



ÉCOLE NATIONALE  
DES SCIENCES  
GÉOGRAPHIQUES

Ecole Nationale des  
Sciences Géographiques

UQÀM | ISE  
Institut des sciences  
de l'environnement  
FACULTÉ DES SCIENCES  
Université du Québec à Montréal

Institut des Sciences de  
l'Environnement de  
l'Université du Québec à  
Montréal

Rapport de stage pluridisciplinaire

Cycle des Ingénieurs diplômés de l'ENSG 2<sup>ème</sup> année

---

## Conception et développement d'outils de planification, d'optimisation et de visualisation pour les recensements en Afrique subsaharienne

---



ÉCOLE NATIONALE  
DES SCIENCES  
GÉOGRAPHIQUES



Erasmus+



**Victorien Ollivier**

Août 2023

Non confidentiel    Confidential IGN    Confidential Industrie    Jusqu'au ...

ÉCOLE NATIONALE DES SCIENCES GÉOGRAPHIQUES  
6-8 Avenue Blaise Pascal - Cité Descartes - 77420 Champs-sur-Marne  
Téléphone 01 64 15 31 00 Télécopie 01 64 15 31 07

## **Jury**

### **Commanditaire :**

Claude Codjia

### **Encadrement de stage :**

Claude Codjia

### **Enseignant référent :**

Cécile Duchène

### **Responsable pédagogique du cycle Ingénieur :**

Jean-François Hangouët

### **Gestion du stage :**

?

© ENSG

## **Stage pluridisciplinaire du 22/05/2023 au 12/08/2023**

**Diffusion web :**  Internet  Intranet Polytechnicum  Intranet ENSG

### **Situation du document :**

Rapport de stage pluridisciplinaire présenté en fin de 2<sup>ème</sup> année du cycle des Ingénieurs

**Nombres de pages :** 51 pages dont 3 d'annexes

**Système hôte :** L<sup>A</sup>T<sub>E</sub>X

### **Modifications :**

EDITION	REVISION	DATE	PAGES MODIFIEES
1	0	09/2016	Création

# Remerciements

---

Je tiens tout d'abord à exprimer ma gratitude envers Claude Codjia pour m'avoir accueilli en tant que stagiaire au sein de son institut. Mon expérience au cours de cette période a été enrichissante à bien des égards.

Je tiens également à remercier l'équipe pédagogique de l'ENSG pour le suivi dont j'ai bénéficié durant ces trois mois. Les conseils avisés de Cécile Duchêne, particulièrement, m'ont été d'une grande aide pour mener à bien ce stage.

Je remercie l'organisme Erasmus+ et la fondation ENSG-Géomatique, sans qui je n'aurais pas pu effectuer ce stage dans les meilleures conditions.

Je remercie enfin mes parents. Leur soutien constant et leurs encouragements ont joué un rôle essentiel dans la réalisation de ce stage et de ce mémoire.

## Résumé

Dans un contexte de croissance démographique en Afrique Subsaharienne, planifier et optimiser les recensements de population est devenu essentiel. Pour répondre à ce besoin, nous avons développé des outils informatiques destinés à la planification, à l'optimisation et à la visualisation de ces recensements. Notre travail consiste en la création d'outils pour aider à chaque étape du processus de recensement. Tout d'abord nous avons conçu un plugin QGIS dédié aux administrateurs du recensement. Ce plugin implémente des fonctions de parcours optimal et permet de construire un itinéraire précis pour chaque secteur de dénombrement d'une zone à recenser. De plus, nous avons développé une application Android destinée aux opérateurs sur le terrain. Cette application leur offre la possibilité de visualiser les parcours de recensement et de suivre les itinéraires calculés par le plugin, guidés en temps réel. Nous présentons la modélisation des fonctionnalités essentielles de ces logiciels et une méthode pour les implémenter. Nous mettons en avant la modélisation détaillée de ces logiciels et décrivons leur processus de conception et de développement. Notre contribution repose sur la mise en œuvre de fonctionnalités essentielles visant à faciliter la planification stratégique des recensements. Nous proposons également une méthodologie claire pour la mise en place de ces outils.

**Mots clés :** Recensement, parcours optimal, itinéraire, guidage, algorithme glouton, plugin, QGIS, Android

## Résumé

In the context of demographic growth in Sub-Saharan Africa, planning and optimizing population censuses have become essential. To address this need, we have developed computer tools aimed at planning, optimizing, and visualizing these censuses. Our work involves creating tools to assist at each stage of the census process. First, we designed a QGIS plugin dedicated to census administrators. This plugin implements optimal traversal functions and enables the construction of precise routes for each enumeration area in a survey area. Furthermore, we developed an Android application intended for field operators. This application provides them with the ability to visualize census routes and track the routes calculated by the plugin, guided in real-time. We present the modeling of the core functionalities of these software tools and a method to implement them. We highlight the detailed modeling of these software tools and describe their design and development process. Our contribution is based on the implementation of essential features aimed at facilitating strategic census planning. We also propose a clear methodology for the deployment of these tools.

**Key words :** Census, optimal route, itinerary, guidance, greedy algorithm, TSP, plugin, QGIS, Android



# Table des matières

<b>Glossaire et sigles utiles</b>	<b>5</b>
<b>Introduction</b>	<b>7</b>
<b>1 Analyse du besoin et contexte</b>	<b>9</b>
1.1 Recueil et Analyse du besoin . . . . .	9
1.2 Contexte Géographique et données associées . . . . .	10
1.3 Gestion de projet et démarche suivie . . . . .	12
<b>2 Conception et Développement d'un logiciel de planification et d'optimisation des recensements</b>	<b>15</b>
2.1 Analyse fonctionnelle . . . . .	15
2.2 Mise en oeuvre . . . . .	20
2.3 Résultats . . . . .	26
<b>3 Conception et développement d'un outil de guidage et de visualisation des recensements</b>	<b>31</b>
3.1 Analyse fonctionnelle . . . . .	31
3.2 Mise en oeuvre . . . . .	33
3.3 Résultats . . . . .	36
<b>Conclusion</b>	<b>39</b>
<b>A Processus de recensement</b>	<b>49</b>
<b>B Statistiques sur l'optimalité</b>	<b>51</b>



# Glossaire et sigles utiles

---

**ENSG** École Nationale des Sciences Géographiques

**UQAM** Université du Québec A Montréal

**SIG** Système d'Information Géographique

**LGA** *Local Government Area*

**OSM** *Open Street Map*

**Planification** Processus qui fixe, après études et réflexion prospective, les objectifs à atteindre, les moyens nécessaires, les étapes de réalisation et les méthodes de suivi du recensement.

**Optimisation** Démarche consistant à rendre optimal le recensement, c'est-à-dire, dans ce contexte, à minimiser le taux de redondance que présente un itinéraire.

**Secteur de dénombrement** Région géographique dénombrée par un recenseur

**Bbox** La *Bounding Box* est un système de coordonnées rectangulaire utilisé pour définir une zone géographique. Elle est déterminée par deux ensembles de coordonnées - latitude et longitude - qui établissent l'emplacement des quatre coins du rectangle. Les coordonnées des coins créent quatre lignes, formant un rectangle. Tous les points situés à l'intérieur de la boîte sont considérés comme faisant partie de la *Bbox*



# Introduction

---

Selon les projections de l'ONU de 2023, la population de l'Afrique subsaharienne devrait augmenter considérablement, passant de 1,1 milliard d'habitants en 2020 à 2 milliards en 2050. Cette croissance démographique aura un impact important sur les pays de la région, notamment sur le Nigeria, dont la population pourrait doubler voire tripler.

Face à cette perspective, les pays de l'Afrique subsaharienne doivent faire face à d'importants défis en matière de recensement des populations en raison du manque de moyens humains pour opérer sur le terrain et de la présence de groupes djihadistes dans certaines zones. Ces recensements demeurent pourtant essentiels pour la planification économique, l'organisation administrative et pour l'aménagement de l'espace urbain. C'est dans ce contexte de raréfaction des ressources, de tensions politiques, économiques et sociales qu'apparaît la nécessité d'une optimisation des recensements pour réduire les coûts, la pollution et la mise en danger des opérateurs.

Le laboratoire de Télédétection et des SIG de l'UQAM joue un rôle clé dans ce contexte. Spécialisé dans l'analyse spatiale et dans la reconnaissance de formes, ce laboratoire se consacre au développement d'outils informatiques visant à automatiser diverses activités humaines.

L'objectif principal de ce stage est de modéliser et de développer un outil permettant la planification et l'optimisation des recensements. Cet outil doit calculer le parcours optimal pour toute zone à recenser en tenant compte des limites administratives, des réseaux hydrographiques et de la topographie du terrain. Cela se concrétise dans notre cadre par la conception et l'implémentation d'un plugin QGIS. Un autre objectif consiste à modéliser et à développer un outil permettant la visualisation d'un parcours de recensement et un guidage en temps réel des enquêteurs. Cet outil se doit d'être disponible sur tablette Android. Aussi, il doit fournir en parallèle toute information utile comme celles relatives à la praticabilité ou la dangerosité d'une zone.

Pour atteindre ces objectifs, notre démarche se décline en plusieurs étapes. D'abord, l'étude du processus de recensement est essentielle afin de mieux comprendre l'insertion de nos travaux dans leur contexte d'utilisation. Ensuite, il est nécessaire de récupérer les données pour le développement du logiciel, et de les analyser afin de prendre en compte leurs caractéristiques et aspérités. Puis, vient le développement du plugin, comprenant l'insertion de ces données, leur prétraitement, la construction de parcours optimaux sous contrainte, et la représentation des résultats de façon ergonomique. Finalement, il faut entreprendre le développement de l'application Android à partir de ce qui aura été créé par le plugin QGIS. Cela permettra de représenter les parcours avec un guidage et un suivi en temps réel, et de communiquer les informations nécessaires au bon déroulement du recensement.

Cette étude vise ainsi à répondre aux défis actuels et futurs des recensements en Afrique sub-

## **8 TABLE DES MATIÈRES**

---

saharienne en fournissant des outils efficaces pour faciliter ces opérations cruciales. En réduisant les coûts, les risques, les impacts environnementaux et en étant accessibles, ces solutions contribueront à une meilleure planification et à un déroulement facilité des recensements. Le travail que je fournis durant ce stage répond à un besoin d'ingénierie logicielle demandé par l'UQAM et sera réutilisé et affiné par la suite.

Afin de présenter la solution logicielle que nous avons conceptualisée et développée, notre étude commencera par mettre en exergue les différents besoins d'un tel logiciel et le contexte dans lequel sa conception est mise en oeuvre. Par la suite, nous reviendrons sur l'outil de planification et d'optimisation des recensements, en montrant comment nous l'avons analysé, mis en œuvre et en présentant les résultats qu'il produit et leur qualité. Nous terminerons par montrer le processus d'analyse fonctionnelle et d'implémentation de l'outil de visualisation des recensement et de guidage, puis présenterons les résultats qu'il donne.

# ANALYSE DU BESOIN ET CONTEXTE

---

CHAPITRE  
**1**

Pour commencer, il est essentiel de présenter le recueil des besoins du commanditaire effectué et l'analyse qui en a été faite. Bien sûr, cette analyse implique de détailler le contexte de mise en œuvre des solutions et méthodes proposées.

## 1.1 Recueil et Analyse du besoin

Le premier objectif du stage est de concevoir et de développer un outil de planification et d'optimisation des recensements. Par concevoir, nous entendons analyser et modéliser une solution logicielle. Par développer, nous considérons le sens d'implémenter et de coder l'outil, interface comprise. L'outil, dans le cadre de notre stage, est un plugin QGIS. Concernant les notions de planification et d'optimisation, nous les avons définies dans le glossaire. Il est essentiel de préciser que cet outil se destine à une utilisation par un « administrateur » du recensement, c'est-à-dire une personne qui est responsable de planifier celui-ci sur une zone définie. Cette zone, par ailleurs, est constituée de plusieurs secteurs de dénombrement. Il est nécessaire que l'administrateur ait quelques compétences avec le SIG QGIS pour une bonne utilisation du plugin. Des discussions entre le commanditaire et moi-même ont permis d'élaborer le recueil du besoin suivant concernant la réalisation du plugin QGIS.

Celui-ci doit être capable d'accepter plusieurs types de données en entrées, tels que le réseau routier, l'hydrographie, la topographie, ainsi que la zone d'étude. Cette zone d'étude est constituée des secteurs de dénombrement. Pour chaque secteur de dénombrement, il faudra calculer l'itinéraire optimal. Il s'agit d'un parcours passant par chaque route présente à l'intérieur du secteur. L'itinéraire doit prendre en compte l'hydrographie et la topographie : s'il y a quelque chose qui fait qu'un itinéraire en plusieurs parties doit être créé sur une zone, cela doit être pris en compte. À titre d'exemple, une rivière scindant le secteur de dénombrement en deux et ne disposant daucun pont permettant de l'enjamber doit être prise en compte et donner lieu à la construction de deux itinéraires sur un même secteur. L'itinéraire doit être visualisable et exportable dans un format facile d'utilisation.

L'autre objectif est de concevoir et de développer un outil de visualisation des parcours de recensement. Il doit aussi permettre un guidage en temps réel, fonction de la localisation. Ainsi, cet outil se destine à une utilisation par un enquêteur, qui opère sur le terrain. L'utilisation préalable du plugin et la production de résultats via celui-ci est nécessaire puisque ce sont ces résultats qui servent de données d'entrée pour l'application et le guidage. En considérant la nécessité de mobilité de l'outil en raison de son contexte d'utilisation qui est le terrain, l'outil prendra la forme d'une application Android. Le recueil du besoin précis effectué est le suivant.

