Rapport de TP : Développement d'une application Android

MathQuiz

Nom : Wissal Kabbour Groupe : GINF3

3 décembre 2025

Introduction

Dans le cadre du module de développement mobile, l'objectif de ce TP est de concevoir une application Android simple nommée **MathQuiz**. Elle permet de générer deux nombres aléatoires et d'effectuer des opérations arithmétiques de base : addition, soustraction et multiplication. Ce TP constitue une introduction à Android Studio, à la structure d'un projet Android, aux fichiers de ressources XML et à la programmation événementielle en Java.

Objectifs du TP

Les objectifs pédagogiques sont les suivants :
— comprendre la structure d'un projet Android (manifest, activités, ressources, Gradle) ;
— manipuler les fichiers de mise en page XML ;
— relier les composants de l'interface au code Java via les identifiants `R.id.*` ;
— implémenter des écouteurs d'évènements avec `OnClickListener` ;
— générer aléatoirement des données et les afficher dans l'interface ;
— compiler, lancer et tester l'application sur un émulateur ou un appareil réel.

Conception de l'interface utilisateur

L'interface a été conçue dans le fichier `activity_main.xml` en utilisant un `LinearLayout` vertical. Elle comporte les éléments suivants :
— un titre présentant le nom de l'application ;
— deux `TextView` affichant les nombres aléatoires ;
— un `TextView` indiquant le résultat de l'opération ;
— trois boutons pour les opérations : $+$, $-$ et $\times$;
— un bouton **Générer** permettant de produire une nouvelle paire de nombres.

Les ressources textuelles sont définies dans le fichier `strings.xml`, ce qui permet une meilleure organisation et une éventuelle traduction.

Logique applicative (Java)

La logique principale se trouve dans la classe `MainActivity.java`. Lors du lancement de l'application, la méthode `onCreate()` initialise les composants graphiques grâce à `findViewById()` et génère une première série de nombres aléatoires.

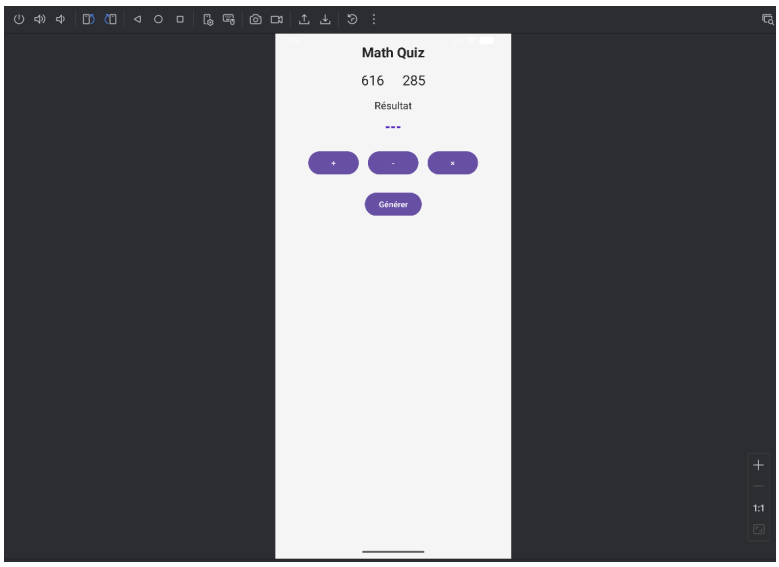

FIGURE 1 – Interface utilisateur de l'application MathQuiz.

Génération de nombres

Les deux entiers sont générés dans l'intervalle $[111, 999]$:

```
number1 = random.nextInt(889) + 111;
number2 = random.nextInt(889) + 111;
```

Ils sont directement affichés dans les deux `TextView` associés.

Gestion des événements

Chaque bouton possède un écouteur permettant de déclencher l'opération correspondante. Par exemple, pour l'addition :

```
btnAdd.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        int result = number1 + number2;
        tvResult.setText(String.valueOf(result));
    }
});
```

Une logique similaire est appliquée pour la soustraction et la multiplication. Le bouton **Générer** rappelle la méthode `generateNumbers()` et réinitialise le résultat.

Problèmes rencontrés et solutions

Durant le développement, plusieurs problèmes ont été rencontrés :

- **Erreur de synchronisation Gradle** : causée par des processus bloqués et résolue en supprimant le cache Gradle.
- **Erreur** `Cannot resolve symbol R` : due à une erreur dans un fichier XML. Résolution : corriger les ressources puis reconstruire le projet.
- **Problèmes de push GitHub** : incompatibilité entre la branche locale et distante. Solution : effectuer un `git pull -allow-unrelated-histories`.

Ces difficultés ont permis de mieux comprendre le fonctionnement de l'environnement Android et des outils associés.

Résultat final

L'application obtenue fonctionne correctement :
— les nombres sont générés dynamiquement ;
— les trois opérations sont exécutées sans erreur ;
— l'interface est simple, lisible et adaptée pour l'usage prévu ;
— l'utilisateur peut générer autant de combinaisons qu'il le souhaite.

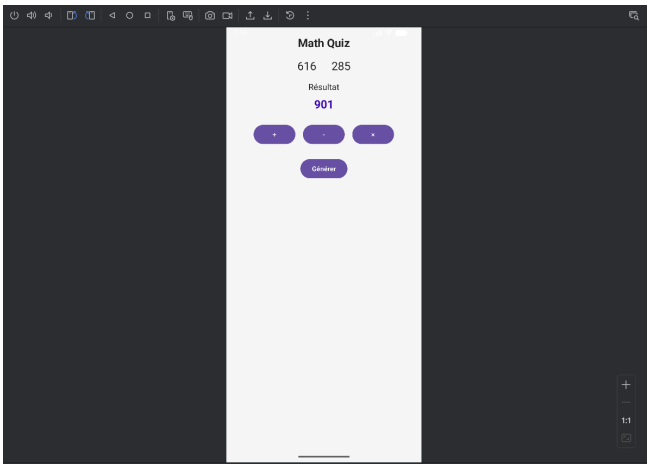

FIGURE 2 – Affichage d'un résultat après une opération.

Conclusion

Ce TP a permis d'acquérir les bases du développement Android : gestion d'interface, manipulation d'évènements, interaction XML/Java et résolution d'erreurs liées à Gradle ou Git. L'application **MathQuiz** est fonctionnelle et respecte les objectifs demandés. Cette première approche ouvre la voie vers des projets plus complexes intégrant des layouts avancés, des activités multiples ou une logique métier plus poussée.