PROJET : my-way
un outil simple et efficace pour la gestion d'itineraire dans une ville donnee

Ecole Nationale Supérieure Polytechnique de Yaoundé (ENSPY)
Département de GENIE INFORMATIQUE

SUPERVISEURS :
Pr DJOTIO
M. Juslin KUTCHE

7 avril 2024

Etudiants :
NOMO BODIANGA Gabriel Nasaire Junior (23P753)
KAMDEM POUOKAM Ivann Harold (21P254)
DONCHI Tresor Leroy (21P360)
DJONGO FOKOU Ariel Sharon (21P360)
classe de 3ème année

Table des matières

1 Introduction

Dans le cadre du développement des infrastructures de transport en milieu inter-urbain, la mise en place d'une plateforme de gestion des itinéraires se présente comme une solution innovante pour répondre aux défis de mobilité urbaine. Ce projet vise à concevoir une application capable de faciliter la planification des voyages et d'optimiser les parcours au sein de la ville de Yaoundé.

La ville de Yaoundé, caractérisée par son dynamisme et sa croissance démographique, se trouve confrontée à des enjeux majeurs en termes de gestion de trafic et de planification des déplacements. L'objectif de ce système est de proposer des itinéraires optimaux, en tenant compte des divers facteurs tels que le temps de parcours, la distance, et la densité de trafic, afin d'améliorer l'expérience des usagers et de contribuer à la fluidité du transport interurbain.

Le présent cahier d'analyse et de conception documentera de manière exhaustive les besoins, les exigences, et les spécifications techniques du système. Il servira de référence tout au long du cycle de vie du projet, depuis la phase d'analyse préliminaire jusqu'à la validation finale du produit. Ce projet, tout en se concentrant initialement sur la gestion des itinéraires dans la ville de Yaoundé, pourra éventuellement s'intégrer dans un projet plus grand de gestion des voyages en milieu interurbain.

2 Contexte & Besoins

Il s'agit ici pour nous de présenter la situation actuelle de mobilité dans la ville de Yaoundé, et de relever les besoins qu'une application de gestions des itinéraires serait amenée à résoudre.

2.1 Contexte

Vu la structure centrée de la ville de Yaoundé et de l'usage exclusif de moyens de transports à faible capacité (taxis et motos), celle-ci est très fréquemment le théâtre de congestion et de difficultés de déplacements, ce qui est un problème qui prend de plus en plus de l'ampleur. D'autant plus que l'usage des bus (plus appropriés pour résoudre le problème), ne se ressent quasiment plus.

Face à une ville avec une structure inadéquate pour les transports de masse, et très souvent aussi à l'incivisme des conducteurs d'automobile, une plate-forme accessible à tous, qui permettrait de repartir de manière uniforme le trafic dans la ville s'avère alors utile.,Mais alors, quelles fonctionnalités devrait avoir une telle solution en vue de résoudre optimalement le problème posé, et en vue de mieux s'intégrer dans la vie de tous les jours ? C'est ce dont nous en parlerons dans la suite

2.2 Fonctionnalités Clés

Au vu de la situation de mobilité décriée plus haut il s'agit pour nous ici de relever des besoins en matière d'application de gestions des itinéraires dans le cadre de lla ville de Yaoundé

- **Calcul d'itinéraire intelligent :** L'application devrait permettre aux utilisateurs de saisir leur point de départ et leur destination, puis générer une liste d'itinéraires classés par ordre d'optimalité en tenant compte des modes de transport disponibles (bus, taxis, moto-taxis, etc.). L'application pourrait aussi proposer le calcul d'itinéraire par filtre (distance, coût, surcharge de la route, état de la route, etc.).
- **Calcul d'itinéraire en temps réel :** Le calcul d'itinéraire optimal pour l'utilisateur devrait pouvoir se faire automatiquement en temps réel lorsqu'il se déplace par rapport à son point d'arrivée.
- **Cartographie détaillée :** - Afficher des cartes interactives avec des points d'intérêt, des arrêts de transport, des stations-service, etc. - Proposer des vues 3D pour faciliter la navigation.
- **Informations en temps réel :** Fournir des mises à jour en temps réel sur les conditions de circulation, les retards et les incidents sur les routes pour permettre aux utilisateurs de planifier leurs déplacements en conséquence, et même de changer d'itinéraire en cas de problème.
- **Indications précises :** Fournir des indications précises et détaillées, y compris les points d'intérêt, les intersections, les changements de direction, etc., pour aider les utilisateurs à naviguer facilement dans la ville.

