Rapport Projet Android Locate Wind

DUT INFORMATIQUE

IUT BELFORT-MONTBELIARD

Réalisé par: Comtois Anthony- Yves Darosey

Sommaire

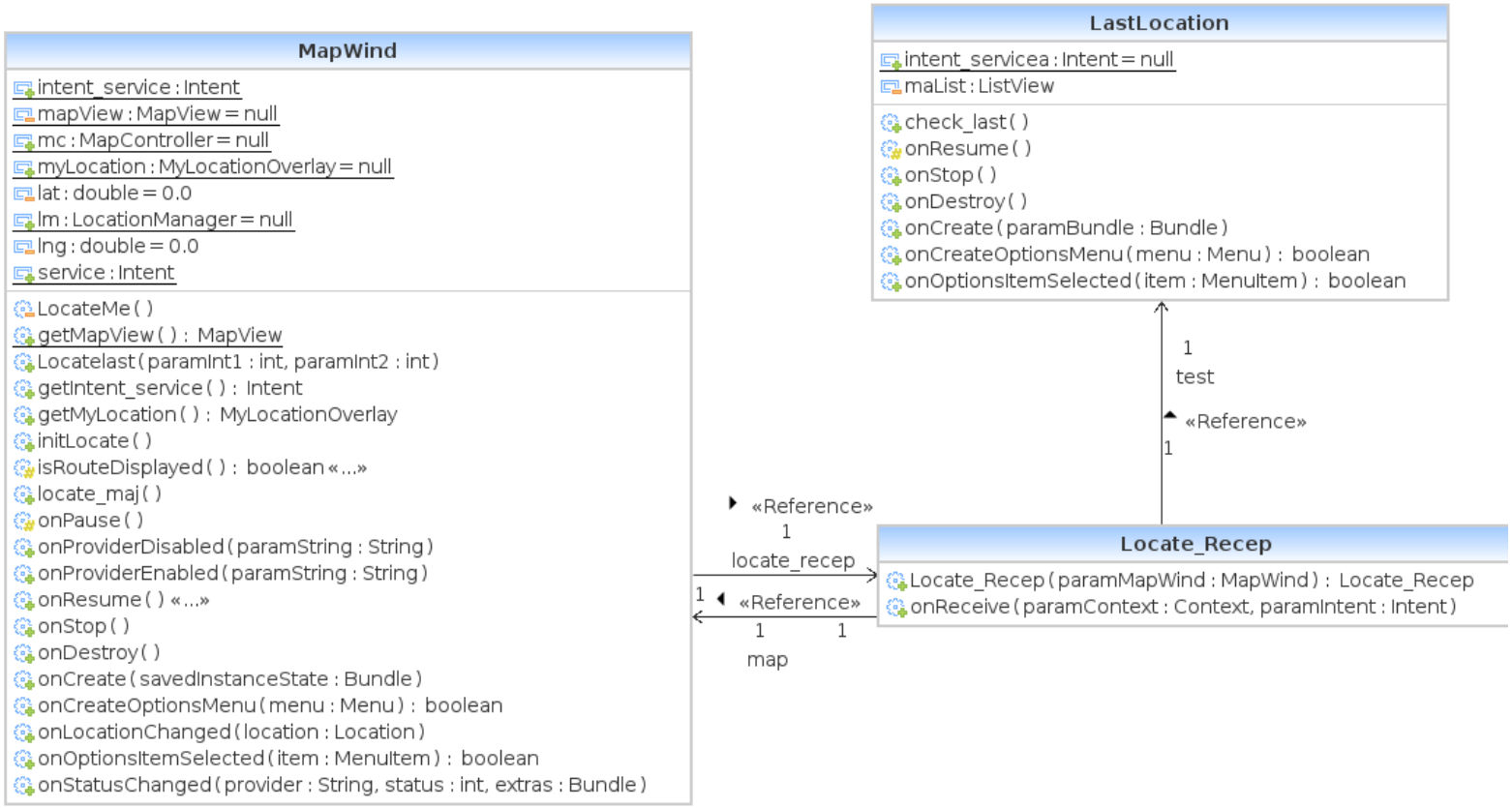
1.Introduction

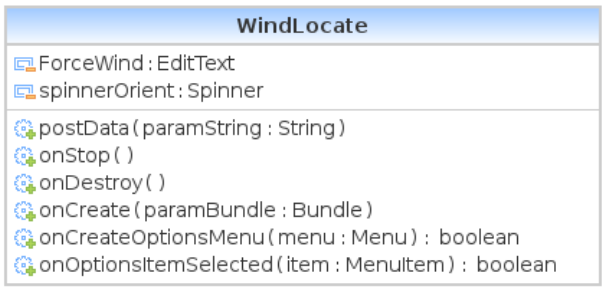
Lors du 4 ème semestre du DUT de Belfort-Montbéliard, il nous a été demandé de réaliser une application de collaboration pour échanger la force des vents ainsi que l'orientation. Nous pouvions nous inspirer de l'application à Bon Entendeur.

