

Emploi du temps FDS

Par Stéphane Wouters

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

EMPLOI DU TEMPS

FACULTÉ DES SCIENCES

PROCHAIN COURS

L1 - Série 5 Gr.AL1 - Série 5 Gr.BL1 - Série 5
Gr.CL1 - Série 5 Gr.DL1 - Série 5 Gr.EL1 - Série 5
Gr.FHLBE202 Clés et outils pour les SVT

📅 Le 18/01 de 11:30 à 13:00

📍 Amphi 5.01(275p)



Calendrier



Mes UEs



Réglages



A propos

🔍 RETOUR

VOS UES

Vous êtes abonné aux cours des UEs suivantes :

🎓 HLBE204

L1 - Série 5 Gr.CHLBE204 clés et
outils pour la biologie et l'écologie



🎓 HLBE202 6A

L1 - Série 6 Gr.AHLBE202 Clés et
outils pour les SVT



🎓 HLBE202

L1 - Série 5 Gr.FHLBE202 Clés et
outils pour les SVT



+ S'ABONNER À UNE NOUVELLE UE

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 options comme des alertes personnalisées ou un stockage « offline » de l'emploi du temps.

Cahier des charges des fonctionnalités

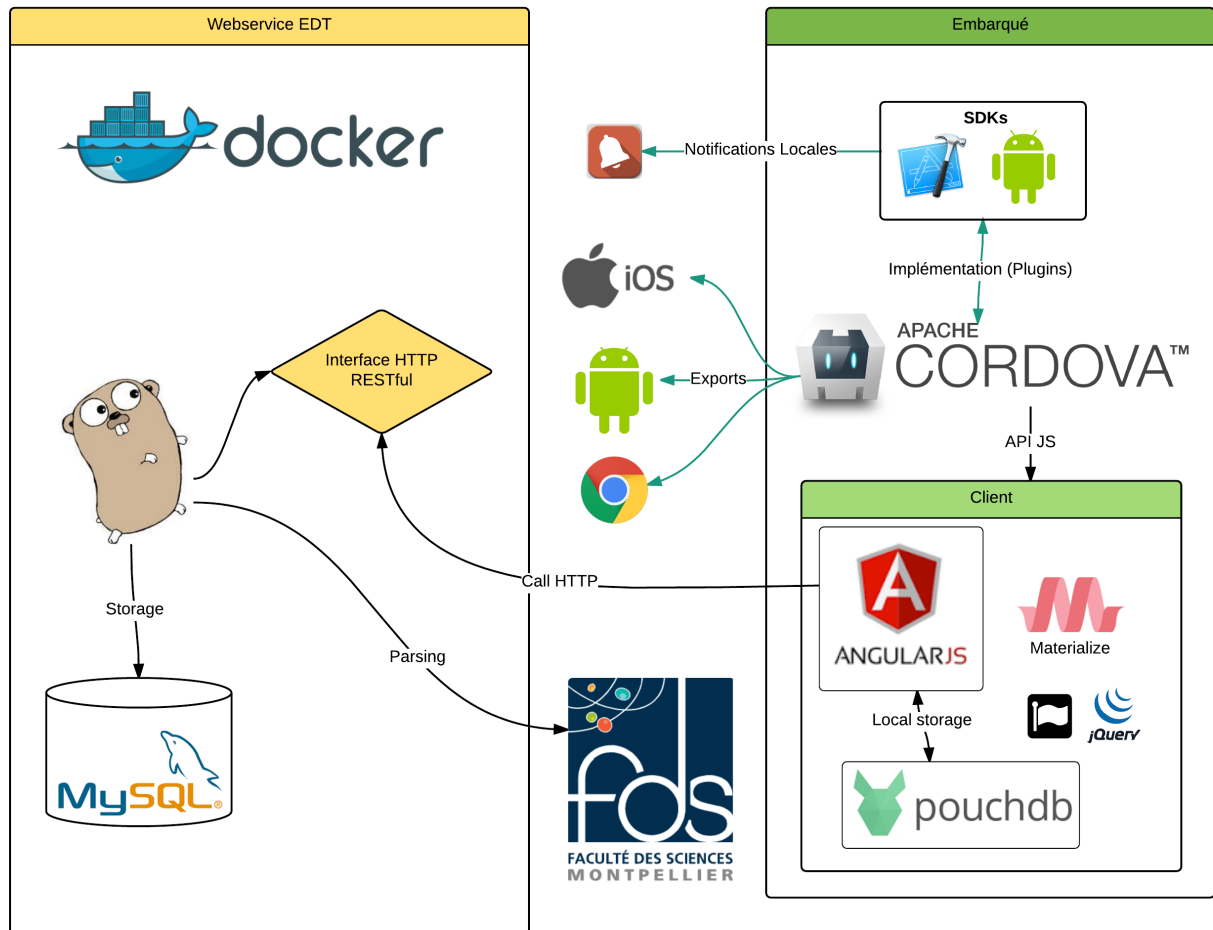
Liste des fonctionnalités, pour l'utilisateur :

