**Sommaire**

**Cahier des charges**

SAÉ S2 – 2021 - 2022

Steve PENNEC & NGUYEN Denis | G5S2

I.U.T Lyon1  Département Informatique

1. **Présentation du projet …......................................................................2**
2. **Définition des termes………………………………………………………………………….2**
3. **Le contenu de l’application …...............................................................2**
4. **Les objectifs de l’application ….............................................................3**
5. **Les cibles …..........................................................................................3**
6. **L’équipement des cibles.......................................................................4**
7. **L’interface graphique...........................................................................4**
8. **Les fonctionnalités...............................................................................5**
9. **Les ressources utilisées……………………………………………………………………….7**
10. **Présentation du projet :**

Dans le cadre d’une de nos SAÉ du deuxième semestre de notre B.U.T Informatique, nous avons comme projet le développement d’une application pour ordinateur sous le nom de **GRAMA**, diminutif de GRAph Map Analysis.

Ce projet, qui s’effectue en binôme, présentera un graphe modélisant la ville de résidence de notre groupe, soit la ville de **Lyon** en France.

1. **Définition des termes :**

Voici ci-dessus les termes techniques abordés dans ce cahier, qui suscite avoir besoin d’être expliqué pour comprendre avec clarté le contenu de ce document.

**Graphe** : Un graphe est une structure constituée d’objet, illustrés par un figuré ponctuel, qui peuvent potentiellement reliés par des liens, illustrés par un figuré linéaire.

**Non-orienté/orienté** : Un graphe peut être non-orienté, c’est-à-dire que les liens de celui-ci sont à double-sens, contrairement un à un graphe orienté, où les liens sont à sens unique.

**Arête/Arc** : Ce sont à eux deux des types de liens d’un graphe. Une arête est un lien pour un graphe non-orienté, représenté par un trait, tandis qu’un arc est un lien pour un graphe orienté, représenté par une flèche.

**Sommet/Noeud** : Ce sont les appellations données aux objets qui constitue un graphe.

**Pondéré** : Un graphe dit pondéré signifie que les liens de ce graphe possèdent une valeur définie, appelé aussi pondération. Cela permet de représenter avec plus de détail le contexte du graphe en question, comme par exemple, définir une distance, une durée ou le nombre d’élément pouvant circuler entre les sommets.

**Chaîne/Chemin** : Ces deux notions représente une séquence de sommets menant un sommet à un autre (appelé **extrémités**) en utilisant les liens existants.

**Longueur** : C’est le nombre de d’arcs ou d’arêtes que contient un chemin ou une chaîne.

**Connexe** : Un graphe dit connexe est un graphe où pour tous couples de sommets distincts (*u, v*) qui le composent, il existe une chaîne ou un chemin entre *u* et *v*.

**Distance** : Dans le plan d’un graphe, une distance exprime la longueur du plus court chemin entre deux sommets.

**Liste** : Une liste est une structure de données qui permet de regrouper des données de façon à ce qu’on puisse accéder à une ou plusieurs de ces données librement, peu importe la façon dont elle est ordonnée dans la liste en question.

1. **Le contenu de l’application :**

Le graphe de l’application GRAMA, ici qu’on nommera **Graphe-MAP**, sera un graphe **non-orienté** et **pondéré** en arêtes. Autrement dit, on modélisera une ville où les routes seront illustrées par les arêtes et seront à double-sens. Puis nous indiquerons une distance en **kilomètre** sur les arêtes entre les nœuds du graphe, qui illustreront les différents lieux de la ville en question.

De plus, le graphe-MAP sera **connexe** et contiendra **au moins 30 nœuds** dont chaque nœud et chaque arête seront répartis en **trois catégories**. Chaque catégorie devra aussi respecter une répartition prédéfinie étant décrite ci-dessous :

Il faudra aussi préciser qu’il sera possible d’avoir **plusieurs routes** entre les deux mêmes villes, peu importe leur type. Par conséquent, le graphe peut être multi-graphe.

Concernant le format du fichier pour pouvoir exploiter le graphe-MAP, on utilisera un **fichier CSV** et ce fichier devra être chargé en mémoire dans une structure de données dynamique de type **liste**.

Les données du fichier CSV seront sous le format de cet exemple :

V, Macon: A,30::R,Les Echets;N , 50::V, Villeurbanne;N,50::V, Villeurbanne;A, V,Meyzieu ;;

V,Meyzieu:A,60::V,Macon;D,5::McDo-Decines;;

R,McDo-Decines:D,5:: V,Meyzieu…

…..

1. **Les objectifs de l’application :**

L’application GRAMA étant de type utilitaire, son objectif consiste à pouvoir explorer et analyser un graphe représentant une carte par le biais d’une interface IHM.

En parallèle, son autre objectif et de pouvoir mettre le binôme en charge du projet en une situation similaire à celle d’une entreprise que ce soit par la préparation, l’organisation, la mise en place ou encore la finalisation du projet, afin de se projeter dans un milieu professionnel.

1. **Les cibles :**

Les cibles primaires de cette application sont les **professeurs** qui seront en charge de nous évaluer lors de ce projet, mais aussi à ceux qui nous encadrent et nous accompagnent au long de ce projet.

En plus des professeurs en tant que cibles primaires, nous visons aussi un **public professionnel** en imaginant un milieu de travail. Le type de public professionnel seront par exemple spécialisé dans le domaine touristique, de travaux publics ou encore de transport…

Enfin, comme cible secondaire, on pourrait supposer que les **étudiants**, dont notre binôme y compris, auront la possibilité de tester l’application pour leur propre usage, qu’ils soient en B.U.T Informatique ou venant d’autres formations.

