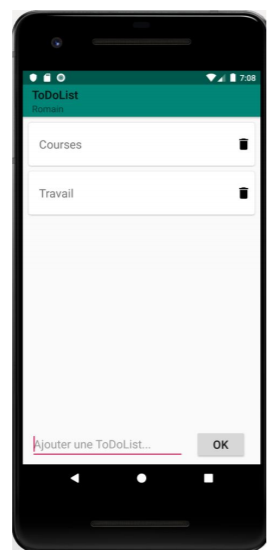


**Séquence 3 :**

**Stockage des données sous SQL pour notre application To Do List**



# Objectifs

L’objectif de ce TEA était de modifier notre application de type « To Do List » de manière à utiliser une base de données pour un mode hors connexion. Le cahier des charges était le suivant :



* En mode en ligne : stocker toutes les données récupérées dans la base SQL grâce à ROOM.
* En mode hors ligne : utiliser les données récupérées de la base SQL pour afficher des To do Lists et des Tasks
* Si en mode hors ligne on modifie l’état d’une checkbox d’une Task, l’information est stockée en SQL et dès que la connexion est établie, les données sur l’API sont mises à jour.

# Equipe et organisation

L’équipe est composée de deux élèves :

* MABBOUX Romain
* GERVASI Pierrick

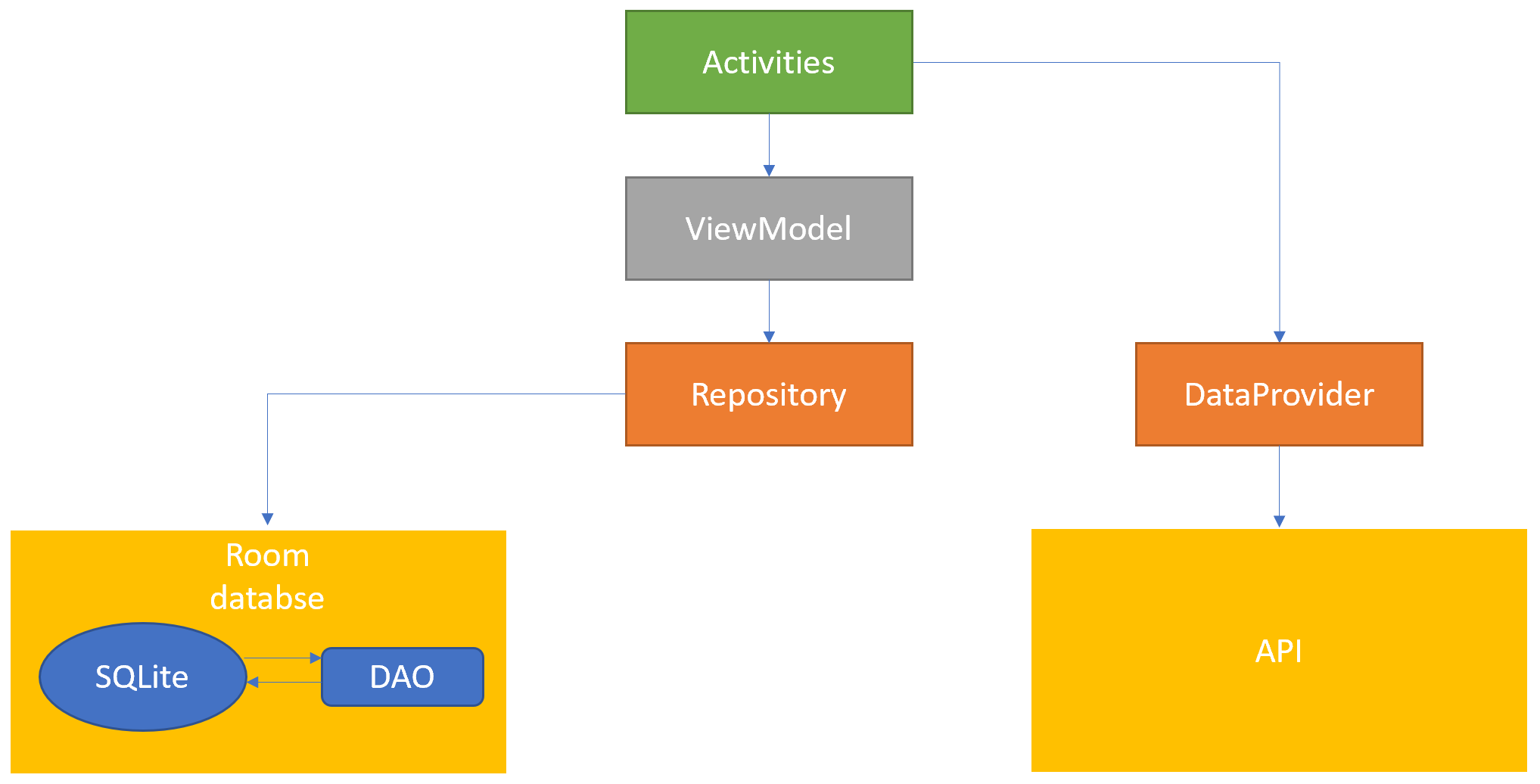


**GitHub**

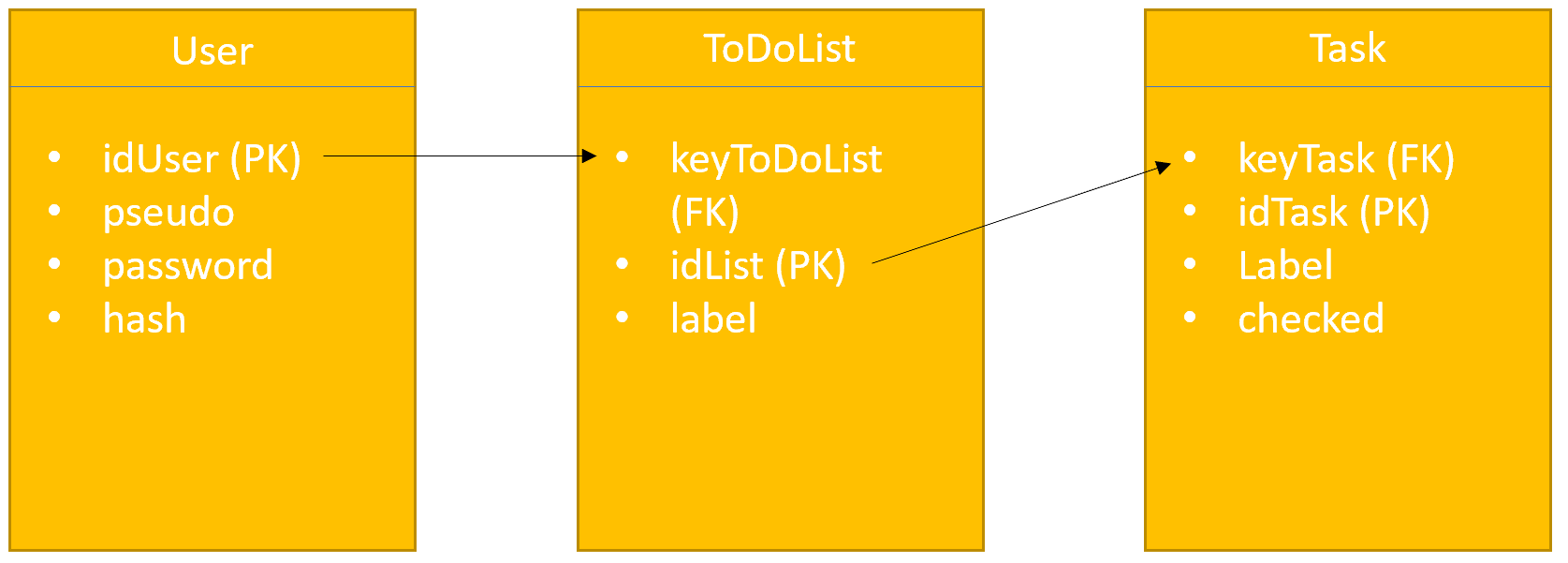