L'application doit permettre de prendre en entrée l'itinéraire créé par le plugin au préalable. Elle doit permettre la visualisation de l'itinéraire, mais aussi son guidage en temps réel. Elle doit permettre d'informer de l'état recensé d'une partie de l'itinéraire et de modifier sa visualisation si c'est le cas. Aussi, elle doit mettre en exergue les informations essentielles au bon déroulement d'un recensement sur la zone en question : informations sur la topographie, sur l'hydrographie, sur la politique et le caractère à risque ou non de la zone recensée.

Ce que l'on peut conclure, après ce recueil du besoin, c'est donc de la nécessité de concevoir et de développer deux solutions logicielles. L'une permet de planifier et d'optimiser les recensements, l'autre de visualiser un des itinéraires créés par la première et de pourvoir un guidage en temps réel. Ces deux solutions logicielles présentent une utilité dans deux maillons différents de la chaîne du recensement. Prenons pour exemple le processus de recensement en France. Celui-ci se déroule en trois étapes : avant la collecte, pendant la collecte, après la collecte comme on peut le voir en annexe A.

On peut affirmer que l'outil de planification et d'optimisation, *i.e* le plugin, sera surtout utilisé dans la première partie du processus, « Avant la collecte » tandis que l'outil de guidage, *i.e* l'application Androïd, trouvera son utilisation dans la deuxième partie : « Pendant la collecte ».

Il est intéressant de noter que les procédures peuvent varier selon les pays et les administrations en terme général. En effet, nous remarquons ici que le processus de recensement français repose sur un réseau d'adresses à visiter. Une autre logique est de parcourir chaque route pour être sûr de qu'aucune habitation ne sera oubliée. C'est vers cette solution que nous avons axé notre réflexion, notamment en raison de la croissance démographique dans les pays d'Afrique subsaharienne qui implique une vitesse de construction de domiciles élevée.

Il est intéressant de noter que les procédures peuvent varier selon les pays et les administrations en terme général. En effet, nous remarquons ici que le processus de recensement français repose sur un réseau d'adresses à visiter. Une autre logique est de parcourir chaque route pour être sûr qu'aucune habitation ne sera oubliée. C'est vers cette solution que nous avons axé notre réflexion, notamment en raison de la croissance démographique dans les pays d'Afrique subsaharienne qui implique une vitesse de construction de domiciles élevée.

## 1.2 Contexte Géographique et données associées

### 1.2.1 Contexte géographique et cas d'application

Il est d'abord nécessaire de présenter le contexte géographique au sein duquel les outils à développer trouveront leur utilisation. En effet, les processus de recensements sont différents en fonction des pays et sont rattachés en général à l'institut national de statistiques. Cependant, chaque pays a sa propre méthode. L'idée de l'outil développé est qu'il puisse être utilisable avec toute forme de découpage administratif, sous réserve d'avoir les données nécessaires correspondantes. Plus précisément, la conception de ces outils se fait plutôt dans l'optique de leur utilisation dans des zones où la croissance démographique s'annonce forte à partir des années 2030, et où les moyens économiques et humains sont à utiliser de façon optimale. Cela concerne notamment les pays d'Afrique subsaharienne.

Nous avons tout de même effectué des recherches sur la nature des découpages administratifs utilisés dans le cadre des recensements afin de déterminer une méthode généralisable. En effet, les recensements, dans n'importe quel pays, se basent en premier lieu sur des découpages administratifs de la région. En France, à titre d'exemple, le découpage utilisé est un découpage infra-communal, nommé « secteur de dénombrement » ou « îlot de recensement », que l'on trouve dans les données IRIS de l'INSEE. Afin de concevoir et de développer notre outil de telle sorte qu'il soit adapté à des données de pays d'Afrique subsaharienne, nous avons choisi un cas d'application qui est celui du Nigeria.

Nous avons donc commencé par nous demander comment allait être choisie la zone à recenser,

sous la responsabilité de l'administrateur. Quelques recherches nous en apprennent davantage sur le découpage administratif du Nigeria. Celui-ci se découpe en premier lieu en états, dont le nombre est actuellement de trente-six, auxquels il faut ajouter un territoire au statut particulier pour la capitale Abuja. Chaque État est découpé en *Local Government Areas* (LGA), zones de gouvernement local. Chacune de ces zones de gouvernement local est découpée en *wards*, dont le nombre par LGA varie entre dix et vingt. La responsabilité de mettre en œuvre le recensement est attribuée à l'administration d'un LGA. Nous avons donc considéré que l'administrateur en charge de répartir les zones le ferait à l'échelle du LGA et que, par commodité dans notre cas d'application, chaque *ward* de cet LGA consisterait en un secteur de dénombrement. Bien sûr, leur superficie peut parfois être élevée. Aussi, il est du devoir de l'administrateur de créer ou récupérer lui-même le découpage administratif en secteur de dénominbrements convenable.

Afin de pouvoir concevoir et développer sur des données concrètes, nous avons choisi de façon empirique de travailler sur les *wards* du LGA Gada, qui se situe dans l'État de Sokoto.

### 1.2.2 Données associées à ce contexte

Le contexte géographique d'application implique de réaliser un état de la donnée dans les zones étudiées. En effet, l'état de la donnée géographique n'est pas le même partout et il peut être moins précis que dans les pays occidentaux, c'est notamment le cas dans les pays d'Afrique subsaharienne. En raison du contexte de travail, nous avons utilisé uniquement des données libres d'accès, c'est-à-dire majoritairement des données mises à disposition par des organisations comme *Open Street Map* (OSM) ou par les sites gouvernementaux. Cependant, il faut mettre en exergue les différences entre ces données et celles des pays occidentaux pour en décrire la qualité.

Considérons tout d'abord la correspondance entre les données et la réalité du terrain. Il faut souligner qu'elle n'est pas avérée dans les pays d'Afrique Subsaharienne ou du moins au Nigeria en particulier. C'est notamment les cas pour les données gratuites et collaboratives. En effet, il est fréquent de voir que rien qu'en comparant les données vectorielles cartographiques OSM avec celles de Google Map, les mêmes informations n'y figurent pas nécessairement. Les données de Google Map présentent plus de routes, plus de cours d'eau recensés que celles d'OSM. Ainsi, les données OSM ne sont pas exhaustives dans certains contextes géographiques. Il serait intéressant de comparer les données Google Map, plus souvent mises à jour, avec la réalité du terrain directement pour analyser la correspondance.

Ensuite, concernant la caractérisation des données, Nous pouvons constater qu'elle est absente dans de telles zones géographiques. En effet, là où OSM fournit dans les pays occidentaux les noms de rues et va jusqu'à recenser le nom des commerce et les petits aménagements (lampadaire, bancs, etc.), il ne fournit que la géométrie des objets dans la majorité des pays d'Afrique subsaharienne.

Enfin, il est à noter que la précision des données est variable mais plutôt faible dans la zone étudiée. En effet, à titre d'exemple, les découpages administratifs ne coïncident pas toujours entre eux. Aussi, la résolution des Modèles Numériques de Terrain (MNT) est de l'ordre d'environ trente mètres, pour les meilleurs que nous avons pu trouver.

Ainsi, les données sont parcellaires, peu annotées et caractérisées, et peu précises. Bien sûr, c'est le cas pour les données *open source*, peut-être que des données de sources privées pourraient avoir une qualité supérieure. Cependant, en raison de la destination des outils à concevoir, il est nécessaire de prendre en compte ces particularités et de construire le logiciel en connaissance de cause. Cela implique par exemple qu'un géocodage pour définir le point de départ du recensement est impossible ou du moins peu pertinent. Aussi, les courbes de niveau pourraient être imprécises. Enfin, cela signi-

## 12 Analyse du besoin et contexte

fie qu'il faut être vigilant concernant l'attribution des zones en raison des limites plutôt floues pour éviter qu'une zone soit recensée plusieurs fois et qu'une autre soit oubliée.

### 1.3 Gestion de projet et démarche suivie

Pour la gestion du temps lors de ce stage, je me suis appuyé sur des plusieurs outils classiques. Les premiers sont le rétroplanning ainsi que le diagramme de Gantt. Le rétroplanning ci-dessous a donc été réalisé en début de stage.

Jour	Mai	Jun	Jul	Août
1 lun		jeu Transformations du réseau en graphe	sam	mar en Java
2 mar	ven	dim	mer	
3 mer	sam	lun Finir de coder le plugin	jeu Débuguer	
4 jeu	dim	mar	ven	
5 ven	lun Rédaction du rapport	mér QGIS	sam	
6 sam	mar d'installation	jeu	dim	
7 dim	mer Fonction de parcours	ven	lun Livrables et Rapport	
8 lun	jeu optimal	sam	mar	
9 mar	ven	dim	mer	
10 mer	sam	lun Définir les fonctionnalités	jeu	
11 jeu	dim	mar de l'application	ven	
12 ven	lun État de l'art sur l'influence	mer Android et trouver	sam	
13 sam	mar de l'hydro et de la topo	jeu les sources de données	lun	
14 dim	mer sur la praticabilité	sam	mar	
15 lun	jeu d'une route	dim	mer	
16 mar	ven	lun Définir l'architecture de l'application	jeu	
17 mer	sam	mar	ven	
18 jeu	dim	lun Définir la fonction de parcours	mer Coder l'application	sam
19 ven	lun et ajouter cela au code	jeu en Java	dim	
20 sam	mar Recherche de données utilisables et visualisation	lun Coder l'application	jeu	
21 dim	mer	mar Android en	ven	
22 lun	Découverte du sujet et de son contexte.	jeu de chaque route	sam	mar
23 mar	ven et ajouter cela au code	dim	mer	
24 mer	Recherche de données utilisables et visualisation	lun	jeu	
25 jeu	mer	mar	lun	
26 ven	sur QGIS	lun Débuguer	mer Java	sam
27 sam	mar et finir la mécanique	jeu	dim	
28 dim	mer Commencer à coder le plugin	ven	lun	
29 lun	Constitution d'une architecture technique	jeu	sam	mar
30 mar	ven	dim	mer	
31 mer	viable du plugin	lun Coder l'application	jeu	

FIGURE 1.1 – Rétroplanning élaboré en début de stage

J'ai au cours du stage mis à jour un diagramme de Gantt qui est plus représentatif du déroulement de celui-ci. Il est ci-dessous.

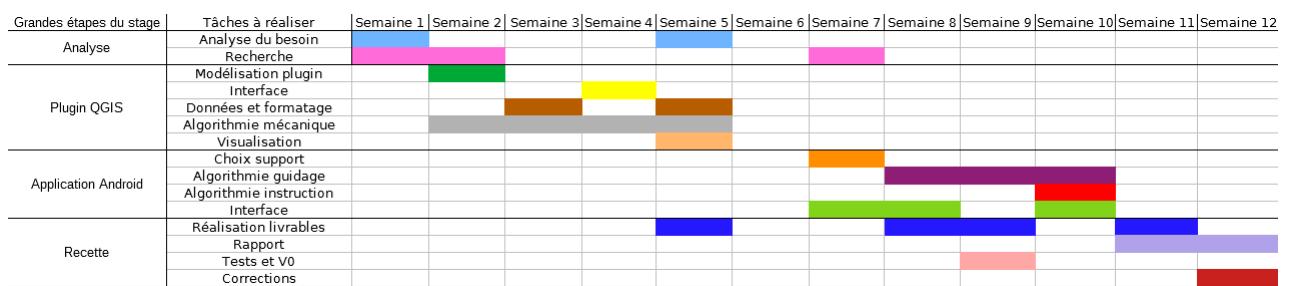


FIGURE 1.2 – Diagramme de Gantt complété au fur et à mesure du stage

Comme on peut le voir, il ne correspond pas tout à fait au rétroplanning, j'ai notamment passé plus de temps que prévu sur le développement de l'algorithme de parcours optimal qui m'a pris toute une semaine, ce que je n'avais pas prévu.

Je me suis aussi appliqué à recenser les actions à mener dans un tableau des actions classiques, sur tableur. Enfin, j'ai tenu à jour toute la durée du stage un journal de bord qui m'a permis de recenser toutes les actions journalières à effectuer, les objectifs à valider, les problèmes à résoudre, les objectifs hebdomadaires, etc.

Concernant la gestion générale du stage, j'ai échangé plusieurs fois avec ma professeur référente, Mme Cécile Duchêne, avec qui nous avons pu discuter des avancées du sujet et qui a pu répondre aux questions que je posais.

Mon commanditaire, Claude Codjia, était aussi présent pour répondre à mes questions et m'aguiller sur le sujet. Nous avons organisé une phase de recette en fin de stage afin que je puisse avoir ses retours sur les outils développés et que je puisse les modifier en conséquence.

Concernant ma démarche, pour chaque tâche à réaliser, je commence par faire quelques recherches sur l'objectif à réaliser. Puis, une fois que j'ai récupéré les informations essentielles, je réalise une modélisation avec quelques schémas et protocoles. Quand la modélisation est terminée, j'applique sur un exemple ce modèle. Puis, je conceptualise le modèle de façon informatique, par exemple par le biais d'une analyse UML, afin de rendre cette portion de code généraliste et adaptable, et je l'implémente. Bien sûr, ont fait exception à cette démarche la phase de recherche approfondie en début de stage et la phase de recette à la fin de celui-ci.



# CONCEPTION ET DÉVELOPPEMENT D'UN LOGICIEL DE PLANIFICATION ET D'OPTIMISATION DES RECENSEMENTS

CHAPITRE  
**2**

Nous allons dans cette sous-partie décrire la conception, la modélisation et l'implémentation de l'outil de planification et d'optimisation des recensements.

## 2.1 Analyse fonctionnelle

### 2.1.1 Données d'entrée, données de sortie

Pour rappel, nous modélisons ici le logiciel destiné à l'administrateur, pour planifier et optimiser les recensements. Ce logiciel implique que l'administrateur ait à disposition les données suivantes, essentielles à son bon fonctionnement.

