Auteur : Keil Laurent 2012-2013
Version 1.1 du module "Ccampus"-côté serveur

Documentation : guide de l'utilisateur

Client : Jean-Noël Colin

1. Introduction.

Ceci est un projet d'**application mobile d'e-learning**, plus précisément pour l'instant une extension de la plateforme WebCampus sur le système d'exploitation pour mobile Androïd. Il contient deux parties bien distinctes, une **partie côté client** (non abordé ici, probablement celle que vous,utilisateur, devez concevoir), traitant de l'interface utilisateur et de l'accès aux données récoltées par la **partie côté serveur** abordée dans cette documentation. L'intérêt de cette séparation est de permettre une abstraction concernant l'utilisation des données.

2. Structure du projet.

Pour une vue globale du projet, référez-vous au dossier « uml diagram » ou l'image en fin de document. Il est important de comprendre la hiérarchie de l'api afin de l'utiliser d'une manière correcte. Une documentation détaillée est également accessible dans le dossier « javadoc ».

Je vais ainsi aborder brièvement le package main et l'api de mon projet personnel auquel vous pouvez accéder et les fonctionnalités envisageables.

2.1. Package main :

Il est le package principal utilisé pour démarrer l'application.

Main est la classe principale, elle permet dans le cas d'une utilisation interne au projet individuel de lancer un exemple de lecture des données soit en mode mock*, soit en mode WebService**.

Example propose un **exemple d'utilisation des services de l'application** en mode mock* et un exemple en mode WebServices**.

Parameters est utilisé pour déclarer les paramètres de l'application : mode MOCK* ou WS**

FactorySession lance une session de l'application en mode MOCK* ou WS** suivant les paramètres définis dans *Parameters.*

***Mock :** objet simulant le comportement d'objet réel de manière contrôlée, permettant de tester le comportement du programme sans avoir un accès aux objets réels.

Dans notre cas, pour ne pas devoir se connecter en ligne à un e-learning pour récolter des données, on crée un mock qui simule ces données, afin de tester le comportement de notre application sans données réelles.

****WS = WebServices :** Cette alternative donne un accès serveur à l'e-learning, pour l'instant ChouetteCampus. Elle offre donc une session réelle et on peut donc manipuler des données réelles avec l'application.

Attention : L'accès n'est permis sur Androïd que par le mock et l'application se déploie seulement sur ChouetteCampus.

2.2. Package api :

Ensemble d'interfaces de fonctionnalités consommant les WebServices (ou les données du mock) auxquelles le module client peut accéder. Je vais détailler chacune des interfaces afin de pouvoir les utiliser.

2.2.1. ***Session*** est la première interface sur laquelle la partie cliente accède. Un objet Session vous est crée, soit en mode *mock* ou *ws*. En ws, le client est **authentifié** avec son username et password donné en entrée. Cet objet permet d'accéder à d'autres objets et outils de l'application. Vous pouvez vous référer à la javadoc pour une compréhension détaillée de chacune des classes et chacune des méthodes.

Chaque méthode : *getUserData, getUpdates et getCourses* renvoie un objet contenant des informations particulière sur l'utilisateur de la session en cours.

2.2.2. L'interface ***UserData*** permet d'obtenir des informations séparément sur l'utilisateur.

2.2.3. L'interface ***Updates*** permet d'obtenir des informations sur les updates pour l'utilisateur connecté depuis sa dernière connexion. Elle contient une méthode *nbUpates* étant le nombre d'update de l'utilisateur (cfr. interface *Update*) et *getUpdatesTab* qui renvoie un tableau de *nbUpates* objets *Update.*

2.2.3.1. ***Update*** contient des informations séparément sur une modification particulière d'une ressource d'un module d'un cours de l'utilisateur connecté depuis sa dernière connexion.

2.2.4. L'interface ***Courses*** permet d'obtenir des informations sur les cours de l'utilisateur. Son objectif est de fournir un objet *Course* approprié par une des deux méthodes, en fournissant deux méthodes renvoyant des tableaux définissant chaque cours disponible. La méthode à utiliser est au choix de l'utilisateur.

2.2.4.1. ***Course*** permet d'obtenir des informations séparément sur un cours particulier de l'utilisateur. Entre-autre, l'interface permet d'obtenir 3 objets qui contiennent différentes informations sur le cours, plus précisément un objet *CourseTool* détaillant les outils disponibles du cours en question(pour ce projet seulement les annonces et les documents), un objet *Annonces* et un objet *Docs*.

