AM4L: Applications Mobiles
Réalisation d'une application Android
ECAlendar

ECAM Bruxelles

Charles Vandevoorde (13019)
Lorenzo Riga (13018)
Antoine Vander Meiren (12088)
Gaetano Giordano (12054)
Sylvain Alonso (12150)

May 1, 2017

1 Introduction

Dans le cadre du cours d'applications mobiles, nous avons réalisé une application Android "ECAlendar" à l'aide d'Android Studio. Cette dernière utilise l'API de "Ecam Calendar" afin de permettre aux étudiants de consulter leurs horaires directement sur leurs GSM.

Dans ce rapport, nous allons présenter la manière dont nous nous sommes organisés en groupe mais aussi la structure de notre application ainsi qu'un listing de ses fonctionnalités.

2 Organisation du groupe

Nous avons hébergé notre projet sur Github de manière à faciliter le travail en groupe et nous sommes répartis les tâches comme suit lors de la première séance:

Charles chargement des données depuis l'API, DAO ainsi que gestion de la *RecyclerView*.

Lorenzo gestion du layout, style de l'application.

Antoine gestion des préférences, analyse est création des modèles requis et gestion de recherche de calendrier

Gaetano gestion du layout et style de l'application.

Sylvain intégration de la base de données SQLite et gestion des clics (affichage des détails lorsque l'utilisateur appuie sur un élément).

3 Présentation de l'architecture

3.1 Création des modèles

Avant de commencer à développer l'application, nous avons du analyser ce qui est renvoyé par l'API proposé par *ECAM Calendar* afin de réaliser un modèle des données que nous allons devoir stocker. Nous avons décidé de créer deux modèles différents: *Schedule* et *CalendarType*. *Schedule* stocke toutes les informations relative à un calendrier en particulier: local, date et heure, professeur, note, nom de l'activité, personne concernée par groupe, ... *CalendarType* quant à lui permet de sauver les différents calendrier disponible via un ID et une description du l'année/section, du professeur ou du local.

Tout au long de l'application nous utiliserons ces modèles pour créer des objets utilisables par chacune des parties.

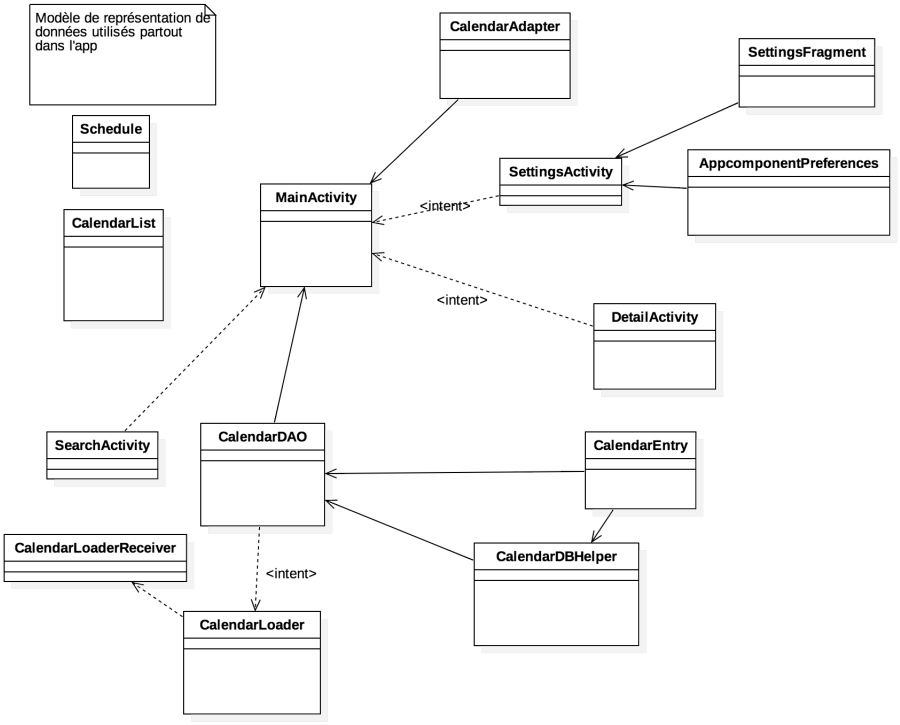


Figure 1: Diagramme UML de classe représentant notre application

3.2 Gestion des préférences

Pour ce qui est des préférences, nous avons simplement effectué une *PreferenceActivity*, accessible via le menu Android de l'application, et composée d'un fragment de préférence affichant une liste d'horaire disponible qui permet à l'étudiant de sélectionner son horaire personnel, en l'occurrence l'année dans laquelle il se trouve et sa série. Au premier lancement de l'application, l'activité principale n'affichera pas de calendrier et nous demandera de nous rendre dans l'onglet préférence afin de sélectionner l'année désirée. A partir de ce menu, nous avons développé une "Up Navigation" qui permet de retourner directement à la l'activité principale montrant le calendrier sélectionne. Notons que nous pouvons également utilisé le bouton back nativement présent sur les smartphones Android.

