École Centrale de Lille
MIN
Mathématiques -Informatique

Séquence 2

CR TP 2

Élèves: Cristopher Pérez
Asunción Gómez Colomer
Enseignant: Thomas Bourdeaud'huy

Programmation mobile et réalité augmentée
Date: 7 de junio de 2021
Villeneuve d'Ascq, 59650

Table des Matières

Liste des figures

Liste des Codes

1. Introduction

Ce travail a comme but améliorer les quatre activités développés a la séquence 2, en récupérant les listes et les items des utilisateurs depuis une API Rest

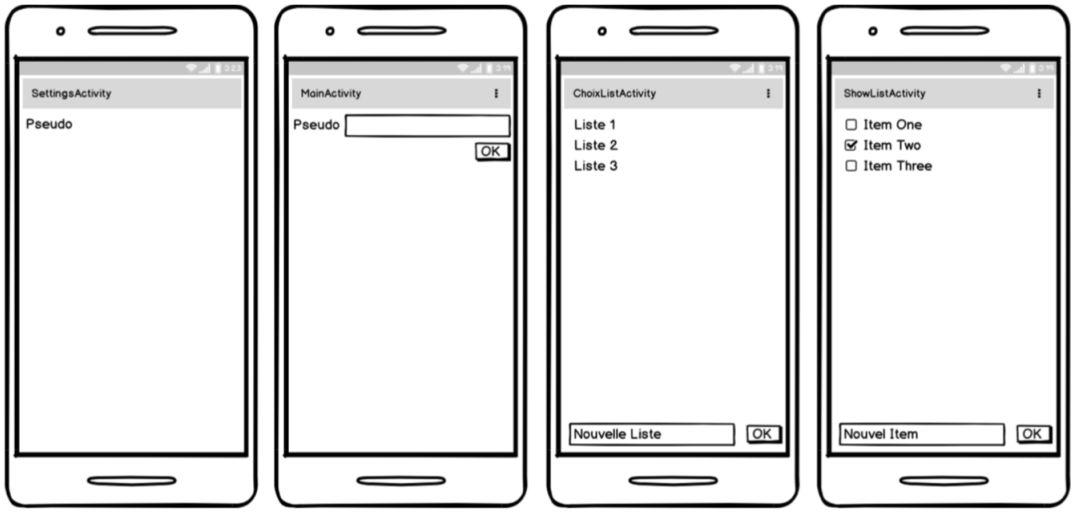

Figure 1: Storyboard

2. Main Activity

Pour le Main Activity la différence principal c'est que maintenant on aurait un login avec un mot de passe alors notre écran on le voit dans l'image 2.

Cette fois-ci le bouton Ok va récupérer les donnes, c'est à dire le pseudo et le mot de passe pour après utiliser ces donnes comme token pour faire la requête a l'API. Si la requête est correcte, on reçoit: 'D/OkHttp: "version":1.1,"success":true,"status":202,"hash":"25760fd2eec0ce18a85d998b6e73de1c

<– END HTTP (85-byte body)'

Figure 2: MainActivity

Pour faire tout ça on a fait une fonction login qu'on peut observer dans (1)

Code 1: Fonction Login

```
fun login(){
        //Log.i("PMR", "clickok")
        Toast.makeText(this, "ok", Toast.LENGTH_SHORT).show()
        //val l = sp.getString("login", "gf")
        activityScope.launch {
            try{
                Toast.makeText(this@MainActivity, "ok2", Toast.LENGTH_SHORT).show()
                val hash: String = connexion(Pseudo.toString(), Mdp.toString())
                if (!hash.isEmpty()) {
                    //Garder dans shared preferences
                    editor.putString("login", Pseudo?.text.toString())
                    editor.commit()
                    val l=sp.getString("login","null")
                    Pseudo?.setText(l.toString())
                    //Changer Activite
                    val versSecondAct: Intent =
                        Intent(this@MainActivity, ChoixListActivity::class.java)
                    //Envoyer donnes
                    //versSecondAct.flags = Intent.FLAG_ACTIVITY_NEW_TASK or
    Intent.FLAG_ACTIVITY_CLEAR_TASK
```

```
                    versSecondAct.putExtra("pseudo", Pseudo?.text.toString())
                    versSecondAct.putExtra("hash", hash)
                    // todo
                    versSecondAct.putExtra("id_user", '1')
                    startActivity(versSecondAct)
                } else {
                    Toast.makeText(this@MainActivity, "error", Toast.LENGTH_SHORT).show()
                }
            } catch (e: Exception){
                Toast.makeText(this@MainActivity, "{e.message}",
    Toast.LENGTH_SHORT).show()
            }
        }
    }
```

3. SettingsActivity

Pour cette activité on a utilisé un fichier 'preferences.xml'. Le code Kotlin de cette activité est écrit dans le code 2 et une visualisation se montre dans la figure 3. Il faut mentionner que la classe 'PreferenceCategory' est déplorée.