#### 1. Zones

- La première donnée nécessaire est le découpage administratif de la zone sur laquelle on souhaite planifier le recensement. La zone est découpée en secteurs de dénombrement.
- Techniquement, il s'agit d'une couche QGIS au format Esri Shapefile. Elle est composée de plusieurs polygones qui couvrent la zone en question.
- Dans notre cas d'utilisation, il s'agit de la couche dont chaque polygone représente un *ward* du LGA de Gada, comme le montre la figure ci-dessous.

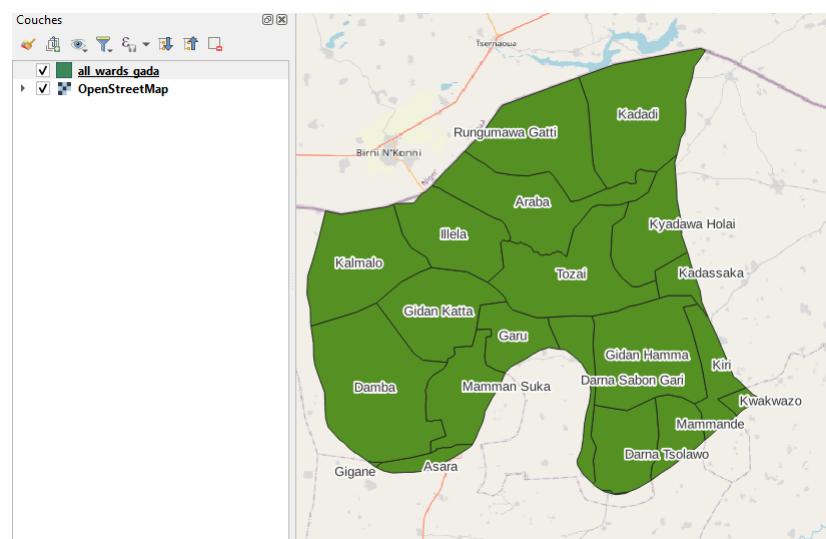


FIGURE 2.1 – Secteurs de dénombrement du Gada, extrait de QGIS

## 16 Conception et Développement d'un logiciel de planification et d'optimisation des recensements

### 2. Routes

- Ensuite, il est nécessaire d'avoir à disposition le réseau routier couvrant la totalité, ou plus, de la zone étudiée.
- Techniquement, il s'agit d'une couche QGIS au format Esri Shapefile, composée d'entités linéaires. Chaque entité est une polyligne.
- Dans notre cas d'utilisation, il s'agit de la couche du réseau routier couvrant le LGA de Gada, comme on peut le voir sur la figure ci-dessous.

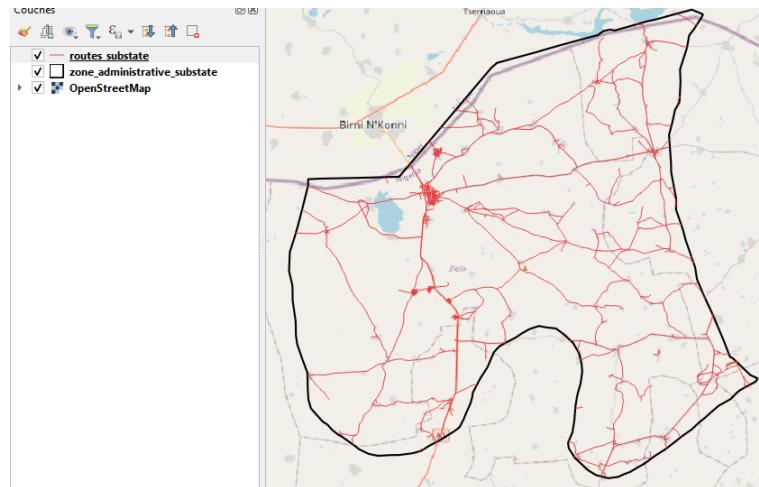


FIGURE 2.2 – Réseau routier du Gada, extrait de QGIS

### 3. Hydrographie

- Il est aussi nécessaire d'avoir le réseau hydrographique couvrant la totalité, ou plus, de la zone étudiée.
- Techniquement, il s'agit d'un fichier au format Esri Shapefile composé d'entités linéaires. Chaque entité est une polyligne.
- Dans notre cas d'utilisation, il s'agit de la couche du réseau hydrographique du réseau hydrographique couvrant le LGA de Gada, comme le montre la figure ci-dessous. On peut noter que ce réseau hydrographique n'est pas complet, ce qui fait écho à ce que nous avons expliqué précédemment concernant l'état et la qualité des données.

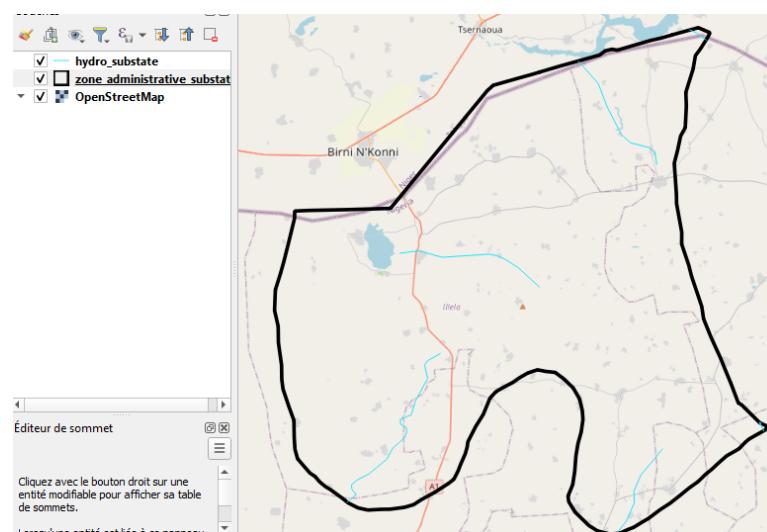


FIGURE 2.3 – Hydrographie du Gada, extrait de QGIS

### 4. Topographie

- La topographie du terrain est aussi nécessaire, c'est-à-dire une modélisation de l'altitude

sur la zone étudiée.

- Techniquement, il s'agit d'une donnée raster, un Modèle Numérique de Terrain (MNT), au format *.hgt*.
- Dans notre cas d'utilisation, il s'agit du MNT couvrant la zone de Gada, venant de la SRTM, comme le montre la figure ci-dessous.

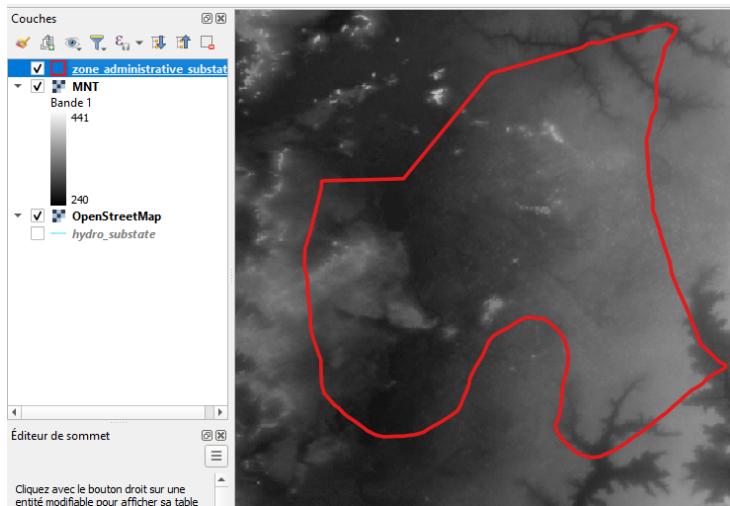


FIGURE 2.4 – Modèle Numérique de Terrain issu de la SRTM couvrant l'état de Sokoto, extrait de QGIS

##### 5. Points de départs

- Les points de départ souhaités de chaque zone, exprimés en coordonnées longitude et latitude, sont requis.
- Techniquement, il s'agit de renseigner une liste de coordonnées.
- Dans notre cas d'étude, il s'agit de renseigner un point par *ward* situé dans le Gada.

##### 6. Nombre d'itérations

- Un nombre d'itérations pour l'algorithme, qui augmente en corrélation avec l'optimalité du parcours que ressort le logiciel (nous y reviendrons en partie 2.2).
- Techniquement, il s'agit d'un nombre entier.

Une fois renseignées ces données, on souhaite que le lancement du plugin QGIS puis son traitement permette de récupérer les données suivantes en sortie :

##### 1. Visualiser les données d'entrées

- Une visualisation de certaines données d'entrée, c'est à dire précisément la zone, l'hydrographie et le réseau routier.
- Techniquement il s'agit de couches au format Esri Shapefiles, affichées sur QGIS.
- Dans notre cas d'étude, cela permet de visualiser les *wards* du Gada, son hydrographie, son réseau routier.

##### 2. Topographie

- Une visualisation des courbes de niveau pour représenter de façon épurée la topographie du terrain.
- Techniquement, il s'agit d'une couche Esri Shapefile qui représente les courbes de niveau, entités linéaires qui sont des polylignes.
- Dans notre cas d'application, il s'agit des courbes de niveau du terrain pour chaque *ward* du Gada.

##### 3. Itinéraire

## 18 Conception et Développement d'un logiciel de planification et d'optimisation des recensements

- Une visualisation de l'itinéraire optimal à parcourir pour recenser chaque *ward*, avec l'ordre de passage des routes, et des instructions pour passer d'une route à une autre.
- Techniquement il s'agit d'une couche Esri Shapefile avec des entités linéaires qui ont une symbologie en flèches. Chaque entité (chaque route) a donc un attribut de rang (expliquant en combienème position elle doit être parcourue en regard du parcours total) et d'instruction (expliquant comment un recenseur devra faire pour passer de cette route à la suivante).

### 2.1.2 Choix des logiciels, des langages et de l'architecture

Tout d'abord, l'outil doit prendre la forme d'un plugin QGIS en raison d'une demande du commanditaire. Cette forme est tout à fait justifiée en raison des nécessités logicielles, notamment celles qui impliquent d'utiliser et de représenter des données géospatiales.

Aussi, le langage choisi est le langage Python car le SIG QGIS est notamment codé en Python et propose des possibilités de construction de plugin assez aisées pour ce langage.

Concernant l'architecture, nous avons décidé de réaliser une analyse UML du plugin en amont du projet par le biais de multiples diagrammes, qui ont bien sûr évolués avec le temps.

Globalement, pour visualiser les grandes fonctionnalités du système, nous pouvons d'abord utiliser un diagramme de cas d'utilisation :

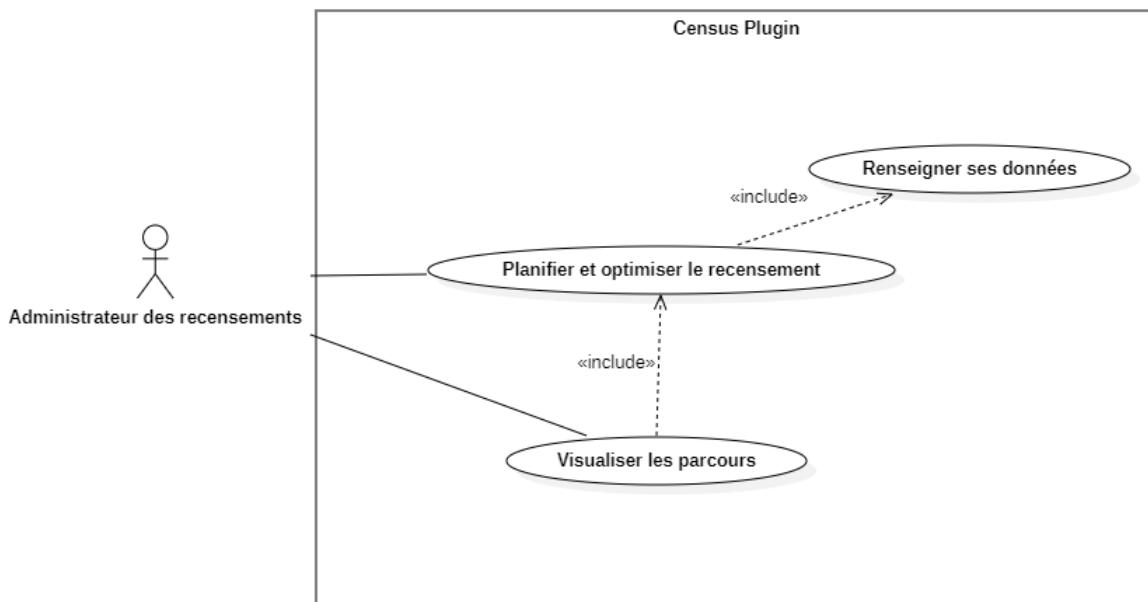


FIGURE 2.5 – Diagramme de cas utilisation du plugin QGIS

Ensuite, pour expliquer l'organisation et l'architecture des composants logiciels de l'outil, nous avons réalisé le diagramme de composant suivant :

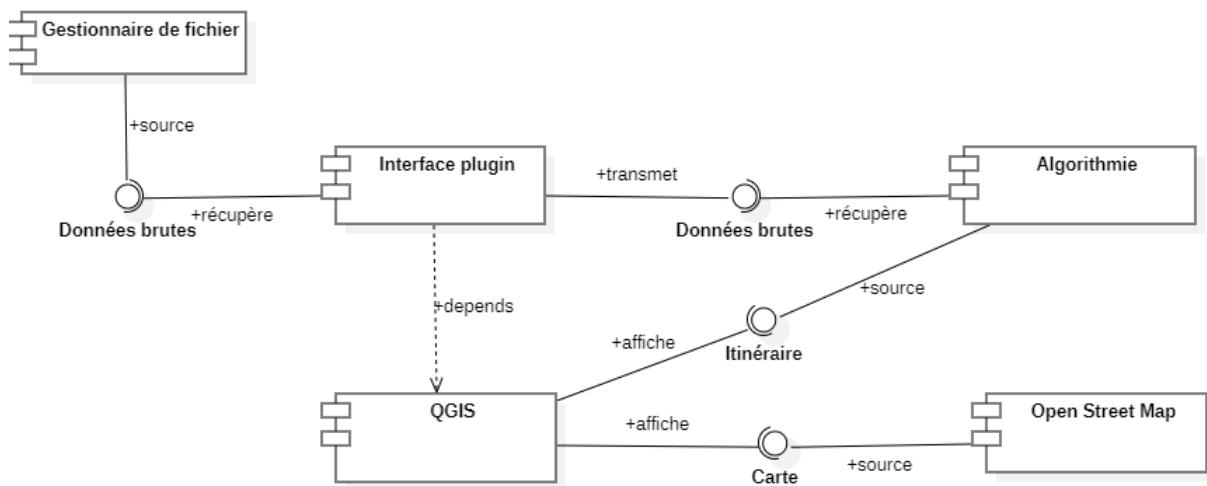


FIGURE 2.6 – Diagramme de composant du plugin QGIS

Comme nous pouvons le voir, le logiciel que nous devons développer comprend de travailler sur la partie « interface plugin » et sur la partie « Algorithmie ». L’interface, qui est utilisable sur le SIG QGIS, permet de choisir ses données d’entrées, et de lancer l’algorithme dont les résultats seront visualisables et récupérables directement sur QGIS.