— **Navigation hors ligne :** Permettre aux utilisateurs de télécharger des cartes et des itinéraires pour une navigation hors ligne dans les zones où la connectivité Internet est limitée.
— **Collaboration avec d'autres services :** L'application devrait permettre de collaborer avec d'autres services tels que la collecte des clients sur un itinéraire, le suivi d'un voyage, et autres.
— **Statistiques de déplacement et IA :** L'application pourrait enregistrer les différentes données statistiques de déplacement de ses utilisateurs qui pourraient aider à des prises de décision et même aux prédictions sur le temps de parcours, la surcharge de certaines routes, et proposer un modèle d'intelligence artificielle pour différentes prévisions et conseils.
— **Suivi personnalisé et Synchronisation avec le calendrier :** L'application devrait pouvoir utiliser les données et préférences de déplacement d'un utilisateur pour lui proposer un service client personnalisé automatique et lui offrir des conseils ; cette partie peut également être basée sur l'IA. L'application devrait également utiliser le calendrier des utilisateurs pour gérer les itinéraires de voyage en fonction de leur emploi du temps.
— **Proposition de réservations :** Afin d'éviter les désagréments souvent causés par des routes bloquées par le gouvernement lors des journées nationales ou lors du déplacement du chef de l'État, l'application devrait pouvoir permettre au gouvernement de réserver des routes à cet effet afin d'éviter de pénaliser les usagers et d'exploiter au mieux les routes libres dans ces situations.
— **Partage d'itinéraire :** Permettre aux utilisateurs de partager facilement leur itinéraire de voyage avec des amis ou des membres de la famille.
— **Intégration des avis :** L'application devrait incorporer des avis et des recommandations d'autres voyageurs sur des étapes spécifiques de l'itinéraire.

3 Modélisation mathématique du problème

Notre modelisation mathematique commence par le choix des outils mathematiques utilises (**les graphes**), passe par le formalisme et les analogies et se terminent par le choix des algorithmes permettant de resourdre le probleme.

Cependant,justifons d'abord le choix de la **theorie des graphes**

1. Tout d'abord,nous devons souligner que la theorie des graphes est un outil simple,facile a comprendre et qui illustre mieux comment representer une carte,des routes et autres.
2. De plus,le probleme que nous devons resourdre s'assimile directemnet aux poblemes de parcours en theorie des graphes
3. par ailleurs,La théorie des graphes offre des outils puissants pour modéliser les connexions entre les différents points du réseau routier et les déplacements des véhicules d'un point à un autre.

3.1 Un peu de théorie des graphes

A fin de faciliter la comprehension de notre modele,nous allons bievrement definir expliquer les concepts de base de la theorie des graphes utiles pour notre modelisation.

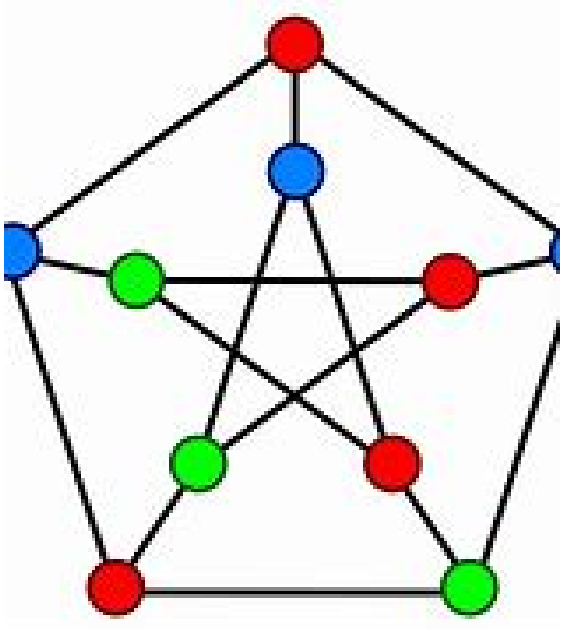


FIGURE 1 – La theorie des graphes
?

3.1.1 Qu'est ce que de théorie des graphes ?

La théorie des graphes est une branche des mathématiques qui étudie les relations entre les objets. Les objets sont représentés par des points, appelés sommets, et les relations entre eux sont représentées par des lignes, appelées arêtes. Voici quelques concepts de base :

3.1.2 Terminologie

— **Graphe :** Ensemble de sommets et d'arêtes.
— **Sommets (ou nœuds) :** Les points d'un graphe.
— **Arêtes :** Les lignes reliant les sommets.
— **Graphe orienté :** Un graphe dans lequel les arêtes ont une direction.
— **Graphe non orienté :** Un graphe dans lequel les arêtes n'ont pas de direction.
— **Graphe mixte :** Un graphe dans lequel certaines arêtes peuvent être orientées et d'autres non, et où les sommets peuvent être attribués à différents types ou classes.
— **Graphe pondéré :** Un graphe dans lequel chaque arête est associée à un poids ou une valeur. Ces poids peuvent représenter des distances, des coûts, des temps, etc., et sont utilisés pour optimiser les algorithmes de recherche de chemins tels que Dijkstra et A* qui seront presentes ci bas.

