Rapport de l'application Jeu de math

ISSOUFFA Ambdouloihi Ibnou et MEHONG SHIT LI Matthieu, L3 informatiqe

2 mai 2021

Résumé

Dans ce projet d'application mobile, nous verrons l'intégration d'une application android et de sa version iOS. Ces deux applications sont toutes deux responsives. Nous verrons notamment, une nouvelle manière d'utiliser les canvases.

1 Introduction

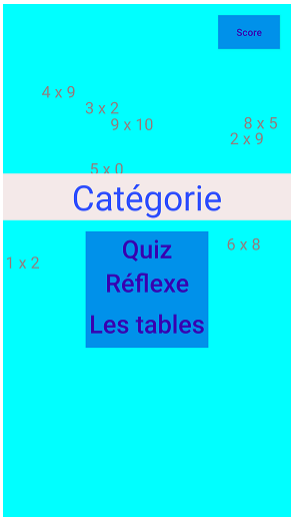
Nous avons tous fait du calcul mental durant notre enfance. Pour la plupart d'entre nous ce n'était pas vraiment agréable. Aujourd'hui, nous vous proposont notre application. Une application mobile, qui permet d'apprendre à retenir ses tables de multiplication tout en jouant. Nous allons découvrir ensemble ce que nous avons codé.

2 Description générale de l'application

Vous avons 8 ecrans en tous :

1. Acceuil
2. Catégorie
3. Quiz
4. Réflexe
5. Table de Multiplication
6. Score
7. Question du Quiz
8. Resultat au Quiz

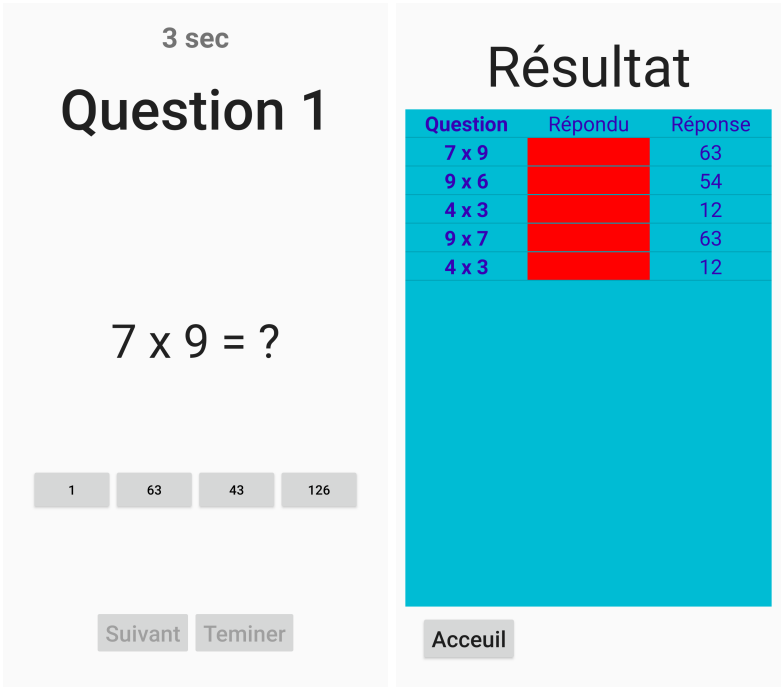
Voici une capture d'écran de la page **catégorie** :



Sur cette page, nous pouvons acceder à :
— la page du **quiz** de calcul mental
— la page des score au jeu de **réflexe**
— la page du jeu de **réflexe**
— la page des **tables** de multipliction

3 Intégration des fonctionnalités

3.1 Le jeu de Quiz



Le jeu de quiz est la seul fonctionnalité qui est disponible sur les deux plateformes. Les questions sont défilées sur un seul écran. Cet écran est mise à jour à chaque fois que nous cliquons sur le bouton **suivant**. Il y'a 5 questions en tout. L'utilisateur à 5 secondes pour chaque question. À la fin du quiz, une page **résultat** affiche les bonnes réponses de l'utilisateur. Grace à nos algorithmes, les pieges permettent d'ajouter une dificulter au jeu.

