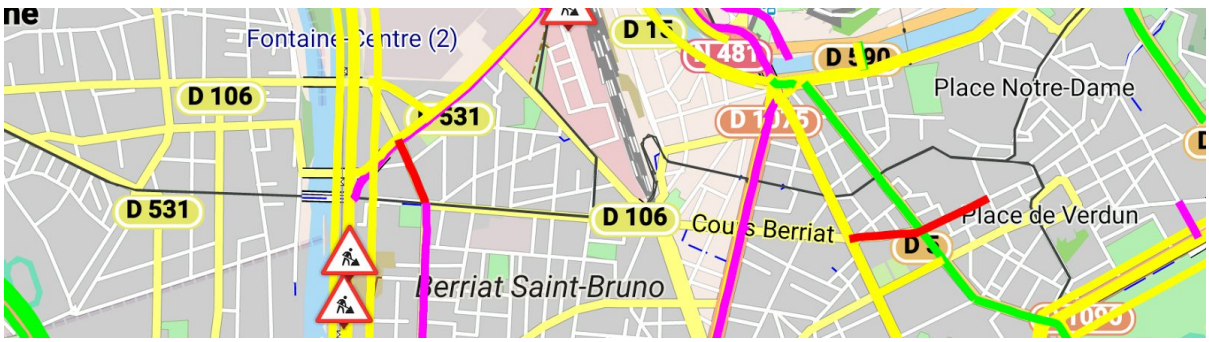
POLYTECH GRENOBLE

Rapport
Grenoblois Futé



DELAPORTE Adrien - MOURET Adrien - LUCIDARME Benjamin

Sommaire

I°/ Présentation du projet

A°/ Contexte

OsmAnd est une application mobile de navigation GPS. Elle permet la navigation hors-ligne grâce aux données OpenStreetMap. Il est également possible d'utiliser des tuiles ou des services en ligne.

B°/ But

Le but de notre projet est de réaliser un Plugin pour OsmAnd permettant d'afficher le trafic en temps réel au sein de l'agglomération Grenobloise. Il affichera également les travaux présents sur les routes de l'agglomération. Le plugin réalisé devra s'intégrer dans l'application déjà existante et utilisera les données Open Sources de la Métromobilité pour indiquer le niveau de trafic.

II°/ Réalisation

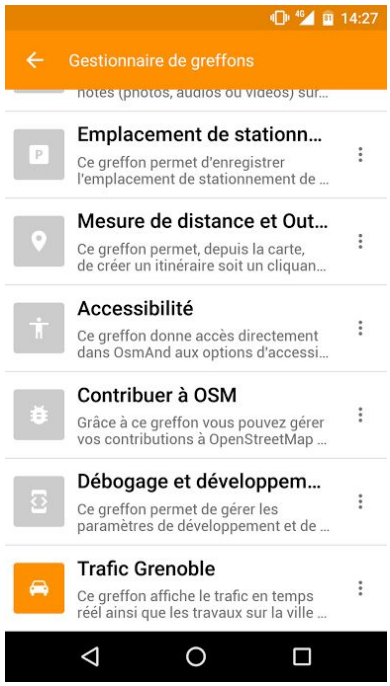
A°/ Création du module

Le module ne se crée pas à l'extérieur d'Osmand, contrairement à ce que l'on pourrait penser. En effet, tous les codes de tous les plugins se trouvent directement dans le code source de Osmand. Cependant, tous les plugins ne s'affichent pas dans le gestionnaire de greffons, et il faut télécharger un petit module d'à peine 50 lignes, parfois payant, afin de l'afficher. Pour créer un greffon à Osmand, il faut donc créer un package à l'intérieur d'Osmand ainsi qu'un petit module qui permet d'activer le greffon.

Ainsi, on crée un dossier dans le package net.osmand.plus, qui contiendra toutes les classes nécessaires à la réalisation du plugin. On crée ensuite la classe principale du plugin, Trafic.java, qui hérite de OsmandPlugin. On rajoute ensuite la ligne:

```
checkMarketPlugin(app, new Trafic(app), false, Trafic.COMPONENT, null);
```

dans le fichier "OsmandPlugin.java" pour créer un bouton d'activation du plugin dans le gestionnaire de greffons sur l'application. L'image suivante montre le résultat:



B°/ Parsers

1/ *JSON / GEOJSON*

Le parser GEOJSON récupère des données statiques sur le site de la métromobilité pour avoir les coordonnées des routes. Ces coordonnées sont associés à un code et un niveau d'embouteillage nul. Le parser JSON complète le précédent en récupérant des données dynamiques qui associent un niveau d'embouteillage réel pour chaque code de tronçon de route. On peut ensuite afficher une couleur en fonction du niveau d'embouteillage sur le tronçon dont nous avons les coordonnées statiquement.