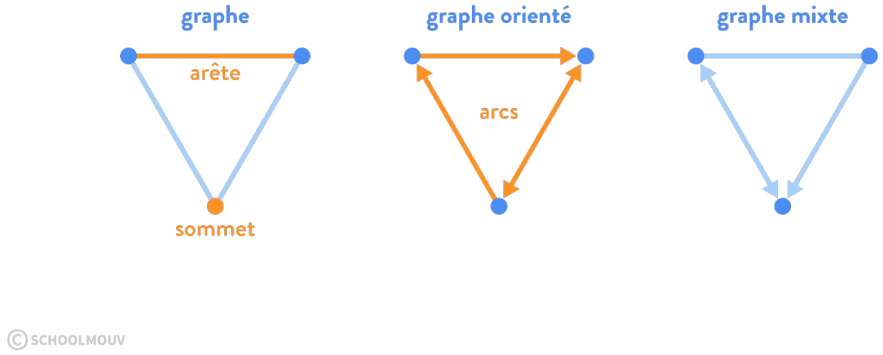


FIGURE 2 – Terminologie theorie des graphes
?

3.1.3 Types de graphes

Il existe plusieurs types de graphes, dont les plus courants sont :

Graphe simple : Un graphe sans boucles ni arêtes multiples entre les mêmes sommets.

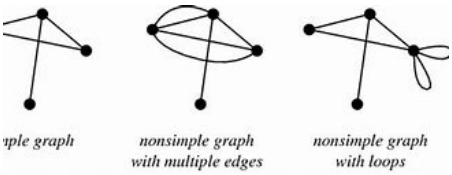


FIGURE 3 – graphe simple

?

Graphe complet : Un graphe dans lequel chaque paire de sommets est reliée par une arête.

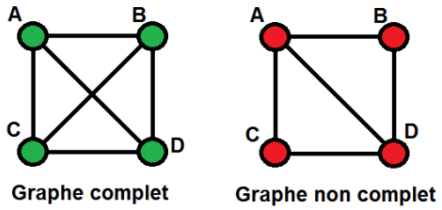


FIGURE 4 – graphe complet

?

Graphe biparti : Un graphe dont les sommets peuvent être divisés en deux ensembles disjoints, et où chaque arête relie un sommet d'un ensemble à un sommet de l'autre ensemble.

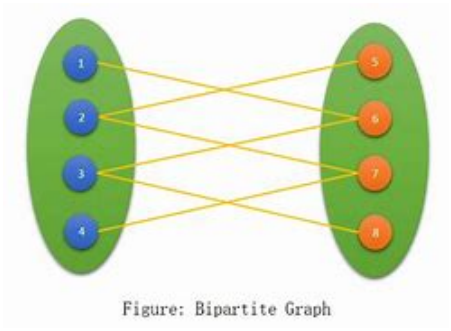


FIGURE 5 – graphe bipartite

La théorie des graphes offre de nombreux outils et concepts pour modéliser et résoudre une variété de problèmes dans divers domaines, tels que les réseaux sociaux, les réseaux informatiques, les itinéraires de transport, etc.

3.2 Formalisme et modelisation de l'environnement

A fin de mieux modeliser notre probleme,nous allons faire une analogie avec **la theorie des graphes**.Ainsi,les entites principales de notre carte sont :

- **Les points d'arret :** qui representent simplement les differents points d'arret d'un vehicules/clients sur une carte.Il s'agit des carrefours,stations services,pharmacies,restaurants,ba qui pourraient servir de destination dans une ville donnee.
- **Les routes** qui representent simplent les liens entre ces points,une route peut etre vue comme un support permettent de se deplacer d'un point a un autre
- **les itineraires** qui representent simplement des sucsessions ordonnes de routes peremttant d'aller d'un poin a un autre. **les contraintes** comme **le distance,le temps,la rigueur d'une route,l'encombrement,....**

Maintenant ,voici notre facon de modeliser ces entites :

- **Notre carte** sera assimillee a **un graphe** mathematique.
- **Les points d'arret** seront assimiles aux **sommets** du graphe.
- **Les routes** seront vues ici de facon basique comme des **arretes** et des **chemins sur le graphe**.**le deplacment sur une route est vu comme une relation entre 2 sommets**
- **les itineraires** seront simplement vus comme une succession de points lies.IL s'agira simplement d'une notion assimilable a une route mais un peu plus large.
- **les contaraintes sur une route** serviront ici a construire **le poids du chemin correspondant**.A chaque fois, en fonction des contraintes utilisees pour un filtre de recherche,nous construirons une metrique composite des contraintes associees qui permettra de definir le poids du chemin.