Afin de continuer à travailler efficacement à deux, nous avons décidé de continuer à utiliser GitHub pour gérer le code de l’application (cf. Rapport de la séquence 1 pour la formation que nous avons suivie).

## Première architecture

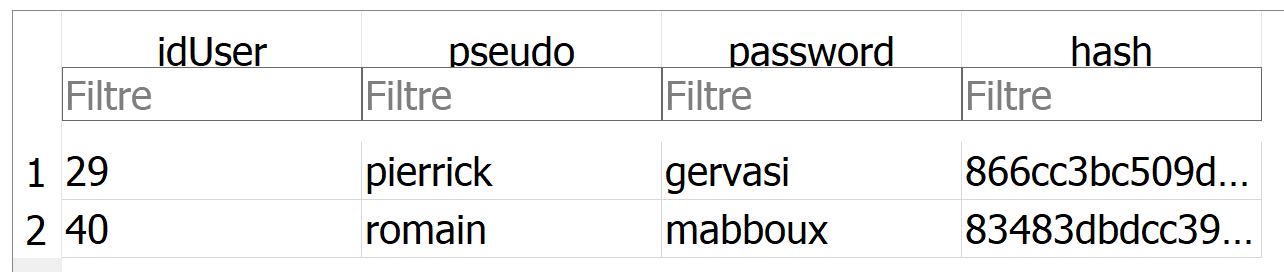
Nous avons basé la première version de notre application pour cette séquence 3 sur un tutoriel assez complet (10 vidéos d’une dizaine de minutes chacune), d’une chaine YouTube que nous avions l’habitude de consulter tout au long de l’électif. Ce tutoriel nous a conduit à créer des classes selon le schéma suivant :



Concernant notre base de données SQL, nous l’avons construite de façon simple selon l’architecture suivante :



De cette façon nous avons réussi à développer une application capable de récupérer les attributs d’un user sur l’API, et de les ajouter à la base SQL lors de la connexion :



Cependant, nous avons rencontré de nombreux problèmes comme :