3.3 Recherche d'un second calendier

L'application permet à l'utilisateur de sélectionner deux calendrier à la fois: le sien comme principale via les préférences et un second pour pouvoir comparer deux calendrier par exemple. Pour en sélectionner un second, nous avons ajouté une petite loupe dans le menu Android de l'application qui nous amène à une autre activité nommée *SearchActivity*. Cette activité nous montre une liste, scrollable, de tous les calendriers disponible via l'API calendrier de l'ECAM. En cliquant une seconde fois sur cette loupe nous faisons apparaitre un clavier qui permet de rentrer du texte afin de rechercher un calendrier particulier. La recherche permettra à l'utilisateur de visualiser au fur et à mesure qu'il tape, les calendriers disponible en fonction du texte qu'il complète afin de simplifier la sélection désirée.
L'utilisateur pourra ensuite cliquer sur le calendrier de son choix, il recevra une confirmation de ce choix via un message *Toast*, et pourra retourner à l'activité principale via la "Up Navigation" ou encore le bouton back natif android. La valeur du calendrier sera alors stockée dans un *SharedPreference* disponible pour le reste de l'application. Une fois sur l'activité principale, le second calendrier sera disponible pour l'utilisateur. A chaque redémarrage de l'application ou changement de la section dans les préférences de l'application (*onDestroy* de la *MainActivity*), cette *SharedPreference* sera remise à zéro pour ne plus apparaitre dans le calendrier principale de l'utilisateur.

3.4 Récupération des données depuis l'API REST

Notre application pour fonctionner a besoin de données provenant de l'API REST du site `calendar.ecam.be`. Après une analyse des requêtes faites par l'application web Calendar, nous avions tous les outils en main pour démarrer l'implémentation de l'API dans notre application.

Pour éviter de bloquer l'interface utilisateur pendant le chargement des données, nous avons utilisé un *IntentService*. Celui-ci est un service spécial d'Android qui s'arrête dès que la tâche est finie. Nous avons choisi ce type de

service à la place de système comme les *AsyncTaskLoader*, ... pour plusieurs raisons.

- Meilleur séparation des affaires puisqu'un *IntentService* ne doit pas obligatoirement être appelé de l'activité principale.
- API plus propre grâce à l'utilisatation d'*Intent*.

Notre service peut être appelé pour récuperer deux types de données: la liste des calendriers disponibles (profs, locals, séries, ...) et un calendrier. Pour chaque type, nous envoyons la requête via un *Intent* contenant les données nécessaires à la récupération des données.

Les requêtes HTTP sont faites synchrone à l'aide de *future* puisqu'Android ne supporte pas le multi-threading imbriqué. Pour ce faire, nous avons utilisé la librairie *Volley* qui est une nouvelle libraire permettant de faciliter les requêtes HTTP.

L'utilisation des *futures* vient du fait que *Volley* est asynchrone de base.

Le parsing des données ICS (format de donnée de calendrier) est effectué par une libraire appelée *iCal4j*. Le parsing est effectué dans l'*IntentService* pour éviter des ralentissements inutiles de l'interface utilisateur.

Finalement, les données sont renvoyés via un *Broadcast* jusqu'à un récepteur spécialement écrit pour ce service. Nous détaillerons plus en détail ce récepteur dans la section 3.4.

3.5 Le DAO

DAO signifie *Data Access Object*. Cette classe permet d'être le seul point d'accès aux données et décide de quel source les données seront tirées. Dans notre cas, les sources sont la base de donnée et l'API REST du site `calendar.ecam.be`.

La sélection de la source est assez trivial, nous regardons d'abord si les données sont en base de données et si elles sont assez nouvelles (moins de 1 semaines) aussi non, nous allons utilisés notre *IntentService* pour charger nos données et les sauver dans la base de donnée.

Mise à part la création, le DAO est responsable de la sauvegarde et de la reprise de données en base de données.

Le DAO devant gérer l'asynchronisme de l'*IntentService*, nous avons utilisés un système de "notifieurs" qui demande d'être notifié dès qu'une donnée est arrivée. Le DAO garde donc une liste de classe implémentant une certaine interface.

Une autre caractéristique du DAO est que celui-ci est implémenté comme un singleton parce que nous devions avoir le contexte d'une activité pour sauvegardé les données en base de données. Lorsque le *CalendarLoaderReceiver* (le récepteur de l'*IntentService*) reçoit les données, nous devons accéder au DAO mais le récepteur n'a pas de contexte. Il faut donc avoir un contexte statique (mauvaise pratique en général) ou alors implémenter le DAO entant que singleton. Cette décision nous a permis également d'être beaucoup plus flexible quand au chargement des données.

