Emploi du temps FDS

Par Stéphane Wouters

Pour l'UE HMIN303 Développement pour l'embarqué
Le 6 Janvier 2015

Présentation

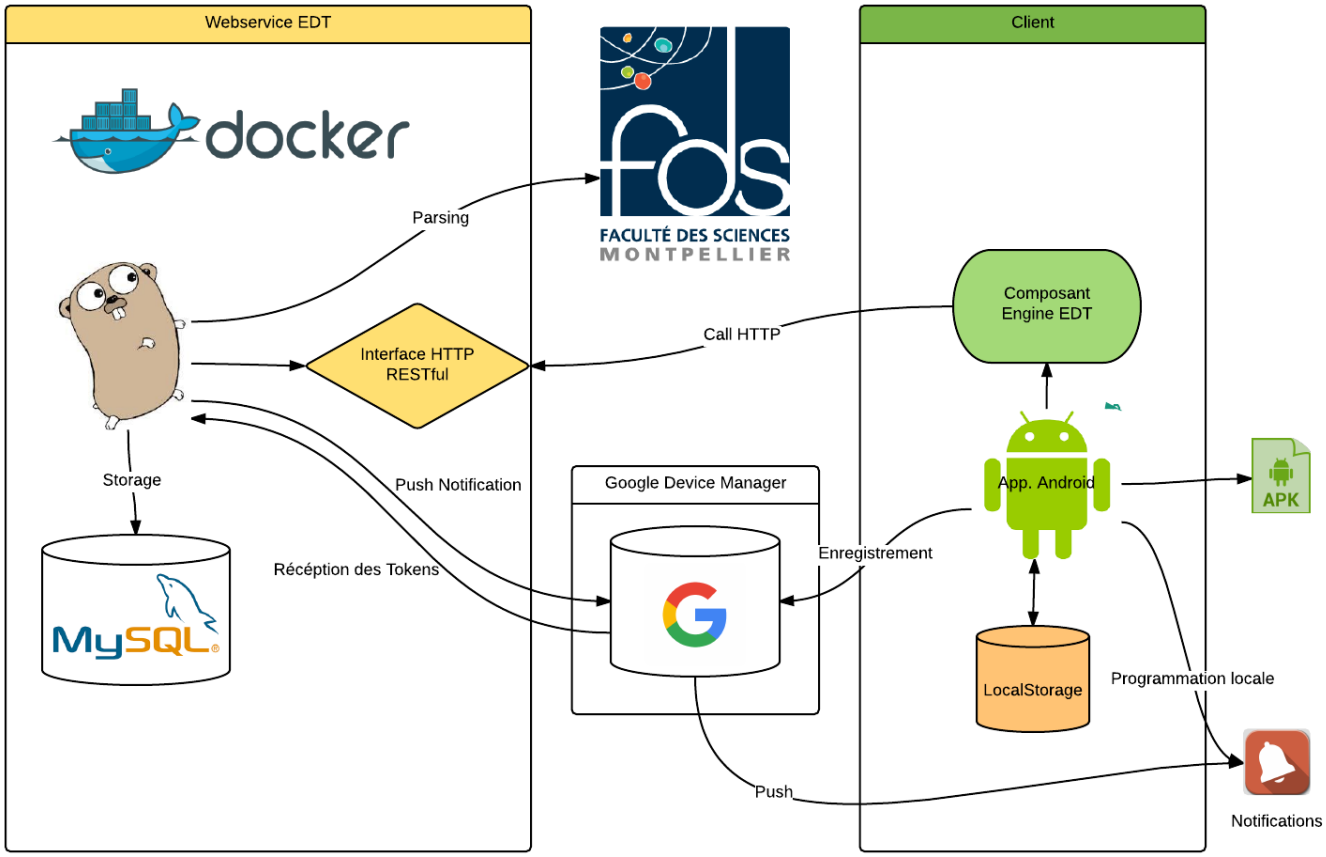
Cette application permet aux étudiants et enseignants de **consulter l'emploi du temps de la FDS**, de manière personnalisée, dans une application mobile. Elle communique avec l'emploi du temps de la faculté des sciences et offre une information la plus à jour possible. Elle dispose de plusieurs option comme des alertes personnalisées ou un stockage « offline » de l'emploi du temps.

