Électif PMR - Séquence 2

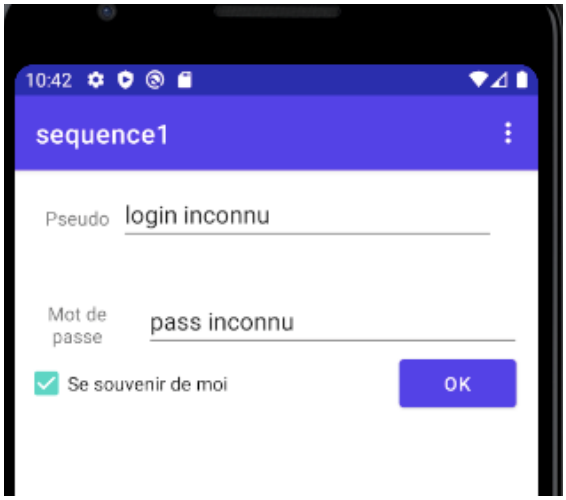
Matyas Macudzinski
Paul Croizat-Viallet

Github de notre travail :
https://github.com/Nanok09/sequence2.git

1 – Introduction : Prise en main de l'application

Ouverture de l'application :

(image de droite)



On peut rentrer son pseudo et son mot de passe,

cocher la checkbox «se souvenir de moi »,

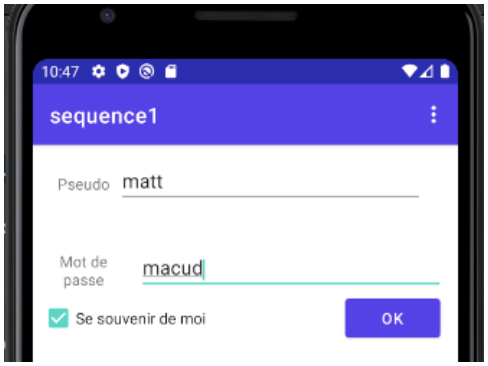
cliquer sur le menu actionbar (voir séquence 1),

et cliquer sur le bouton OK.

Le click du bouton OK ne permet pas ici d'accéder à la suite.

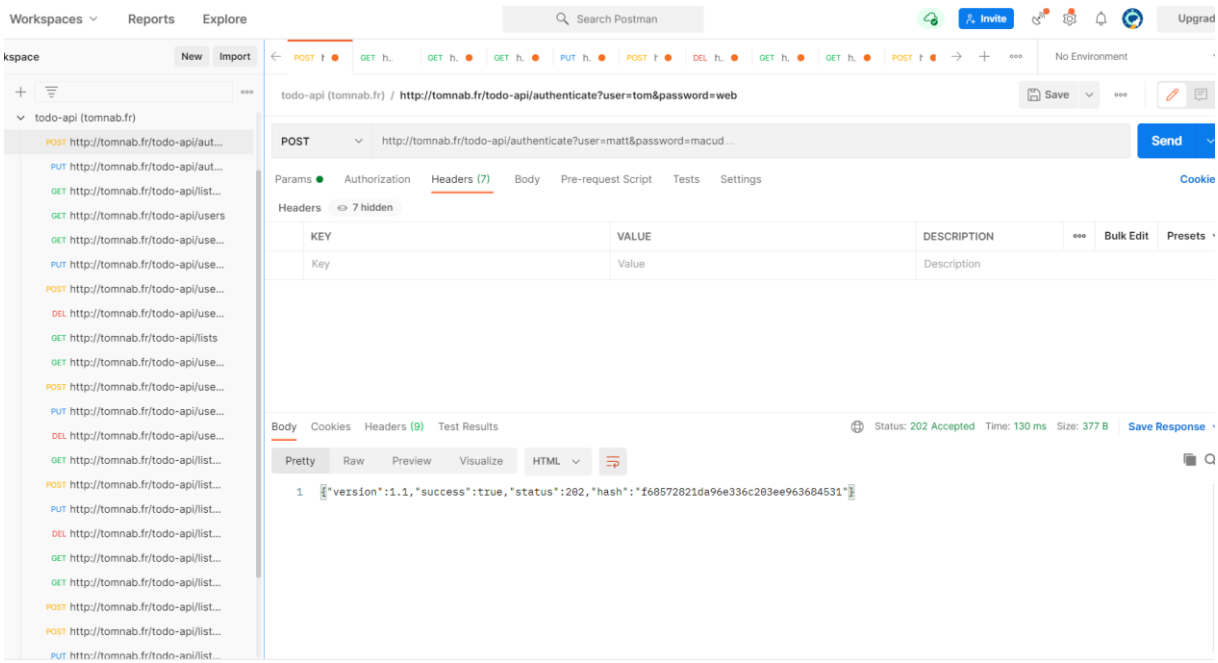
```
D/EGL_emulation: eglMakeCurrent: 0xeb9ab440: ver 3 0 (tinfo 0xe0598170)
I/OpenGLRenderer: Davey! duration=701ms; Flags=1, IntendedVsync=7767124342
I/EDPMR: Error: HTTP 401 Unauthorizedlogin inconnupass inconnu
D/EGL_emulation: eglMakeCurrent: 0xeb9ab440: ver 3 0 (tinfo 0xe0598170)
D/EGL_emulation: eglMakeCurrent: 0xeb9ab440: ver 3 0 (tinfo 0xe0598170)
```