2/ *XML*

Le parser XML récupère les données des travaux de l'agglomération sur le site de la métromobilité. Il permet de récupérer plusieurs informations dont les latitudes et longitudes des emplacements de travaux. On peut ensuite afficher, grâce à la classe TraficLayer, l'icône de travaux aux emplacements récupérés.

C°/ Affichage sur OsmAnd

1/ *Couche de carte - Layer*

Les layers permettent de dessiner sur la carte de Osmand. La classe TraficLayer hérite de OsmandMapLayer et implémente les fonctions initLayer(...) et onDraw(...) ou onPrepareBufferImage(...). Dans notre cas, nous initialisons les variables et nous récupérons les données de la métromobilite dans la fonction initLayer(...), afin d'avoir l'assurance que nous ne récupérons les données une seule fois. Ensuite, nous dessinons dans la fonction onPrepareBufferImage(...).

L'objet TraficLayer doit être appelé dans la classe principale du plugin, Trafic.java, dans la fonction registerLayers(...). Une fois l'objet créé, il doit être ajouté aux autres layers. Sa place parmi les autres layers est définie par un entier. Dans notre cas, nous utilisons l'entier "12" pour que notre layer soit toujours supérieur aux 11 layers de bases.

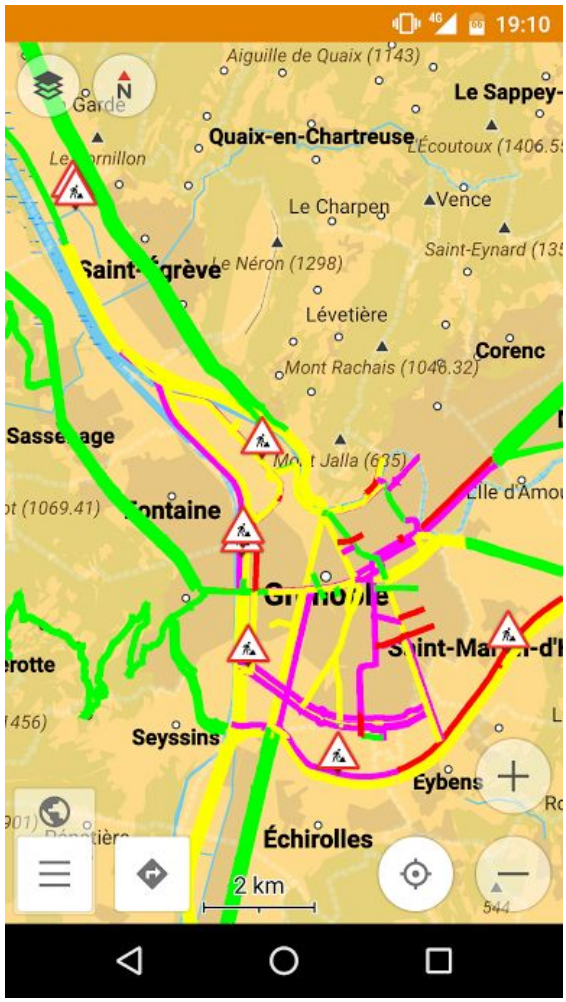
```
traficlayer = new TraficLayer(app);
activity.getMapView().addLayer(traficlayer, 12);
```

2/ *Style de carte - Render*

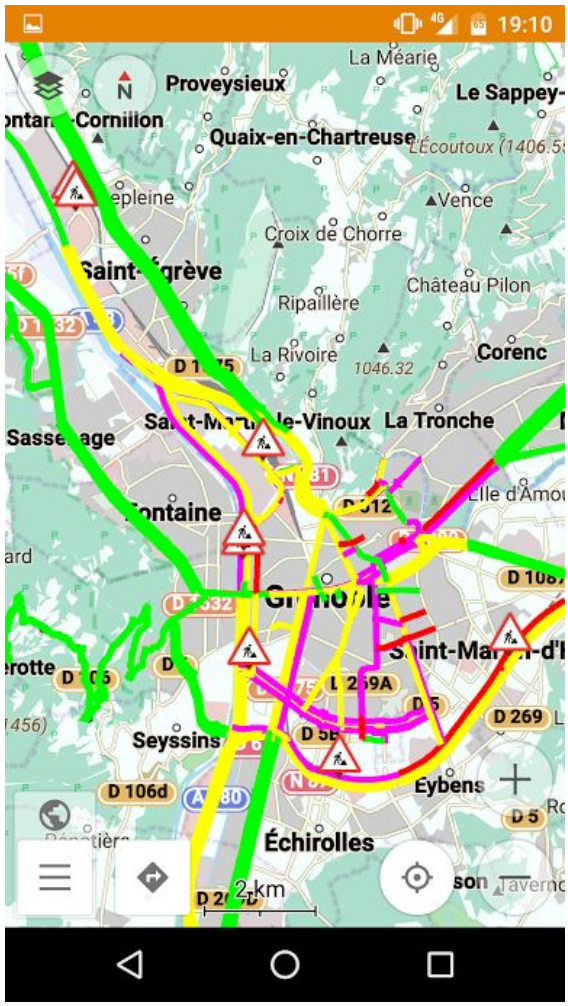
Il est possible, sur Osmand, de créer son propre style de carte. En effet, il suffit d'enregistrer son propre style de carte dans la classe RendererRegistry.java :

```
public final static String TRAFIC_RENDER = "Trafic"; //$NON-NLS-1$
```