Cahier des charges des fonctionnalités

Liste des fonctionnalités, pour l'utilisateur :

- **Choix de ses UEs** parmi toutes celles proposées par la FDS
- **Consultation des cours programmés** pour ses UEs sélectionnées : Nom du cours, description, date et heure du cours, emplacement…
- **Information synchronisée** en temps réel avec la FDS
- **Stockage des cours 3 semaines à l'avance** sur le téléphone au cas où le serveur ne serait pas accessible (pas de connexion Internet ou serveur surchargé)
- **Reception d'alertes hors-lignes avant le début d'un cours**, même quand l'application est fermée (configurable : désactivation et choix de la durée)
- **Reception d'alertes « en lignes » en cas de changement exceptionnel** sur l'emploi du temps (exemple : changement de salle).

Architecture



Étant donné la masse d'échanges existants dans le système d'information, une bonne architecture n'est pas à négliger.

Si contre le schéma de l'architecture réseau.

On peut distinguer trois grandes parties :

- **Webservice EDT**, qui correspond à la partie serveur écrite en GO
    - Elle est encapsulée dans un container Docker
    - Elle parse régulièrement l'emploi du temps de la FDS
    - Elle communique avec un serveur MySQL pour stocker les informations sur les UEs et sur les créneaux
    - Elle propose une interface HTTP qui servira au client
- **La partie cliente**
    - Développée pour Android, elle contient le fonctionnement de l'application, les vues…
        - Son module « Engine EDT » est un ensemble de classes Java qui communique avec le serveur
        - Elle dispose d'une base de stockage locale pour permettre une consultation « offline » de l'emploi du temps
        - Elle programme des notifications pour les cours à venir
- **La partie « Google Device Manager »** qui permet d'envoyer des notifications push à tout moment (le serveur envoi une information au téléphone, sans que l'application n'est besoin d'être ouverte ou de se synchronisation
    - Quand l'utilisateur ouvre son application, son « token id » est envoyé au serveur de Google et au serveur Golang.
    - Quand le serveur veut envoyer une « push notification », il envoi le message et le token id au serveur de Google
    - Le serveur de Google envoi la notification au téléphone

API RESTful

Le serveur propose une API HTTP RESTFul privée, utilisée par le client (module totalement indépendant qui pourrait très bien être utilisé pour d'autres types de clients). Cette API retourne du contenu au format JSON.

Elle dispose des flux suivants :

	Parametres	
GET /list-ue	Aucun	Retourne un tableau des UES disponibles au format JSON' [{Name, Description}]
GET /list-times	- ues: Liste des UEs à filtrer, séparés par les « , » Exemple : HMIN303,HMIN302,FMIN305	Retourne un tableau JSON des créneaux horaires correspondants aux UEs voulus [Name, Descrition, Location, DateStart, DateEnd]

Partie client Android

Dans une optique de réutilisation, une stratégie composants a été employée.
Le code source est composé de deux grandes parties :

- Le composant Engine EDT qui s'occupe de la communication avec le serveur et qui propose des interfaces Java pour être utilisé avec d'autres modules ;
- Le composant Application Android qui utilise les interfaces du composant Engine EDT et qui réalise l'affichage et contrôle les actions de l'utilisateur

Les fonctionnalités suivantes de l'API Android ont été exploités :

- Appels réseau
- Enregistrement des préférences utilisateurs
- Stockage de données dans une « flat database »
- Programmation de notifications locales
- Réception de « push notifications »

Les points principaux qui ont demandés le plus d'attention en programmation pour le client :

- Gestion du « CRUD » : Interfaces d'ajout, modification, suppressions de données utilisateurs (Pour les UEs par exemple) ;
- Synchronisation entre le serveur et le contenu local pour diminuer les appels réseau tout en conservant des données à jour ;
- La programmation de notifications ;
- La synchronisation des services, parfois asynchrones, pour avoir un affichage fluide tout en ayant des processus de traitement continus (Actualisation de l'emploi du temps pendant sa consultation…) ;
- Utilisation du « Material Design » pour des vues agréables visuellement

Résultats

La totalité des fonctionnalités du cahier des charges a été développée, hormis le système d'alerte en cas de changements exceptionnel (de salle par exemple).

