Romain MABBOUX
Pierrick GERVASI

Séquence 2 :

Utilisation d'une API pour notre application To Do List

Romain MABBOUX
Pierrick GERVASI

Objectifs

L'objectif de ce TEA était de modifier notre application de type « To Do List » de manière à utiliser une API pour importer des données. Le cahier des charges était le suivant :

- Permettre la sauvegarde et l'édition de l'url de base de l'API
- Rendre impossible l'appui du bouton OK si le réseau n'est pas disponible
- Récupérer les hash de chaque utilisateur et les stocker dans les préférences
- Permettre d'accéder aux To do List de l'utilisateur
- Permettre d'accéder aux Tâches de chaque To Do List, et permettre l'ajout de tâches.

Nous avons choisi d'ajouter les fonctionnalités :

- Ajout d'une nouvelle To Do List à l'utilisateur

Equipe et organisation

L'équipe est composée de deux élèves :

- MABBOUX Romain
- GERVASI Pierrick

Dès le début, nous avons décidé de nous organiser afin de trouver une bonne répartition du travail pour être efficaces. Nous avons décidé de nous organiser de la manière suivante :

- Pierrick devait s'occuper de mettre à jour tous les éléments de FrontEnd, de la gestion du réseau (désactivation/activation du bouton OK), et de rendre tous les objets de l'interface « Parcelable ».
- Romain devait s'occuper des requêtes POST et GET avec l'API, en utilisant des requêtes asynchrones.

GitHub 

Afin de continuer à travailler efficacement à deux, nous avons décidé de continuer à utiliser GitHub pour gérer le code de l'application (cf. Rapport de la séquence 1 pour la formation que nous avons suivie).

Présentation de l'application

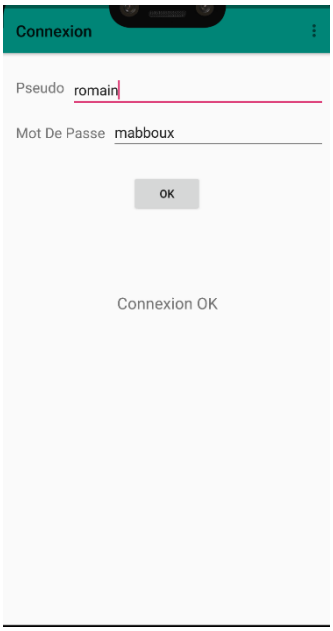


Figure 1

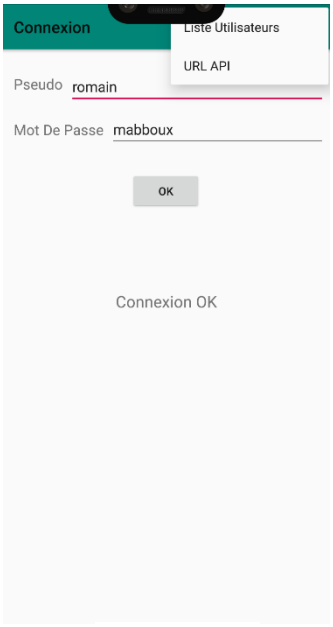


Figure 2

Figure 3

Figure 4

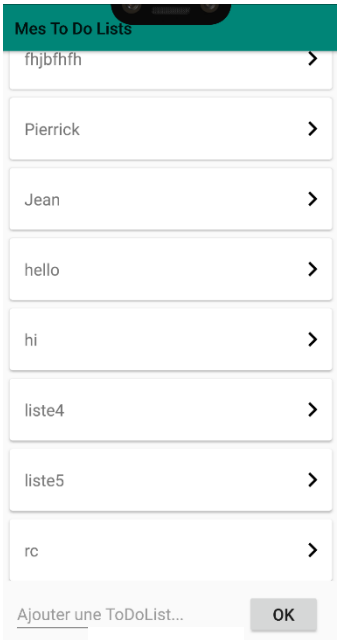


Figure 5

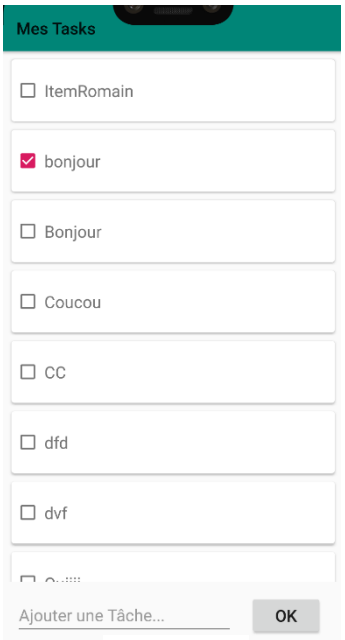


Figure 6

Figure 1 : MainActivity qui permet à un utilisateur d'entrer son pseudo et son mot de passe. S'ils sont erronés, il en est informé.

