Document d'architecture technique (DAT) JCertif Mob

1. Principe de base

L'application devra être conçue sur deux principes :

- C'est à destination de la plateforme Android, donc devra répondre aux contraintes de l'embarqué (cache et sécu).
- Générique. L'objectif étant de faire de JCertif App, l'application pour l'organisation de conf, qui tout le monde pourra télécharger, alors le client mobile devra être suffisamment générique et paramétrable pour être utilisée par d'autres instances de JCertif App.

2. Fonctionnalités à implémenter

2.1. Affichage des informations de base

L'application devra être pensée générique afin de permettre une utilisation par d'autres installations de JCertif App. On permettra aux utilisateurs de JCertif Mob de fournir l'url de base des services. On pourra ainsi, afficher des informations basiques d'une conf : nom de la conf, lieu, dates, texte de présentation

Prérequis :

1. Avoir un service rest qui permet de récupérer un objet conférence (nom, dates, description, lieu)

URL du service	[URL_FACADE]/api/conference/list
Exemple en pré-prod	http://jcertif.baamtu.com/jcertif-facade/api/conference/list
Informations remontées	• id • nom • website • dateDebut • dateFin • details
Reste à faire	• Créer un objet façade Conference qui ne contient pas les FAQ • Modifier le retour de ConferenceFacade pour envoyer le nouvel objet

2.2. Lister des speakers

Cette fonctionnalité permet d'afficher la liste des speakers avec une petite description (sur une ligne). En cliquant sur un speaker, on affiche une description plus complète. La liste des speakers est à stocker dans la base de données locale (sqlite). On pourra mettre en place un mécanisme qui périodiquement, 1 fois par jour par exemple, va mettre à jour la liste de speakers locale avec celle remote.

Prérequis :

1. Avoir un service rest qui permet de récupérer l'ensemble des speakers

URL du service	[URL_FACADE]/api/speaker/list
Exemple en pré-prod	http://jcertif.baamtu.com/jcertif-facade/api/speaker/list
Informations remontées	• id • nom • prenom • compagnie • photo • bio
Reste à faire	Rien. Service opérationnel.

2.3. Calendrier global avec détails par session

Comme dans l'application web, avoir un calendrier global de toutes les sessions. On doit avoir la possibilité d'afficher le détail.

Prérequis :

1. Avoir un service rest qui permet de récupérer la liste des sessions (date, short description, long description, speaker)

URL du service	[URL_FACADE]/api/event/list
Exemple en pré-prod	http://jcertif.baamtu.com/jcertif-facade/api/event/list
Informations remontées pour un item	• id • nom • dateDebut • dateFin • salle • sommaire • description • speakersId : Liste ids speaker séparés par ",". Exemple : 12,35,25,45. • motcle • sujets : Liste de libellés sujets séparés par « , ». Exemple : java,jee,android
Reste à faire	• Créer l'objet métier • Créer le service

2.4. Inscription et connexion

Permettre à un utilisateur de s'inscrire puis se connecter. On stocker les informations de connexion en local (SharedPreference)

URL du service	[URL_FACADE]/api/user/create
Exemple en pré-prod	http://jcertif.baamtu.com/jcertif-facade/api/user/create
Description	Créer un participant avec les informations de connexion
Informations	• civilité

envoyées	• nom • prenom • email • password • role • type • compagnie • site web • téléphone fixe • téléphone mobile • ville • pays
Reste à faire	• Créer l'objet métier • Créer le service

URL du service	[URL_FACADE]/api/user/{email}
Exemple en pré-prod	http://jcertif.baamtu.com/jcertif-facade/api/user/toto@gmail.com
Description	A partir d'une adresse email, retourne un utilisateur
Informations remontées pour un item	• civilité • nom • prenom • email • password • role • type • compagnie • site web • téléphone fixe • téléphone mobile • ville • pays
Reste à faire	• Créer l'objet métier • Créer le service

2.5. Inscription à des sessions

Permettre à un utilisateur de s'inscrire à une session.

