Barou Romain
Chanson Chloé

Dossier Android – Preuves
Application « Reing rise »

29/03/2020

Documentation :

Je sais décrire le contexte de mon application, pour que n'importe qui soit capable de comprendre à quoi elle sert :

Voir le fichier « dossier Android » à la première page, nous avons expliqué comment fonctionne l'application de son lancement à sa fermeture, ainsi que le but du jeu,

Je sais faire un diagramme de cas d'utilisation pour mettre en avant les différentes fonctionnalités de mon application.

Voir le diagramme de cas d'utilisation dans le fichier « dossier Android » à la première page. (Tous les packages et classes de l'application sont présents, le diagramme est un minimum clair malgré les nombreuses relations interclasses.).

Je sais concevoir un digramme UML de qualité représentant mon application.

Voir le fichier « dossier Android » de la page 2 à 6. Nous avons décrit la façon dont sont séparés les packages, quelles classes sont appelées à quelle moment et de quelle façon tout au long de l'utilisation de l'application.

Je sais décrire un diagramme UML en mettant en valeur et en justifiant les éléments essentiels.

Voir le fichier « dossier Android » à la page 7. Nous avons décrit la façon dont sont séparés les packages, quelles classes sont appelées à quelle moment et de quelle façon tout au long de l'utilisation de l'application.

Code :

Je sais utiliser les intent pour faire communiquer deux activités.

Le pseudo du joueur circule d'une activité à l'autre grâce aux Intents, le score circule de la gameActivity à l'activité de tableau des scores.

Preuve : Voir les lignes suivante dans le code : 66 et 68 de la classe GameActivity, dans la classe ClikManageur de la ligne 44 a 69, et dans la classe ActivityTableauDesScores de la ligne 38 à 41.

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

L'application a été créé en utilisant un SDK compatible sur tous les portables, ce dernier a été augmenté uniquement en cas de besoin pour implémenter des fonctionnalités qui étaient grandement nécessaires à l'application. Preuve :

Dossier Android – Preuves
Application « Reing rise »

Je sais distinguer mes ressources en utilisant les qualifier.

XXX

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

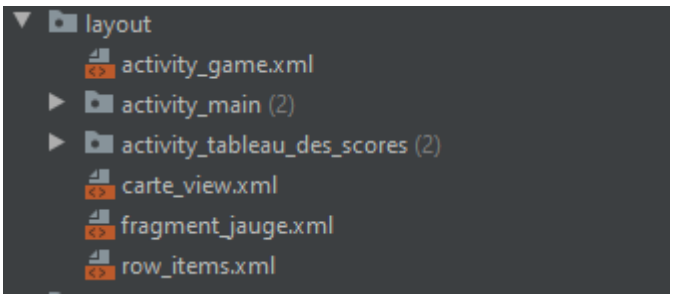
Le manifest a été modifié afin de rajouter les permissions nécessaires à l'utilisation du GPS (Internet et Access_Fine_Location) et afin de bloquer l'orientation de GameActiity. Preuve :Voir le dossier manifest de l'application les ligne suivante :

```
18        <activity android:name=".Vues.GameActivity" android:screenOrientation="portrait"/>
```

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

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

Nous avons produit un code XML dans le fichier layout pour chaque activité ainsi que pour la vue des cartes. Nous avons utilisé divers layout tel que des ListView, LinearLayout, CardStackView, RelativeLayout… Preuve :Voir le dossier layout dans res :



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

Les activités font appel à un Controller pour les clics. Preuve :Voir le package « Vue » de l'application.

Je sais coder une application en ayant un véritable métier.

Les scores et les cartes sont stockées à l'aide de classes « Score » et « Carte » dans le package « Métiers » de notre application. Preuve :Voir le package « Metier » de l'application.

Je sais parfaitement séparer vue et modèle.

Les vues transmettent ce qui est nécessaires au « Modele » et minimisent le traitement de données. Preuve :Voir le package « Modele » de l'application.

Dossier Android – Preuves
Application « Reing rise »

Je maitrise le cycle de vie de mon application.

La sauvegarde du pseudo, et des scores est bien réalisé lorsqu'on passe par des OnDestroy ou OnStop, l'expérience utilisateur n'est pas altérée lors de l'évolution du cycle de vie de l'application. Preuve : Voir dans le code des ligne 58 à 72, puis 75 à 79, puis 84 à 89 et de 98 à 111 de la MainActivity.

Je sais utiliser le findVIewById à bon escient.

Le findViewById est utilisé pour récupérer des champs des vues, des id d'actions etc...
Preuve : Voir dans le code aux lignes 83, 139, 71 de la MainActivity par exemple.

Je sais gérer les permissions dynamiques de mon application.

Nous avons mis en place un système demandant une autorisation pour le GPS.
Preuve :

```
100         if(ContextCompat.checkSelfPermission(
101                 getApplicationContext(), Manifest.permission.ACCESS_FINE_LOCATION
102         )!= PackageManager.PERMISSION_GRANTED){
103             ActivityCompat.requestPermissions(
104                     activity: MainActivity.this,
105                     new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
106                     REQUEST_CODE_LOCATION_PERMISSION
107             );
108         }else{
109             gps.getCurrentLocation( activity: this);
110         }
```

Je sais gérer la persistance légère de mon application.

