GAILLARD Victor
BROQUET Mattéo

Preuves Maitrises des Compétences

Documentation : (dossier doc)

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

→ Le contexte se trouve dans le README.

2. **Je sais concevoir et décrire un diagramme de cas d'utilisation pour mettre en avant les différentes fonctionnalités de mon application.**

→ Les explications du diagramme classe se trouve dans le dossier Documentation

3. **Je sais concevoir un diagramme UML de qualité représentant mon application.**

→ Le diagramme de classe se trouve dans le dossier Documentation

4. **Je sais décrire mon diagramme UML en mettant en valeur et en justifiant les éléments essentiels.**

→ La description du diagramme de classe se trouve dans le dossier Documentation

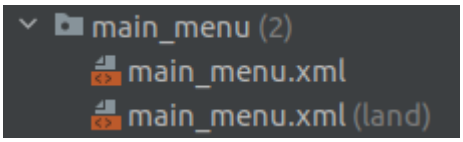
Code :

1. **Je sais utiliser les Intent pour faire communiquer deux activités.**

→ Transmission du pseudo entre le menu et le jeu

2. **Je sais développer en utilisant le SDK le plus bas possible.**
3. **Je sais distinguer mes ressources en utilisant les qualifier**

→ Nous avons utilisé le qualifier *land* pour posséder une vue qui soit différente en portrait et en paysage



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

→ Nous avons fait des vues pour le menu, les paramètres, l'affichage des scores et une dernière pour la partie. Nous avons utilisé le constraint layout en tant que composant principal de ces vues car il est responsive et s'adapte donc à tout type d'écran

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

→ La classe *ActivityGame* est notre activité principale. Elle ne prend aucune décision à la place du modèle

6. **Je sais coder une application en ayant un véritable métier**

→ Notre model métier n'interagit pas directement avec la vue

7. **Je sais parfaitement séparer vue et modèle**

→ Tout le modèle est séparé de la vue. En effet, le modèle a été réutilisé d'un projet qui n'était pas sous android, donc la partie vue se devait d'être séparée. Si l'on regarde la classe *ActivityGame* on remarque que cette classe de la vue de prend aucune décision sur le modèle

8. **Je maîtrise le cycle de vie de mon application**

→ Arret du Thread et reprise de celui ci

9. **Je sais utiliser le findViewById à bon escient**

→ Nous avons utilisé *findViewById* pour récupérer des éléments de la vue dans le code behind. Pour ajouter une action aux boutons du menu, nous avons dû récupérer le bouton dans le code behind.

```
Button buttonPlay = findViewById(R.id.menu_buttonPlay);
buttonPlay.setOnClickListener(this::onClickPlay);
```

10. **Je sais gérer les permissions dynamiques de mon application**
11. **Je sais gérer la persistance légère de mon application**

→ Game Activity

12. **Je sais gérer la persistance profonde de mon application**

→ Nous avons utilisé la persistance profonde pour sauvegarder les scores des précédentes parties, malgré la fermeture de l'application.

```
loader = new FileLoader();
try {
    save.save(openFileOutput(PATHToScores, MODE_PRIVATE), scores);
} catch (FileNotFoundException e) {
    Log.e(getPackageName(), msg: "Save failed");
}
```

13. **Je sais afficher une collection de données**

→ Nous avons utilisé une *RecyclerView* afin d'afficher la liste des meillleurs scores. Elle se trouve dans la vue *ScoreActivity*.

```
scoresRecyclerView = findViewById(R.id.scoresRecyclerView);
scoresRecyclerView.setLayoutManager(new LinearLayoutManager(getApplicationContext()));

textViewHUD = findViewById(R.id.textViewHUD);

scoresView.addAll(scores);

scoresRecyclerView.setAdapter(new ArrayToView(scoresView));
```

14. **Je sais coder mon propre adaptateur**

→ Afin de pouvoir afficher une *ArrayList* dans notre *RecyclerView*, nous avons été obligé de faire un adaptateur *ArrayToView.*

```java
@NonNull
@Override
public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    LinearLayout layoutScore = (LinearLayout) LayoutInflater.from(parent.getContext()).inflate(R.layout.score_field, parent,  attachToRoot: false);
    return new ViewHolderScore(layoutScore);
}

@Override
public void onBindViewHolder(@NonNull RecyclerView.ViewHolder holder, int position) {
    GameState scoreCourant = scores.get(position);
    Log.d( tag: "Pseudo",scoreCourant.getPseudo());
    Log.d( tag: "Score", String.valueOf(scoreCourant.getScore()));
    ((ViewHolderScore)holder).getPseudoViewScore().setText(scoreCourant.getPseudo());
    ((ViewHolderScore)holder).getScoreViewScore().setText(String.valueOf(scoreCourant.getScore()));
}
```

15. Je maîtrise l'usage des fragments

→ Nous avons un fragment *MasterScoreFragment* pour l'affichage des scores.

```java
@Override
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);

    activiteParente = (ScoreActivity) getContext();
    RecyclerView laListView = view.findViewById(R.id.laListView);
    laListView.setLayoutManager(new LinearLayoutManager(activiteParente));
    laListView.setAdapter(new ArrayToView(activiteParente.getScores()));
}
```

16. Je maîtrise l'utilisation de Git

→ Nous avons utilisé plusieurs branches lorsque nous avons fait des tests, ou des améliorations, qui demandait de modifier des choses déjà implémentée ou qui aurait pu créer des bugs. Ceci permet de garder une version stable sur le *master*

Application :

1. **Je sais développer une application sans utiliser de librairies externes.**

→ Aucune librairie n'a était utilisée pour ce projet. Ce qui est utilisé vient de l'ancien projet et des outils fournis par android.

2. **Je sais développer une application publiable sur le store.**
3. **Je sais développer un jeu intégrant une boucle de jeu threadée observable.**
4. **Je sais développer un jeu graphique sans utiliser de SurfaceView.**

→ Nous n'avons pas utilisé de SurfaceView mais des ImageView