TP Channel Messaging

But du TP : Création d'une application de messagerie instantanée fonctionnant par channel

I. Création de l'application

Dans Android studio, sélectionnez « New project ».

Application name : ChannelMessaging

Company domains : nom.prenom

Pour phone and tablet avec minimum SDK 13.

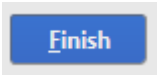
Avec une Blank Activity.

Activity name : LoginActivity

Layout name : activity_login

Title : Connexion

Menu Resource Name : menu_login



II. Connexion

A. Le layout

Ajoutez dans votre layout 2 Textview qui seront le titre des 2 EditText.

La première Textview servira pour l'identifiant, la deuxième pour le mot de passe.

Ajouter un Button nommé « Valider » en bas du layout.

En s'appuyant sur les linearlayout avec orientation horizontal/vertical et le poids de leur contenu, votre layout devra ressembler à ceci ->

B. La classe LoginActivity

Définir dans la classe des attributs de classe correspondants aux layout puis les remplir avec le layout.

Créer un événement personnalisé s'occupant de gérer le webservice (Utiliser CustomEvents, AsyncTask, HttpPost des android cheat sheet, (autres détails ci-dessous))

Pour cela, créer d'abord une interface OnDownloadListener contenant une méthode « onDownloadComplete(String downloadedContent) » et une méthode « onDownloadError(String error) ».
Créer ensuite une classe HttpPostHandler qui contiendra une arraylist de listeners. (avec une méthode addOnDownloadListener)
Cette classe héritera de l'AsyncTask. Gérer dans la méthode doInBackground l'appel POST.
Récupérer le résultat dans le onPostExecute puis le renvoyer au listeners.
Tous les détails concernant le webservice sont disponibles à cette adresse :
http://raphaelbischof.fr/messaging/
Vos identifiants de connexion sont :

- Identifiant = première lettre du prénom + 4 premières lettres du nom (exemple rbisc, si moins de 4 lettres, toutes les lettres du nom)
- Mot de passe = prénom complet en minuscule + nom complet en minuscule (exemple raphaelbischof)

Utiliser la classe HttpPostHandler pour gérer l'appel webservice qui permet de se connecter.
Utiliser la librairie GSON pour désérialiser les différentes réponses du webservice

Stocker l'access token dans les SharedPreferences.

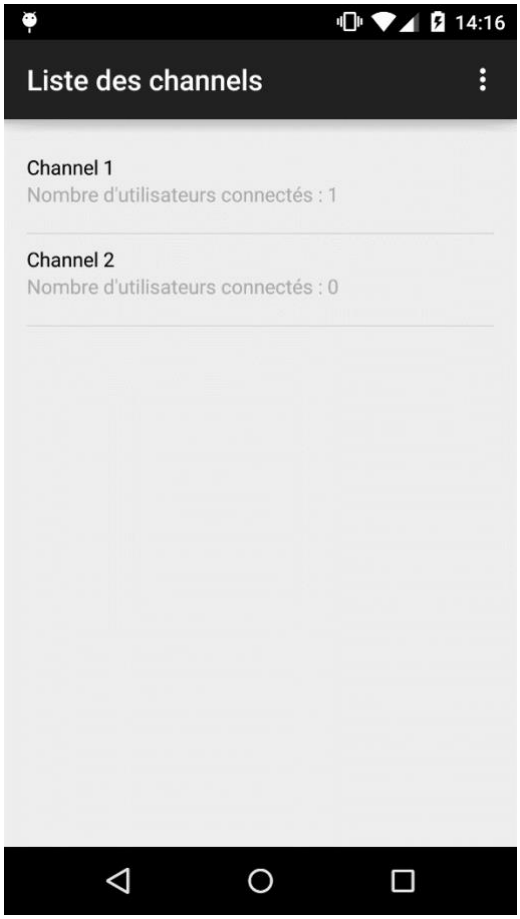
Faire que le clic sur le bouton fasse cette appel web service, en cas de réussite, il démarrera une nouvelle activité « ChannelListActivity ». En cas d'échec, il affichera un Toast d'erreur (Info ici)

III. Liste des channels

Afficher les différents channels dans une Listview comme dans la capture d'écran ci-dessous :
(Méthode du WS utile : getchannels)

Récupérer l'accesstoken dans les sharedpreferences pour l'envoyer au WS.

Faire que le clic sur un item de la listview démarre une nouvelle activité « ChannelActivity » qui s'occupera d'afficher les messages du channel. Dans l'intent qui démarrera l'activité prendra en extra le channelID que l'on souhaite voir.



IV. Affichage/envoi de messages

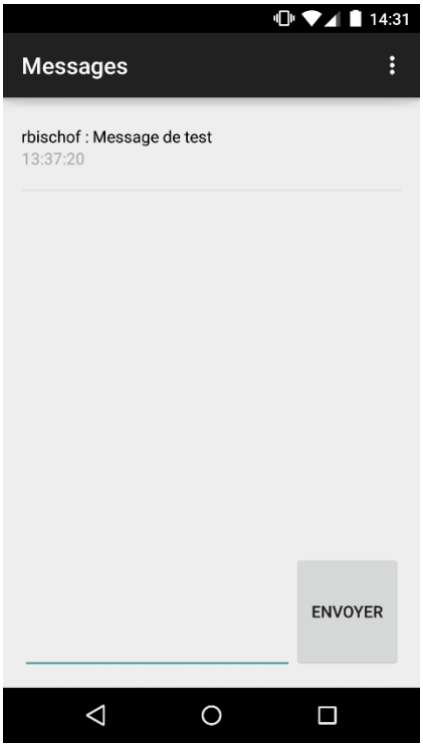
Dans l'activité ChannelActivity, récupérer le channelID de l'intent (getIntent()).

L'utiliser pour récupérer les messages à intervalle de temps régulier de 1 secondes. (Utiliser pour cela un Runnable)

Remplir le layout comme la capture d'écran ci-jointe.

(Un relative layout contenant une listview et un linearlayout horizontal, ce linearlayout contenant un edittext et un bouton)

Faire que lors d'un appui sur le bouton, le texte soit envoyé.



V. Amélioration de l'affichage des images

Modifier la listview pour qu'elle puisse aficher les images des personnes ayant écris un message.

(CustomAdapter, Chargement des images dans une asynctask, génération d'un événement lorsque l'image est chargé qui met à jour l'imageview)

Afficher cette image de manière circulaire en s'aidant du code disponible ici.