Ainsi, la conception de l’outil se divise en deux parties, une première qui est celle de l’interface et une deuxième qui est celle de la mécanique.

Concernant la partie sur l’algorithmie, celle-ci a nécessité de créer plusieurs modèles avant de l’implémenter. En effet, tout d’abord, il a fallu modéliser son *workflow*, c’est à dire son activité. Pour cela, nous avons réalisé plusieurs diagrammes d’activités qui permettent de bien comprendre ce en quoi la mécanique du plugin doit consister pour passer des données d’entrée aux données de sortie.

De façon générale, voici une modélisation de la chronologie et des actions entre les objets principaux du logiciel via un diagramme de séquence :

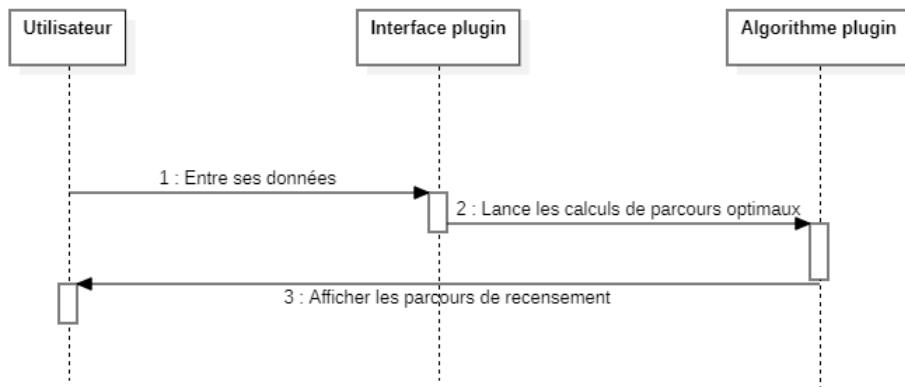


FIGURE 2.7 – Diagramme de séquence du plugin QGIS

Pour préciser ce fonctionnement, reprenons le *workflow* de l’application qui est le suivant : l’administrateur clique sur l’icône du plugin, dont l’interface s’affiche et lui permet de renseigner ses données. Il clique ensuite sur « ok », ce qui lance les fonctions de traitement du plugin. Celles-ci commencent par créer la couche « zone », c’est à dire la couche qui comprend dans notre cas

## 20 Conception et Développement d'un logiciel de planification et d'optimisation des recensements

d'application chaque *ward* de Gada. Puis, les fonctions calculent le parcours optimal de recensement pour chacun de ces *wards*. À chaque fois que celui-ci est calculé, il permet la visualisation sur QGIS du *ward*, de son réseau routier, de son hydrographie, de sa topographie (sous forme de courbes de niveau), et du parcours à suivre pour recenser le secteur de dénombrement. S'il reste des *wards* non-traités à l'issu du traitement d'un *ward*, alors le plugin doit continuer le traitement sur le *ward* non-traité suivant, et il ne s'arrête que lorsque tous les secteurs sont traités. Nous avons représenté ce fonctionnement général dans le diagramme d'activité suivant.

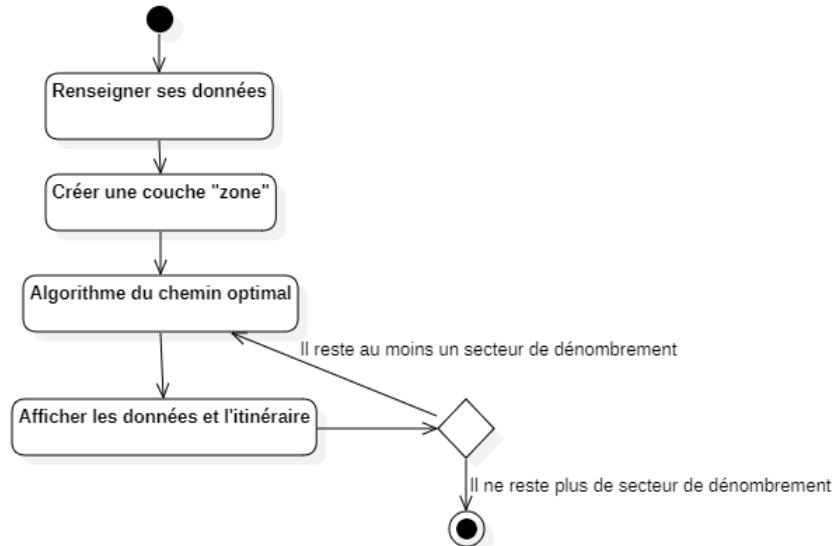


FIGURE 2.8 – Diagramme d'activité du fonctionnement général du plugin QGIS

## 2.2 Mise en oeuvre

### 2.2.1 Définition et implémentation de fonctions de parcours optimal

La première chose qu'il a été nécessaire de coder a été une fonction de parcours optimal sur un jeu test. Cela a impliqué plusieurs recherches et modélisations en amont.

Pour faciliter la compréhension du problème, nous présentons la situation suivante. Nous avons à disposition une zone, à titre d'exemple un centre-ville, et son réseau routier. Nous souhaitons trouver un itinéraire qui parcourt chaque route de la ville, avec comme objectif de minimiser la distance totale parcourue par rapport à la distance totale du réseau routier. Une modélisation du réseau en graphe semble appropriée. Celle-ci permet de représenter chaque intersection ou fin de route en tant que nœud et chaque tronçon de route comme une arête.

Ce problème d'optimalité se rapproche d'un autre bien connu, celui du voyageur de commerce. En informatique, le problème du voyageur de commerce est un problème d'optimisation qui consiste à déterminer, étant donné un ensemble de villes et les distances entre toutes les paires de villes, un plus court circuit qui passe par chaque ville une et une seule fois.

Celui-ci est peut être résolu par deux méthodes. La première est la résolution exacte. C'est-à-dire qu'il faut construire chaque trajet potentiel et tous les comparer pour trouver la solution optimale. Cependant, cette méthode a une complexité en temps importante, de  $O(n!)$  et en fonction du nombre de villes et de paires de villes, le temps de calcul pour la résolution peut être véritablement énorme.

La seconde méthode est celle de la résolution approchée, en trouvant une solution sous-optimale, mais qui reste tout à fait correcte. Ce sont les méthodes dites heuristiques. De nombreuses méthodes heuristiques ont été pensées, et répondent sous le nom d'algorithme génétique, d'algorithme tabou, ou encore d'algorithme glouton, entre autres.

Il y a des différences entre mon objectif et ce problème connu, comme par exemple le fait que les graphes construits ne sont pas nécessairement complets, puisqu'ils modélisent un réseau routier, et que les voies sans issue sont fréquentes. Il est aussi à noter que, dans notre cas, il est possible d'emprunter plusieurs fois la même route, il faut simplement ne la recenser qu'une seule fois. Mais par exemple, si une portion de route est très courte et qu'elle est empruntée de multiples fois, cela n'affecte pas énormément l'optimalité si l'on raisonne en termes de distance parcourue par rapport à la distance totale du réseau. En effet, nous savons qu'il est impossible de parcourir toute la ville en empruntant une seule fois chaque route, il faut donc accepter d'avoir un certain taux de redondance, bien qu'on cherche à le minimiser. Cependant, le principe de résolution par méthode heuristique s'est avéré être une piste intéressante pour notre problème.

Nous nous sommes focalisés sur le fonctionnement des algorithmes gloutons. Cette famille d'algorithme suit le principe de réaliser, étape par étape, un choix optimum local, afin d'obtenir un résultat optimum global. Dans notre cas présent, une façon de définir l'optimalité à l'échelle locale pourrait reposer sur la méthode du plus proche voisin.

Ainsi, pour l'idée générale, on pourrait considérer que lorsque le recenseur est à une intersection, il préfère choisir la route la moins longue qui s'offre à lui. Cependant, quand on se replace dans le contexte de notre logiciel, on se rend compte que la priorité du recenseur est plutôt de choisir une route qui n'a pas été recensée. Cela nous a menés à édifier une disjonction de cas. Nous allons expliquer la logique de l'algorithme en utilisant un vocabulaire de théorie des graphes. Pour rattacher cela au contexte réel, une arête représente une route. Un noeud, représente une intersection, ou une fin de route. Le noeud d'intérêt est dans la réalité l'intersection où se situe le recenseur. L'aspect visité ou non d'une arête représente l'aspect recensé ou non de la route qu'elle représente.

Voici la définition que nous pouvons élaborer d'un algorithme glouton reposant sur un principe de plus proche voisin, dans notre contexte :

- Si, au noeud d'intérêt, il n'y a qu'une arête adjacente, alors c'est l'arête qui va être parcourue. Il s'agit, dans la réalité, d'un cas de début de trajet ou de demi-tour.
- Si le noeud d'intérêt a deux arêtes adjacentes, comprenant celle qui vient d'être visitée, alors on visite l'autre option.
- S'il y a plusieurs arêtes adjacentes, alors on priorise en fonction de l'aspect visité ou non de celles-ci. En effet, s'il y a une seule arête qui n'est pas visitée, alors c'est celle-ci qui est choisie. Si aucune des arêtes n'est visitée, alors on raisonne selon la méthode du plus proche voisin, et on choisit l'arête dont l'autre noeud est le plus proche du recenseur par rapport aux autres arêtes.

Cela permet déjà de construire des portions d'itinéraires assez longues. Cependant, comme expliqué précédemment, on accepte de repasser par certaines arêtes puisqu'on sait que cela sera nécessaire. Ainsi, on a établi un raisonnement par degré de voisinage. Cela implique d'étoffer l'algorithme en raisonnant à des degrés de voisinage supérieurs à un.

Si jamais toutes les arêtes disponibles sont recensées, alors on récupère les noeuds adjacents du noeud d'intérêt, et on récupère les noeuds adjacents de chacun d'entre eux. Ainsi, si une arête adjacente de degré 2 présente un aspect non-recensé, alors on choisira l'arête voisine (donc de degré

## 22 Conception et Développement d'un logiciel de planification et d'optimisation des recensements

---

1), qui y mène. Cette situation est représentée dans le schéma ci-dessous. Le noeud rouge correspond à l'intersection où se situe le recenseur. Les nœuds verts sont les voisins de degré 1 et les nœuds jaunes les voisins des nœuds verts, soit les voisins de degré 2. En rouge sont représentées les arêtes recensées et en vert clair celles qui ne le sont pas. Dans ce cas, l'algorithme choisira le nœud vert menant aux arêtes vert clair.

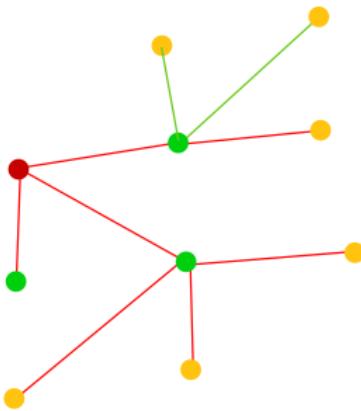


FIGURE 2.9 – Schéma d'un noeud et de son voisinage

Par la suite, nous avons raisonné en acceptant une visibilité de l'algorithme jusqu'à des voisins de degré 4. Si jamais un nœud est isolé, c'est-à-dire que toutes les arrêtes adjacentes autour de lui (jusqu'à un degré de voisinage égal à 4) sont visitées, alors on utilise la récursivité et on relance l'algorithme sur le graphe à partir d'un point choisi aléatoirement, mais de telle sorte que ses arêtes adjacentes soient non visitées.

Ce raisonnement nous a permis de construire plusieurs itinéraires, qui correspondent au parcours optimal. Ci-dessous, le diagramme d'activité explique la modélisation de cette fonction de parcours optimal.

