Nathan BOISARD
Maxime LERAITRE

Réalisation d'une application de gestion de ToDoList
- ToDouDou -

Introduction

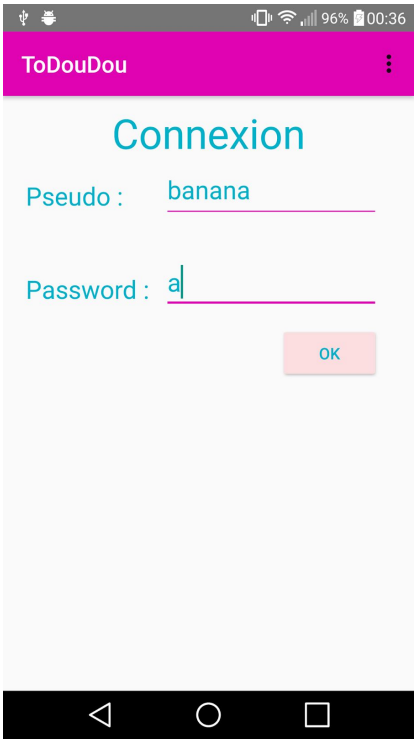
L'objectif de la séquence 2 est de reprendre le travail de la séquence 1 en changeant le moyen de stocker les données. Au lieu de se servir des préférences de l'application, on utilisera une API Rest pour stocker les ToDoList.
Nous avons implémenté le travail demandé ainsi que les 2 premiers points facultatifs, tout en ajoutant la possibilité de supprimer une liste/un item.

Analyse

Fonctionnalités :

- ✓ Connexion:

  Pour que l'utilisateur se connecte, il faut qu'il ait accès à internet. Dans ce cas, le bouton "OK" est visible. En donnant son pseudo et son mot de passe, l'utilisateur peut se connecter. Cette action envoie une requête POST à l'API, et si le pseudo et le mot de passe sont correct, l'application reçoit en retour un "hash" qu'elle conserve dans les préférences de l'application;
  De plus, si l'utilisateur s'est déjà connecté antérieurement (et qu'internet est disponible), l'application lance automatiquement la connexion.
  Exemple : ici nos identifiants pour nos tests étaient "banana" pour le pseudo et "a" pour le password.

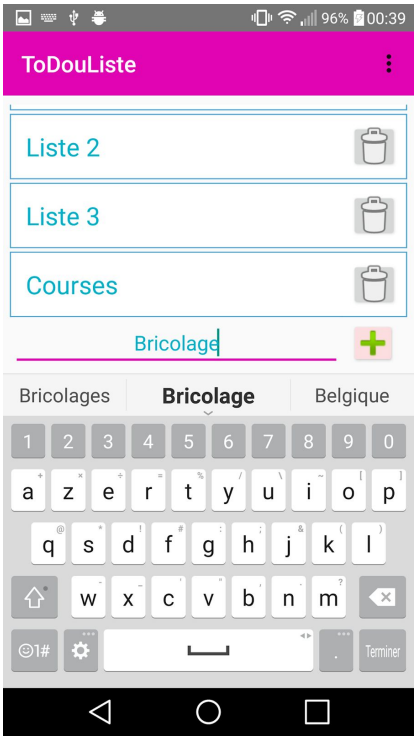


Nathan BOISARD
Maxime LERAITRE

✓ Listes:

Après une connexion réussie, l'utilisateur a accès à l'ensemble de ses listes que l'application récupère grâce à une requête GET à l'API. L'utilisateur peut alors :

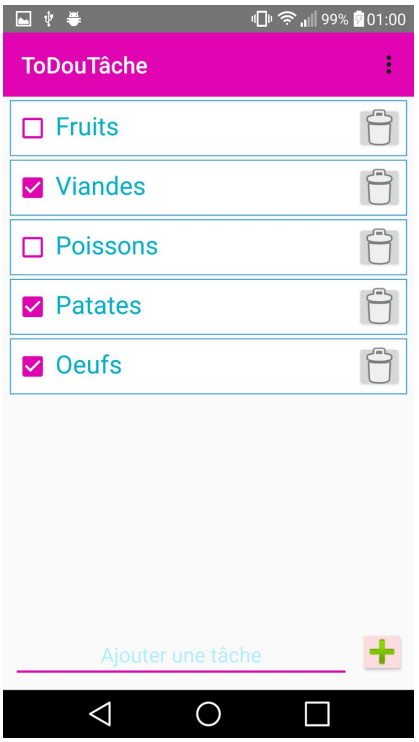
1. Ajouter une liste en saisissant son nom puis en appuyant sur le bouton "+". L'application envoie alors une requête POST pour créer la liste dans l'API.
2. Supprimer une liste en appuyant simplement sur l'icône poubelle à droite. L'application envoie alors une requête DELETE pour supprimer cette liste de l'API.
3. Accéder aux tâches d'une liste désirée en cliquant sur celle-ci. L'application change d'activité et fais passer l'id de la liste choisie à la nouvelle activité.



