

Ministère de l'Enseignement Supérieur et de la Recherche et de l'Innovation
Université de Lille
École Centrale de Lille



Programmation mobile et réalité augmentée

Sujet de rapport : **Développement de l'application android ToDoList version 2**

Rédigé Par : **Imen Kerkeni**
Wissem Chabchoub



Année universitaire : **2018 - 2019**

Table des matières

Contexte	1
1 Analyse	2
1.1 Activités	2
1.1.1 Main Activity	3
1.1.2 ChoixListActivity	3
1.1.3 ShowListActivity	4
1.2 RecyclerView	5
1.2.1 item_list.xml et item_item.xml	5
1.2.2 ItemAdapter.java :	5
1.3 La librairie Volley	6
1.4 Diagramme UML	7
Conclusion	8

Contexte

N'oubliez plus jamais une tâche importante, Organisez votre vie puis profitez-en!!” La vie peut parfois sembler écrasante, mais ça n’a pas à être le cas. Avec “ToDo”, dans sa deuxième version on vous garantie la tranquillité d’esprit et la sauvegarde de vos données dans notre API . Notre application android “ToDo” qui permet aux utilisateurs de lister,ajouter, sauvegarder et mettre à jour leurs "ListToDo" en toute simplicité.

La description détaillée de fonctionnalité de l’application en sa deuxième version se trouve dans les paragraphes suivantes.

Mots-clés : List, Activités, API,Requêtes, RecyclerView, Volley

1.1 Activités

L'application se base sur trois activités principales :

- ◇ MainActivity : dans laquelle l'utilisateur saisit son login et on ajouté dans cette version un champ pour mot de passe.
- ◇ ChoixListActivity : dans laquelle on visualise une liste des ListToDo de l'utilisateur en question.
- ◇ ShowListActivity : dans laquelle on visualise les itemToDos de la liste sélectionnée.

Dans l'activité Settings on a ajouté l'url de base de l'API `http://tomnab.fr/todo-api/` et qu'on peut l'éditer.

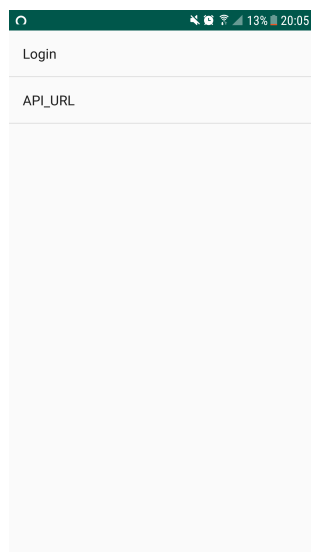


FIGURE 1.1 — Settings Activity

1.1.1 Main Activity

Une vérification de l'accès au réseau se lance lors de la démarrage de l'application et qui permet l'activation du bouton Ok si le terminal utilisé peut accéder au réseau. L'activité principale permet à l'utilisateur de saisir son login dans un champ Auto-Complete ainsi que son mot de passe. Cette connexion à l'api permet de récupérer le token d'identification et de le stocker dans les préférences de l'activité. Après l'étape de l'identification l'activité ChoixListActivity se lance.

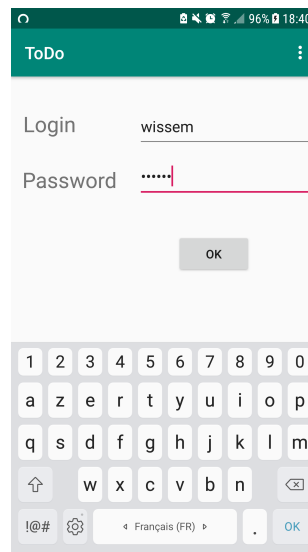


FIGURE 1.2 — Main Activity

1.1.2 ChoixListActivity

Dans cette activité, on affiche les ListeToDos de l'utilisateur connecté en utilisant un RecyclerView. Le bouton add permet d'ajouter une nouvelle liste dont le nom est saisi dans le EditText. Les données dans l'API sont alors mises à jour. Le glissement d'un item permet de le supprimer. Un snack bar s'affiche dans ce cas et propose de remettre la liste supprimée. Le clique sur une liste permet de passer l'activité ShowListActivity en passant le id de la liste sélectionnée.