URL du service	[URL_FACADE]/api/event/adduser/{idevent}/{email}
Exemple en pré-prod	http://jcertif.baamtu.com/jcertif-facade/api/event/adduser/123/toto@gmail.com
Description	Ajouter un event dans l'agenda personnalisé de l'utilisateur
Informations envoyées	• civilité • nom • prenom • email

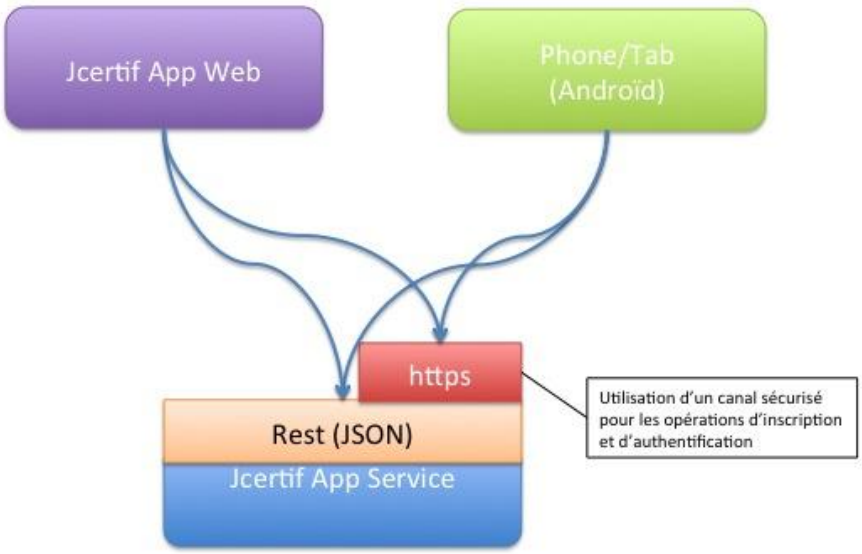
	• password • role • type • compagnie • site web • téléphone fixe • téléphone mobile • ville • pays
Reste à faire	• Créer l'objet métier • Créer le service

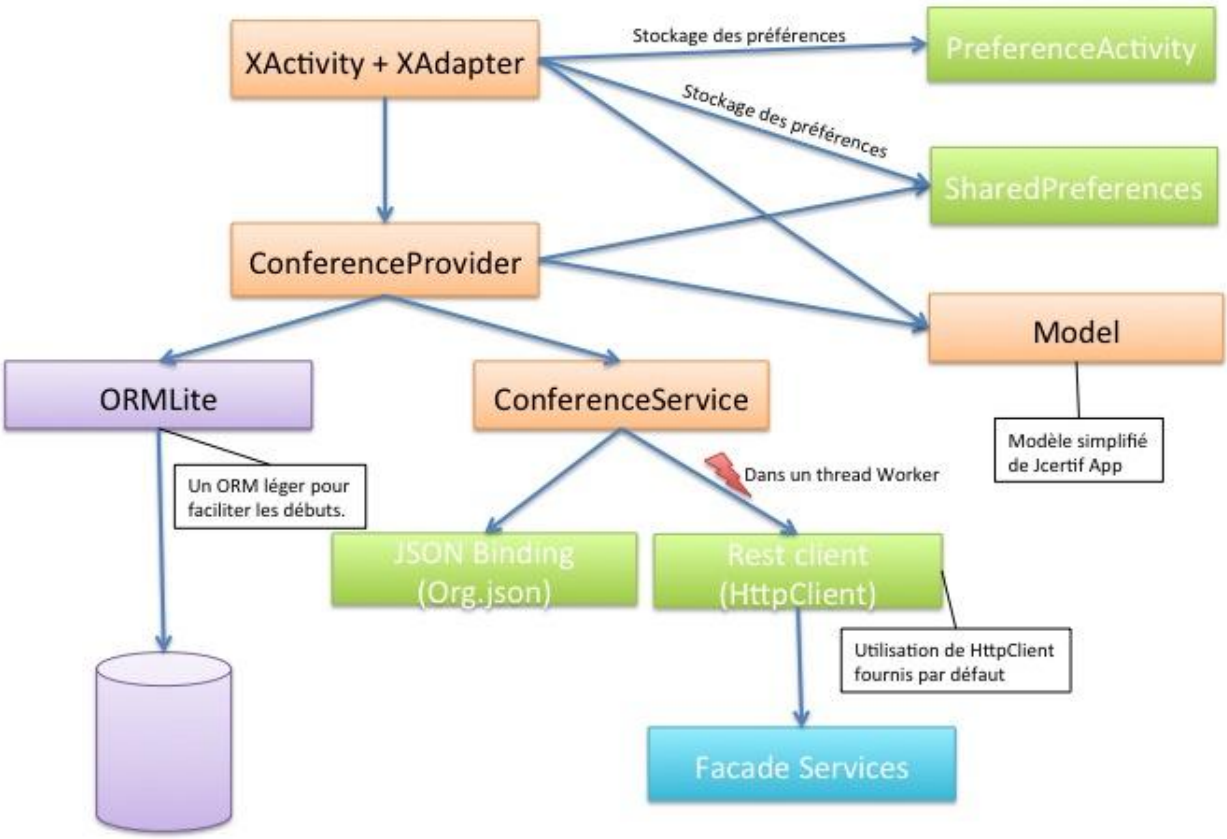
2.6. Affichage d'un calendrier personnel