Dans cette lancee,notre probleme de gestion et de parcours d'itineraire se ramene simplement au probleme de **gestion et parcours d'un graphe en mathematique.**

3.3 Algorithmes de Parcours d'un Graphe

les differents facteurs nous permettant de choisir les algorithmes de parcours de notre graphe sont :

- **l'architechture reseau de la ville de yaounde :** en effet la ville de yaounde a une architechture reseau en etoile centree.
- **les besoins de parcours lies aux filtres de recherches** tel que la distance,l'optimalite,l'encombrement.
- **les contraintes d'equite :** pour eviter que les clients se dirigent tous vers le meme lieu au meme moment.

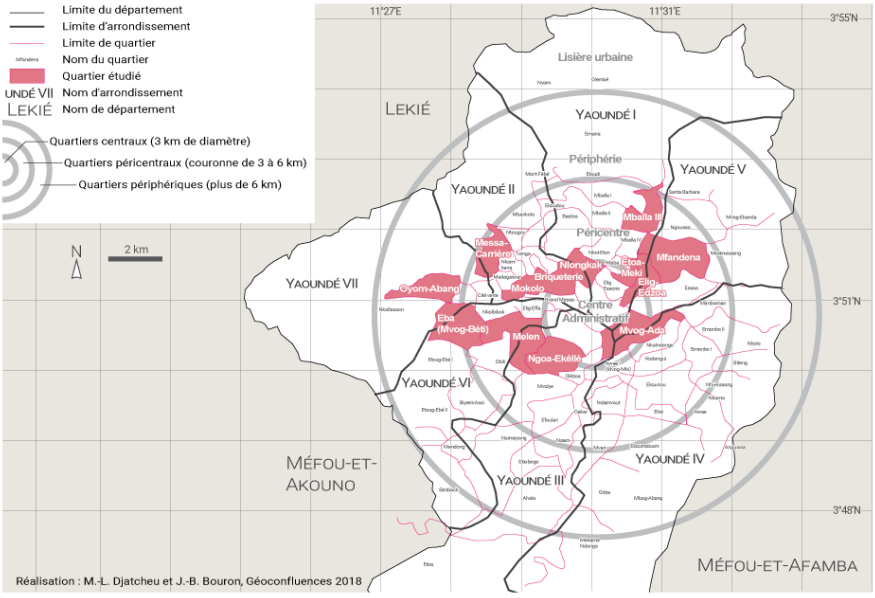


FIGURE 6 – architechture reseau ville de yaounde

-Parlons maintenant des differents algorithmes de parcours qui nous interessent :

Parcours en Largeur (BFS - Breadth First Search)

- **Principe** : Le BFS explore les nœuds du graphe en utilisant un ordre de visite en largeur. Il commence par le nœud de départ, puis visite tous ses voisins avant de passer aux voisins de ces voisins, et ainsi de suite.
- **Spécificités** :
  - Utilise une file (FIFO) pour gérer l'ordre de visite des nœuds.
  - Trouve le plus court chemin entre deux nœuds non pondérés.
  - Convient pour la recherche de la solution la plus rapide.

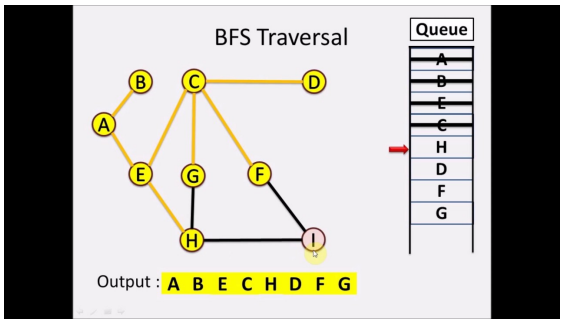


FIGURE 7 – algorithme BFS

Parcours en Profondeur (DFS - Depth First Search)

- **Principe** : Le DFS explore les nœuds du graphe en utilisant la récursivité. Il visite les nœuds les plus "profonds" en premier, puis remonte progressivement dans le graphe.
- **Spécificités** :
  - Utilise la pile (LIFO) pour gérer l'ordre de visite des nœuds.
  - Peut être utilisé pour détecter des cycles dans un graphe.
  - Ne garantit pas le plus court chemin.