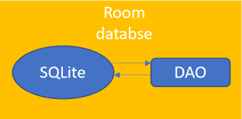
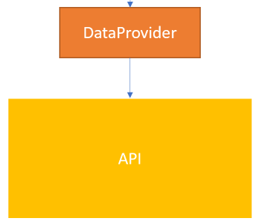
* Problème avec la version de Room
* Obligation d’utiliser la méthode allowMainThreadQueries lors de la création de la database
* Problème de clé étrangère
* …

Mais le problème majeur été le nombre de classe trop important pour le travail demandé. Cela a contribué à nous ralentir et fausser notre compréhension de l’application souhaité. C’est lorsque nous avons voulu gérer la partie connexion (API) / hors connexion (database) que nous avons constaté que notre architecture ne convenait pas. En effet, l’ensemble de requêtes combinées sur l’API + la database prenait parfois jusqu’à 30 secondes.

# Deuxième architecture

Notre seconde et dernière architecture n’est pas très optimale et ne fonctionne pas sous forme de blocs indépendants et agencés les uns avec les autres. Nous avons plutôt quelque chose de peu organisé (beaucoup de classes sont dépendantes des autres), mais c’est fonctionnel.

Activities

On se sert de la classe « Data Provider » pour récupérer des informations de l’API, et des DAO pour récupérer des informations du DAO.

# Ecoute de l’état de la connexion internet en continu

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 durant la séquence 2. Nous avons pu trouver du code notamment sur Stack Overflow (cf bibliographie) qui nous a aidé à réaliser les actions désirées.

Toutefois, le networkStateReceiver n’est implémenté que dans la MainActivity : on veut éviter que la connexion soit écoutée en continu dans les TodoList et TaskActivities ce qui ferait entrer en conflit les modes hors connexion et en connexion.

# Gestion des modes hors connexion et Connecté

Afin de gérer les modes hors connexion et Connecté, nous avons ajouté l’attribut « connexionOK » dans chacune des activités. Cela nous a aidé à faire notamment 2 cas correspondant aux deux modes : connecté ou hors connexion.

Nous avons choisit de ne pas mettre « connexionOK » en public static en raison du « NetworkStateReceiver » implémenté dans la MainActivity, parce que nous ne savions pas si le NetworkStateReceiver continuerai de fonctionner même si on se trouve dans la ToDoList activity. SI c’était le cas, cela reviendrait à modifier « connexionOK » mise en public static qui entrerait en conflit avec ce qui a été dit plus haut. Ainsi, chaque activity passe à l’autre dans l’intent l’attribut « connexionOK ».

# Modifications par rapport à la séquence 2

* On doit récupérer en plus l’ID du user après avoir récupéré son Hash en faisant une requête à l’API afin de stocker cette clé primaire.
* Nous avons modifié les classes Task, TodoList et User, pour avoir des clés étrangères dans la classes ToDoList (vers un User) et dans la classe Task(vers une ToDoList).

# Fonctionnement de l’application pour les modes hors connexion et connecté

En mode connecté :

* Dès qu’on charge une liste de ToDoLists d’un User en faisant une requête à l’API :
* On supprime du SQL la liste des ToDoLists correspondant à ce User
* On sauvegarde la réponse de l’API.
* Dès qu’on charge une liste de Tasks d’une ToDoList en faisant une requête à l’API :
* On charge du SQL la liste des Tasks de cette ToDoList
* On compare la liste provenant du SQL, et la liste provenant de l’API. Si certaines checkbox sont cochées dans la liste du SQL, on met à jour celles de l’API.
* On supprime du SQL la liste des Tasks correspondantes à cette To Do List
* On sauvegarde la réponse de l’API dans le SQL.
* Si on clique sur une checkbox d’une Task : on sauvegarde cet état dans le SQL et dans l’API

En mode hors connexion :

* On affiche les listes de To Do Lists et les Listes de Tâches et leur état qui sont enregistrées dans le sQL.
* Si une Tâche est cochée dans le SQL, son état est enregistré dans le SQL.

# Résultats :

En téléchargeant l’application, vous verrez que tout fonctionne, sauf l’état de la checkbox. Il manque très peu mais nous n’y sommes pas arrivés à temps.

# Conclusion

En conclusion, nous pouvons dire que cette troisième séquence nous a donné du fil à retordre au premier abord. En reprenant bien le code, et en adaptant les outils de ROOM à notre architecture, cela s’est avéré moins ardu. Il nous manque très peu pour mettre à jour les checkbox depuis le mode hors ligne, mais l’essentiel est compris.

# 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.
* Ajouter la possibilité de créer des ToDoLists ou des tâches en mode hors connexion, qui seront ensuite ajoutées à l’API dès lors qu’on se connecte.

# Bibliographie

* Google Codelabs pour reprendre l’utilisation de ROOM

<https://codelabs.developers.google.com/>

Site très bien avec énormément de tutoriels extrêmement bien faits. Nous le conseillons pour tous les étudiants qui chercheraient à apprendre à coder sur téléphone mobile.