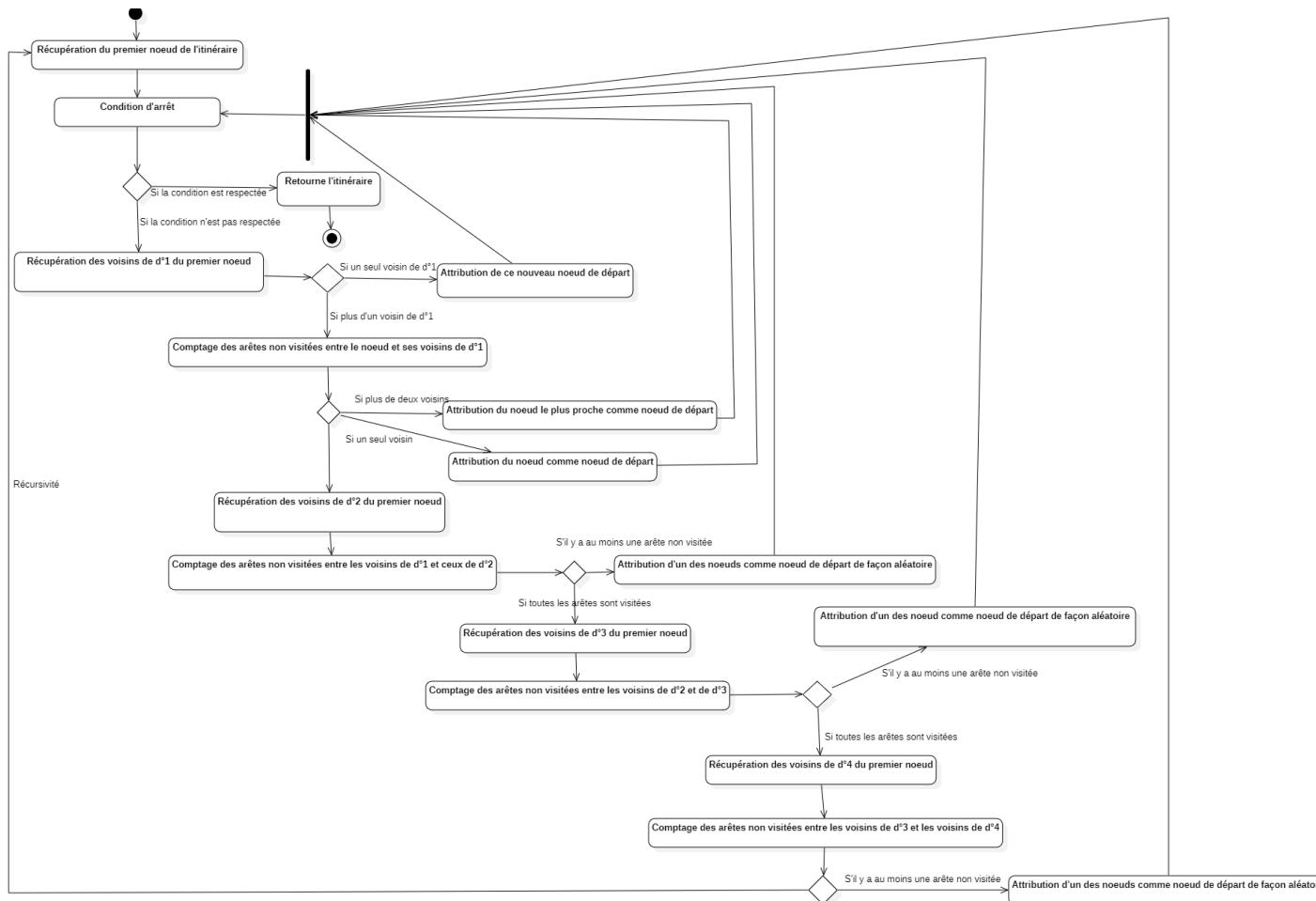


FIGURE 2.10 – Diagramme d'activité de la fonction de parcours optimal

## 2.2.2 Définition du meilleur itinéraire unique et implémentation

Une fois créés ces multiples itinéraires optimaux à l'aide de la méthode expliquée précédemment, le secteur de dénombrement est couvert de portions d'itinéraires optimales, mais non rattachées. Il s'agit donc de rattacher ces portions de la meilleure façon possible pour construire le trajet final.

Tout d'abord, nous avons commencé par observer les résultats, sur plusieurs jeux de données tests, obtenus à l'étape précédente. Nous nous sommes rendus compte que deux types d'itinéraires sont créés : les petits (composés de huit routes ou moins) et les grands (composés de plus de huit routes). Nous avons donc décidé de séparer les petits itinéraires des grands en fixant le seuil à huit routes pour les différencier. Ce seuil est empirique.

Pour chaque petit itinéraire, on vérifie si son point de départ ou d'arrivée est un noeud confondu avec un noeud appartenant à l'un des grands itinéraires. Auquel cas, nous calculons l'algorithme de Dijkstra, algorithme du plus court chemin, sur le graphe entre le point final du petit itinéraire et son point de départ, confondu avec un point d'un grand itinéraire. Ainsi, on insère une petite boucle dans un grand itinéraire. Cette petite boucle ne présente pas une augmentation du taux de redondance significative, mais permet de réduire le nombre d'itinéraires à rattacher. Bien sûr, si aucune des extrémités du petit itinéraire n'est confondue avec un point d'un des grands itinéraires, alors on le conserve tel quel.

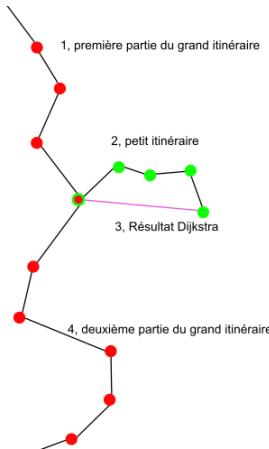


FIGURE 2.11 – Schéma expliquant le rattachement d'un petit itinéraire à un grand

Une fois ce premier tri effectué, nous remarquons que nous avons laissé beaucoup de place à des choix aléatoires : au cours de l'algorithme de parcours optimal, au cours du choix d'un point aléatoire si jamais il y a de la récursivité dans l'algorithme de parcours optimal, etc. Au cours du tri entre les petits et les grands itinéraires et de leur rattachement, cet aspect aléatoire est toujours. En effet, en fonction des choix aléatoires faits précédemment, le rattachement des petits itinéraires aux grands peut être très différent.

Pour contrer cet aspect aléatoire et avoir la meilleure solution possible, nous avons décidé de réaliser les étapes précédentes plusieurs fois, c'est-à-dire la fonction de parcours optimal et la fonction de tri des itinéraires entre les grands et les petits, et d'évaluer le résultat produit. Ainsi, nous avons implémenté une fonction qui réitère les fonctions précédentes le nombre de fois que l'utilisateur le souhaite. Il s'agit de l'entier dont nous avons parlé lorsque nous avons abordé les données d'entrée en 2.1.2. Ainsi, on peut produire  $n$  solutions du problème. Mais il faut trouver quelle solution parmi toutes celles générées est la meilleure.

Dans cet objectif, nous avons défini un critère d'optimalité. Pour cela, nous avons réfléchi à ce qui faisait l'optimalité d'un trajet entier et unique.

- Le premier critère est la distance parcourue. On somme, lors de l'exécution de la fonction de parcours optimal, la distance parcourue au cours de cet itinéraire. On stocke par la suite cette distance.
- Le second critère est le nombre d'itinéraires créés par la fonction de parcours optimal. En effet, qui dit une multitude d'itinéraires dit qu'il y aura un gros travail de rattachement et donc potentiellement une augmentation du taux de redondance par la suite.

Nous avons donc construit notre critère d'optimalité  $I$  en essayant de minimiser la distance parcourue et le nombre d'itinéraires créés. Il est exprimé comme ceci :

$$I = \alpha * x + \beta * y$$

Avec  $x$  le nombre d'itinéraires créés et  $y$  le poids de la relance, c'est-à-dire la distance totale parcourue.

Dans notre application, nous avons verrouillé la valeur d'alpha à 0.75 et la valeur de beta à 0.25. En effet, en tâtonnant, nous nous sommes rendus compte que ces valeurs donnaient la meilleure solution.

Une fois sélectionnée la solution avec le critère d'optimalité le plus faible, nous pouvons procéder au rattachement des itinéraires en un seul. Pour cela, pour chaque portion d'itinéraire, nous sélec-

tionnons ses points d'extrémité puis, pour chacun de ces deux points, nous calculons un algorithme de Dijkstra avec chaque extrémité des autres itinéraires encore disponibles. Parmi tous les chemins les plus courts calculés, nous choisissons celui dont la distance est la plus courte et l'itinéraire auquel il mène. Nous commençons alors à construire le trajet final en ajoutant chaque portion d'itinéraire dans le bon sens et les chemins les plus courts calculés qui les relient les uns avec les autres. Puis nous réitérons jusqu'à obtenir le trajet final complet.

Une amélioration possible serait d'appliquer ces derniers calculs, ce rattachement en un seul itinéraire non pas sur la, mais sur les meilleures solutions ressorties à l'étape précédente. Cela permettrait de choisir l'itinéraire avec le taux de redondance final le plus faible. En effet, en fonction de la disposition spatiale de la meilleure solution, le taux de redondance de la distance parcourue pourrait augmenter ou diminuer lors du rattachement. À titre d'exemple, si dans la solution optimale avec le moindre taux de redondance sélectionné les extrémités de chaque portion d'itinéraire sont très espacées, peut-être que le taux de redondance va considérablement augmenter lors du rattachement chaque portion. Cependant, peut-être que d'autres configurations des portions d'itinéraires présenteraient un taux de redondance plus élevé mais avec une configuration spatiale des extrémités plus resserrée qui fait que, lors du rattachement en un itinéraire final, le taux de redondance serait moins important.

Par ailleurs, il est à noter que l'on conserve dans tous les cas le premier itinéraire calculé par la fonction de parcours optimal tel quel, afin de ne pas perdre le point de départ choisi par l'utilisateur. La solution trouvée reste un trajet très correct, dont nous reviendrons sur la qualité en partie suivante.

Aussi, il peut y avoir une hydrographie ou une topographie qui empêche de construire un itinéraire unique. Par exemple, s'il y a une rivière qui traverse le secteur de dénombrement, alors on ne peut construire un seul itinéraire. En théorie des graphes, cela est assez visible, sans que l'on ait à considérer de façon explicite la topographie ou l'hydrographie. En effet, il s'agit en réalité d'un cas où le graphe est non connexe et constitué d'au moins deux sous-graphes. Dans ces cas-là, on construit autant d'itinéraires qu'il y a de sous-graphes.

### 2.2.3 Mise en oeuvre du plugin QGIS

Le calcul de parcours optimal sur graphe est la méthode pour calculer le trajet à effectuer par le recenseur, cependant, il faut construire le reste du logiciel pour que l'outil soit utilisable.

Ainsi, nous avons d'abord créé une interface de plugin à l'aide de deux autres plugins QGIS. Le premier est *Plugin Builder* qui initialise l'interface vierge, la boîte de dialogue et le script python de la mécanique. Le second est *Plugin Reloader*, qui permet de recharger le plugin, de le mettre à jour directement via QGIS et de prendre en compte au fur et à mesure les modifications du code Python exécuté. Pour l'ergonomie de l'interface, nous avons utilisé le logiciel QT Design.

Une fois l'interface réalisée, nous avons codé la récupération des données d'entrée. Puis, nous les traitons pour sélectionner uniquement les données sur la zone concernée.

Pour appliquer la méthode développée précédemment en partie 2.2.1 et 2.2.2, nous avons commencé par utiliser la fonction « explosion des lignes » de QGIS sur chaque route de notre réseau routier. La raison est simple : tel qu'il était, deux routes qui se croisaient ne présentaient pas de façon explicite d'intersection, et donc ne permettait pas de détecter un nœud. Ainsi, nous avons divisé chaque route en ses plus petites entités géométriques afin de facilement détecter chaque intersection. Puis, nous avons simplifié notre graphe en supprimant les nœuds non-nécessaires, c'est-à-dire qui les nœuds n'étant ni des extrémités de route, ni des intersections. Cela a permis de construire les

## 26 Conception et Développement d'un logiciel de planification et d'optimisation des recensements

arrêtes : chaque route construite possède deux extrémités, qui sont dans la réalité des bouts de route ou des intersections, et dans le graphe des nœuds. Et entre ces nœuds il y a la modélisation de la route, c'est-à-dire dans le graphe l'arête. Une fois ces transformations réalisées, nous avons utilisé la bibliothèque *Networkx* et avons instancié le graphe qui modélise le réseau routier traité. Puis, nous avons appliqué à ce graphe les fonctions de parcours optimal dont nous avons parlé précédemment. Ensuite, nous avons appliqué la méthode expliquée en 2.2.2 pour construire un itinéraire unique. Puis, nous avons codé des fonctions de visualisation pour que les données utiles et l'itinéraire créé pour le secteur de dénombrement en question soient représentés.

Ce *workflow* est représenté ci-dessous par un diagramme d'activité.

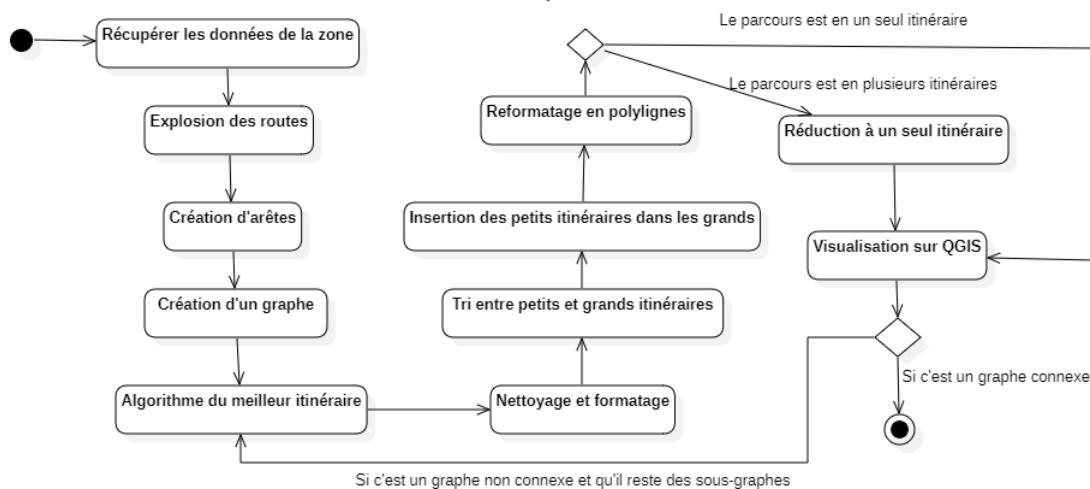


FIGURE 2.12 – Diagramme d'activité du workflow du plugin QGIS

## 2.3 Résultats

### 2.3.1 Résultats visuels

Pour présenter les résultats, voici tout d'abord les visuels obtenus pour le plugin. Commençons par l'interface en elle-même, que l'on peut afficher directement sur QGIS, qui se présente comme sur la figure ci-dessous.