Code 2: SettingsActivity

```
class SettingsActivity :PreferenceActivity(){
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        addPreferencesFromResource(R.xml.preferences);
    }

}
```

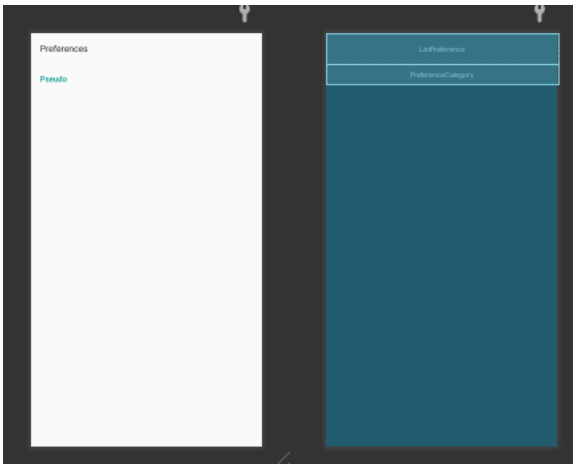

Figure 3: SettingsActivity

4. ChoixListActivity

Pour cette activité on a utilisé une 'RecyclerView'. 'RecyclerView' est le ViewGroup qui contient les vues correspondant à vos données. Chaque élément individuel de la liste est défini par un objet view holder. Lorsque le view holder est créé, aucune donnée ne lui est associée. Une fois le view holder créé, le RecyclerView le lie à ses données. On définie le view holder en étendant RecyclerView.ViewHolder. Le RecyclerView demande ces vues, et lie les vues à leurs données, en appelant les méthodes de l'adaptateur. On défini l'adaptateur en étendant RecyclerView.Adapter. Après pour détecter l'élément tapé il fallait utiliser le 'listener' de notre adaptateur, et override la fonction avec la tache désire que dans notre cas est afficher les éléments de la liste dans un nouvelle activité Et pour ajouter des nouveaux elements a la liste on a utilise un 'EditText' et un'Button' que dans le moment qu'on tap sur le button, le texte écrit s'ajoute a notre liste.

Dans cette partie, on a utilisé le hash récupéré lors de l'activité principal, l'identifiant de l'utilisateur et son pseudo. De cette façon, on a réussi à faire une requête de type GET pour avoir les listes de l'utilisateur. Si la requête est bien faite, on reçoit 'D/OkHttp: "version":1.1,"success":true,"status":200,"lists" $--ENDHTTP(798-bytebody)'$.

Puis, on a aussi implementé la création d'une nouvelle liste grâce à une requête POST (voir code **??** et 4).

Code 3: POST création d'une liste

```
// creation of a new list
    @POST("users/{id}/lists")
    suspend fun createList(@Path("id")id: Int,
                           @Query("label")label:String,
                           @Header("hash")hash: String): List
```

Code 4: Adapter: création d'une nouvelle liste

```
b.setOnClickListener {
        // to change --> with the user
        activityScope.launch {
            try{
                Toast.makeText(this@ChoixListActivity, "For id user: $id_user",
    Toast.LENGTH_SHORT).show()
                if(id_user_int!=null && hash!=null){
                    recyclerView.visibility = View.GONE
                    val newListName = t.text.toString()
                    Toast.makeText(this@ChoixListActivity, newListName,
    Toast.LENGTH_SHORT).show()
                    // add the new list
                    val new_list = createList(id_user_int, newListName, hash)
                    val listReady : List<com.example.tp1.data.model.List> =
    listOf(new_list)
                    adapter.addData(listReady)
                    //val lists = DataProvider.getListsFromApi(hash)
                    //adapter.addData(lists)
```

```
                    t.setText("")
                    recyclerView.visibility = View.VISIBLE
                }
            }catch (e: Exception){
                Toast.makeText(this@ChoixListActivity, "${e.message}",
    Toast.LENGTH_SHORT).show()
            }
        }

    }
```

Lorsqu'on click sur une liste, on passe à l'activité 'Show List Activity'.