Figure 2 : Depuis cette MainActivity, on peut également accéder à un menu contenant deux items : un qui envoi vers la liste des utilisateurs, l'autre vers l'URL de l'API utilisée.

Figure 3 : Dans la liste de utilisateurs, on retrouve la liste des pseudos des utilisateurs ayant déjà utilisé l'application, ainsi que leur hash. Bien qu'il y ait un icon « poubelle », l'application ne permet pas de supprimer un utilisateur.

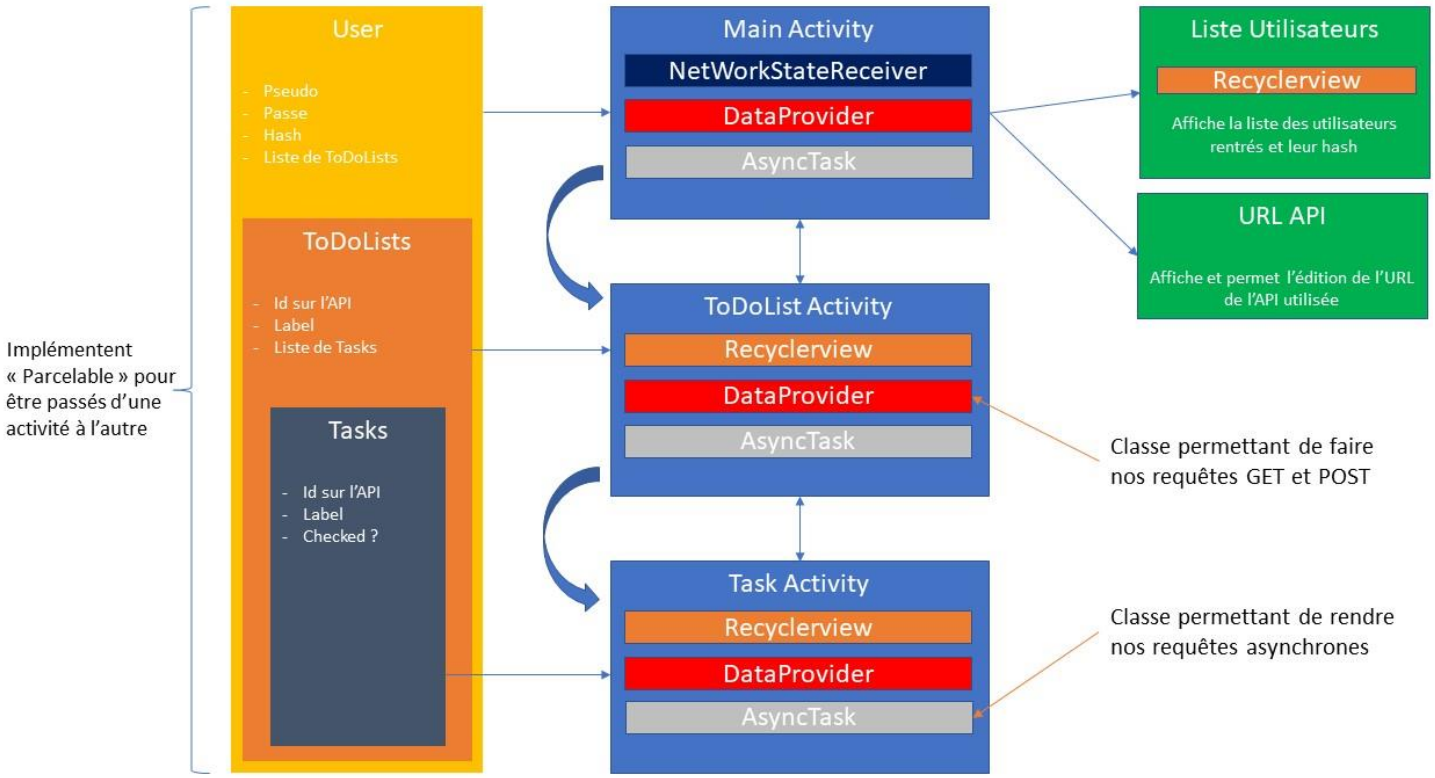
Figure 4 : Dans cette activité on peut sois même changer l'URL de l'API que l'on souhaite utiliser. Par mesure de sécurité, il est possible de récupérer l'URL de l'API de base, c'est-à-dire celle que l'on utilise pour cet exercice.