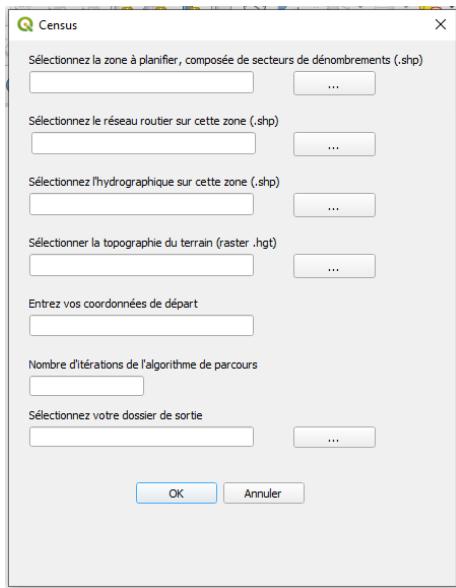


FIGURE 2.13 – Interface du plugin QGIS créé

Comme nous pouvons le voir, les données demandées sont celles spécifiées en partie 2.1. Une fois renseignées ces données et appuyé sur OK, voici ce que l'on peut observer, respectivement à l'échelle de la zone à planifier (le LGA de Gada), puis à l'échelle d'un secteur de dénombrement (*ward* du Gada), et enfin à l'échelle de la ville qui est au sein d'un secteur de dénombrement.

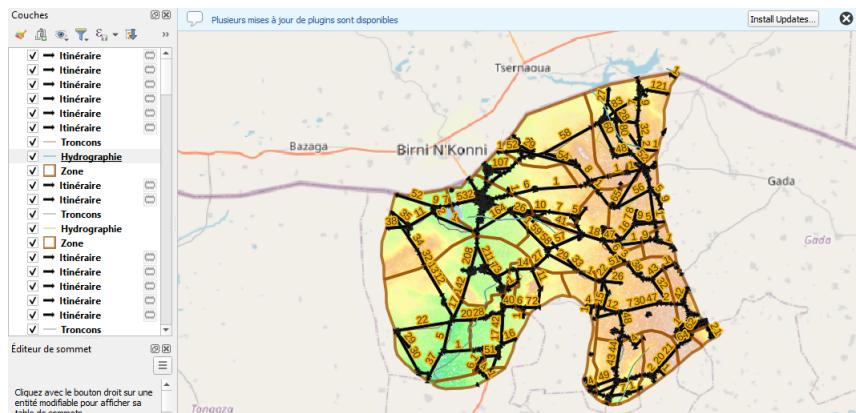


FIGURE 2.14 – Extrait de QGIS présentant les sorties du plugin sur la zone totale à planifier

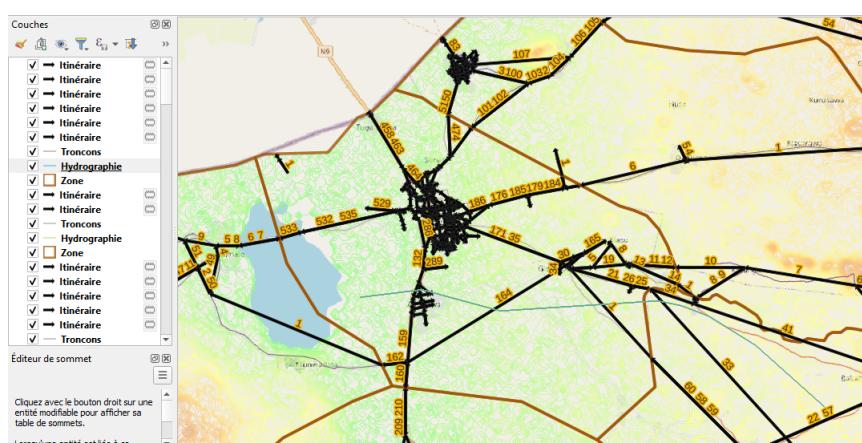


FIGURE 2.15 – Extrait de QGIS présentant les sorties du plugin sur un secteur de dénombrement



FIGURE 2.16 – Extrait de QGIS présentant les sorties du plugin au niveau d'un centre-ville

Les zones délimitées par une ligne marron correspondent à chaque *ward*. Le fond de carte est un fond cartographique offert par OSM directement sur QGIS. Les flèches noires sont au-dessus du réseau routier : il s'agit de l'itinéraire. Pour éviter les confusions en raison de fréquents aller retour, nous avons ajouté des étiquettes, qui indiquent le rang de la route, c'est-à-dire dans quel ordre elle doit être parcourue. Les figurés linéaires dans les tons verts, jaune et orange sont des courbes de niveau extraites du MNT renseigné en entrée. Les figurés bleus linéaires correspondent à l'hydrographie du terrain.

### 2.3.2 Etude de l'optimalité

Une question légitime sur les résultats est celle de leur optimalité. Bien évidemment, l'aspect optimal au sens de « meilleur » ne peut être obtenue qu'avec une étude cas par cas de toutes les possibilités qui impliquerait une complexité très haute, ou par un coup de chance. Notre algorithme donne un algorithme sous-optimal, mais dont les résultats produits semblent correcte. Pour vérifier cela, nous avons réalisé une étude statistique des résultats en fonction du taux de redondance. Celui est un pourcentage qui exprime le rapport entre la distance totale du parcours et celle parcourue, comprenant alors les aller-retour. Si nous avons un taux de redondance de 33%, cela signifie que 33% du parcours est parcouru plus d'une fois. Nous avons essayé de dessiner à la main des parcours de recensement et nous sommes rendu compte que, sur des zones petites à moyenne, on obtenait globalement les mêmes résultats de taux de redondance avec le plugin qu'à la main.

Nous avons cependant un paramètre que l'on peut modifier pour améliorer l'optimalité : c'est le cas du nombre d'itérations choisi. En effet, si l'on choisit de faire trois tirages d'itinéraire, ou mille tirages, il y a plus de chance d'obtenir un meilleur itinéraire dans les mille tirages. Cependant, cela dépend aussi de la taille du réseau sur lequel on travaille. Cela nous amène à l'étude suivante.

Nous avons choisi quatorze secteurs de dénombrement différents, d'une taille variant de 1 route à 356. Pour chacune, nous avons lancé l'algorithme avec différents nombres d'itérations donnés, 1, 5, 10, 20, 50, 100, 500, 1000. Nous avons à chaque fois calculé le taux de redondance. Les résultats observés sont mis en exergue en annexe B.

Ce qu'on observe, c'est que pour des petits réseaux (moins d'une dizaine de routes), rien ne sert d'avoir trop d'itérations qui augmentent le temps de calcul, une itération est largement suffisante. Cependant, pour des réseaux plus grands, plus on augmente le nombre d'itérations, plus le taux de redondance diminue bien que cela plafonne à partir d'un certain nombre d'itérations. Ce seuil est corrélé au nombre de noeuds du réseau. Bien sûr, on parle de corrélation et non de causalité, d'autres facteurs doivent être pris en compte pour établir une règle plus précise, comme la configuration

spatiale du réseau. En effet, si chaque nœud a beaucoup d'arêtes adjacentes ou non, cela influence aussi le taux de redondance.

On remarque que l'on peut améliorer l'optimalité en jouant sur ce paramètre. Il est donc intéressant de souligner que plus les secteurs de dénombrement sont petits, plus la solution présentée par l'algorithme est optimale dans le sens de la « meilleure ». Et, plus le secteur de dénombrement contient des routes, plus un nombre d'itération élevé permettra d'avoir de bons résultats, en règle général.

Comment conserver l'optimalité lorsque l'on rattache chaque portion d'itinéraire aux autres pour construire le trajet final ? Pour conserver l'optimalité, nous avons, pour rappel, effectué ce rattachement en deux temps, d'abord en rattachant les petits itinéraires aux grands, puis en rattachant chaque grosse portion l'une à l'autre. Nous avons utilisé un algorithme de plus court chemin, l'algorithme de Dijkstra, pour effectuer cela au mieux. Cependant, on note que l'influence sur le taux de redondance dépend alors de l'éloignement des extrémités des itinéraires les uns par rapport aux autres. C'est pourquoi une perspective intéressante à analyser serait de conserver plusieurs des solutions optimales calculées précédemment, et de recalculer un indice d'optimalité basé sur le taux de redondance uniquement après le rattachement des itinéraires pour chacune des solutions envisagées. Cela permettrait de prendre en compte la totalité des facteurs pouvant influencer le taux de redondance. Cependant, sur des petits secteurs de dénombrement, ces calculs seraient superflus.

### 2.3.3 Perspectives d'amélioration

Il y a certains aspects du plugin qui peuvent être améliorés. Tout d'abord, à propos des codes de visualisation, il serait possible d'améliorer les codes de symbologie. En effet, certains détails peuvent être ajoutés comme la représentation directe des étiquettes, l'adaptation des couleurs, le fait de conférer la bonne symbologie aux courbes de niveau, etc.

Ensuite, pour l'optimalité, on pourrait améliorer l'algorithme. Actuellement, si un nœud a un voisin de n'importe quel degré qui présente une ou plusieurs arête(s) non visitée(s) lui étant adjacente(s), le choix est réalisé aléatoirement. Il pourrait être intéressant d'ajouter certaines vérifications. Par exemple, au lieu de choisir aléatoirement, regarder pour chaque voisin d'un degré précis (s'il présente plus d'une arête non visitée), laquelle mène vers d'autres arêtes non visitées pour éviter des demi-tours et des redondances inutiles. Ainsi, l'algorithme peut encore être affiné.

Il pourrait être très intéressant de multiplier les tests pour créer un indicateur qui se base sur le nombre d'arêtes, le nombre de nœuds d'un graphe modélisant un réseau et la taille en mètres du réseau pour déterminer le nombre d'itérations à effectuer pour avoir les meilleurs résultats, et d'attribuer par défaut ce nombre d'itérations lors de l'exécution de l'algorithme glouton.

Il pourrait être intéressant, au lieu d'entrer les coordonnées des points de départ souhaités, de pouvoir cliquer sur la zone de départ envisagée avec la souris, en raison de la destination de l'outil qui n'est pas nécessairement pour des experts en géomatiques.



# CONCEPTION ET DÉVELOPPEMENT D'UN OUTIL DE GUIDAGE ET DE VISUALISATION DES RECENSE- MENTS

---

CHAPITRE  
**3**

La conception et le développement de l'outil de planification et d'optimisation par le biais d'un plugin QGIS était axée sur une utilisation par un administrateur du recensement. Il a permis, pour chaque secteur de dénombrement, de construire un itinéraire optimal. Nous souhaitons maintenant récupérer cet itinéraire créé au préalable, et permettre son utilisation directe par le recenseur, opérant sur le terrain. Nous allons donc expliquer dans cette partie la conception et le développement d'une application Android qui permet au recenseur de suivre en direct l'itinéraire qu'il doit parcourir et d'être guidé tout au long de celui-ci.

## 3.1 Analyse fonctionnelle

### 3.1.1 Données d'entrée, données de sortie

Une seule donnée est demandée à l'utilisateur en entrée. Il s'agit de l'itinéraire sur le secteur de dénombrement créé par le plugin QGIS au préalable. A chaque morceau de l'itinéraire est attribué un rang et une indication. L'indication permet de passer d'une route à l'autre. Nous reviendrons sur la génération de cette indication dans cette partie. Bien que techniquement elle est générée à l'aide du plugin, nous avons décidé de présenter son algorithmie dans cette partie puisque c'est ici qu'elle est utilisée. Techniquement il s'agit d'un fichier GeoJSON sauvegardé depuis QGIS et enregistré dans les dossiers de l'appareil qui porte l'application. Dans notre cas d'étude, il s'agit d'un itinéraire calculé sur l'un des wards de Gada, nous avons choisi Tozai.

La reste des données provient de différents services, comme ceux de tuiles cartographiques

Le traitement effectué par l'application Android permet de récupérer ces données en sortie :

1. Carte
  - On souhaite obtenir une visualisation cartographique centrée sur la localisation de l'utilisateur.
  - Techniquement, il s'agit d'un fond de carte offert par OSM.
  - Dans notre cas d'application, nous avons simulé la position dans le Gada.
2. Instructions
  - Nous souhaitons aussi obtenir une instruction composée d'une indication pour rejoindre la prochaine route de l'itinéraire, la distance à cette route restant à parcourir et une flèche qui permet de visualiser la direction à suivre.

### 3. Itinéraire

- Enfin, nous souhaitons une visualisation de l'itinéraire sur le fond cartographique : il faut une visualisation de tout l'itinéraire à parcourir, et différencier ce qui a été parcouru.
- Techniquement, il s'agit du fichier GeoJSON dont les polylignes sont représentées sur la carte.

#### 3.1.2 Choix des langages et modélisation UML

Initialement, nous avions pensé utiliser QFIELD, version mobile de QGIS sensément adaptée à une utilisation pour le terrain. Cependant, ce support est peu malléable et ne laisse pas beaucoup de possibilités pour adapter, modifier une visualisation, afficher des instructions, etc. En conclusion, QFIELD ne permettait que d'afficher le GeoJSON créé, ce qui n'était pas suffisant.

Nous avons donc décidé de changer de support en considérant les besoins de fonctionnalités de l'application énoncés précédemment, et en considérant la nécessité d'interopérabilité de l'application sur tablettes, téléphones, qui sont sous le système d'exploitation Android.