3.6 Le *RecyclerView*

Le *RecyclerView* nous permet d'afficher une grande liste de calendriers sans problème de performances. Dans notre cas, nous avons créer une *RecyclerView* avec deux types de données. Nous pouvions soit avoir un slot horaire, soit avoir une information sur la date des slots horaires suivant.

Dans notre *ItemAdapter* qui gère la création à la volée des informations affichées, nous avons donc deux différents *ViewHolder*.

La *RecyclerView* ne supportant qu'une seule liste de donnée, nous avons créer une liste d'*Object*. Cette approche pourrait être changé dans le futur si nous rajoutons plusieurs types de vues différentes. Dans notre cas, cette solution est la plus facile et la plus rapide à implémenter.

4 Listing des fonctionnalités

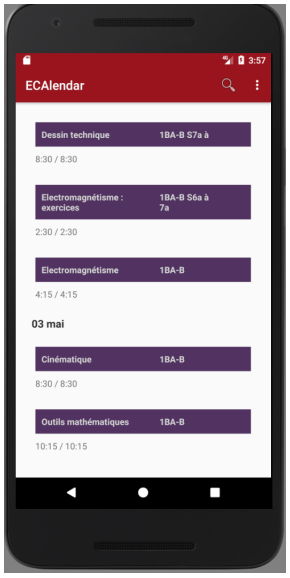
- Affichage de calendrier reprenant son horaire personnel
- Recherche parmi les calendrier disponible via l'API de l'ECAM
- Affichage simultané d'un second calendrier au choix pour comparer des horaires (local, professeur et étudiant)

4.1 Icône

Que serait une application mobile, sans une icône appropriée. C'est pour cela que nous avons crée une icône ECAlendar spécialement customisée pour notre application.

4.2 La page d'accueil

Comme vous pouvez le voir sur l'image, en entrant dans l'application ECAlendar nous avons une page qui nous indique les cours pour l'année sélectionnée. Autrement dit elle indique la date ainsi que les cours donnés. Elle indique également l'heure du début et l'heure de fin du cours.

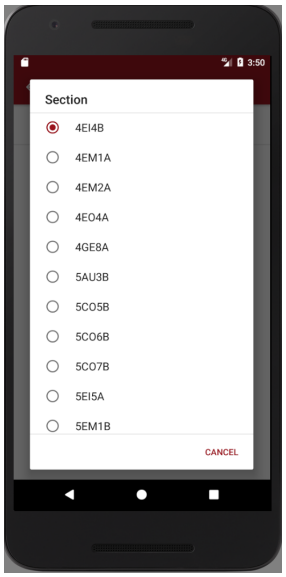


4.3 Détail d'un cours

En cliquant sur l'un des cours on entre dans une nouvelle page qui nous permet d'afficher les détails concernant ce cours. C'est a dire le local, l'enseignant et la série.

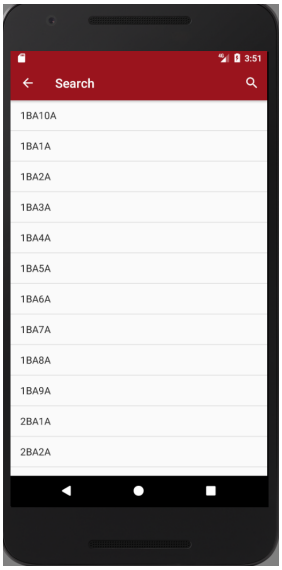
4.4 Choix de l'année

Bien évidement, notre application nous permet de choisir également l'année d'étude pour lequel nous aimerions voir l'horaire. C 'est pour cela que sur la page d'accueil en cliquant sur les 3 points en haute à droite, cela nous redirigera vers une page qui nous permettra de cocher une année et ainsi la mettre dans nos préferences.



4.5 Recherche

Nous pouvons également choisir l'année sur la page d'accueil, en cliquant simplement sur la petite loupe située en haut. Cela nous redirigera vers la page ci-dessous.



4.6 Remarques

Comme vous pouvez le voir sur certaines images, certains bugs interviennent comme par exemple l'heure de fin du cours. Nous tenons a préciser que ces bugs ont été corrigés et fonctionnent désormais parfaitement.

5 Conclusion

La réalisation de cette application nous aura permis de nous familiariser avec les notions vues lors des différents cours théoriques ainsi que d'améliorer notre organisation en groupe dans le cadre d'un projet de développement.

Nous avons atteint les objectifs que nous nous étions fixés en réalisant une application fonctionnelle et user-friendly reprenant l'ensemble des fonctionnalités auxquelles nous avions pensé lors de la première séance.