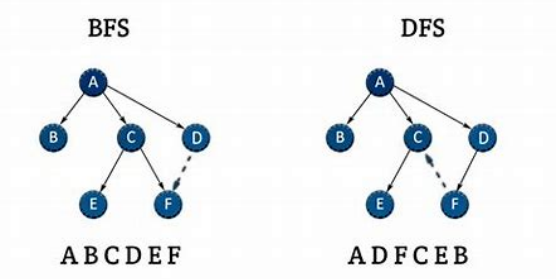


FIGURE 8 – algorithme DFS

Algorithme de Dijkstra

— **Principe** : L'algorithme de Dijkstra permet de trouver le plus court chemin entre deux sommets d'un graphe (orienté ou non orienté). Il choisit le sommet non visité avec la distance la plus faible, calcule la distance à travers lui pour chaque voisin non visité, et met à jour la distance du voisin si elle est plus petite.
— **Spécificités** :
  — Utilise un tableau de mémoire pour garder en mémoire les distances minimisées.
  — Convient pour les graphes orientés pondérés par des réels positifs.
  — Peut également calculer un plus court chemin entre un sommet de départ et un sommet d'arrivée.

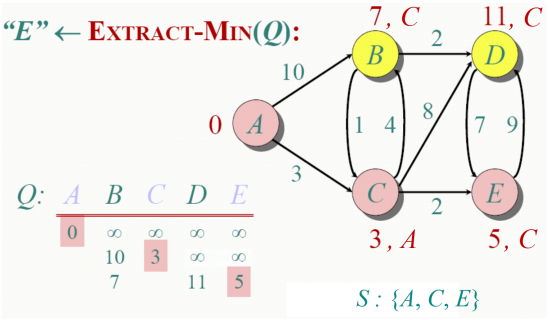


FIGURE 9 – Algorithme de Dijkstra

Algorithme A*

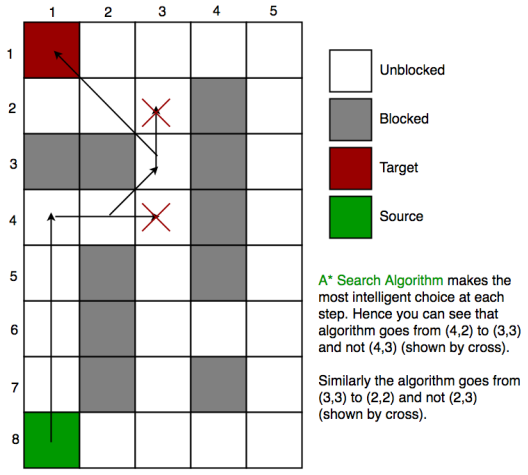


FIGURE 10 – Algorithme A*

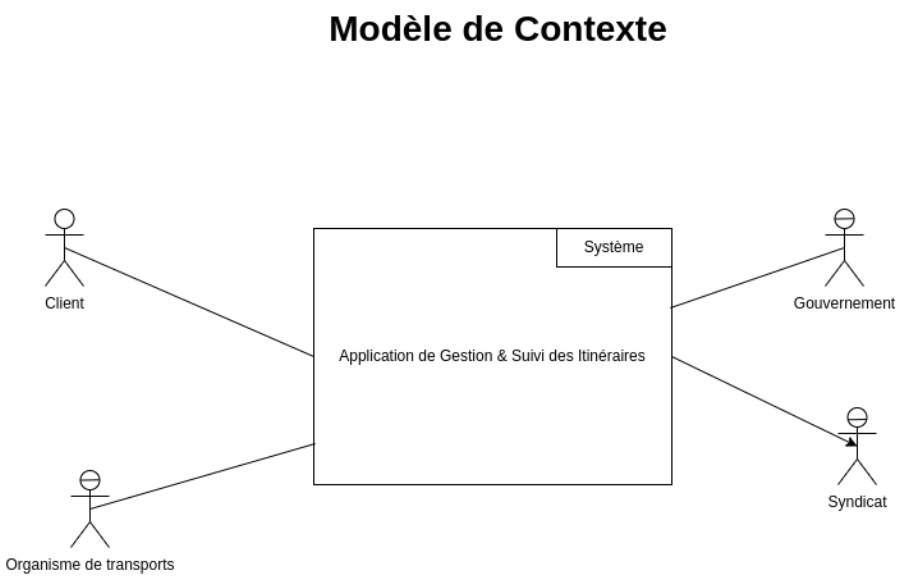
— **Principe** : L'algorithme A* utilise une fonction heuristique pour estimer le coût restant à parcourir pour atteindre le sommet cible à partir du sommet actuel. Il combine cette estimation avec le coût réel parcouru jusqu'à présent pour évaluer les sommets à explorer en priorité. A* explore les sommets avec le coût total le plus faible en priorité.
— **Spécificités** :
  — Utilisation d'une fonction heuristique : L'algorithme A* utilise une fonction heuristique qui fournit une estimation du coût restant pour atteindre le sommet cible. Cette fonction doit être admissible (ne jamais surestimer le coût restant) pour garantir l'optimalité de l'algorithme.
  — Efficace pour la recherche de chemins dans les graphes avec des coûts non uniformes : L'algorithme A* peut être plus efficace que Dijkstra dans les graphes où les coûts des arêtes varient et nécessitent une exploration plus intelligente.