✓ Tâches:

Après avoir choisie une liste, cette activité affiche l'ensemble des tâches de cette liste grâce à une requête GET envoyée à l'API. L'utilisateur peut alors :

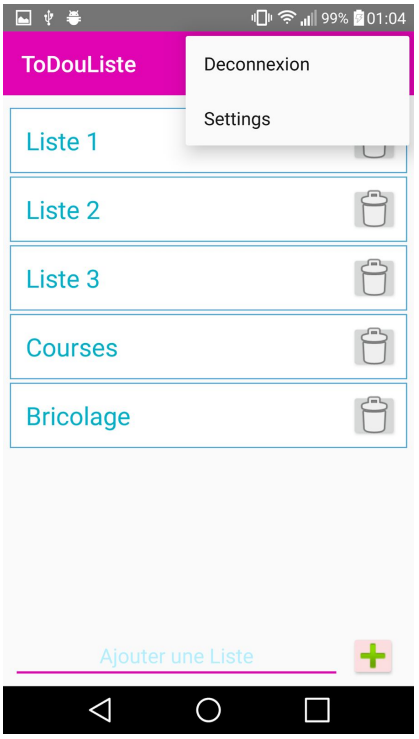
1. Ajouter une tâche en saisissant son nom puis en appuyant sur le bouton "+". L'application envoie alors une requête POST pour créer la tâche dans l'API.
2. Supprimer une tâche en appuyant simplement sur l'icône poubelle à droite. L'application envoie alors une requête DELETE pour supprimer cette tâche de l'API.
3. Cocher/décocher une tâche en appuyant sur la checkbox de la tâche à sa gauche. L'application envoie une requête PUT pour modifier la tâche dans l'API.



Nathan BOISARD
Maxime LERAITRE

✓ Déconnexion:

A tout moment de sa session, l'utilisateur peut se déconnecter en appuyant sur "Deconnexion" dans le menu. Cela effacera le mot de passe enregistré dans les préférences de l'application pour désactiver la connexion automatique et renverra l'utilisateur à l'activité de connexion.



✓ Settings:

L'utilisateur peut se rendre dans "Settings" (disponible dans le menu) pour modifier l'URL de l'API. Cette URL est enregistrée dans les préférences de l'application pour être récupérée par la classe qui gère les requêtes. Elle sert alors d'url de base dans toutes les requêtes envoyées à l'API.

Nathan BOISARD
Maxime LERAITRE

Précisions :

Pour envoyer des requêtes à l'API, nous disposons d'une classe "Reseau" qui contient une méthode pour vérifier si internet est disponible et 4 méthodes pour effectuer des requêtes GET, POST, PUT et DELETE. Ces 4 méthodes utilisent la classe "HttpURLConnection" pour communiquer avec l'API.
A chaque fois que l'on en aura besoin, cette classe "Reseau" sera appelée dans un autre thread que le thread principal à l'aide d'une AsyncTask. Chaque activité définit autant de classes internes AsyncTask qu'elle en a besoin. Par exemple, l'activité gérant les listes de l'utilisateur dispose de 3 classes internes AsyncTask : une récupérant les listes avec une requête GET, une autre ajoutant une liste avec une requête POST, et une autre supprimant une liste avec une requête DELETE.

Perspectives

Actuellement, l'application comporte un grand nombre d'activités auxquelles on accède en appuyant sur les différents boutons et naviguer de l'une à l'autre est peu agréable. Une solution pourrait être l'utilisation d'un drawer et de fragments qui rendraient l'expérience utilisateur plus fluide.

On pourrait également ajouter des options de couleurs pour les listes afin que l'utilisateur puisse les catégoriser selon son envie, une fonctionnalité pour glisser les items dans la liste et les réordonner, un moyen d'annoter les listes et tâches, donner la possibilité à l'utilisateur d'ajouter une échéance à une liste et de l'inclure dans son calendrier, de partager ou mutualiser ses listes en ligne avec d'autres utilisateurs.

En ce qui concerne l'envoie de requête vers l'API Rest, l'application utilise la classe HttpURLConnexion. Cette méthode étant très verbeuse, cela nous a permis de bien comprendre les mécanismes à l'oeuvre pour utiliser l'API. Cependant, cela rend la programmation lourde et fastidieuse. L'utilisation d'une librairie adaptée telle que Retrofit permettrait de contourner ces problèmes.

En ce qui concerne sécurité, d'après le dicton NEVER TRUST USER INPUT, il serait souhaitable de prendre des mesures pour éviter des failles, notamment au moment où l'utilisateur rentre le nom d'une liste ou d'une tâche. De plus, le hash est transmis dans l'url donc une autre personne peut facilement s'en emparer et accéder aux données de l'utilisateur.

Nathan BOISARD
Maxime LERAITRE

Conclusion

La réalisation de cette application nous a permis d'appréhender le protocole HTTP ainsi que la notion d'API Rest.

Cela a été aussi l'occasion d'améliorer la première version de l'application en proposant la suppression individuelle de liste ou d'item (plutôt qu'une suppression de tous les items et listes).

Encore une fois nous avons souvent été confronté à de nombreux problèmes durant la programmation. Cependant, nous sommes plus efficaces pour les surmonter car nous utilisons maintenant les ressources à notre disposition (stackoverflow.com, developer.android.com, javadoc etc.) de manière plus efficace.

Bibliographie

Sitographie :

- Documentation de l'API : https://documenter.getpostman.com/view/375205/S1TYVGTa?version=latest
- Requête HTTP : https://www.supinfo.com/articles/single/2592-android-faire-requetes-http-simplement
- AsyncTask : https://stackoverflow.com/questions/9671546/asynctask-android-example

Ressource :

- Code utilisant HttpURLConnection sur Moodle