3.1.1 Android

— Le Timer, vient de cette page [3],et il a été adapter au code.
— Les buttons sauvegardent la dernière sélection de l'utilisateur en changeant de couleur.
— Les lignes de colonnes de **"répondu"**(ce qui a été répondu) sont en vert si la réponse de l'utilisateur est correcte sinon elle est en rouge.
— La page résultat implémente la ListViews
— Les données nécessaires pour afficher la page résultat sont envoyées via un bundle. Ce bundle est composé de clé qui sont les numéros de ligne et d'une liste de 3 strings(question, réponse du joueur, réponse de la question).

— Les réponses aléatoires sont générées par L'algorithme **setAnswerList** ci-dessous :

```
private fun setAnswerList() {
        val list = ArrayList<Int>()
        val max = if (calcul.answer!=0) calcul.answer + calcul.answer / 2 else 10
        var piege: Int
        for (i in 1..2) {piege=getRandInt(max); list.add(piege)} // 2 reponses pieges
        piege = if (calcul.answer!=0) 2*calcul.answer else getRandInt(10)
        list.add(getRandInt(list.size-1),piege) // reponse piege
        list.add(getRandInt(list.size-1), calcul.answer) // la reponse
        answerList = list
    }
```

3.1.2 iOS

— Le Timer, vient de cette page [4],et il a été adapter au code.

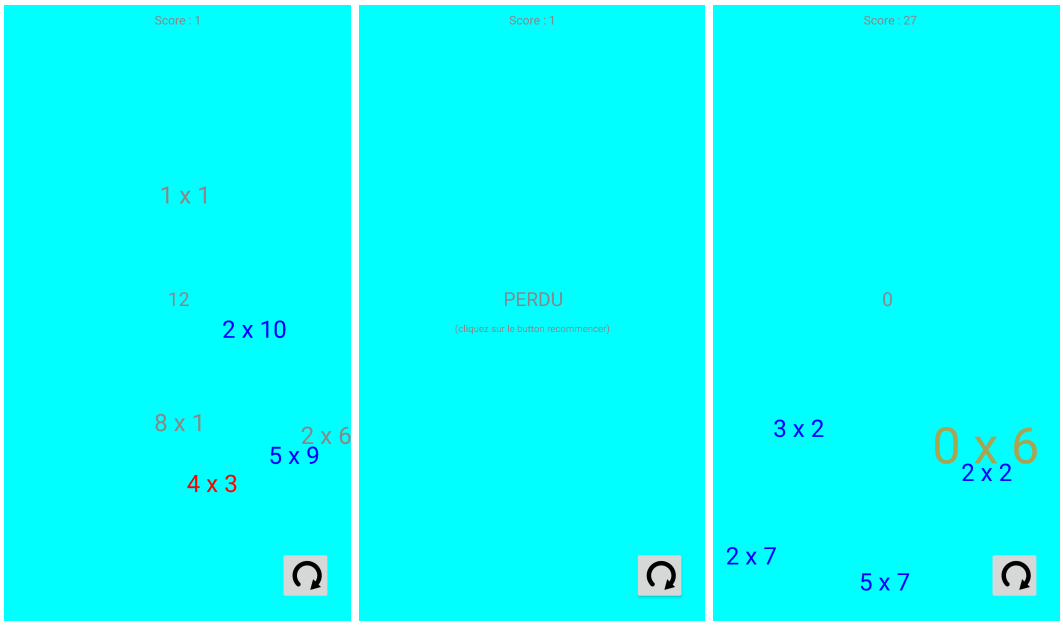
— Les buttons sauvegardent la dernière sélection de l'utilisateur car ce sont de Segmented-Control.