En résumé, le choix de l'algorithme dépend du problème spécifique que vous essayez de résoudre. Le BFS est idéal pour les chemins les plus courts, tandis que le DFS est plus souple pour explorer toutes les possibilités. L'algorithme de Dijkstra est puissant et polyvalent pour résoudre les problèmes de plus court chemin. L'algorithme A*, quant à lui, offre une approche efficace pour trouver des chemins dans les graphes avec des coûts non uniformes, en utilisant une fonction heuristique pour guider la recherche. Celui que nous allons souvent utiliser dépendra à chaque fois de la tâche que nous voudrons accomplir lors de la gestion des sommets sur notre graphe.

4 Modélisation UML

En vu d'être claire et concis sur ce que fournira notre plateforme, il s'est avéré necessaire, une fois la modélisation mathématique terminée, de s'interesser à une modélisation orientée objet en UML. C'est ainsi que pour un début, nous avons conçu les diagrammes d'analyse statique qui suivent :

4.1 Diagramme de contexte

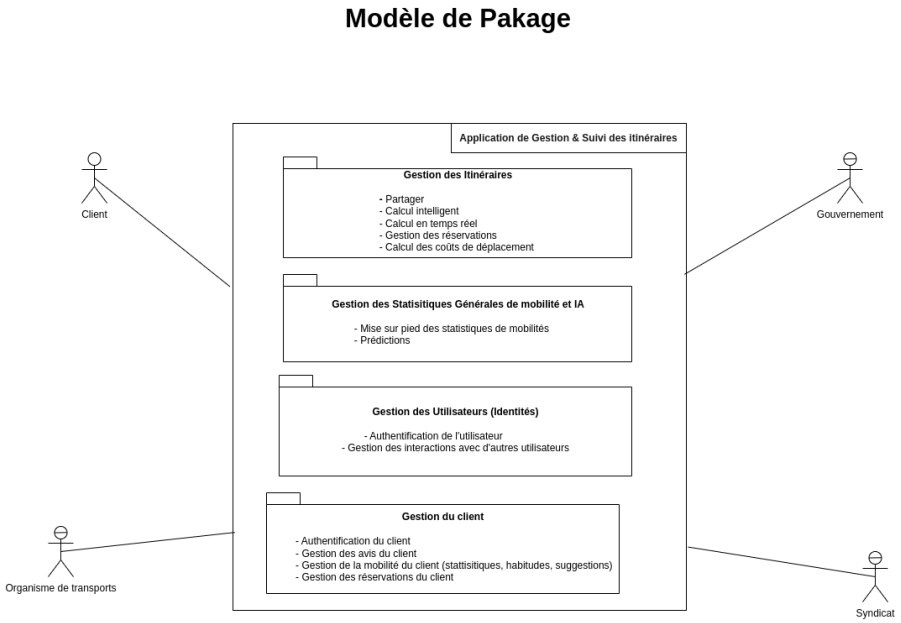


Ce diagramme nous permet de représenter notre système dans son environnement, d'avoir une vue externe, globale de ses interactions avec différents types d'acteurs. Au vu de ce diagramme, on distingue :

- **Système :** Il s'agit de notre application de gestion itinéraires, représenté comme une boite noire, dont le contenu se spécifiera avec les diagrammes qui suivront
- **Acteurs principaux :** Il s'agit des acteurs pour qui est dédié le système. On distingue ainsi
  - Le Client : Il s'agit de celui qui vient sur l'application pour avoir le ou les meilleurs itineraire(s) pour se déplacer d'un point A à un point B. Dans le cadre du projet entier, il s'agira d'un conducteur tout court.
  - Les Organismes de Transport : On voit ici d'éventuelles agences de transports qui pourront faire appel à nos services pour optimiser les déplacements de leurs véhicules
- **Acteurs secondaires :** Ce sont les acteurs qui pourront échanger des informations avec le système, pour son bon fonctionnement. On y retrouve
  - Le Gouvernement : Ce sera bien sûr, le régulateur des activités ; il pourra même nous fournir des données en matière de mobilité dans un zone spécifique,

en vue de l'optimisation des itinéraires.
— Syndicat : On parle ici du syndicat des transporteur, qui sera également d'une aide précieuse en vue d'assurer la conformité des conducteurs utilisant la plateforme.