2.2.4.1.1. ***CourseTool*** permet d'obtenir des informations séparément sur les outils disponibles et nouveautés du cours concerné de l'utilisateur de la session.

2.2.4.1.2. ***Annonces*** permet d'obtenir des informations sur les annonces d'un cours particulier de l'utilisateur. L'interface, de la même manière que l'interface *Courses*, de fournir deux tableaux définissant chaque objet annonce pouvant être récupérer par une des deux méthodes *getAnnonce.*

2.2.4.1.2.1. L'interface ***Annonce*** contient des informations sur une annonce particulière d'un cours particulier de l'utilisateur.

2.2.4.1.3. ***DocsAndFolders*** permet d'obtenir des informations sur les documents et dossiers de documents d'un cours particulier de l'utilisateur. Elle peut fournir un tableau des noms des dossiers et un des noms de documents du cours concerné. L'utilisateur peut alors obtenir l'objet représentant ce document ou dossier par les deux méthodes associées.

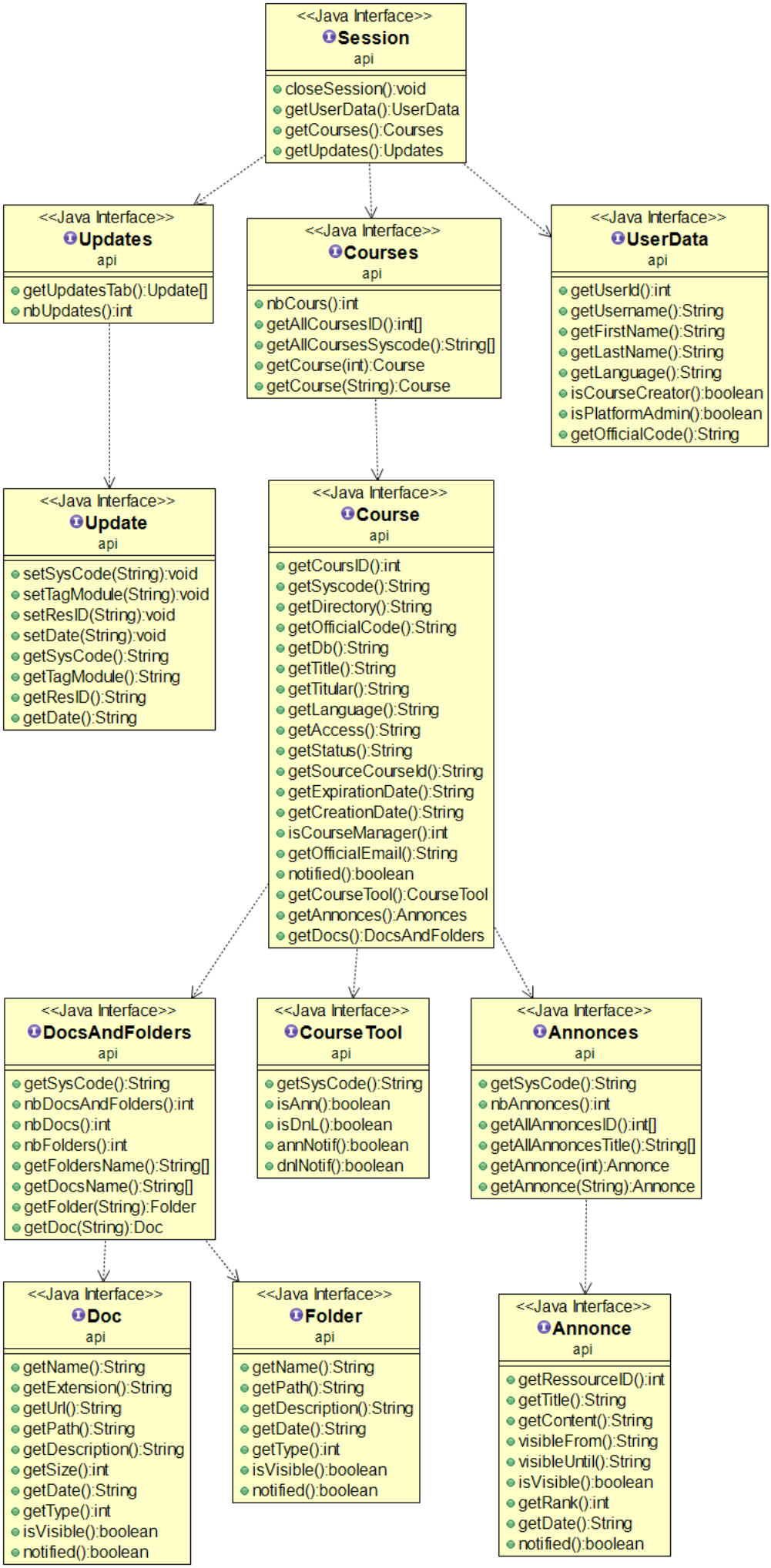
2.2.4.1.3.1. L'interface ***Doc*** contient des informations sur un document particulier d'un cours particulier de l'utilisateur.