Pour pouvoir répondre au besoin, nous avons donc choisi de coder une application pour Android avec un langage adapté : le Java. Cela nous permet de réaliser nos fonctions. Cela apporte aussi un intérêt qui est la programmation orientée objet, qui va nous permettre une exécution rapide et en direct des scripts lors de l'utilisation de l'application. Nous avons effectivement la contrainte d'utilisation en direct qui est importante : le temps de chargement ne peut pas excéder plusieurs secondes.

Nous pouvons mettre en exergue les grandes fonctionnalités attendues de l'application par le biais du diagramme de cas d'utilisation ci-dessous.

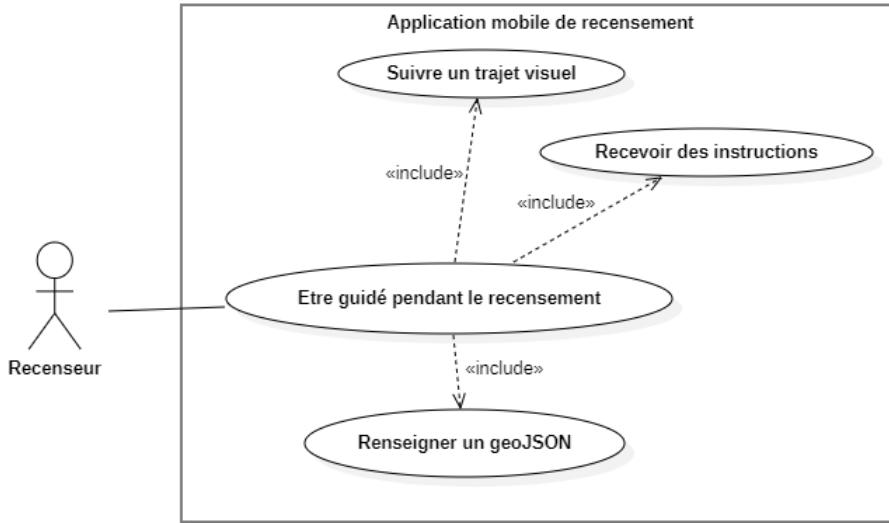


FIGURE 3.1 – Diagramme de cas utilisation de l'application Android

On remarque qu'il y aura la nécessité d'actualiser la localisation en fonction de la position du recenseur, et d'adapter à chaque fois les indications. Pour mettre cela en avant, on peut observer le diagramme de séquence suivant. Comme on peut le voir, la modification de la position et des indications se fait via une mise à jour régulière des informations de localisation du recenseur.

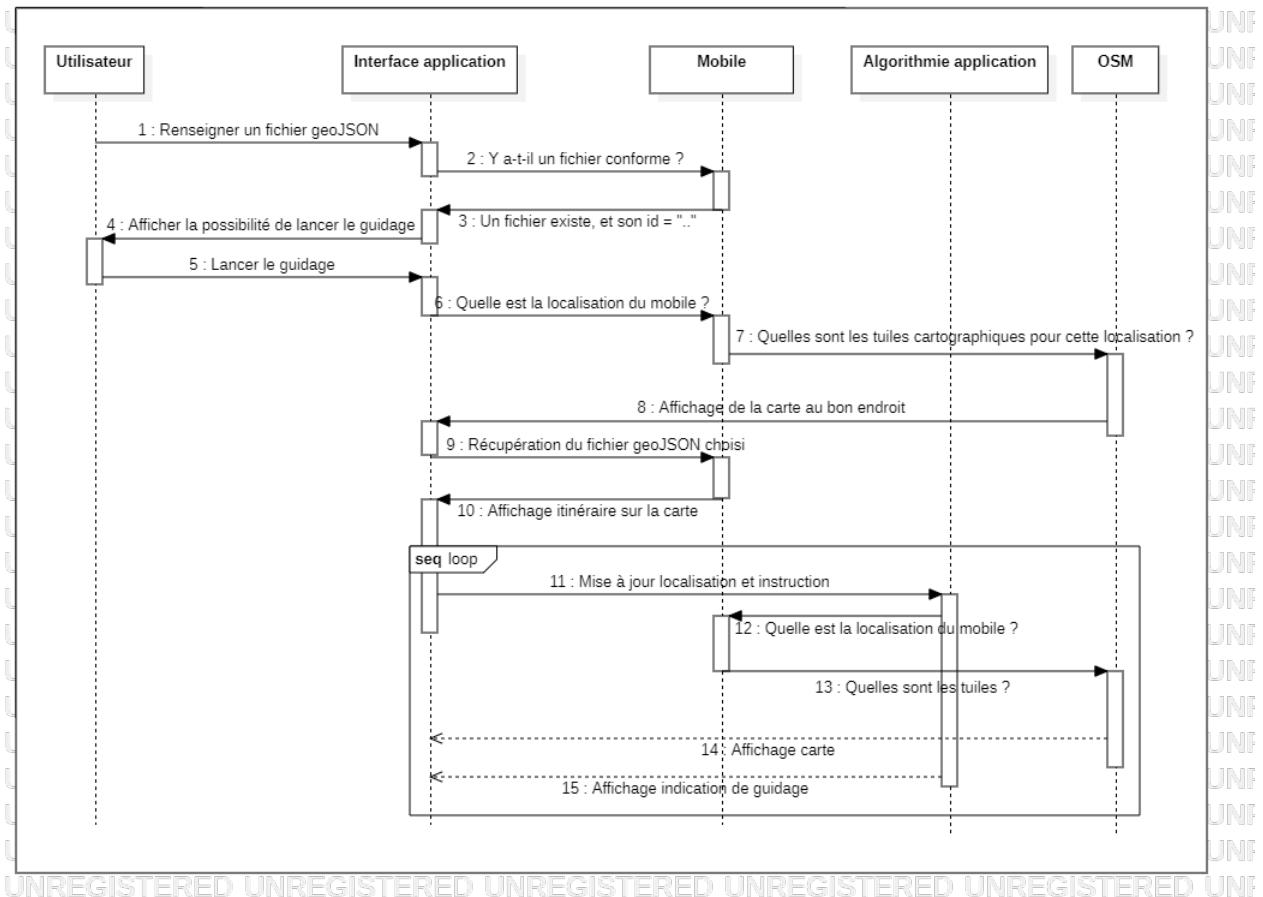


FIGURE 3.2 – Diagramme de séquence de l'application Android

## 3.2 Mise en oeuvre

### 3.2.1 Développement d'un algorithme de guidage

Pour répondre au besoin recueilli, il y a la nécessité d'ajouter à notre application un guidage en temps réel du recenseur. Pour cela, nous nous sommes renseigné sur les algorithmes existant qui permettraient éventuellement de prendre en entrée notre trajet et la localisation de l'utilisateur afin de pourvoir un guidage. N'ayant pas trouvé de solution concluante et facile à mettre en place, nous avons décidé de construire un algorithme de guidage simple mais efficace.

Nous considérons que pour guider le recenseur, il lui faut des instructions pour passer d'une étape du trajet, soit d'une route, à une autre. Pour ce faire, certaines indications sont nécessaires comme la distance restant à parcourir sur la présente étape, et l'orientation à prendre pour passer à la suivante. Par exemple : « Dans 300 mètres, tournez à gauche ».

Pour cela nous avons conçu une méthode reposant sur un calcul d'azimut.

Nous considérons seulement deux étapes pour générer une instruction. Cela implique que nous considérons trois points A, B et C avec B l'intersection entre les routes AB et BC, comme l'exempleifie la figure ci-dessous, qui montre différentes configurations possibles. On remarque par ailleurs que dans la troisième configuration, le point A et le point C sont confondus. Dans la réalité, il s'agit d'une situation de dmi-tour. Nous allons commencer par générer l'instruction d'orientation pour passer de

## 34 Conception et développement d'un outil de guidage et de visualisation des recensements

la première route à la seconde. Par la suite, il suffit d'appliquer la même méthode pour les tronçons BC et CD, et ainsi de suite.

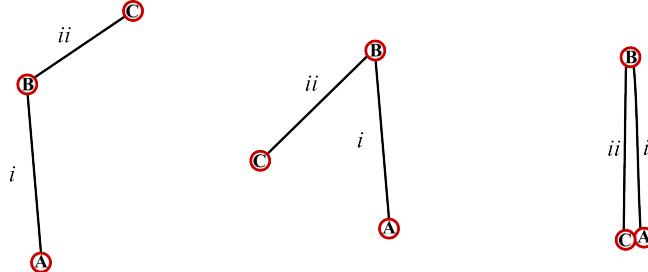


FIGURE 3.3 – Schéma de deux routes consécutives dans l'itinéraires, avec plusieurs configurations possibles

Considérons notre cas d'étude avec les points A,B et C dont on connaît les coordonnées. On commence par calculer les différences de longitudes et les azimuts de chaque tronçon en appliquant la formule suivante aux tronçons AB puis BC :

$$az\_AB = \arctan 2 (\sin(\Delta\text{lon}\_AB) \cdot \cos(\text{lat}\_B), \cos(\text{lat}\_A) \cdot \sin(\text{lat}\_B) - \sin(\text{lat}\_A) \cdot \cos(\text{lat}\_B) \cdot \cos(\Delta\text{lon}\_AB))$$

On réalise ensuite une différence entre l'azimut d'AB et celui de BC. Cela nous permet de calculer l'angle entre AB et BC. Nous pouvons ensuite effectuer une disjonction de cas en fonction des différentes mesures d'angle obtenues. Cette disjonction de cas nous permet de séparer les possibilités en six cas différents qui expliquent ce qu'il faut faire pour passer d'AB à BC, par exemple tourner à gauche, tourner à droite, faire demi-tour, etc. Nous présentons l'ensemble des possibilités dans le tableau ci-dessous.

Valeur de la différence d'azimuts (en °)	Indication d'orientation
Entre 0° et 15°	Allez tout droit
Entre 15° et 55°	Serrez à droite
Entre 55° et 125°	Tournez à droite
Entre 125° et 165°	Tournez franchement à droite
Entre 165° et 195°	Faites demi-tour
Entre 195° et 235°	Tournez franchement à gauche
Entre 235° et 305°	Tournez à gauche
Entre 305° et 345°	Serrez à gauche
Entre 345° et 360°	Allez tout droit

TABLE 3.1 – Tableau des indications d'orientation en fonction de la valeur de la différence des azimuts

Cela nous permet de récupérer les instructions pour chaque étape. À la dernière étape, on génère l'instruction « recensement terminé ».

Étant donné les nécessités de rapidité de calcul à la volée de l'application Android, nous avons décidé d'implémenter cet algorithme dans l'outil de planification, donc dans le plugin QGIS, et d'ajouter les instructions comme attributs de chaque étape de l'itinéraire créé. Ainsi, il est facilement récupérable dans l'application via le fichier geoJSON, mais ce choix s'explique surtout par sa commodité pour ne pas saturer les calculs à la volée.

### 3.2.2 Algorithmes de visualisation

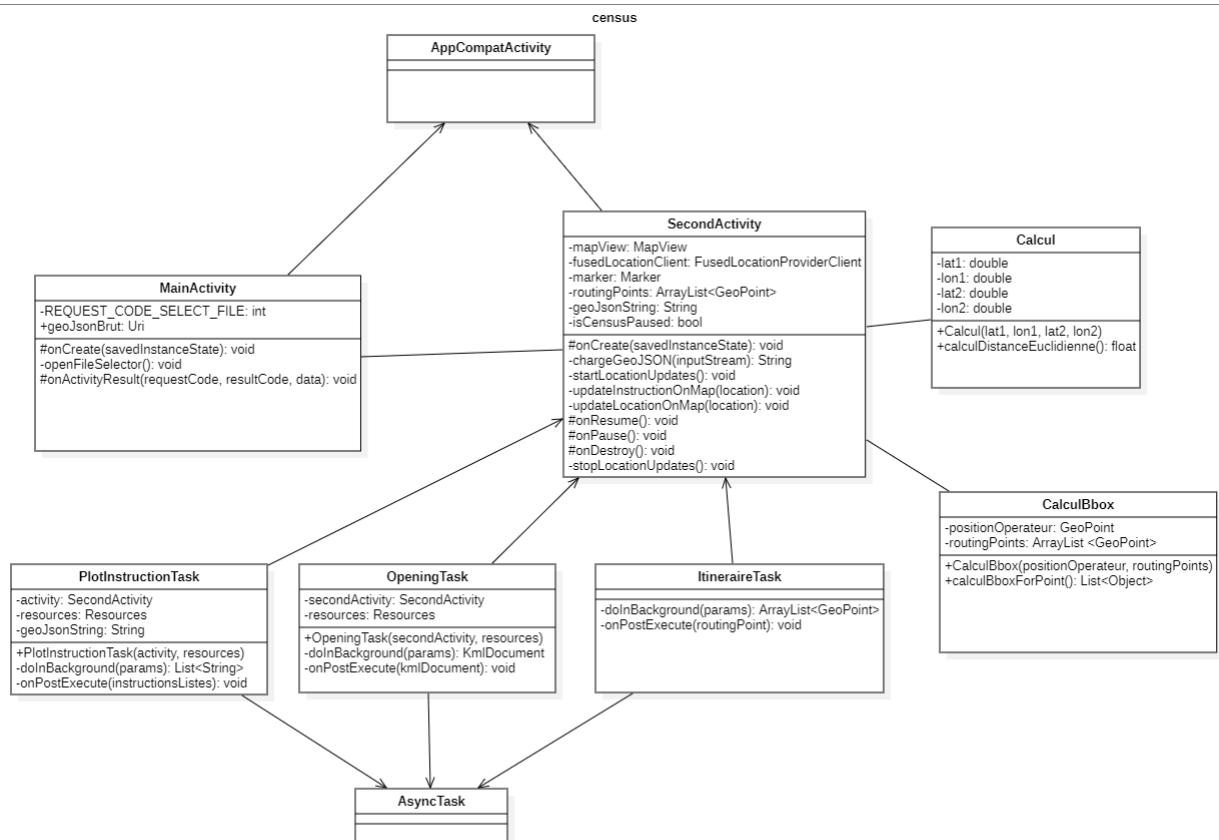
Le fichier geoJSON qui a été récupéré doit être représenté : il s'agit de l'itinéraire qui doit être suivi. Pour ceci, nous avons utilisé la bibliothèque Android Studio OSMdroid, qui implémente des fonctions appliquées aux fonds de cartes OSM. Ces fonctions nous ont permis de parcourir notre fichier geoJSON et de représenter chaque entité à partir de leur géométrie sur le fond de carte.