— Les colonnes de répondues sont en vert si la réponse de l'utilisateur est correcte sinon elle est en rouge. page résultat implémente la TableView

— Les données de la page quiz sont partagées par la méthode **prepare**

```
    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        if segue.identifier == "goResult" {
            let vc=segue.destination as! ResultatViewController
            vc.allRepUser.append(contentsOf: self.allRepUser)
            vc.allRep.append(contentsOf: self.allRep)
            vc.allCalc.append(contentsOf: allCalc)
        }
    }
```

3.2 Le jeu de Reflexe sous Android



Le jeu de réflexe consiste à cliquer sur le calcul correspondant au nombre affiché au centre de la page. Si l'utilisateur clic sur le bon calcul associé à la réponse, son score augmente et la couleur du calcul devient bleue. Sinon, la couleur devient rouge. Et si la couleur du calcul est rouge et qu'il est cliqué, celle-ci redevient gris qui est sa couleur de base. Lorsque tous les calculs qui viennent du haut de l'écran disparaissent en bas, le jeu

Nous avons réussi à mettre tous le code du jeu de réflexe dans la classe View. En utilisant les méthodes **onDraw**, **onSizeChanged** et **onTouchTvent**. Le bouton **restart** est le seul qui est se situe dans le layout XML associé au jeu de réflexe.
Pour initialiser le bouton restart, voici le code :

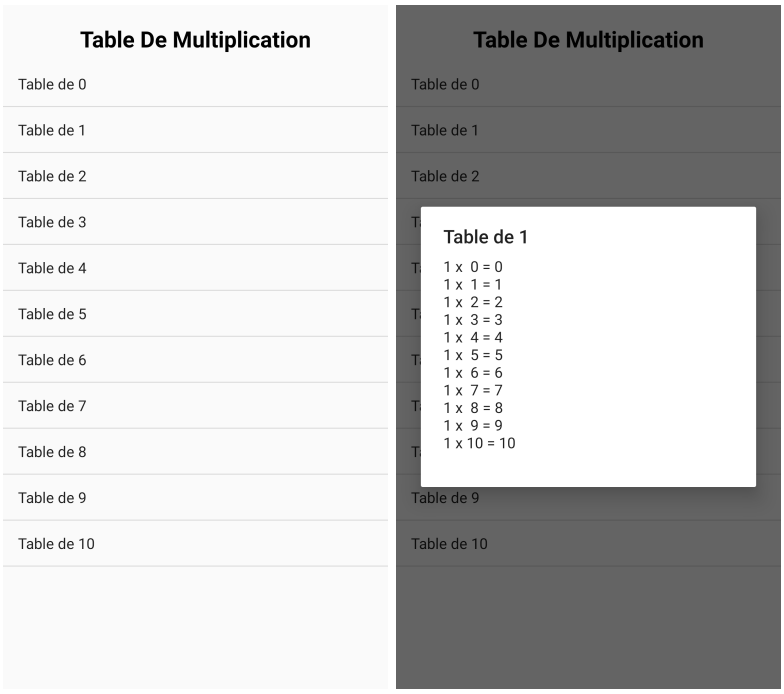
```
val canvasParent = parent.parent as ConstraintLayout
    btnRestart = canvasParent[1] as Button
    btnRestart.setOnClickListener { restart() }
    btnRestart.isEnabled = false
```

Ce genre de code nous aurait pu permettre de gérer les d'autres view de la page XML et intégrere le modèle MVC. La méthode **onSizeChanged** permet d'initialiser les variables et d'obtenir la taille du canvas. onTouchEvent permet de gérer les clics de l'utilisateur et **onDraw** dessine ce nous avons besoin d'afficher.

Il est normal de penser que le score maximum est majoré au nombre du calcul donner. Mais heureusement, nous avons une fonctionnalitée cachée qui permet d'aller au-delà. Il suffit de cliquer plusieurs fois sur le dernier bon calcul devenu bleu, avant qu'elle arrive à en bas de l'écran. La taille du calcul, et le score augmentera. Le calcul changera de couleur aussi aléatoirement.