2.2.4.1.3.2. L'interface ***Folder*** permet d'obtenir des informations sur un dossier particulier d'un cours particulier de l'utilisateur.

3. Utilisation.

Comme énoncé précédemment au point 2.1., un exemple d'utilisation est disponible dans la classe *Example* du package *main*. Le principe général est de fournir des méthodes permettant d'obtenir un objet comportant les informations souhaitées. Dans certains cas, des méthodes utilitaires comme le nombre d'éléments dans un objet ou un tableau d'identifiants de sous-objet à l'objet en cours est disponible. Ce **tableau** permet donc d'obtenir une série d'identifiants que l'on doit ensuite fournir en paramètre à une méthode qui permet elle d'obtenir l'objet voulu. Pour l'instant, seul une lecture d'information est permise (getter).

Exemple : L'utilisateur devra d'abord se logger en fournissant son username et password par la méthode *getSession* de *FactorySession*. On obtient alors un objet *Session* qui permet d'accéder à l'objet *Courses* qui contient 2 méthodes *getCourse* auxquelles on doit fournir un identifiant du cours désiré pouvant être obtenu par la méthode *getAllCoursesID* ou *getAllCoursesSyscode*. L'objet *Course* renvoyé permet ensuite d'obtenir des annonces et des cours de la même manière. Il est alors possible par-exemple d'obtenir le titre du cours de l'utilisateur par la méthode *getTitle()*. Pour vous déconnecter, il suffit de faire appel à la méthode *closeSession* de l'interface Session.


<<Java Interface>>
Session
api
closeSession():void
getUserData():UserData
getCourses():Courses
getUpdates():Updates
<<Java Interface>>
Updates
api
getUpdatesTab():Update[]
nbUpdates():int
<<Java Interface>>
Courses
api
nbCours():int
getAllCoursesID():int[]
getAllCoursesSyscode():String[]
getCourse(int):Course
getCourse(String):Course
<<Java Interface>>
UserData
api
getUserId():int
getUsername():String
getFirstName():String
getLastName():String
getLanguage():String
isCourseCreator():boolean
isPlatformAdmin():boolean
getOfficialCode():String
<<Java Interface>>
Update
api
setSysCode(String):void
setTagModule(String):void
setResID(String):void
setDate(String):void
getSysCode():String
getTagModule():String
getResID():String
getDate():String
<<Java Interface>>
Course
api
getCoursID():int
getSyscode():String
getDirectory():String
getOfficialCode():String
getDb():String
getTitle():String
getTitular():String
getLanguage():String
getAccess():String
getStatus():String
getSourceCourseId():String
getExpirationDate():String
getCreationDate():String
isCourseManager():int
getOfficialEmail():String
notified():boolean
getCourseTool():CourseTool
getAnnonces():Annonces
getDocs():DocsAndFolders
<<Java Interface>>
DocsAndFolders
api
getSysCode():String
nbDocsAndFolders():int
nbDocs():int
nbFolders():int
getFoldersName():String[]
getDocsName():String[]
getFolder(String):Folder
getDoc(String):Doc
<<Java Interface>>
CourseTool
api
getSysCode():String
isAnn():boolean
isDnL():boolean
annNotif():boolean
dnlNotif():boolean
<<Java Interface>>
Annonces
api
getSysCode():String
nbAnnonces():int
getAllAnnoncesID():int[]
getAllAnnoncesTitle():String[]
getAnnonce(int):Annonce
getAnnonce(String):Annonce
<<Java Interface>>
Doc
api
getName():String
getExtension():String
getUrl():String
getPath():String
getDescription():String
getSize():int
getDate():String
getType():int
isVisible():boolean
notified():boolean
<<Java Interface>>
Folder
api
getName():String
getPath():String
getDescription():String
getDate():String
getType():int
isVisible():boolean
notified():boolean
<<Java Interface>>
Annonce
api
getRessourceID():int
getTitle():String
getContent():String
visibleFrom():String
visibleUntil():String
isVisible():boolean
getRank():int
getDate():String
notified():boolean