Ensuite, il faut créer un fichier XML dans un dossier "render", qui contient des balises connus par Osmand afin de modifier la couleur. Par exemple, la balise "motorwayRoadColor" permet de modifier la couleur des autoroutes. Dans le cas de notre plugin, nous n'avons modifié seulement la couleur des autoroutes. Cependant, avec les styles des autres plugins, nous pouvons obtenir des rendus intéressants:



Nautical



Winter and ski

III°/ Problèmes rencontrés

A°/ <u>Création du module</u>

Nous avons mis un peu de temps à trouver comment créer un module puisque le module ne se créé pas à part comme nous l'avions imaginé.

B°/ <u>Git</u>

Du fait de la taille de notre projet, nous avons eu quelques soucis avec Git. De plus, à la suite d'un crash d'un de nos ordinateurs pendant un push, nous avons du créer un nouveau dépot.

C°/ Intégration des parsers dans Android Studio

Nous avons eu des problèmes pour intégrer les parsers dans Android Studio. En effet, nous avons codé les parsers sous eclipse, comme un projet JAVA. Il a donc fallu modifier un peu le code des parsers, en rajoutant des autorisations pour qu'ils puissent fonctionner dans notre projet.

D°/ Affichage des segments

L'affichage de segments requière des fonctions et des objets pas forcement évidents. Par exemple, il est nécessaire de dessiner dans un objet Canvas et non directement dans la fonction onDraw.

E°/ AndroidStudio

Le logiciel AndroidStudio demande beaucoup de ressources, et nos machines ont eu beaucoup de mal à faire tourner ce logiciel correctement.

IV°/ Application obtenue

A°/ Fonctionnalités implémentés

Nous avons créé un nouveau plugin qui apparaît dans le gestionnaire de greffons. En y accédant, on peut voir une description du plugin, une capture d'écran et surtout l'activer.

Une fois le plugin activé, il affiche plusieurs informations sur la carte. Premièrement, il affiche le trafic de l'agglomération. Pour cela, il colorie les routes dont on a des informations en fonction du trafic. Si la route est verte, il n'y a pas d'embouteillage. Si la couleur est jaune, violet ou rouge, l'embouteillage est de plus en plus important.

Ensuite, le plugin affiche des panneaux aux endroits de l'agglomération où il y a des travaux.

Le plugin est par défaut en anglais, mais si le mobile de l'utilisateur est en français, le plugin sera traduit en français.

B°/ Performances

Notre plugin a des performances comparables à celles des autres plugins. En effet, le chargement de la coloration des routes et des panneaux mets un peu de temps mais les autres plugins qui affichent également des informations sur la carte, ont aussi un certain temps de chargement.
Cela est notamment dû au fait que l'application OsmAnd de base est très lourde.

V°/ Suites possibles du projet

A°/ Extension sur d'autres villes

Une des suites possibles au projet serait d'étendre le plugin à d'autres villes. Les données étant dans des fichiers JSON et XML, il suffit d'avoir des fichiers construits de la même façon pour pouvoir adapter le plugin à d'autres villes.

B°/ Routage

Une autre suite possible serait de faire un routage sensible au trafic. Le GPS de l'application pourrait alors prendre en compte le trafic pour éviter les zones trop embouteillées dans le calcul d'un itinéraire.

C°/ Réglages supplémentaires

On pourrait rajouter dans les options du Plugin, un réglage pour choisir le zoom à partir duquel les informations de trafic ou les travaux apparaissent sur la carte.

Concernant les travaux, on pourrait aussi rendre l'image "cliquable" pour avoir plus d'information sur leurs natures.

D°/ Mode hors-ligne

Enfin, on pourrait rajouter un mode hors-ligne qui afficherait le trafic en faisant un calcul des affluences moyennes pour une certaine période de la journée ou en gardant en cache les dernières informations reçues lors de la dernière connexion à internet.