TP Channel messaging - GPS

But du TP : Gérer la récupération des coordonnées GPS dans une appli, les réutiliser pour afficher une carte

I. Récupération et utilisation du service de localisation

A. Installation des google-play-services

La librairie utilisée pour récupérer les coordonnées GPS est la librairie Google Play services.
Pour installer cette lib, allez dans **File**, **Project structure**.
Sélectionnez ensuite votre module dans la liste de gauche. Puis allez dans l'onglet « **Dependencies** ».
Appuyez sur le « **+** » situé à droite des dépendances et sélectionnez « **Library dependency** ».
Sélectionnez « **play-services (***)** » puis cliquez sur Ok.

(Remarque : Il est possible d'accéder directement au matériel sans utiliser les google play services, cette méthode est néanmoins désormais deprecated car elle est plus coûteuse en énergie et ne permet pas de récupérer la localisation la plus récente enregistrée)

B. Mise en place des google-play-services

Ajoutez à l'android Manifest la permission pour accéder à la localisation précise (**ACCESS_FINE_LOCATION**) et celle pour accéder à la localisation approximative (**ACCESS_COARSE_LOCATION**).

Créez une nouvelle classe **GPSActivity** qui héritera d'**ActionBarActivity**.

Ajoutez un attribut **mGoogleApiClient** que vous initialiserez dans un override de la méthode **onCreate**.
Pour initaliser le GoogleApiClient, il vous faudra utiliser la classe **GoogleApiClient.Builder**, définir à ce builder les callbacks (évenements) de réussite de connexion et d'échec de connexion (**addConnectionCallbacks** et **addOnConnectionFailedListener**), l'API utilisée (**addApi(LocationServices.API)**) et enfin d'appeler la méthode « build » du builder.
Connectez vous ensuite via la méthode **connect** de **mGoogleApiClient**.

C. Récupération de la dernière coordonnée GPS reçue

Dans le cas où la google api client s'est bien connecté, il vous est possible de recevoir les dernières coordonnées reçues via

```
Location lastLocation = LocationServices.FusedLocationApi.getLastLocation(
                mGoogleApiClient);
```

Attention : Bien penser à vérifier si la google api client est bien connectée via mGoogleApiClient.isConnected()

D. Demander des mises à jour des coordonnées GPS

Pour récupérer de nouvelles coordonnées GPS, il faut créer une demande de mise à jour via la classe LocationRequest que l'on utilisera avec les locations services. Cette demande se fera dans le cas où la google api client s'est bien connectée aussi.

```
    LocationRequest mLocationRequest = new LocationRequest();
    mLocationRequest.setInterval(10000);
    //Correspond à l'intervalle moyen de temps entre chaque mise à jour des
coordonnées
    mLocationRequest.setFastestInterval(5000);
    //Correspond à l'intervalle le plus rapide entre chaque mise à jour des
coordonnées
    mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
    //Définit la demande de mise à jour avec un niveau de précision maximal
    LocationServices.FusedLocationApi.requestLocationUpdates(
            mGoogleApiClient, mLocationRequest, this);
```

Pour plus de détails sur la priority, voir ici.

this dans ce cas-là devra être capable d'écouter les événements de l'interface **LocationListener**.

E. Arrêter la demande de mise à jour des coordonnées GPS

Pour arrêter la demande de mise à jour des coordonnées GPS, il vous suffit d'utiliser la méthode **removeLocationUpdates** de la classe **FusedLocationApi.** Afin d'éviter de consommer de la batterie inutilement, cette méthode devra être utilisée dans un override de la méthode **onStop**.

```
    LocationServices.FusedLocationApi.removeLocationUpdates(
            mGoogleApiClient, this);
```

F. Gestion des cas spécifiques

En prenant en compte le cycle de vie d'une activity, il est possible que l'on réaffiche l'activity sans même passer dans le **onCreate**. Il est donc probable que la variable **mGoogleApiClient** ne lève pas l'évenement **onConnected**. C'est pourquoi il faut ajouter dans un override de la méthode **onResume** une demande de mise à jour des coordonnées GPS.
Il faudra toutefois vérifier si une demande de mise à jour des coordonnées GPS n'est pas déjà active. Pour ce faire, utiliser un boolean qui aura en valeur par défaut false et qui passera à true lors d'une demande de mise à jour. Ce boolean repassera donc à false lors de l'arrêt de la demande de mise à jour.

G. Stockage des coordonnées et héritage de la GPSActivity

A chaque **onLocationChanged**, stocker la valeur dans une variable d'instance **protected** nommée **mCurrentLocation**.

Faire hériter l'activité de liste des channels et l'activité des messages de la GPSActivity nouvellement créée.

Dans le cas de l'utilisation d'un émulateur, il sera nécessaire de passer par l'android device monitor pour pousser de fausses coordonnées GPS (Tools, Android, Android device monitor).

H. Utilisation des coordonnées

Ajouter lors de l'appel webservice **sendMessage** les coordonnées gps dans le cas où celles-ci sont différentes de **null** (voir descriptif webservice).

I. Récupération des coordonnées des messages

Lors de la méthode **getMessages** les coordonnées GPS ont été ajoutées, ajoutez donc à votre objet de désérialisation les propriétés **latitude** et **longitude**.

