# Rapport de projet de Recherche opérationnelle

# Base de donnée 14

**ASI 4**
Thomas Boucher
Simon Brunet
Tom Masson

Année : 2017-2018
À l'attention de M.Mainguenaud

# Table des matières

# Chapitre 1

# Introduction

Dans le cadre de l'EC Théorie des Graphes nous avons eu un projet à réaliser. L'objectif de ce projet était de prendre en main la bibliothèque de graphes java GraphStream dans le but de tracer un graphe. Ce graphe est issu des données mises à disposition sur le serveur asi-pg.insa-rouen.fr. Utilisées via un JDBC postgres et la base de donnée orange-14, le graphe représente les données pour les différentes unités de temps (minutes, heures, jours ou mois) dans un contexte de spacialisation(communes ou départements).

# Chapitre 2

# Conception

Avant de commencer à coder le projet il nous a fallut effectuer un tutoriel pour comprendre comment utiliser la bibliothèque GraphStream. Nous avons ainsi apprit à implémenter des nœuds, les relier entre eux, construire des graphes. Une fois ce tutoriel effectué et acquis par chacun nous avons pu commencer à implémenter le projet.

Il se compose des classes CustomGraph, Database et Point, deux énumérations UniteSpatiale et UniteTemporelle et enfin notre Main pour exécuter le programme.

## 2.1 Classes

### 2.1.1 CustomGraph

La classe CustomGraph est la classe permettant de créer des graphes et de les afficher. Elle est la classe principale de notre projet.Elle est composée d'un constructeur basique qui prend en entrée une Database, une UniteTemporelle et une UniteSpatiale. Une méthode display() qui construit et affiche le graphe voulu en fonction des paramètres choisis. et une méthode toString().

### 2.1.2 Database

La classe Database dispose d'un constructeur qui permet une connexion à notre base de données, une méthode logout() pour se déconnecter, une méthode createView() qui permet de créer ou remplacer une vue à partir d'une UniteTemporelle et une UniteSpatiale elle utilise la méthode getRequest() qui va générer une requête suivant l'UniteTemporelle et renverra la requête SQL associée. La méthode getCommunes() qui permet de retourner la liste des points représentant les frontières des communes et départements. La méthode getDepartements() retourne la liste

des points représentant les frontières des départements. Et les deux dernières méthodes getPointsLocalisation() et getPointsFrontiere() permettant d'obtenir respectivement les points de la localisation et des villes et frontières et de les stocker dans une collection.

### 2.1.3 Point

Cette classe permet simplement de créé un point avec des coordonnées x et y utilisé par la suite dans les classes CustomGraph et Database.

### 2.1.4 Enumerations

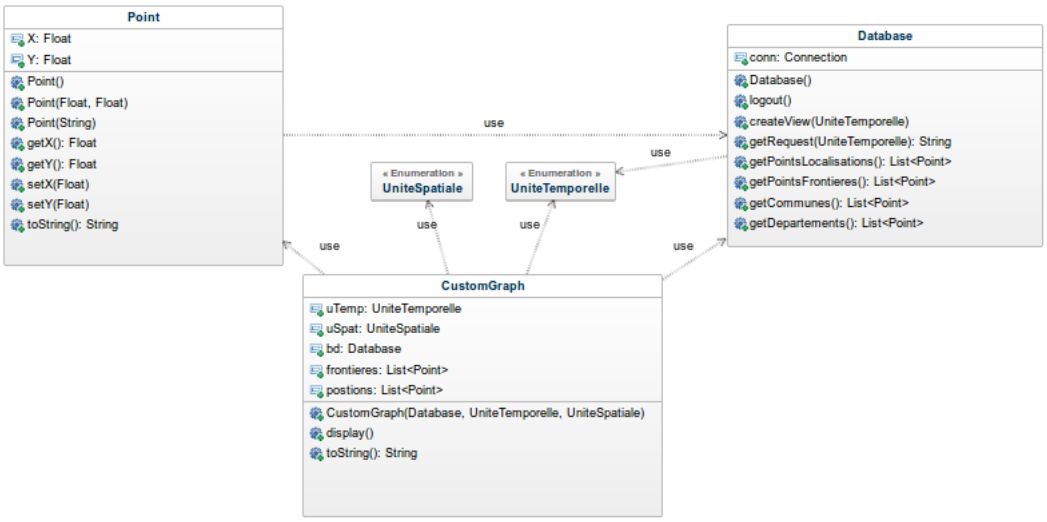
L'énumération UniteTemporelle est composée de MOIS, JOUR, HEURE, MINUTE et l'énumération UniteSpatiale est composée de COMMUNES et DEPARTEMENTS.

### 2.1.5 Main

Notre Main qui lance le programme est composé d'un menu qui permet de choisir les paramètres voulus et enfin fait appel à la méthode display() de CustomGraph afin de construire et afficher le graphe demandé.

## 2.2 Diagramme de classes

FIGURE 2.1 – Diagramme de classes



# Chapitre 3

# Analyse volumetrique

Grâce à la requête

```
select count(*) from spatialisation;
```

nous pouvons voir que la table spatialisation de la base de donnée orange-14 comporte 258577 tuples. exactement.
À l'aide la requête

```
select count(distinct location) from spatialisation;
```

nous pouvons voir que nous avons 2156 points de localisation différents, et donc que de nombreuses positions sont identiques dans la base. Cela s'explique par le fait que les points sont relevés avec un intervalle de 10 secondes, et que la personne est immobile parfois (la nuit par exemple).

Pour traiter ces données avec la plus grande efficacité, nous avons créé une vue "Points" qui extrait de la table spatialisation les données en fonction des paramètres temporel souhaité.

# Chapitre 4

# Répartition des tâches

TABLE 4.1 – Tableau de répartition des tâches

| **Tâche** | **Responsable** |
|---|---|
| CustomGraph | Simon |
| Database | Thomas, Tom |
| Main | Thomas, Simon, Tom |
| Rapport | Thomas, Simon, Tom |

# Chapitre 5

# Conclusion

Ce projet nous a permis de prendre en main la bibliothèque java GraphStream afin de l'utiliser sur des données réelles stockées en base de données. La prise en main de cette bibliothèque n'a pas été aisée, nous n'avons trouvé que peu d'exemples d'utilisation de celle-ci. Ainsi cela nous a pris du temps afin de pouvoir avancer sur le projet. Une fois cette dernière maitrisée, il nous a fallu accéder aux données via un JDBC. Cependant, des mesures de sécurité prises par l'INSA nous ont empéché d'avoir accès à ces données depuis l'extérieur du réseau. De ce fait, nous avons du passer par un client graphique de type X2GO afin de pouvoir tester notre code. Nous avons géré ces difficultés aussi bien en nous réunissant afin de résoudre de grosses difficultés, qu'en nous répartissant les tâches dans le but d'avoir une répartition équitable.

# Table des figures