Rapport SIG

Jordan FONTORBE
Willy FRANÇOIS
Jérémy MOROSI
Jean-Baptiste PERRIN

16 décembre 2013

Table des matières

1 Introduction

2 Analyse

2.1 Architecture

3 Réalisation

3.1 Commun

3.1.1 Package geometry

3.1.2 Package data

Le package **data** contient le modèle des données ainsi que les fonctions (SQL et XML) nécessaire au prétraitement des données.

Le modèle :
Notre modèle correspond à la structure de notre base de données. Nous avons choisi de ne garder que les nœuds, les trous, les routes, les bâtiments, les bassins et les forêts. Ainsi nous retrouvons :

- une classe **Node** contenant un *id*, une *latitude* et une *longitude*. Cette classe caractérise un point de la carte et va permettre avec ses coordonnées de construire une route, un bâtiment, un bassin ou une forêt.
- une classe **Road** contenant un *id*, un *nom* (pas obligatoire), un *type* (route secondaire, chemin, route piétonne, etc.), la liste des nœuds qui la constitue et sa *géométrie*.
- une classe abstraite **Structure** possédant un *id*, un nom (pas obligatoire), la liste des nœuds qui constitue cette structure, sa géométrie ainsi que la liste de ses trous (peut être vide). Les classes héritant de **Structure** sont :
  - la classe **Building**. Elle représente un bâtiment et à en plus une liste de tous ses nœuds proches (hors ceux constituant le bâtiment).
  - la classe **Basin** décrivant un bassin (lac, étang, etc.)
  - la classe **Forest** décrivant une forêt.
  - la classe **Hole** décrivant un trou d'une structure. Elle possède en plus l'id de la structure à laquelle il appartient.

Le modèle est représenté par le schéma ci-dessous :

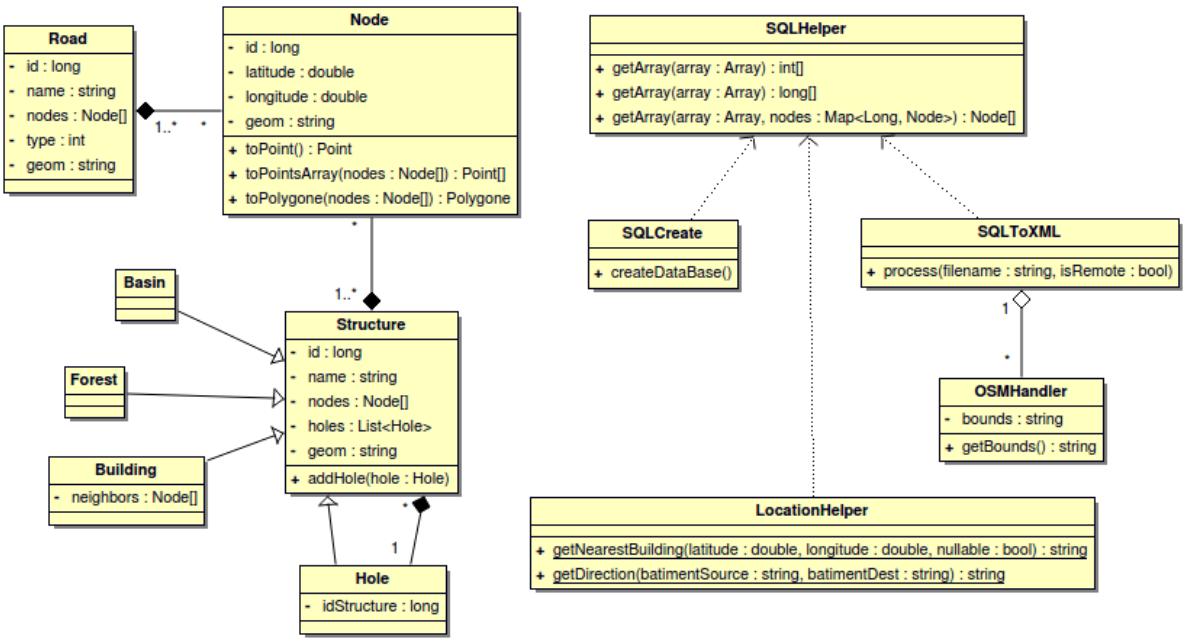

FIGURE 1 – Modèle

Le pré-traitement :

Une étape de pré-traitement à été nécessaire. En effet lorsque nous avons récupérer les données *OpenStreetMap* et importé dans une base de données (à l'aide de l'outil *osm2pgsql*), nous avons dû récupérer seulement les informations qui nous étaient utiles. Ainsi nous avons créé 4 nouvelles tables correspondant à notre modèle (sig1337_nodes, sig1337_roads, sig1337_holes et sig1337_structures) avec seulement les informations nécessaire (coordonnées, géométrie, nom, ...). Cette génération se fait à l'aide de la classe **SQLCreate** du package **data.sql**.
Ensuite, les données vont être récupérées et parsée en XML. On retrouvera notamment les tags :

- **bassin**, contenant un nom, une liste de voisins, et les triangles permettant sa construction en *OpenGL*.
- **foret**, contenant un nom, une liste de voisins, et les triangles permettant sa construction en *OpenGL*.
- **batiment**, contenant un nom, une liste de voisins, et les triangles permettant sa construction en *OpenGL*.
- **route**, contenant la liste des points la décrivant et son type (chemin ou route pour une réprésentation différent sur la carte).

3.2 WebService

Afin de permettre une utilisation de l'application en mode "remote", un *WebService* a été mis en place. Aucune donnée (comme les bâtiments, les routes, l'arbre de décision) n'étant stockée sur le téléphone dans ce mode, c'est l'appel aux méthodes de ce *WebService* qui va nous permettre de récupérer toutes les données nécessaires.
Il permet entre autre d'avoir accès à trois méthodes :

- Une méthode permettant de récupérer les informations de la carte au format XML, c'est-à-dire les nœuds, les routes et les structures ainsi que l'arbre de décision. Cette méthode sera appelée au lancement de l'application en mode "remote". Elle est accessible à partir de l'URL : http ://*IP_ADRESS* :*PORT*/WebService/service/map. La définition de cette méthode est présente dans la classe **MapService**.
- Une méthode permettant à partir d'une latitude et d'une longitude, de récupérer le plus proche bâtiment. Elle sera appelée dans l'application Android, lors d'un appuie sur une zone de la carte afin de pouvoir sélectionner un bâtiment comme point de départ ou d'arrivée pour un itinéraire. Cette méthode est accessible à partir de l'URL : http ://*IP_ADRESS* :*PORT*/WebService/service/location/building/{lat}/{lon}. Elle est définie dans la classe **LocationService**.
- Une méthode permettant à partir de l'identifiant d'un bâtiment de départ et de l'identifiant d'un bâtiment d'arrivée de retourner un itinéraire (au format JSON). La réponse renvoie la liste des coordonnées à parcourir. Cette méthode sera appelée lors de la demande de calcul d'un itinéraire par l'utilisateur et est accessible à partir de l'URL : http ://*IP_ADRESS* :*PORT*/WebService/service/location/direction/{id départ}/{id arrivée}. Cette méthode est définie dans la classe **LocationService**.

La méthode récupérant le XML décrivant la carte étant la plus longue, c'est un tâche asynchrone qui l'appel. Les deux autres méthodes étant relativement rapide (une seconde environ), on attend la réponse du WebService (avec une limite fixée à 3 secondes pour ne pas bloquer l'application trop longtemps).