Ensuite, pour adapter la visualisation au parcours du trajet en temps réel, nous avons développé une modification de couleur de chaque entité à partir du moment où celle-ci est recensée. Ainsi, lorsque le recenseur arrive au bout d'une route, la distance calculée entre sa localisation et le bout de la route passe en dessous d'un seuil, défini empiriquement à trois mètres. Alors, on considère que la route a été parcourue une fois et donc qu'elle est recensée. Celle-ci prend alors la couleur rouge, ce qui veut dire qu'elle a été parcourue et recensée. Cela ne veut pas dire qu'elle ne sera plus parcourue pour un demi-tour ou encore pour rejoindre une autre route. Le recenseur peut tout de même être amené à retraverser la route, auquel cas elle ne changera pas de couleur et restera rouge.

Pour visualiser la position, l'algorithme principal de l'application met à jour la réception de la localisation environ toutes les dix secondes. Lors de cette réception, on met à jour la position d'un *marker* qui représente la localisation sur la carte. Aussi, on lance les fonctions de calcul de distance qui permettent d'adapter la visualisation de l'itinéraire en fonction des routes qui ont été recensées, et de celles qu'il reste à parcourir.

### 3.2.3 Mise en oeuvre de l'outil

Avec les mécaniques expliquées précédemment, nous pouvons assez facilement développer notre outil. Le diagramme de classe ci-dessous explique l'algorithme.



## 36 Conception et développement d'un outil de guidage et de visualisation des recensements

FIGURE 3.4 – Diagramme de classe de l'application Android

Nous avons deux classes "activité", qui représentent chacune une page de l'application. *MainActivity* est une classe qui crée l'application et permet à l'utilisateur de renseigner un fichier geoJSON en permettant l'accès aux fichiers internes du téléphone ou de la tablette. Elle permet également à l'utilisateur de lancer le guidage en cliquant sur un bouton qui entraîne la création de la *SecondActivity*. Celle-ci, visuellement, est ainsi :



FIGURE 3.5 – Extrait de l'application Android

On remarque deux parties. En haut de l'écran, un fond cartographique avec une visualisation de l'itinéraire global et de la position du recenseur. En bas, il y a un encadré avec l'instruction pour rejoindre la prochaine étape et une image qui illustre cette instruction.

Ce qui se passe, concrètement, c'est que au lancement de la *SecondActivity*, on affiche la carte. Puis, la classe asynchrone *OpeningTask* permet de convertir notre fichier geoJSON en format chaîne de caractère, lisible sur le logiciel AndroidStudio. Elle permet aussi de représenter l'itinéraire sur le fond de carte. Puis, on appelle la classe asynchrone *ItineraireTask* qui permet quant à elle d'éplucher le geoJSON pour créer une liste de points : on ne fait qu'y récupérer les longitudes et latitudes de chaque point pour les stocker dans un format plus facile de traitement. Ensuite, à chaque mise à jour de la localisation, on lance plusieurs calculs. La première chose que l'on fait, c'est lancer une fonction qui met à jour la position de l'opérateur sur la carte. On met alors aussi à jour l'instruction.

Pour générer cette instruction, le temps de calcul peut être assez long, c'est pourquoi nous avons aussi décidé de coder la majeure partie de la génération d'instruction de façon asynchrone. Ainsi, lors de la mise à jour de la position, on récupère celle-ci et on construit, pour chaque route, une *bbox*. On vérifie ensuite si la position est bien dans une *bbox*, ce qui permet de définir explicitement sur quelle route se situe l'opérateur. Une fois la route où il se situe déterminée, on peut calculer la distance entre la position de l'opérateur et la fin de la route pour estimer ce qui lui reste à parcourir, en mètres. Cela permet de compléter l'instruction. Tous ces calculs sont réalisés via la classe asynchrone *PlotInstructionTask* et c'est uniquement lorsque les calculs sont terminés que l'instruction s'affiche, et ce sans bloquer l'application.

### 3.3 Résultats

#### 3.3.1 Résultats visuels

L'ergonomie de l'application peut être observée par le biais des images suivantes. Tout d'abord, nous avons la page d'accueil de l'application qui est comme suit :

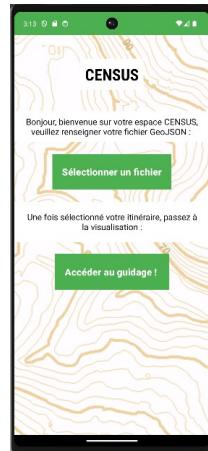


FIGURE 3.6 – Page d'accueil, extrait de l'application Android

Nous remarquons deux boutons, le premier permet de sélectionner son fichier geoJSON, et le deuxième permet de lancer le guidage. Lorsque l'utilisateur clique sur "sélectionner un fichier geoJSON", il peut choisir un fichier dans son téléphone :



FIGURE 3.7 – Sélection du fichier geoJSON, extrait de l'application Android

Une fois que l'utilisateur a cliqué sur le bouton « lancer le guidage », la page suivante s'affiche :



FIGURE 3.8 – Guidage du recenseur, extrait de l'application Android

## 38 Conception et développement d'un outil de guidage et de visualisation des recensements

On peut observer la localisation du recenseur sur la carte. Celui-ci est se trouve au début de l'itinéraire. En bas de l'écran, nous pouvons voir l'indication qui lui dit dans quelle distance il doit effectuer un changement de route et quelle orientation il doit suivre.



FIGURE 3.9 – Itinéraire en cours, extrait de l'application Android

On peut observer qu'au cours du recensement, les routes recensées s'affichent effectivement en rouge et que la distance est affichée en fonction de la localisation, diminuant jusqu'à aller à 0 mètres au bout de la rue.

### 3.3.2 Perspectives d'amélioration

Cette application consiste en une bonne base, mais elle peut être améliorée. D'abord, une amélioration est certaine en raison de la faible qualité des données géographiques représentées. En effet, le fond de carte OSM, qui n'est pas tout à fait à jour dans la zone de l'Afrique Subsaharienne. Par exemple, utiliser les fonds de cartes de Google Map, dont les services sont payants, permettrait une meilleure précision.

Aussi, il pourrait être intéressant d'utiliser un système comprenant une base de données remplie via l'exécution du plugin, et connectée à l'application. Ainsi, il faudrait, au lieu de charger un fichier geoJSON, renseigner son identifiant de recenseur et l'identifiant du secteur de dénombrement. Ces informations seraient suffisantes pour se connecter à une base de données et permettraient de récupérer les fichiers geoJSON calculés par le plugin QGIS. Cela permettrait une utilisation plus aisée de l'application.

La mise à jour de la localisation pourrait être plus fréquente, et la vérification de présence du recenseur sur un tronçon en particulier pourrait être affinée.

Enfin, de nombreuses options seraient intéressantes à ajouter en utilisant des données fournies par divers services. Par exemple, des informations sur l'état de danger d'une zone, des informations sur la praticabilité des routes en fonction des intempéries et de la météo en général, ou encore une visualisation des routes déjà recensées par d'autres opérateurs, pourraient ajouter beaucoup d'intérêt à l'application.

# Conclusion

---

Notre étude avait deux objectifs principaux. Le premier était la modélisation et la conception d'un outil de planification et d'optimisation des recensements. Le deuxième était la modélisation et la conception d'un outil de visualisation et de guidage des recensements. Nous souhaitions obtenir une analyse approfondie de ces outils, c'est-à-dire une modélisation UML, une réflexion sur les langages et les architectures. Nous souhaitions aussi avoir des logiciels implémentés et fonctionnels.

C'est ce travail de recherche, de conception et de mise en œuvre que nous avons mis en exergue au cours de ce rapport. Nous avons présenté une analyse fonctionnelle de l'outil de planification et d'optimisation, une modélisation UML et une méthode d'implémentation. Nous avons aussi présenté des fonctions de parcours optimal, le logiciel conçu, et des études des résultats produits. De la même manière, nous avons présenté l'analyse fonctionnelle et la modélisation UML de l'outil de visualisation et de guidage. Nous avons aussi montré la méthode suivie pour son implémentation, le logiciel conçu et les résultats obtenus.

Des perspectives d'amélioration sont à noter. Il est possible, comme nous l'avons expliqué au fil de ce rapport, d'améliorer sensiblement les analyses et implémentations qui ont été faites. À long terme, il serait intéressant d'adapter ce logiciel à la zone traitée. En effet, pour l'instant, les logiciels conçus sont adaptés pour des zones en Afrique subsaharienne, mais ils pourraient être utilisables pour différentes zones. Aussi, les fonctions de parcours optimal sont intéressantes et pourraient, en plus d'être affinées, réutilisées dans d'autres cadres. Par exemple, elles pourraient être utilisées dans la construction d'itinéraires pour la collecte des déchets.



# Bibliographie

---

- [1] E. BAMPIS. *Algorithmes gloutons*. Rapp. tech. UPMC, 2017.
- [2] Nicolas ISOART. "Le problème du voyageur de commerce en programmation par contraintes". In : *Université de la Côte d'Azur* (2021).
- [3] G. LABIALE. "Approche psycho-ergonomique des systèmes d'information et de guidage routier des conducteurs automobiles". In : *IFSTTAR* (1984).
- [4] Jérôme MONNOT. "Le voyageur de commerce et ses variations : un tour d'horizon de ses résolution". In : *Hermes science* (2007), p51-93.
- [5] QGIS. *Documentation PyQGIS*. PyQGIS. 2022.
- [6] SPYHUNTER99. *Documentation osmdroid*. OSM. 2022.



# Table des figures

1.1	Rétroplanning élaboré en début de stage . . . . .	12
1.2	Diagramme de Gantt complété au fur et à mesure du stage . . . . .	12
2.1	Secteurs de dénombrement du Gada, extrait de QGIS . . . . .	15
2.2	Réseau routier du Gada, extrait de QGIS . . . . .	16
2.3	Hydrographie du Gada, extrait de QGIS . . . . .	16
2.4	Modèle Numérique de Terrain issu de la SRTM couvrant l'état de Sokoto, extrait de QGIS . . . . .	17
2.5	Diagramme de cas utilisation du plugin QGIS . . . . .	18
2.6	Diagramme de composant du plugin QGIS . . . . .	19
2.7	Diagramme de séquence du plugin QGIS . . . . .	19
2.8	Diagramme d'activité du fonctionnement général du plugin QGIS . . . . .	20
2.9	Schéma d'un noeud et de son voisinage . . . . .	22
2.10	Diagramme d'activité de la fonction de parcours optimal . . . . .	23
2.11	Schéma expliquant le rattachement d'un petit itinéraire à un grand . . . . .	24
2.12	Diagramme d'activité du workflow du plugin QGIS . . . . .	26
2.13	Interface du plugin QGIS créé . . . . .	27
2.14	Extrait de QGIS présentant les sorties du plugin sur la zone totale à planifier . . . . .	27
2.15	Extrait de QGIS présentant les sorties du plugin sur un secteur de dénombrement . . . . .	27
2.16	Extrait de QGIS présentant les sorties du plugin au niveau d'un centre-ville . . . . .	28
3.1	Diagramme de cas utilisation de l'application Android . . . . .	32
3.2	Diagramme de séquence de l'application Android . . . . .	33
3.3	Schéma de deux routes consécutives dans l'itinéraires, avec plusieurs configurations possibles . . . . .	34
3.4	Diagramme de classe de l'application Android . . . . .	35
3.5	Extrait de l'application Android . . . . .	36
3.6	Page d'accueil, extrait de l'application Android . . . . .	37
3.7	Sélection du fichier geoJSON, extrait de l'application Android . . . . .	37
3.8	Guidage du recenseur, extrait de l'application Android . . . . .	37
3.9	Itinéraire en cours, extrait de l'application Android . . . . .	38



# Liste des tableaux

3.1	Tableau des indications d'orientation en fonction de la valeur de la différence des azimuts . . . . .	34
B.1	Taux de redondance (en %) du parcours en fonction du nombre de noeud du graphe (colonnes) et du nombre d'itérations (lignes) . . . . .	51



# **Annexes**

A	Processus de recensement	49
B	Statistiques sur l'optimalité	51



# PROCESSUS DE RECENSEMENT

ANNEXE

A



Processus de recensement en France, extrait du site du gouvernement



# STATISTIQUES SUR L'OPTIMALITÉ

ANNEXE **B**

Nombre de noeuds du graphe	1	5	10	20	50	100	500	1000
2	0	0	0	0	0	0	0	0
134	34	34	33	33	33	33	33	33
260	30	28	28	27	27	26	26	25
3	0	0	0	0	0	0	0	0
12	25	25	25	25	25	25	25	25
35	43	42	42	42	42	42	42	42
67	40	39	38	38	38	38	38	38
8	24	24	24	24	24	24	24	24
32	38	37	37	37	37	37	37	37
4	11	11	11	11	11	11	11	11
61	30	29	24	23	23	23	23	23
33	33	32	32	32	32	32	32	32
42	30	28	28	28	28	28	28	28
53	35	35	35	35	35	35	35	35

TABLE B.1 – Taux de redondance (en %) du parcours en fonction du nombre de noeud du graphe (colonnes) et du nombre d'itérations (lignes)