II. Affichage d'une carte

Pour intégrer une carte dans une appplication, il est aussi nécessaire d'avoir la librairie google-play-services.

A. Création de l'activity de la carte

Faire clic droit sur le package contenant vos activités, puis New, Google, MapsActivity.
Remplir les champs avec comme nom d'activité « MapActivity », puis laisser tel quel.
Android studio fait alors :

- Le layout de l'activity qui contient un MapFragment
- L'activity MapActivity qui instancie la GoogleMap
- Un fichier google_maps_api.xml dans le dossier debug qui contient les informations pour créer la carte
- Des modifications dans le manifest pour récupérer les informations nécessaires du MapFragment

B. Récupération de la clef google map

Connectez-vous avec votre compte google.

Suivez ensuite le lien noté dans le fichier google_maps_api.xml. Celui-ci permettra d'obtenir une clef de la part de Google qui autorisera l'affichage de la carte avec votre signature d'application en debug.
Attention, bien vérifier que le nom du package qui se situe dans l'android manifest (<manifest …. Package= XXX) corresponde au nom du package de l'url.

Pour information : il faudra donc obtenir aussi une clef dans le cas d'une release de l'application. Cette clef devra être créée de telle manière : empreinte SHA1 de votre signature **;**nom du package

C. Démarrage de la map activity

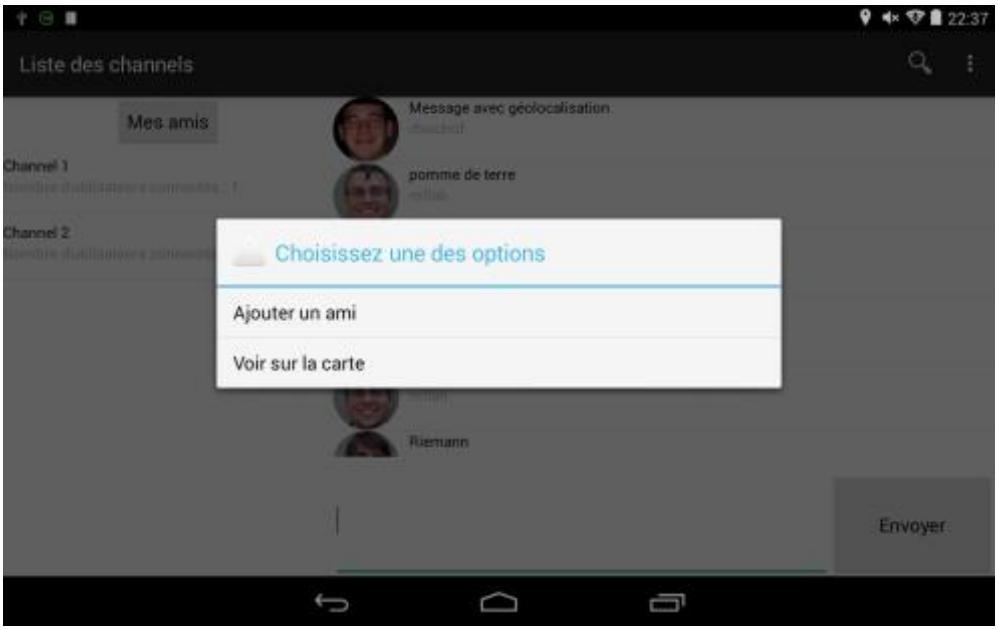
Lors du touch sur un des messages du MessageFragment, afficher un AlertDialog (cf AlertDialog plus bas) proposant :

- Soit l'ajout en ami du contact
- Soit l'affichage de l'emplacement d'écriture du message

L'application devra alors démarrer la MapActivity ou ajouter l'ami.

Le webservice répond 0 en latitude et longitude si les messages reçus n'ont pas de coordonnées GPS lors de leur création.

```
new AlertDialog.Builder(getActivity())
     .setIcon(android.R.drawable.ic_dialog_alert)//drawable de l'icone à gauche du titre
     .setTitle(R.string.make_a_choice)//Titre de l'alert dialog
     .setItems(arr, new DialogInterface.OnClickListener() {
         @Override
         public void onClick(DialogInterface dialog, int which) {//which = la position de
l'item appuyé
             if (which == 0) {
                 //Do some stuff (1st item touched)
             } else {
                 //Do some over stuff (2nd item touched)
             }
         }
     })//items de l'alert dialog
     .show();
```



D. Affichage d'un marker sur la carte

Modifier dans la méthode setUpMap autogénérée le marker pour qu'il corresponde au marker de notre message. Mettre en titre l'username de la personne qui a écrit le message.

E. Centrage de la carte sur le marker

Créer une instance de la classe **LatLng** qui contiendra les coordonnées.

Créer une instance de la classe **CameraPosition** qui prendra en paramètres l'instance de **LatLng**, le zoom level (0 -> Vue totalement dézoomée, 19 -> Zoom le plus fin, dans notre cas nous utiliserons un zoom level de 14), l'inclinaison de la carte (dans notre cas 0), et l'angle de la carte par rapport au nord (dans notre cas 0).

Créer une instance de la classe CameraUpdate via la méthode CameraUpdateFactory.newCameraPosition(CameraPosition cameraPosition).

Déplacer la caméra via la méthode monInstanceDeLaGoogleMap.moveCamera(CameraUpdate cameraUpdate).