Bonjour Michel voice un petit résumé qui va t’aider a bien plonger dans l’app et comprendre son fonctionnement. En espérant que la transition soit sans embuches.

Si tu as des questions je suis pas loin lache un texte.

J’ai une bonne base de couler même s’il manque plusieurs chose elle seront facile a faire car tu n’as qu’a utiliser ce qui est déja la

Des que tu as des services fonctionels , parcours toutes les classes à la recherché de TODO car j’ai hard coder quelques objects ici et la afin de pouvoir travailler avec quelques données.

# Ce qui est fait :

la connection : quand un utilisateur se connecte , le drawer affiche toute les options et affiche quelque info sur l’utilisateur. La page de profile est afficher.

La déconnection : quand il se déconnecte, le drawer retire les options non disponible et propose une option pour se connecter .L’application est redirigé vers l’accueil.

Se rapel des session : si un utilisateur connecté quitte l’application, la reconnection se fait automatiquement au prochain lancement. Si l’utilisateur se déconecte , il doit reentrer ses identifiant a la prochaine réouverture du programe. Si le mot de passe a change alors que l’application est fermé, il devra se reconnecter.

La creation des utilisateur devrait marcher. Je n’ai pas pu tester car il manquait des services

# Ce qui est a faire :

Programmer le core de “ObjectConvertor.stringBas64ToBitMap()”

Changer les appels au service pour se conformer a ce que tu as fait.

Blocquer l’orientation en mode portrait CRUTIAL sinon bug

Gérer les fonctions 14 a 18

# Recycler view :

affin d’être modulaire et de gérer toutes les listes, 2 classe travaille en tandem :

* “ListRecycler extends RecyclerView.Adapter”
* “ListControler”

Il faut dire au controleur quelle est notre action.

Puis on initie le Recycler avec une liste d’object

L’adapteur adapte automatiquement son contenu pour :Avatar, ListeLecture et Musique.

Le principe est modulaire on pourait ajouter d’autre type d’object si besoin.

Quand un element quelquonque est sélectionner , le controleur traite la demande en se référant a l’action.

Cela peu paraitre abstrait mais voici un example concret d’utilisation avec des object de type XX:

fragmentExigantUneListe. onCreateView(){

**activity**.**listControler**.**action** = ListControler.***ACTION\_XX*** Indiquer l’action déclanché ( int )  
**recyclerView** = (RecyclerView) **view**.findViewById(R.id.***id\_du\_recycler\_view***); indiquer dans quell Xml   
**recyclerView**.setLayoutManager( **new** LinearLayoutManager( **activity**.getApplicationContext()) ); //default, obligatoire  
**recyclerView**.setItemAnimator( **new** DefaultItemAnimator() ); //default, obligatoire   
**recyclerView**.setAdapter(**new** ListRecycler( (List<Object>)(List<?>) **listXX** , **activity** ) ); passer une listeXX

ListControler.onDoneSelectingXX( XX pObjectXX) {  
 **switch** (**action**) {  
 **case *ACTION\_XX***

**activity**.**controlerXX**.onActionXX(pObjectXX) Redirige l’object select. vers le bon controleur

# Initiation de Fragment

Beaucoup de mesure de sécurité sont en place affin d’éviter que l’app crash. Il fau savoir que certains fragment ont des conditions pour s’afficher correctement

Certains fragment ont besoin d’effectuer plusieur requête pour remplir leur contenu. C’est le cas par example de “fgrPlaylist” voici comment l’initiencer.

Mise en situation :Une playlist vient d’etre sélectionner par l’utilisateur et elle lui appartient

**public void** onDoneSelectingMyPlaylist(ListesDeLecture pPlaylist) {

**fgPlaylist** = **new** fgrPlaylist();  
**fgPlaylist**.setPlaylist(pPlaylist);  
**fgPlaylist**.setCanEdit(**true**);  
**dataMgr**.getSongListForPlaylist( pPlaylist.getId() );

attendre que les chancon apartenant a la playlist arrive

…

…

**public void** onGetSongListForPlaylistAnswer(List<Musique> pListSong) {

**fgPlaylist**.setListSong(pListSong);  
**activity**.changeFragment( **fgPlaylist** );

Voila ce fragment est maintenant initié et affiché.

Comme tu peu le remarquer fgrPlaylist a besoin que 3 paramêtre soit setter avant de pouvoir être afficher

setListSong(List<Musique>){  
 setCanEdit(**boolean**}

Si un fragment appellee “onCreateView” et que tout les parameter nécessaire a sa creation sont manquant un message d’erreur précis est loggé et une page blanche est affiché.

Log.*e*(**"fgrPlaylist.onCrtView()"**,**"these MUST be set before displaying fragment --> playlist=SET , listSong=UNSET , canEdit=UNSET "** ) );