- **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 cas de changement exceptionnel** sur l'emploi du temps (exemple : changement de salle).
- **Connexion à une application Android partenaire** pour la localisation des salles

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 deux grandes parties :

- **La partie serveur, Webservice EDT**, écrite en GO
 - Elle est encapsulé 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 et iOS, elle contient le fonctionnement de l'application, les vues...
 - Elle communique avec les SDKs via l'API de Cordova (voir partie client, plus bas)
 - Elle dispose d'une base de stockage locale (PouchDB) pour permettre une consultation « offline » de l'emploi du temps
 - Elle programme des notifications pour les cours à venir

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, Description, Location, DateStart, DateEnd]

Partie cliente

Note personnelle à propos de Cordova : J'ai débuté le projet en Android / Java pure et effectué une grande partie de la base de mon projet de cette manière (voir code source : EDTAndroid). Seulement pour des raisons professionnelles, j'ai fait le choix de recommencer et d'effectuer le développement avec d'autres technologies que je serais amené à utiliser dans mes futurs responsabilités. En effet, les technologies d'export cross-plateformes sont à l'ordre du jour, surtout dans le monde du Web.

J'ai utilisé l'API **Cordova** pour développer mon application.

Cordova propose une API Javascript qui permet de développer des application mobiles dans les langages du web et de les exporter vers plusieurs plateformes : Chrome, Android, iOS, Windows Phone, Blackberry...

Cependant, même si la syntaxe utilisée n'est pas celle du Java, l'architecture et les problématiques liées au développement mobile sont les même : Système de Service, Threads, communication réseau, synchronisation des tâches...

Les fonctionnalités suivantes de l'API Android (et iOS) ont été exploitées :

- Appels réseau
- Enregistrement des préférences utilisateurs
- Stockage de données dans une « flat database » (NoSQL)
- Programmation de notifications locales
- Lancement d'autres applications Android avec paramètres

J'ai aussi découvert et utilisé plusieurs technologies émergentes du web. Parmi elles :

- **AngularJS**, puissant framework Javascript
- **PouchDB**, database NoSQL inspirée par Apache CouchDB, écrite en Node.JS
- **Materialize**, framework CSS optimisé pour mobile
- **Font Awesome**, bibliothèque d'icônes

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

- Synchronisation entre le serveur et le contenu local pour diminuer les appels réseau tout en conservant des données à jour ;
- Gestion du « CRUD » : Interfaces d'ajout, modification, suppressions de données utilisateurs (Pour les UEs par exemple) ;
- 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...) ;
- La programmation de notifications ;
- La communication avec l'application partenaire de localisation des salles (Qui demandait un plugin avec Cordova pour lancer les Intents)
- Utilisation du « Material Design » à la mode 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).

La fonctionnalité de push est prête au niveau architecture et a été testée (le serveur parvient à 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 bien fonctionnelles.

La fonction « Localiser la salle » est fonctionnelle et ouvre correctement l'application partenaire, à condition que l'utilisateur l'ai installée (Sinon le bouton « Localiser » ne s'affichera pas). Si cette application partenaire de localisation est publiée sur le Google Play, un lien invitant à l'installer sera ajouté.

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

Mise en production

Au-delà de la conception et du développement de l'application, j'ai pris le temps d'aller jusqu'à 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 oeuvre

- Location et configuration d'un serveur dédié pour l'hébergement du webservice de récupération et de distribution de l'emploi du temps.
 - La tâche 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>

Je compte la publier aussi sur l'Apple Store, mais les délais de validation sont plus longs.

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)
- Cordova

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 gogoedt -port 2000
```

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

Construire le client :

```
cd client/MyApp  
cordova add platform android  
cordova run android
```

Compiler une version Release et signer le fichier .apk pour le store

```
./compile.sh
```