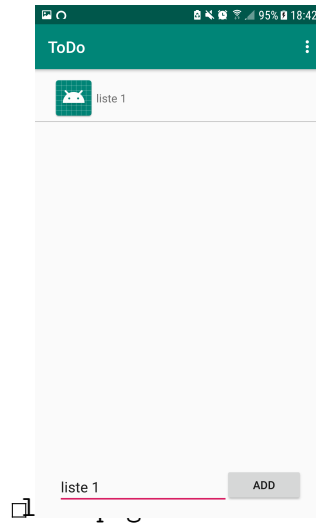


FIGURE 1.3 — ChoixListActivity

1.1.3 ShowListActivity

Dans cette activité, on affiche les `ItemsToDos` de la `ListeToDo` qu'on les a récupéré de l'API par des requêtes GET. Les actions de glissement et le bouton Add permettent respectivement de la suppression et l'ajout de la liste concernée. Les `ItemsToDos` sont équipés avec un `CheckBox`. ainsi on peut cocher/decocher chaque item.

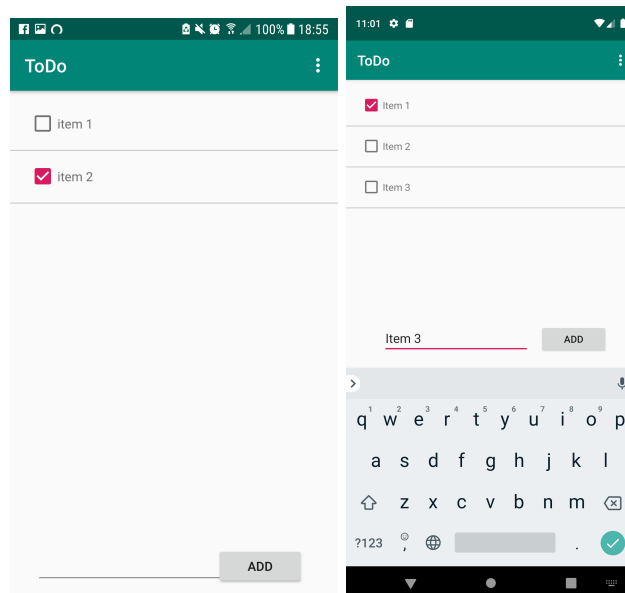


FIGURE 1.4 — ShowListActivity

1.2 RecyclerView

Le recyclerView permet d'afficher des items en optimisant l'utilisation des ressources disponibles. Il se base sur un ItemAdapter (classe Java) et un design des items enregistré sous format xml.

1.2.1 item_list.xml et item_item.xml

Ce fichier xml implémente le design des items du recyclerView.

- i) item_list : Il concerne des items à afficher dans la liste des ListeToDos.
- ii) item_item : Il concerne des items à afficher dans la liste des ItemTodos.

1.2.2 ItemAdapter.java :

Dans ce projet, on a créé deux ItemAdapters en fonction des fichiers item.xml. Leur contenu est le même à l'exception du fichier xml qu'ils utilisent.

L'item adapter contient cinq attributs :

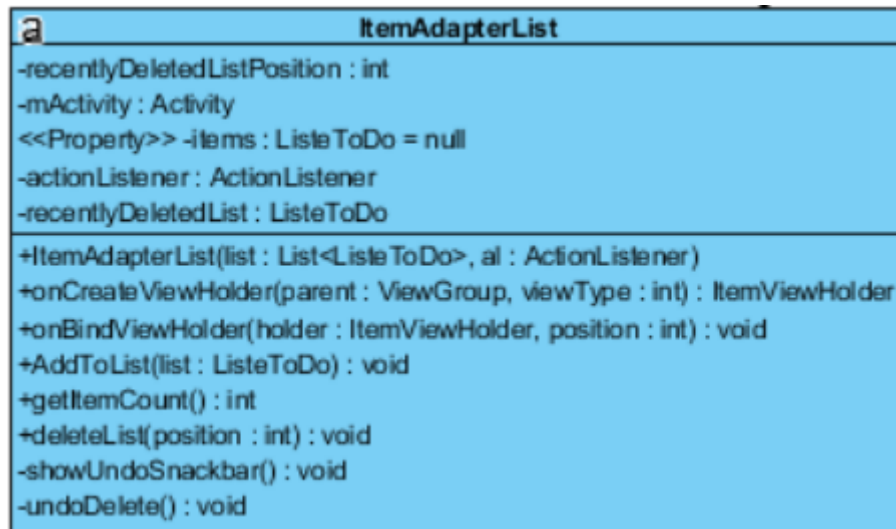
1. La liste des items à afficher
2. Une interface ActionListener qui contient des méthodes à implémenter par l'activité qui contient le RecyclerView.
3. Dernier item supprimé
4. Position du dernier item supprimé
5. L'activité parente.