Note personnelle : La fonctionnalité de push est prête au niveau architecture et a été testée (le serveur parvient a envoyer des messages au téléphone via les tokens), mais la liaison avec l'emploi du temps n'a pas été développée. Ceci demanderait d'envoyer et de stocker tout les choix d'UE des utilisateurs sur le serveur, ainsi que de créer l'algorithme de détection de différence entre deux dates de l'emploi du temps. Cela demandait un temps de développement important supplémentaire que je n'ai pas pu apporter. J'ai préféré me concentrer sur une première version parfaitement finalisée et fonctionnelle.

Les notifications locales pour rappeler des cours à l'avance sont par contre bien fonctionnelles.

Une vidéo de démonstration est mise à disposition pour montrer les fonctionnalités.

Mise en production

Au-delà de la conception et du développement de l'application, j'ai pris le temps d'aller jusqu'a l'étape de la mise en ligne sur le marché mobile.

L'application est donc 100% finalisée en terme de fonctionnement, de montée en charge, d'ergonomie et d'aspect visuel.

Moyens mis en ouvre

- Location et configuration d'un serveur dédié pour l'hébergement du webservice de récupération et de distribution de l'emploi du temps.
    - La tache du parsing est ajoutée en crontab et s'effectue toutes les heures
- Création d'un compte Google développeur et achat des licences nécessaires
- Génération d'un certificat pour l'application et d'un APK signé prêt pour une publication
- Rédaction de la présentation de l'application, des formulaires commerciaux et administratifs pour la mise en ligne sur le store Google Play

Disponibilité

L'application est disponible en téléchargement gratuit sur le Google Play dans sa version 1, sous le nom de « Calendrier FDS ».

Téléchargement : https://play.google.com/store/apps/details?id=fr.doelia.calendrierfds

Pour information, L'API est disponible sur le serveur edt.doelia.fr
Exemple d'appel : http://edt.doelia.fr:2010/list-ue Retournera la liste des UEs au format JSON.

Installation et configuration technique

Pour tester l'application, il n'est pas conseillé de monter le projet en local. L'architecture est composée de nombreux modules et n'est pas aisée à configurer et à mettre en place. Pour un test, il est préférable de télécharger directement l'application sur le Google Play.

Une version compilée de l'application est aussi fournie au format APK dans l'archive.
A titre informatif, quelques directives pour monter le projet :

Dépendances

Le projet nécessite des packages / logiciels suivants :

- Serveur MySQL
- Golang : https://golang.org/doc/install
    - Package github.com/go-sql-driver/mysql
- Docker : https://docs.docker.com/
    - Compatible tout OS
- SDK dernière version (Android 5)

Installation et configuration

Lancer le container Docker:

```
docker run \
  -p 3306:3306 \
  -v $(pwd)/deployed:/docker-entrypoint-initdb.d \
  --name emp-mysql \
  -e MYSQL_ROOT_PASSWORD=gogoedt \
  -e MYSQL_DATABASE=edt \
  -d \
  mysql:latest
```

Ajouter le script .sql fournit au serveur MySQL

Installer les dépendances go :

```
go get github.com/go-sql-driver/mysql
```

Compiler le serveur :

```
cd server
go build
```

Faire un parsing depuis la FDS (devrait remplir la base de donnée) :

```
./server —parse -mhost [IP DOCKER] -muser root -mpass gogoedt
```

Lancer le serveur HTTP :
./server -mhost [IP DOCKER] -muser root -mpass gogoed -port 2000

Vérifier son fonctionnement via l'url http://localhost:2000/list-ue

Puis compiler et lancer l'application Android depuis l'IDE netbeans avec le plugin nbandroid (l'adresse localhost est chargée par défait dans les sources).