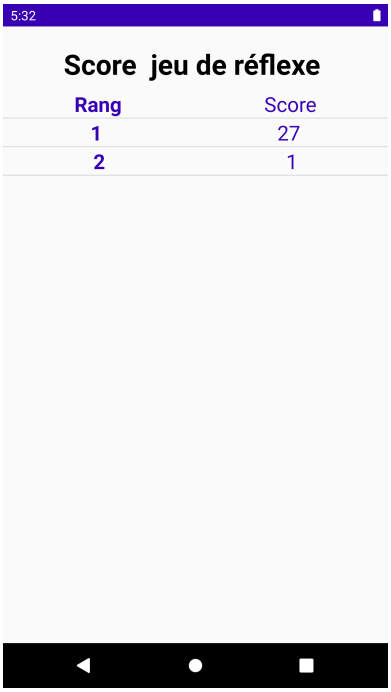
Enfin, nous avons utilisé cette view pour créer un fond d'écran animé disponible dur l'écran d'accueil et de catégorie.

3.3 Les Tables de Multiplication sous Android



Une application basée sur les calculs mentaux se doit d'avoir les tables de multiplication en son sein pour ne pas faire fuir les utilisateurs. C'est pour cette raison que cette page est utile. Cette page affiche la liste des tables de multiplication allant de 0 à 10. En clichant sur un item, un pop-up apparaît avec la table correspondant à l'item.

3.4 La Table des Meilleurs scsore sous Android



Cette table conserve les 3 meilleurs scores de manière persistante. Cela est possible grâce à la classe **SQLiteOpenHelper**. Nous avons suivi ce tutoriel [2] et nous l'avons adapté à notre projet, en rajoutant notamment la méthode **updateData** dans notre class **DataBaseHandler** .

La sauvegarde des scores se fait lorsque l'état du jeu de réflexe **est Fini** est vrai. Le code se trouve dans la méthode **dessinerLesCalculs**. Voici le code :

```
if (estFini) {
        btnRestart.isEnabled = true

        //sauvegarde du score
        var db = DataBaseHandler(this.context)
        val allScore = db.readData()

        when {
            (allScore.size < 3) && (!allScore.contains(score))  -> db.insertData(score)
            (3 <= allScore.size) && (!allScore.contains(score)) -> {
                val lastScore = allScore[allScore.lastIndex]
                db.updateData(lastScore, score)
            }
        }
    }
```

3.5 La ListView sous Android

Grace la classe **ArrayAdapter<T>** nous avons pu facilement afficher mes données avec le style que je voulais. Cet exemple [1] nous a permis d'utiliser layout que nous avons créé pour remplir la ListView. C'est un code Java que nous avons traduit en Kotlin :

```
class ArrayAdaptaterScore: ArrayAdapter<Score>(this, R.layout.my_list_item_2score, R.id.questRes
    override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {
    var view:View= super.getView(position, convertView, parent)
    val text1:TextView = view.findViewById(R.id.questResult)
    val text2:TextView = view.findViewById(R.id.repUserResult)
    text1.setText(listeTable.get(position).rang)
    text2.setText( listeTable.get(position).score )
    return view
    }
}
val arrayAdaptaterScore=ArrayAdaptaterScore()
listView.adapter = arrayAdaptaterScor
```

4 Quelques points délicats/intéressants

4.1 Android

— La version android de l'application est complète. Elle possède même une icône créer à partir d'Illustrator.

— L'icon du bouton restart a été creer sur Illustrer.

4.2 iOS

Malheureusement par manque de temps seul les écrans suivants sont fonctionnels et responsifs :

1. Acceuil
2. Catégorie
3. Quiz
4. Question du Quiz
5. Resultat au Quiz

4.3 Autre

Nous aurions voulu ajouter une page regroupant la liste d'erreur obtenue durant les quiz. Pour que l'utilisateur sache ce qu'il a besoin de reviser.

5 Conclusion

En conclusion nous avons parcouru les fonctionnalités de notre application. Plein d'autres fonctionnalité aurait pu être ajouter, mais dans l'ensemble l'application est déjà assez complète.

Références

[1] https://stackoverflow.com/a/18529511.

[2] https://www.tutorialspoint.com/how-to-use-a-simple-sqlite-database-in-kotlin-android.

[3] https://www.tutorialspoint.com/how-to-set-a-timer-in-android-using-kotlin.

[4] https://learnappmaking.com/timer-swift-how-to/.