Le constructeur de l'adaptateur permet d'initialiser la liste des items et le ActionListener.

Dans la méthode bind, on attribue l'id de l'item au TextView ou au CheckBox pour identifier l'item en cas de clique ou de modification.

La méthode deleteListe permet de supprimer l'élément glissé et enregistrer cet élément et sa position. Si l'utilisateur choisit d'annuler la suppression, la méthode undoDelete, remet l'élément dans la liste des items dans sa position initiale.

Les méthodes deleteListe et undoDelete et le clique sur un item appellent les fonctions de l'interface ActionListener qui enregistrent les modifications dans les préférences



```
class ItemAdapterList {
    -recentlyDeletedListPosition : int
    -mActivity : Activity
    <<Property>> -items : ListeToDo = null
    -actionListener : ActionListener
    -recentlyDeletedList : ListeToDo

    +ItemAdapterList(list : List<ListeToDo>, al : ActionListener)
    +onCreateViewHolder(parent : ViewGroup, viewType : int) : ItemViewHolder
    +onBindViewHolder(holder : ItemViewHolder, position : int) : void
    +AddToList(list : ListeToDo) : void
    +getItemCount() : int
    +deleteList(position : int) : void
    -showUndoSnackbar() : void
    -undoDelete() : void
}
```

1.3 La librairie Volley

Afin de réaliser les requêtes HTTP, on choisi d'utiliser la librairie Volley. C'est une bibliothèque HTTP qui rend la mise en réseau des applications Android plus facile et surtout, plus rapide. Elle offre plusieurs avantages comme la planification automatique des requêtes réseau, connexions réseau multiples et simultanées et la mise en cache transparente de la réponse du disque et de la mémoire avec cohérence standard du cache HTTP.

Nous avons utilisé Volley en créant un RequestQueue et en lui passant des objets Request. La RequestQueue gère les fils de discussion des travailleurs pour exécuter les opérations réseau, lire et écrire dans le cache, et analyser les réponses. Les requêtes font l'analyse des réponses brutes et Volley s'occupe de l'envoi de la réponse analysée vers le thread principal pour livraison.

Un RequestQueue a besoin de deux choses pour faire son travail : un réseau pour effectuer le transport des requêtes, et un cache pour gérer la mise en cache. Il y a des implémentations standard de ces derniers disponibles dans la boîte à outils Volley : DiskBasedCache fournit un cache à un fichier par réponse avec un index en mémoire, et BasicNetwork fournit un transport réseau basé sur votre client HTTP préféré.