Si on entre le login : **matt** et le mot de passe : **macud**, on peut accéder à la suite :



On accède aux listes todo vues en séquence 1 :

2 – Utilisation de l'API – Postman



Grâce à l'utilitaire Postman, nous pouvons préparer nos requêtes http avant le développement. On peut ainsi savoir quels paramètres fournir à l'API et quel résultat on va obtenir.

3 – Utilisation de l'API – Retrofit

Tout d'abord il nous faut définir un object Provider. Cet objet a pour attribut l'url de base de l'API : http://tomnab.fr/todo-api/ , un « builder » et un « service ».

Les méthodes de l'objet appellent une méthode de l'interface Service implémenté par l'objet Retrofit. C'est dans cette interface qu'on définit nos requêtes http en se référant à postman.

Il nous faut également définir des classes réponses pour la librairie Gson (utilisé par Retrofit). Ces (data) classes déterminent en quel objet la réponse reçue en format Json sera converti.

Ainsi, dès qu'on veut faire une requête HTTP, on crée une coroutine, on appelle dans le scope de cette coroutine une méthode dans la classe Provider défini auparavant et on récupe un objet réponse qui contient les informations reçues par l'API.

4 – Bilan des résultats atteints

- SettingsActivity : sauvegarder et permettre l'édition de l'url de base de l'API
- MainActivity :
    - Vérifier l'accès au réseau, et activer le bouton OK si le terminal utilisé peut accéder au réseau
    - Ajouter un champ mot de passe
    - Lors du clic sur le bouton OK, demander une identification auprès de l'API en lui envoyant pseudo/mot de passe
    - Récupérer le token d'identification sous forme d'un hash et le stocker dans les préférences de l'application
        - Vous renverrez ce token à chaque requête à l'API, dans un header "hash: <token>", ou dans une chaîne de requête
- ChoixListActivity : Récupérer les listes d'items associés à l'utilisateur connecté, et les afficher dans l'activité.
    - Lors du clic sur une des listes, passer à l'activité suivante en lui fournissant l'identifiant de la liste sélectionnée.
- ShowListActivity : Récupérer les items associés à la liste sélectionnée et les afficher dans l'activité.
    - Permettre de cocher/décocher chaque item,
    - Permettre l'ajout d'un nouvel item dans cette liste

Tous les objectifs exigés ont été atteints. (Pas les objectifs facultatifs).

5 – Bibliographie

En plus des cours dans le cadre de l'électif, nous avons utilisé d'autres ressources comme :

- Une vidéo tutoriel sur les RecyclerView
https://www.youtube.com/watch?v=Vyqz_-sJGFk

- Des articles sur l'utilisation de la librairie Gson en Kotlin :
https://bezkoder.com/kotlin-parse-json-gson/

https://medium.com/@hissain.khan/parsing-with-google-gson-library-in-android-kotlin-7920e26f5520

- La documentation officielle de android studio et du langage kotlin