L'application Locate Wind cible toute personne désirant faire une activité liée au vent. La principale fonctionnalité de Locate Wind consiste à fournir une carte avec des indications qualitatives des vents.

Ce rapport a pour but d'expliquer comment l'application fonctionne, et rendre compte du travail effectué.

2.Diagramme de classe


MapWind
intent_service : Intent
mapView : MapView = null
mc : MapController = null
myLocation : MyLocationOverlay = null
lat : double = 0.0
lm : LocationManager = null
lng : double = 0.0
service : Intent
LocateMe ( )
getMapView ( ) : MapView
Locatelast (paramInt1 : int, paramInt2 : int)
getIntent_service ( ) : Intent
getMyLocation ( ) : MyLocationOverlay
initLocate ( )
isRouteDisplayed ( ) : boolean «...»
locate_maj ( )
onPause ( )
onProviderDisabled (paramString : String)
onProviderEnabled (paramString : String)
onResume ( ) «...»
onStop ( )
onDestroy ( )
onCreate (savedInstanceState : Bundle)
onCreateOptionsMenu (menu : Menu) : boolean
onLocationChanged (location : Location)
onOptionsItemSelected (item : MenuItem) : boolean
onStatusChanged (provider : String, status : int, extras : Bundle)
LastLocation
intent_servicea : Intent = null
maList : ListView
check_last ( )
onResume ( )
onStop ( )
onDestroy ( )
onCreate (paramBundle : Bundle)
onCreateOptionsMenu (menu : Menu) : boolean
onOptionsItemSelected (item : MenuItem) : boolean
1
test
«Reference»
1
«Reference»
1
locate_recep
1
«Reference»
1
1
map
Locate_Recep
Locate_Recep (paramMapWind : MapWind) : Locate_Recep
onReceive (paramContext : Context, paramIntent : Intent)


WindLocate
ForceWind : EditText
spinnerOrient : Spinner
postData (paramString : String)
onStop ( )
onDestroy ( )
onCreate (paramBundle : Bundle)
onCreateOptionsMenu (menu : Menu) : boolean
onOptionsItemSelected (item : MenuItem) : boolean


WindItem
arrayListOverlayItem : ArrayList = new ArrayList() «...»
context : Context
createItem (paramInt : int) : OverlayItem «...»
onTap (index : int) : boolean «...»
size ( ) : int «...»
WindItem (paramDrawable : Drawable) : WindItem
WindItem (paramDrawable : Drawable, paramContext : Context) : WindItem
addOverlayItem (paramOverlayItem : OverlayItem)

Diagramme de classe:

Locate_WindActivity est l'activité principale lançant l'application, elle hérite de TabActivity pour pouvoir créer des onglets. Ces onglets ont tout d'abord été créer simplement via une TabHost dont on lui ajoute des onglets,voir documentation android avec un exemple très simple.

Ces onglets par défaut comporte du texte et une image qu'il est possible de personnaliser ou de ne pas afficher, seulement si l'on n'affiche pas d'image dans l'onglet, celui-ci prend quand même toute la place du texte et de l'image, ce qui perd énormément de place au niveau de l'écran. C'est pour cela que j'ai décidé de créer une vue personnalisé via un layout tab_indicator.xml celui-ci comporte un RelativeLayout avec un TextView à l'intérieur. La place est donc optimisée pour une seule ligne de texte ce que fait gagner de la place sur l'écran tout en gardant l'ergonomie.

Locate_WindActivity lance 3 activitées :

-MapWind.java: affichage de la map
-WindLocate.java: enregistrement de sa position
-LastLocation.java: dernières positions enregistrées

Ces 3 activités seront détaillé plus loin, l'activité principale implémente la classe GestureDetector. Celle-ci va permettre de déclancher des actions suivant les gestes de l'utilisateur, dans le cas présent elle est utilisé pour changer d'onglets via un "swipe", mouvement horizontale de droite à gauche ou inversement.

Celui-ci marche parfaitement avec un léger détail, la map google et la listView du 3eme onglet prennent toute la place sur la vue il faut donc commencer le geste bien au bord de l'écran alors que l'onglet 2 ne pose aucun problème (fonction à tester de préférence sur un téléphone, l'émulateur n'étant pas très réactif).

Une dernière fonctionnalité de cette activité, est la vérification de l'activation du GPS, s'il n'est pas activé un popup s'affiche demandant si l'utilisateur veux l'activer. S'il accepte il est redirigé vers la page activation du GPS dans les paramètre systèmes de l'OS, il coche la case activation puis appuie sur son bouton retour du téléphone et revient sur l'application. S'il refuse l'application continuera sans le GPS et géolocalisera l'utilisateur par triangulisation via le réseau, ce qui est un peu moins précis mais marche tout aussi bien.

3.Map

Lors du lancement de l'application on arrive sur la vue MAPS que google nous fournit via l'API Google, ce projet utilise la version 2.3.3. Cette API permet entre autres d'utiliser MAPS, pour utiliser celle-ci nous devons générer une clé API via un site de Google pour pouvoir l'utiliser, après avoir effectué un checksum MD5 créé grâce au debug certificate pour garantir l'authenticité.