Le pseudo est conservé dans son champ de saisi dans le cas où on retourne la tablette (passage par OnDestroy). Preuve :

```
74      @Override
75      public void onRestoreInstanceState(Bundle savedInstanceState) {
76          editText =(EditText) findViewById(R.id.editTextSurname);
77          namer = savedInstanceState.getString(SAVED_NAME);
78          editText.setText(namer);
79      }
80
81
82      // invoked when the activity may be temporarily destroyed, save the instance state here
83      @Override
84      public void onSaveInstanceState(Bundle outState) {
85          editText =(EditText) findViewById(R.id.editTextSurname);
86          String namer = editText.getText().toString();
87          outState.putString(SAVED_NAME, namer);
88          super.onSaveInstanceState(outState);
89      }
```

Dossier Android – Preuves
Application « Reing rise »

Je sais gérer la persistance profonde de mon application.

Les scores sont sauvegardés en base de données et sont réaffichés en fin de partie, la persistance profonde est gérée grâce à une base de données SQLite afin de conserver les scores et les cartes. Preuve : Voir dans le package BDD.

Je sais afficher une collection de données.

En fin de partie, notre score et celui des autres parties (sur l'appareil) sont affichés dans une « ListView ». Preuve : fichier (dans l'odre des screens ActivityTableauDesScores.java, activity_tableau_des_scores.xml et row_items.xml.

```
42          adapter= new MyAdapter(listeScore.getListeScore(), context: this);
43          listView=(ListView)findViewById(R.id.listView);
44          listView.setAdapter(adapter);
```

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3       android:orientation="vertical"
4       android:layout_width="match_parent"
5       android:layout_height="wrap_content">
6
7       <TableLayout
8           style="@style/frag2TableLayout"
9           android:layout_width="wrap_content"
10          android:layout_height="wrap_content"
11          android:background="@android:color/background_light"
12          android:padding="15dp">
13
14          <TableRow style="@style/frag1HeaderTableRow">
15              <TextView
16                  android:layout_width="match_parent"
17                  android:layout_height="wrap_content"
18                  android:layout_gravity="center"
19                  android:id="@+id/surnameJoueurTab"/>
20              <TextView
21                  android:layout_width="wrap_content"
22                  android:layout_height="match_parent"
23                  android:layout_gravity="center"
24                  android:id="@+id/scoresTab"/>
25          </TableRow>
26      </TableLayout>
27  </LinearLayout>
```

Dossier Android – Preuves
Application « Reing rise »

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TableLayout
        style="@style/frag2TableLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@android:color/background_light"
        android:padding="15dp">

        <TableRow style="@style/frag1HeaderTableRow">
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:id="@+id/surnameJoueurTab"/>
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_gravity="center"
                android:id="@+id/scoresTab"/>
        </TableRow>
    </TableLayout>
</LinearLayout>
```

Je sais coder mon propre adaptateur.

Nous possédons deux adapters : la classe MyAdapter et la classe CardStackAdapter, la première a été réalisée par nous même, pour la deuxième nous nous sommes inspirés du travail de l'utilisateur Github YuyaKaido car l'affichage de cartes Swippables étaient un élément essentiel à l'essence même de notre application et nous n'avions pas les connaissances requises pour le faire seul. Preuve : Voir dans le package « Adaptateur ».

Je maitrise l'usage des fragments.

Les jauges « Richesse », « Puissance » et « Revolte » sont des fragments et sont affichées durant la partie (ainsi que l'évolution de leur remplissage). Preuve : Voir fragment_jauge.xml, FragmentJauge.java et aux lignes dans le fichier GameActivity.java (voir screen suivant)

```
69              ft.replace(R.id.fragment1, fragments[0]);
70              ft.replace(R.id.fragment2, fragments[1]);
71              ft.replace(R.id.fragment3, fragments[2]);
72              ft.commit();
```

Dossier Android – Preuves
Application « Reing rise »

```
51      private FragmentJauge fragment1 = FragmentJauge.newInstance("Pouvoir");
52      private FragmentJauge fragment2 = FragmentJauge.newInstance("Richesse");
53      private FragmentJauge fragment3 = FragmentJauge.newInstance("Révolte");
54      private FragmentJauge fragments[] = {fragment1,fragment2,fragment3};
55
```

Je maitrise l'utilisation de Git.

Nous avons uniquement utilisé Git (pas de périphérique physique pour transférer du code), nous avons réalisé de nombreux commit, push et pull, des conflits ont été corrigés et nous avons utilisé des branches pour chaque fonctionnalité majeure ajoutée. Preuve : Voir la forge.

Application :

Je sais développer une application publiable sur le store.

Voir le Play store.

Je sais utiliser le GPS

Lorsqu'on démarre l'application, on nous demande si on veut être géolocalisé, notre domaine (adresse) est alors affiché si on est d'accord. Preuve : Voir les classes GPS et FetchAddressIntentService dans le package « Modele » et dans MainActivity le screen suivant.

```
146     private class AdressResultReceiver extends ResultReceiver{
147         public AdressResultReceiver(Handler handler) { super(handler); }
150         @Override
151         protected void onReceiveResult(int resultCode, Bundle resultData){
152             super.onReceiveResult(resultCode,resultData);
153             if(resultCode == Constantes.SUCCESS_RESULT){
154                 textAdresse.setText(resultData.getString(Constantes.RESULT_DATA_KEY));
155             }else{
156                 Toast.makeText( context: MainActivity.this,resultData.getString(Constantes.RESULT_DATA_KEY),Toast.LENGTH_SHORT).show();
157             }
158         }
159     }
```