4.2 Diagramme de Package

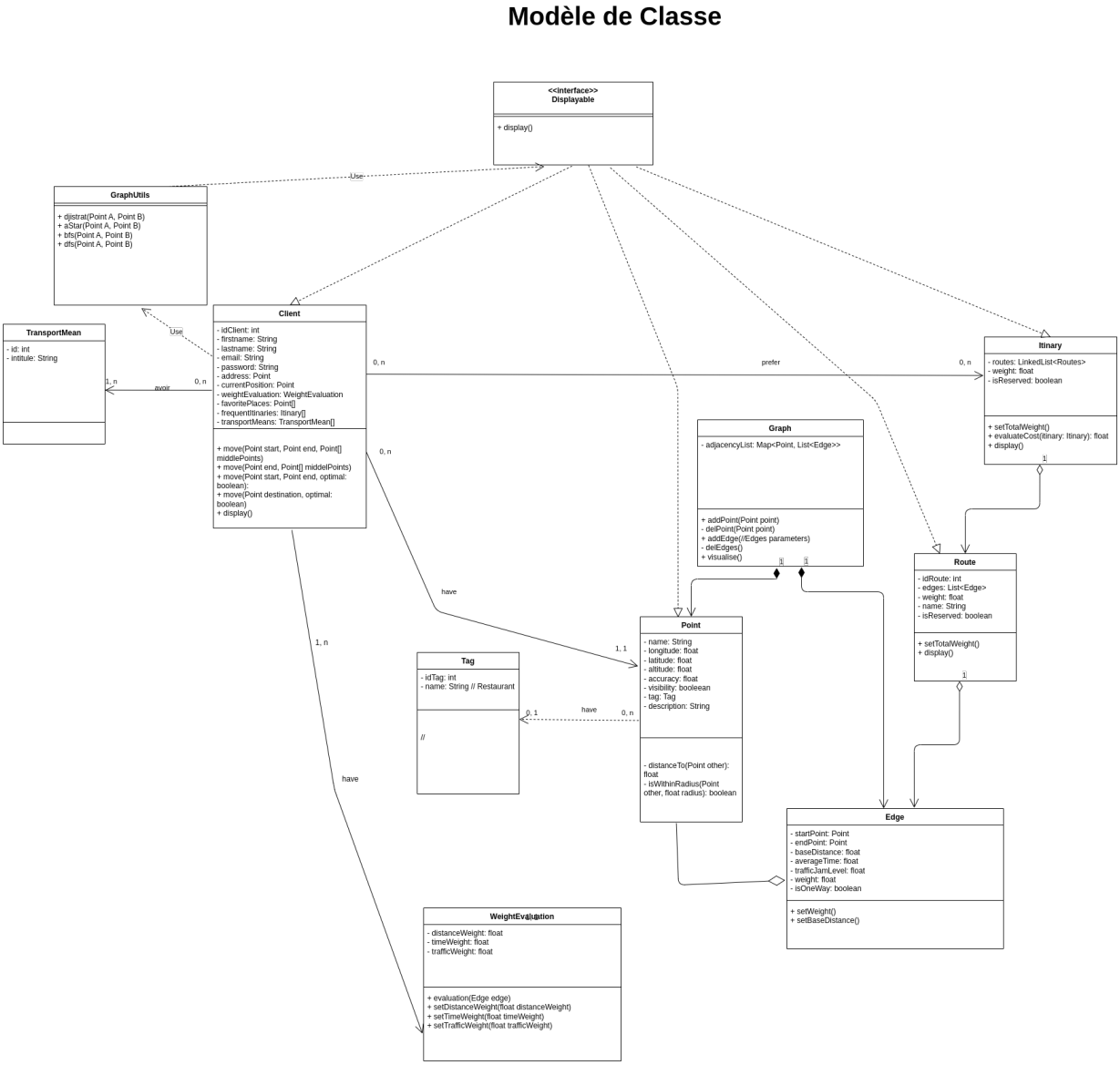


Ce diagramme ci nous permet de voir un peu plus clair dans la boite noire qu'est le système. Comme on peut remarquer sur l'image en amont, on y retrouve :

— textbfLe package de gestion des itinéraires : C'est la package central, qui va nous permettre de définir modéliser la carte, ses lieux et ses routes, comme expliqués dans le modèle mathématique en amont. On y retrouvera aussi des méthodes pour effectivement calculer les meilleurs itineraires suivant différents filtres, pour évaluer les coûts de déplacements sur ces itinéraires, et même une fonctionnalité pour partager un itineraire.
— textbfLe package de Gestion des Statistiques Générales de mobilité & IA : c'est un package ultérieur, qui nous permettra d'ajouter l'IA pour proposer des itinéraires qui tiendront aussi comptes des expériences passées.
— textbfLe package de Gestion des identités : C'est un package qui nous permettra de stocker les informations de connexion de nos utilisateurs, et qui permettra éventuellement des communications et interactions entre ces derniers. Mais dans il ne sera pas géré dans le cadre de ce projet vu qu'il y'a un autre module du plus grand projet de "Gestion des voyages", dédié à cela.
— textbfLe package de Gestion du Client : C'est ici qu'on enregistrera tout ce dont on a besoin pour un client qui utilise la plateforme ; et c'est également ici, on

gérera son interaction avec la plateforme.

