Compte rendu application de todo list v2

Luca Jakomulski et Tristan Buscemi

Introduction
L'application est similaire à la première version, à la différence notable que les listes ne sont plus enregistrées en local mais récupérées d'un serveur grâce à une API.

Analyse
Afin de réussir cette transformation, les classes qui définissent les données ont été modifiées afin que les Items et TodoLists aient un identifiant. De plus, un nouveau package a été créé : le package API. Celui-ci contient des classes Response qui correspondent au format de donnée reçu lorsque l'on fait une requête à l'API. Il contient également une interface dans laquelle on va définir les différents requêtes que l'on peut effectuer, ainsi que les paramètres de celles-ci. Et enfin il contient un classe depuis laquelle on va créé un objet Retrofit. Enfin afin d'utiliser, une fois qu'on reçoit la réponse de l'API, les classes Item et TodoList qui étaient déjà présentes dans la première version de l'application, on prendra soin d'utiliser @SerializedName devant les attributs de ces classes. Cependant ces classes perdent beaucoup de leur méthodes étant donné que les données ne sont plus manipulées en local.

Dans le Manifest, on va ajouter les permissions pour avoir accès à l'état du réseau et à internet et également le paramètre `android:usesCleartextTraffic="true"` qui va nous permettre d'utiliser l'API sans avoir de restriction de sécurité.

Dans GenericActivity, on ajoute un attribut texte qui sera l'URL de l'API elle même enregistrée dans les paramètres et l'autre également un texte mais qui correspond au hash utilisé pour construire les requêtes pour l'API. On ajoute également la méthode connectedToNetwork() qui vérifie si le terminal est connecté à internet (la version est différente de celle proposée par Thomas car elle faisait appel à des méthodes dépréciées). Cette méthode n'est utilisée que dans MainActivity donc on pourrait se demander ce que la méthode fait dans GenericActivity : elle a été placée ici dans le but d'être utilisée par les autres activités afin de vérifier régulièrement l'état du réseau, cependant cela n'est pas été fait et reste à l'état d'amélioration possible.

MainActivity se compose maintenant d'un champ de mot de passe. Le bouton de l'activité est désactivé quand le réseau n'est pas disponible. Inconvénient majeur à l'état actuel : le bouton ne se réactive pas quand le réseau est disponible, il faut redémarrer l'activité car on ne surveille pas en permanence l'état de la connexion au réseau. Lorsqu'on dispose de réseau et qu'on se connecte, on va envoyer un requête pour s'authentifier. On récupère alors un hash que l'on stocke dans les préférences puis on accède aux todo listes.

Note : les classes ListsActivity et ItemsActivity étant structurées de la même manière nous nous contenterons d'expliquer simplement la première des deux classes, le lecteur pourra par lui-même expliquer les différences assez simplement en regardant le code source.

Dans ListActivity on a revu la façon dont on récupère, enregistre, supprime les listes. En effet, auparavant tout s'effectuait en local mais maintenant on fait uniquement appel à l'API donc ce sont ces parties du code qui ont changé. Pour chacune de ces opérations, on passe par l'interface pour envoyer notre requête à l'API (le code est assez répétitif). Finalement la logique du code depuis sa première version a peu évolué.

Enfin on a enlevé la plupart du code qui ne servait à rien dans l'application (la classe UserProfile, les fonctions JSON,...) afin d'avoir un code plus clair.

Conclusion
On a appris à utiliser Retrofit et une API.

Perspective
On pourrait vérifier régulièrement la présence d'une connexion au réseau.
On pourrait donner la possibilité à l'utilisateur de créer un compte.
On pourrait également proposé une connexion automatique.

Sitographie
Documentation de l'API
Documentation de Retrofit