5. ShowListActivity

Pour passer de l'activité 'ChoixListActivity' à cette activité on a implementé le code 5

Code 5: toShowListActivity

```
adapter.setOnItemClickListener(object : AdapterList.OnItemClickListener {
        override fun onItemClick(position: Int) {
                val listName = lists[position].label
                val id_list = lists[position].id

                Toast.makeText(applicationContext, listName, Toast.LENGTH_SHORT).show()
                Toast.makeText(applicationContext, id_list, Toast.LENGTH_SHORT).show()
                change.putExtra("list", listName)
                change.putExtra("id_list", id_list)
                change.putExtra("hash", hash)
                startActivity(change)

        }


    })
```

Et pour le reste l'activité fait les mêmes fonctions que 'ChoixListActivity' donc c'est facile a implémenter en ajoutant les classes, les fichiers et le code déjà utilisé. Dans cette partie, il faut récupérer de la dernière activité l'identifiant de la liste, son nom (pour le titre) et le hash.

Ainsi, on reçoit: 'D/OkHttp: –>GET http://tomnab.fr/todo-api/lists/2/items hash: 25760fd2eec0ce18a85d998b –>END GET'

Code 6: Adapter: création d'un nouvel item

```
b.setOnClickListener {
            // add new item
            activityScope.launch {
                try{
                    if(id_list_int!=null && hash!=null){
                        recyclerView.visibility = View.GONE
```

```
                        val labelItem = t.text.toString()
                        Toast.makeText(this@ShowListActivity,labelItem,
    Toast.LENGTH_SHORT).show()
                        // add the new list
                        val newItem = DataProvider.createItem(id_list_int, labelItem,
    hash)
                        val listReady : List<Item> = listOf(newItem)
                        adapter.addData(listReady)
                        //val lists = DataProvider.getListsFromApi(hash)
                        //adapter.addData(lists)
                        t.setText("")
                        recyclerView.visibility = View.VISIBLE
                    }
                }catch(e: Exception){
                    Toast.makeText(this@ShowListActivity, "${e.message} ",
    Toast.LENGTH_SHORT).show()
                }
            }

        }
```

On a aussi implémenté les requêtes pour cocher et décocher un item grâce à la requête PUT:

Code 7: Cocher et décocher

```
// cocher un item: 1
    // decocher un item:0
    @PUT("lists/{id_list}/items/{id_item}")
    suspend fun cocherDecocherItem(@Path("id_list") id_list: Int,
                                   @Path("id_item") id_item: Int,
                                   @Query("check") check: Int,
                                   @Header("hash")hash: String)
```

6. Analyse

Pour l'amélioration de l'activité 'Settings', il faudrait lire plus la documentation pour implementer une nouvelle classe et ne pas la classe 'PreferenceCategory' qui est dépréciée.

Il a fallu créer un nouveau fichier xml pour la connexion à internet parce qu'elle n'était pas sécurisé (cf. 'networksecurityconfig.xml'.)

On a amélioré l'interface et on a beaucoup utilisé Toast pour voir ce qu'il se passait lors d'un click sur un bouton ou sur une autre chose.

L'utilisation de la librairie Retrofit nous a aidé énormément pour toutes les requêtes. On a apprécie qu'elle était très facile à utiliser.

7. Conclusion

On a vu des différents types d'activités, on a appris à passer d'une activité à l'autre et de faire un type particulier de fichier xml pour chaque activité. On a réussi à bien implémenter l'interface de l'API avec ses fonctions et les requêtes GET, POST et PUT.

On a beaucoup apprécie la documentation d'android studio, qui a été très utile lorsqu'on cherchait comment implémenter une classe ou une autre chose.

Par rapport au Kotlin, comme c'est un nouveau langage appris il y a quelques semaines, il fallait le connaitre un peu plus et aussi le pratiquer, surtout pour la connexion à internet.

On a réussi à bien implémenter les quatre activités mais pas complètement. On n'a pas pu récupérer l'identifiant de l'utilisateur.

Références

[1] RecycleView `https://developer.android.com/guide/topics/ui/layout/recyclerview`