone. Disponible dans la Vue "fragment_add.xml"

Je sais coder mon propre adaptateur

Nous avons donc en relation de la ListView, coder notre propre Adaptateur afin d'afficher une liste de buttonImage.

Extrait de la classe ImageAdapter

```
public ImageAdapter(Activity activity, List<String> listImagePath){
        super(activity, R.layout.image_list_item, listImagePath);
        this.activity = activity;
        this.listImagePath = listImagePath;
}

@Override
public View getView(int position, View convertView, ViewGroup parent) {
   LayoutInflater inflater = activity.getLayoutInflater();
   View view = inflater.inflate(R.layout.image_list_item, null, false);
   ImageButton imageButton = view.findViewById(R.id.image_item);
   String s = listImagePath.get(position);
   Bitmap bm = BitmapFactory.decodeFile(s);
   imageButton.setImageBitmap(bm);
   return view;
}
```

Je maitrise l'usage de fragments

Notre application utilise principalement des fragments, le bottomNavigationMenu utilise pour chaque onglet un fragment.

Extrait de la classe MainActivity

```
private Fragment selectedFragment;

private BottomNavigationView.OnNavigationItemSelectedListener mOnNavigationItemSelectedListener
        = new BottomNavigationView.OnNavigationItemSelectedListener() {

        @Override
                public boolean onNavigationItemSelected(@NonNull MenuItem item) {
                        switch (item.getItemId()) {
                                case R.id.navigation_home:
                                        selectedFragement = HomeFragment.newInstance();
                                        break;
                                case R.id.navigation_creation:
                                        selectedFragement = CreationFragment.newInstance();
                                        break;
                                case R.id.navigation_add:
                                        selectedFragement = AddFragment.newInstance();
                                        break;
```

```
                        }
                        itemSaved = item.getItemId();
                        FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();
                        transaction.replace(R.id.main_fragment, selectedFragement);
                        transaction.commit();
                        return true;
                }
};
```

Je maîtrise l'utilisation de Git

Nous avons essayé de découper le travail avec un système de branch avait d'éviter au maximum de toucher au master.

Application

Je sais utiliser la caméra

Nous avons réussi à implémenter un système de caméra 100% fonctionnel, de l'instanciation à la destruction de celle ci. Les classes CameraPreview, CameraActivity et CameraManager intègrent tout les fonctionnalités d'une caméra. La classe CameraPreview gère l'affichage de la preview de la caméra et c'est celle qui est visible dans la vue activity_camera.xml, gérè par le controlleur CameraActivity. CameraActivity dépend de CameraManager car c'est cette classe qui s'occupe de l'instanciation de la caméra, et qui gère aussi l'orientation de la caméra en fonction de la position du téléphone.