Figure 5 : Cette activité représente la liste des ToDoLists associées à l'utilisateur connecté, et permet l'ajout de nouvelles ToDoLists mais pas leur suppression.

Figure 6 : Cette activité représente la liste des Items associés à la ToDoList sélectionnée, et permet l'ajout de nouveaux items mais pas leur suppression.

Note : pour des raisons inconnues et alors que la méthode fonctionne parfaitement pour les ToDoLists, l'ajout des items fonctionne un fois sur 2 ou 3.

Fonctionnement de l'application



L'application fonctionne de la manière suivante :

Si les informations rentrées de l'utilisateur sont bonnes et que la connexion est disponible, on créé un objet « User » qui possède un hash, un pseudo, et une liste de ToDoList.

On passe l'objet « User » qui est un Parcelable (cf ci-dessous) dans l'activité « ToDoListActivity », puis on effectue une requête asynchrone vers l'API pour récupérer les ToDoLists de ce User, créer les objets « ToDoLists » associés (qui possède un label, un idAPI, et une liste de Tâches), et les afficher dans le recyclerview.

Si on clique sur une ToDoList, on passe le hash de l'utilisateur et la ToDoList dans l'activité « TaskActivity » dans laquelle on effectue une requête asynchrone vers l'API pour récupérer les Tasks de cette ToDoList, créer les objets « Tasks » associés, et les afficher.

A chaque requête GET ou POST asynchrone, on utilise comme URL d'API l'URL d'API qui est présent dans l'activité « URL API » (URL que l'utilisateur peut modifier en cas de besoin).

D'autre part, si l'utilisateur créé une ToDoList ou une Tâche, on effectue une requête POST vers le serveur pour créer l'objet dans l'API, puis on créé l'objet associé et on l'ajoute au Recyclerview.

Utilisation de « Parcelable »

Dans la séquence 1, afin d'afficher les To Do Lists d'un utilisateur, et les tâches associées à ces To Do Lists, nous avions utilisé un attribut public static pour l'utilisateur, ce qui est une très mauvaise pratique.

Pour cette séquence, nous avons décidé d'utiliser uniquement des attributs privés. La difficulté était alors de passer des attributs « User », ou « To Do Lists », d'une activité à une autre. Après quelques recherches, nous avons pu trouver l'existence des entités « Parcelable », qui représentent la même chose que « Serializable », mais qui est beaucoup plus efficace pour faire passer des objets (plus complexes que des String ou des int). Nous avons alors implémenté cette caractéristique dans chacune des classes du modèle (User, ToDoList, Task).

Utilisation d'un BroadcastReceiver

Afin de pouvoir écouter en continu la connexion internet, de manière à activer ou non le bouton « Ok » permettant d'accéder aux To Do Lists, nous avons implémenté la classe « NetWorkStateReceiver » qui écoute en continu la connexion. Nous avons pu trouver du code notamment sur Stack Overflow (cf bibliographie) qui nous a aidé à réaliser les actions désirées.

Conclusion

En conclusion, nous pouvons dire que cette deuxième séquence nous a véritablement permis de revenir sur notre code et de l'améliorer. De nombreux points de code étaient fonctionnels mais le code était très peu maintenable. Nous utilisions des attributs « public static » pour l'utilisateur ce qui est une mauvaise pratique, mais nous nous sommes rattrapés grâce à l'utilisation d'objets « Parcelable ». Nous avons revu entièrement le code et l'avons amélioré et commenté.

Enfin, nous nous sommes rendu compte à quel point l'utilisation d'internet en tant que développeurs est indispensable, et que si nous rencontrons une difficulté il y a de fortes chances que quelqu'un l'ait déjà rencontré. Stack OverFlow est pour cette raison une plateforme très intéressante.

Perspectives

- Ajout de la fonctionnalité « Delete » (utiliser des requêtes « DEL »
- Connecter automatiquement l'utilisateur sans passer par l'activité de connexion en stockant pseudo/passe dans les préférences. Ajouter un menu « déconnexion » dans toutes les activités.
- Ajouter une possibilité de créer un nouvel utilisateur.

Bibliographie

- Codexpedia pour l'implémentation de Parcelable : https://www.codexpedia.com/android/using-android-parcelable-objects/
- StackOverflow pour le BroadcastReceiver : https://stackoverflow.com/questions/6169059/android-event-for-internet-connectivity-state-change