BasicNetwork est l'implémentation réseau par défaut de Volley. Un BasicNetwork doit être initialisé avec le client HTTP que votre application utilise pour se connecter au réseau. Typiquement, il s'agit d'une connexion HttpClient. La figure ci dessous montre l'implémentation de RequestQueue lors du click sur le bouton OK.


```

// Add the request to the queue
RequestQueue queue = Volley.newRequestQueue( context: this);
String url1= setAct.url.getText().toString();
String url = settings.getString( key: "api_url", defValue: "http://tomnab.fr/todo-api").toString() + "/authenticate?user=" + edtLogin.getText().toString();
Log.i( tag: "Web", url);
StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
    (response) => {
        Log.i( tag: "Web", response);
        Gson gson = new Gson();
        Type type = new TypeToken<HashMap<String, String>>() {
        }.getType();
        HashMap<String, String> responseJson = gson.fromJson(response, type);
        SharedPreferences.Editor editor = settings.edit();
        editor.putString("token", responseJson.get("hash"));
        Log.i( tag: "Web", responseJson.get("hash"));

        //save login in preferences
        //SharedPreferences settings = PreferenceManager.getDefaultSharedPreferences(this);
        editor.putString("login", edtLogin.getText().toString());
        editor.commit();

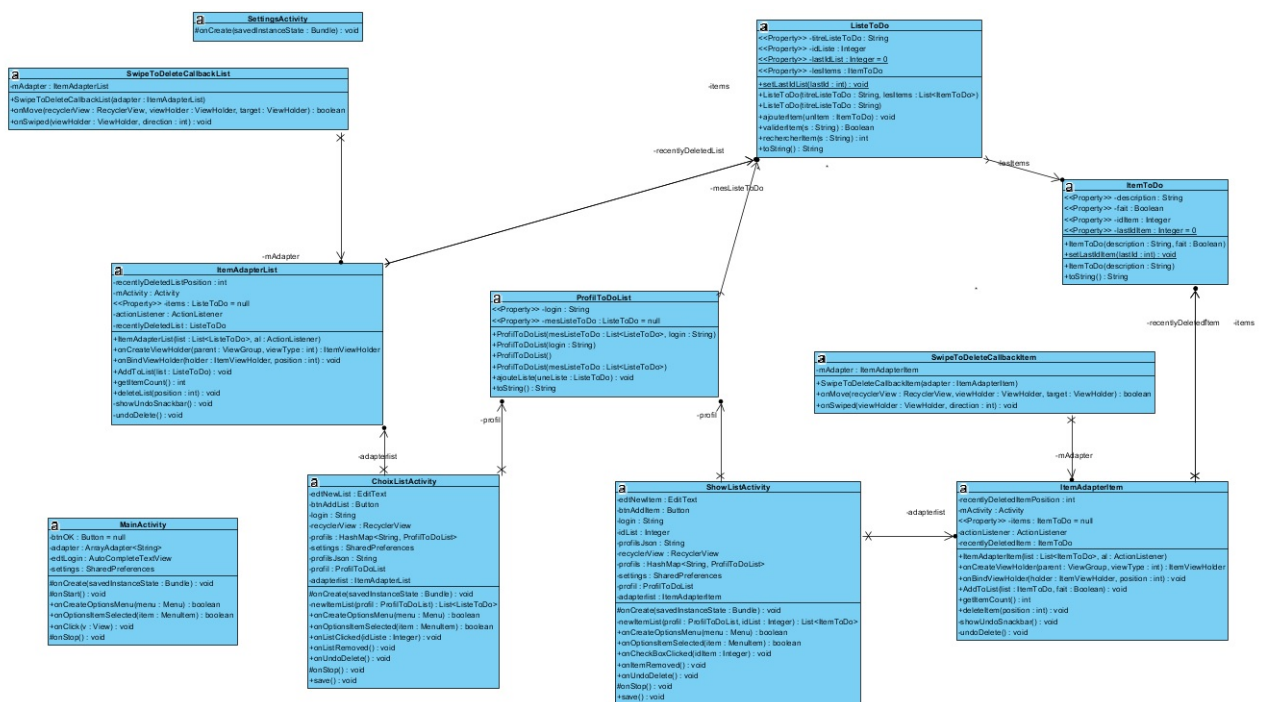
        //Create an intent to change activity
        Intent toSecondAct = new Intent( packageContext: MainActivity.this, ChoixListActivity.class);
        //Bundle data = new Bundle();
        //data.putString("login", edtLogin.getText().toString());
        //toSecondAct.putExtras(data);
        startActivity(toSecondAct);

    }, (error) => {
        Log.i( tag: "Web", msg: "error");
        Toast.makeText(context, text: "Verify your login or your password", Toast.LENGTH_LONG).show();
    });
// Add the request to the RequestQueue.
queue.add(stringRequest);
} else
    Toast.makeText( context: this, text: "Verify your internet connection", Toast.LENGTH_LONG).show();
break;
}

```

1.4 Diagramme UML

Le diagramme UML complet est dans ce lien : [Lien](#)



Conclusion

On a pu dans cette version d'établir la connexion entre l'application et l'API TODO. Durant cette étape d'amélioration on a decouvert une nouvelle librairie qui est la librairie Volley. C'était une bonne occasion de manipuler des nouvelles fonctionnalités et méthodes et de les comparer avec ce qu'on a utilisé dans les séances du TP comme retrofit et asynctasks. Durant l'étape de développement de notre application on a rencontré des différentes difficultés (bugs, blocage, connaissance..) mais on a pu dépasser ces situations grâce à des connaissances et des méthodes qu'on a pu voir durant les séminaires et des recherches et des lectures bibliographiques sur internet.

Bibliographie

- [1] <https://blog.webwag.com/2017/02/14/introduction-a-volley-gson/>
- [2] <https://developer.android.com/reference/android/widget/AutoCompleteTextView.html?fbclid=IwAR1Usva0KqKKjPrl7qeZ7SFUeheiA8>
- [3] <https://openclassrooms.com/fr/courses/4568576-recuperez-et-affichez-des-donnees-distances/4893781-implementez-votre-premiere-recyclerview>
- [4] <https://developer.android.com/reference/android/preference/PreferenceActivity>
- [5] <https://docs.microsoft.com/fr-fr/dotnet/framework/wcf/feature-details/how-to-serialize-and-deserialize-json-data>
- [6] <https://www.lemagit.fr/conseil/Comment-connecter-une-API-JSON-a-un-projet-Android>