Cette vue gère la map, avec une vue satellite, on récupère les coordonnées par GPS ou triangulisation voir plus haut. l'utilisateur est centré sur sa position il peut ensuite se balader sur la carte et cliquer sur les marqueurs. S'il clique sur un marqueur un popup s'affichera avec la force du vent en noeuds, l'orientation, l'heure et la date.

L'utilisateur peut appuyer sur menu (touche du téléphone), 3 options lui seront proposé:

-Me: permet de revenir à sa position sur la map
-Update: permet de forcer la synchronisation avec le serveur
-Quitter: permet de mettre fin à l'application

Ensuite un service ou démon en jargon système, est lancé via le onCreate() donc au lancement de la vue. Celui-ci lance la classe RecupService.java va créer un timer via RecupTimerTask. Ce qui va permettre d'exécuter le service de façon périodique. RecupTimerTask appelle une méthode de RecupService qui va créer un thread. Celui-ci va lui-même lancer une méthode locate_maj() qui va permettre de récupérer les informations dans une base de données, la partie serveur sera expliqué plus tard.

Cette méthode envoie une requête HTTP à un serveur, elle récupère une réponse sous le format JSON, JavaScript Object Notation, modèle structuré se rapprochant du principe du XML avec une syntaxe encore plus légère. Ensuite la réponse est converti en String, puis est parsé via JSONObject ce qui permet de récupérer toutes les données de la réponse.

Ces données sont stockées dans un arraylist de type static d'objet de type Wind, la classe Wind stock des objets contenant la date, l'heure, l'orientation, la force, la latitude et la longitude. Celui-ci est ensuite utilisé par la classe Map_Wind pour peupler la map.

La classe WindItem permet de créer des marqueurs, ainsi qu'une méthode onTap qui affiche le popup avec les informations comme dis précédemment.

Le service lancé pour récupérer les données du serveur s'arrête quand on quitte l'application ou qu'on la met en pause (multi-tache). Et est redémarré dans le onResume.

4.Locate

L’ activité Windlocate, affiché dans le 2 ème onglets, permet de rentrer la force du vents en noeuds ainsi que l’orientation, puis de valider. La méthode postData récupère ses informations ainsi que le latitude et la longitude et les envoie par requête HTTP en POST. Le serveur récupère les données et les insère dans la Base De Donnée. (sur certains téléphone lors du clique dans le champ pour rentrer la force du vent, le clavier ne s’affiche pas, dans ce cas la choisir l’orientation avant puis revenir dans le champ pour rentrer la force du vent).

Pour cette vue au niveau du menu seul l’option quitter est accessible.

5.Last

L’activité LastLocation recense les dernières entrées des utilisateurs dans la base de données sous forme de listView, d’une liste, lorsque que l’on clique sur un des éléments de la liste l’utilisateur est renvoyé sur la map en onglet 1, et centrer sur la position cliquer ce qui permet de savoir automatiquement où se situe les derniers vents rentrés. Ce qui rend l’application plus conviviale. Une autre amélioration que je n’ai pas eu le temps de terminer aurait été de récupérer l’adresse suivant la latitude et la longitude et l’afficher dans la listView. L’utilisateur aurait donc tous les éléments pour savoir où le vent énoncé se situe sans aller voir la carte.

Pour cette vue au niveau du menu seul l’option quitter est accessible.

6.Serveur

Au niveau de la sauvegarde des données, j'avais commencé à enregistrer les données dans un fichier sur le téléphone, pour ensuite les envoyées sur les serveurs Google en les sauvegardant en tant que préférence mais je me suis vite rendu compte que l'usage des préférences n'étaient pas approprié pour un partage de données entre différents terminaux.

J'ai donc mis en place sur mon serveur personnel, une base de donnée ayant une table Locate, et qui contient id_locate latitude longitude force orientation date.
Les champs étant explicite, je rajouterai que date est un datetime c'est-à-dire qu'il contient l'heure et la date et qu'il y a juste à splitter en java pour récupérer ces données séparément.
La latitude et la longitude sont des entiers puisque qu'avant de les envoyer je les multiplie par 10^6 puisque l' API Google fonctionne comme ceci. La force du vent est un entier relatif et l'orientation est une chaine de caractères.

J'ai utilisé PHP5, donc avec de l'objet comme dans l'exemple du court pour l'envoi de message, mon code reprend le même principe. Je crée un objet Wind que j'insère dans la base de donnée. Et pour envoyer les données à l'application j'encode la réponse de la requête SQL en JSON, et je le décode dans la vue MapWind. L'application est donc accessible depuis n'importe quel téléphone, le serveur étant toujours allumé et n'est pas le serveur le l'IUT.

7.Conclusion

Dans cadre des Travaux Pratiques nous nous sommes initiés à la programmation Android. Puis nous avons pu réaliser une application qui ravira toutes les personnes exerçant une activité lié au vents.