Idem que 2.3 sauf qu'ici ceux sont uniquement les sessions auxquelles l'utilisateur participe qui sont affichées.

URL du service	[URL_FACADE]/api/event/user/{email}
Exemple en pré-prod	http://jcertif.baamtu.com/jcertif-facade/api/event/user/toto@gmail.com
Description	Récuperer la liste des évènements d'un utilisateur
Informations remontées pour un item	• idevent : liste d'identifiants d'évènements séparés par « , ».
Reste à faire	• Créer l'objet métier • Créer le service

Architecture technique





Environnement de développement

1. Maven

La version de Maven à utiliser est 3.0.3 (http://maven.apache.org/download.html).

2. Android SDK

Installer le SDK Android (http://developer.android.com/sdk/index.html)

3. Variable d'environnement ANDROID_HOME

Avant de compiler avec Maven il faut ajouter cette variable d'environnement. Elle doit contenir le répertoire d'installation d'Android SDK.

4. EDI & Settings

On va utiliser Eclipse comme IDE avec le plugin ADT (Android Devlopment Tools). Il est conseillé qu'on ait tous la même version d'Eclipse. Je préconise la dernière Helios. Pour

ceux qui encore leur install utilisée pour JCertif App, ils peuvent rester dessus et juste rajouter le plugin ADT
Eclipse : http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/heliossr2
Plugin ADT, utiliser le Repository : https://dl-ssl.google.com/android/eclipse/
Il faudra également installer un **plugin SVN** pour eclipse : je conseillerai Subeclipse
Une fois l'installation terminée, il faut créer un émulateur Android. La version cible sera la 2.3.
Pour plus de détails sur l'environnement Androïd : https://dl-ssl.google.com/android/eclipse/
Plugin maven :

- installer **M2Eclipse** : http://m2eclipse.sonatype.org/
- installer le plugin **Maven Android Plugin** (obligatoire pour fonctionner http://code.google.com/p/maven-android-plugin/wiki/EclipseIntegration)

5. Découpage de l'application (maven et package)

Pour l'instant, nous avons un projet maven avec 2 modules :

- project-jcertifmobile-app
- project-jcertifmobile-test
- avec un pom parent à la racine

I usually separate stuff into sub-packages in my apps.

- com.jcertif.android.app : contient les classes de base et les fonctionnalités globales de l'application
- com.jcertif.android.net - network stuff, network related utils
- com.jcertif.android.data - db helpers, providers, etc
- com.jcertif.android.model - object model

6. Repo

https://project-jcertif-mobile.googlecode.com/svn/trunk/

7. Plateforme d'intégration continue

(en cours)

8. Quelques liens utiles

http://code.google.com/p/maven-android-plugin/wiki/GettingStarted
http://code.google.com/p/maven-android-plugin/wiki/EclipseIntegration
http://developer.android.com/guide/index.html
http://developer.android.com/guide/practices/design/performance.html
http://ormlite.com/sqlite_java_android_orm.shtml