4.3 Diagramme de Classe



Ce diagramme comme on peut le voir nous permet de préciser les classes de notre projet, et leurs relations avec les autres classes.

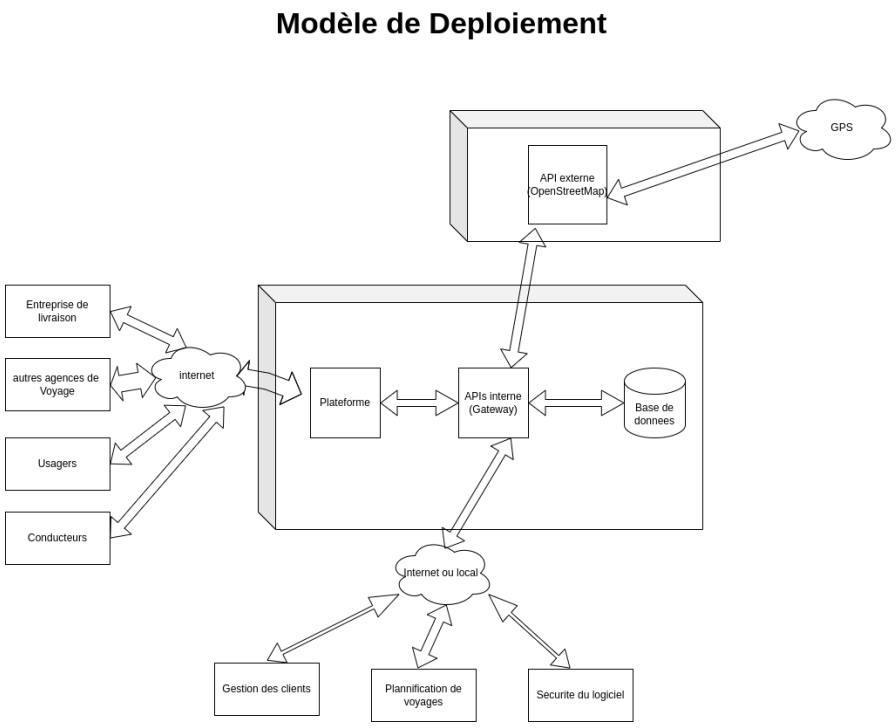
On a d'un coté les classes liées à la gestion du Client, et de l'autre celles liées à la gestion des itineraires. On peut voir qu'un client à un un WeightEvaluation, on l'on règlera ses filtres en matières de mobilité. Aussi la classe Client en plus de ses méthodes

et attributs propres au client, peut utiliser la classe GraphUtils, pour effectuer le hestion de ses itinéraires

A droite, on définit toutes les classes liées à la modélisation mathématique et dont on a parlé en amont, et qui ont les relations mentionnées dans la partie modélisation mathématique.

Remarque : On remarque qu'il y a une tout haut, une interface "Displayable", qui permet de marquer les objets qui l'implémentent comme les itinéraires et les points, sur la carte, pour qu'on puisse les distinguer.

4.4 Diagramme de Deploiement



On arrive à la phase de déploiement où l'on voit de manière synthétique le contenu du système et comment il interagit avec les clients à gauche et avec d'autres services internes, voir d'autres plateformes, via l'API interne, juste en bas. Plus loin on voit que le système communiquera avec l'API d'OpenStreeMap, pour la localisation GPS et l'accès au carte en temps réel.

5 Spécifications Techniques

Il s'agit ici pour nous de faire une tour des outils techniques qui seront nécessaires à l'implémentation de ce projet.

Les infrastructures

— Cloud computing pour la scalabilité et la flexibilité.
— Big data pour l'analyse des données de trafic et de mobilité.
— Intelligence artificielle pour l'optimisation des itinéraires et la prédiction du trafic.
— Blockchain pour la sécurisation des transactions et la lutte contre la fraude.

Plateformes

— Plateformes ouvertes et interopérables pour faciliter l'intégration avec les systèmes existants.
— API pour permettre aux développeurs de créer des applications et des services innovants.

Outils

— Outils de développement open source pour réduire les coûts et favoriser l'innovation.
— Outils de sécurité pour protéger les données des utilisateurs.

Compétences

— Développeurs logiciels expérimentés dans les technologies cloud, big data et IA.
— Data scientists pour l'analyse des données de trafic et de mobilité.
— Experts en sécurité pour protéger les données des utilisateurs.

Voici grosomodo, les aspects techniques qui attireront notre attention, lors de l'implémentation de ce projet.