1. **L’équipement des cibles :**

L’application sera développée et disponible uniquement sur le support **ordinateur** tournant sous Windows, Linux ou MacOS et supportant principalement le langage de programmation **Java**.

1. **L’interface graphique :**

Son interface présentera d’une part, un écran principal qui affichera si le graphe est bien chargé en mémoire ainsi que le graphe en lui-même avec ses composants où :

* Elle affichera le **nom** des nœuds du graphe et ses trois types avec :
  + **V** pour les Villes
  + **L** pour les centres de Loisirs
  + **R** pour les Restaurants
* Elle affichera la **pondération** des arêtes du graphe et ses trois types avec :
  + **A** pour les Autoroute
  + **N** pour les routes Nationales
  + **D** pour les routes Départementales

Exemple :



Ensuite, l’interface va permettre d’autre part de donner un affichage dédié à chaque fonctionnalité disponible de l’application, étant décrite ci-dessous.

Ceci nous donnera un total de 6 écrans différents à afficher.

1. **Les fonctionnalités :**

Via l’interface IHM, l’application va devoir permettre à l’utilisateur d’interagir avec le graphe et de faire un certain nombre de commandes grâce aux fonctionnalités suivantes :

|  |  |  |
| --- | --- | --- |
| **Fonctionnalité** | **Objectif** | **Descriptif** |
|  | Analyser des éléments du graphe-MAP | Donner la possibilité d’afficher uniquement certains éléments du graphe-MAP à la demande sur l’interface IHM :   * Les villes, les centres de loisir et les restaurants. * Les autoroutes, les routes nationales et les routes départementales. * Lister par catégories les nœuds et les liens. * Le nombre de villes, de centres de loisir, de restaurants, d’autoroutes, de routes nationales et de routes départementales. |
|  | Comprendre les contextes des éléments du graphe | Donner la possibilité de connaître les voisins directs pour un nœud donné, c’est-à-dire les villes avoisinantes par une seule route la ville donnée. |
|  | Comprendre les contextes des éléments du graphe | Donner la possibilité de connaître les villes qui relie une arête donnée, soit ses deux extrémités. |
|  | Connaître les distances entre deux sites | Donner la possibilité de savoir si deux nœuds donnés sont à 2-distances ou non, soit qu’il existe une chaîne de longueur 2 entre eux. |
|  | Comparer deux villes | Donner la possibilité de comparer deux villes, autrement dit de pouvoir indiquer si :   * Une ville est plus/moins ouverte qu’une autre 🡪 Si le nombre de villes à 2-distance d’une ville est supérieur/inférieur au nombre de villes à 2-distance d’une autre. * Une ville est plus/moins gastronomique qu’une autre 🡪 Si le nombre de restaurants à 2-distance d’une ville est supérieur/inférieur au nombre de restaurants à 2-distance d’une autre. * Une ville est plus/moins culturelle qu’une autre 🡪 Si le nombre de centres de loisir à 2-distance d’une ville est supérieur/inférieur au nombre de centre de loisir à 2-distance d’une autre. |
|  | **Bonus :**  Connaître les distances entre deux sites | * Donner la possibilité de connaître la distance la plus courte entre deux nœuds donnés * Donner la possibilité de trouver une route traversant une ou plusieurs villes, centres de loisir ou restaurants entre deux nœuds donnés. |

1. **Les ressources utilisées :**

Les ressources se présentent en tant que cours afin de pouvoir élaborer de fond en comble notre projet et de mettre en place l’application **GRAMA**.

* Gestion de Projet :

Les heures de cette matière seront consacrées à l’établissement de plusieurs documents requis lors de la mise en place d’un projet. Ces documents sont le **dossier de gestion projet**, un document de **spécification technique**, un autre pour les **spécifications techniques** et enfin ce document-ci, soit le **cahier des charges**.

* Mathématiques :

Les mathématiques nous permettront d’aborder tout le nécessaire concernant les **graphes** et de comprendre les **notions de base** de ce module. Cela nous permettra ainsi d’être plus technique au niveau **théorique** et de savoir comment on procédera pour établir les fonctionnalités requis du graphe.

* Structure de données :

Cette ressource-ci nous sera utile dans la situation où il faudra créer des **structures de données** et d’utiliser la plus adéquate pour notre application et son fonctionnement. C’est pour cette raison qu’il nous est imposé d’utiliser une structure de données **dynamique** de type **liste** pour pouvoir exploiter un **fichier** **CSV**.

* Java :

Le Java, sollicité en parallèle de Structure de données, nous permettra de rendre fonctionnel le **chargement du graphe** et le **remplissage des structures de données** que l’on utilisera.

De plus, elle représente le langage de programmation primaire pour développer Graphe-MAP au niveau **algorithmique**. Autrement dit, les **fonctionnalités** présentées précédemment seront implémenter lors des cours de Java.

* Interaction Homme Machine :

Appelé sous le diminutif IHM et en parallèle au Java, ce module nous servira à développer les différentes **interfaces** de l’application. Cela a pour but de permettre à l’utilisateur de l’application de pouvoir **interagir** avec le graphe via les fonctionnalités grâce à l’interface réalisé en IHM.

* Qualité Dev :

Surtout sollicité vers la fin du développement de l’application, la tâche attribuée lors de ce module sera d’évaluer la **qualité de conception** de Graphe-MAP, d’effectuer le **développement**, des **tests** et ainsi des **améliorations** pour notre application.