**Introduction:**

Une entreprise spécialisée dans la fabrication de glaces et pâtisseries surgelées veut gérer les travaux de ses centres et la distribution des produits aux marchands qui sont repartis sur le territoire française. Cette entreprise vend plus que 1500 produits et a plus que 87 centres de distribution satisfaisant mensuellement environ 1.000.000 de clients, et a trois canaux de distribution intégrés : vente et livraison à domicile, magasins spécialisés et Internet.

Cette entreprise recherche une solution qui répond à ses besoins et qui facilite le travail de ses employés et organise le travail de ses livreurs.

Le but de notre projet est de faciliter le travail de cette entreprise et d’organiser son travail. Plus concrètement il s’agit de trouver un plan de livraison optimal d’un livreur, qui peut visiter tous les marchands dans le plan par un plus court chemin. Pour atteindre ce but il faut modéliser le problème sous forme d’un TSP (*Travelling Salesman Problem)* afin de trouver la solution optimale qui permet de minimiser la longueur de la route. L’algorithme GA (*Genetic Algorithm)* est utilisé pour résoudre ce problème et Les librairies GMap.Net.Core et GMap.Net.WindowsForm sont utilisées pour l’implémentation du programme.

[Cloud Computing]

**Environnement du projet :**

Nous développons ce logiciel chez “*Telepaty Holding”* sous la direction de Mr. Marwan Hassan directeur technique, et la surveillance de Dr. Hala Naja Coordinatrice du Master en Génie Logiciel (Université Libanaise).

**Profile de *Telepaty Holding:***

*Telepaty Holding* développe à l’origine entre la France, le Liban et la Belgique. C’est un pionnier des technologies de l’information apportant des solutions efficaces aux entreprises et jouant un rôle important dans le gain financier. Elle offre une vaste gamme de services dédiés aux entreprises dont le but [Le rôle de telepaty]



**Organisation du rapport:**

Le rapport est divisé en les parties suivantes :

* Le chapitre 1 : Le Cahier de charge où on identifie les acteurs et leurs rôles, les entités principales dans notre application et le travail demandé.
* Le chapitre 2: Dans ce chapitre on expliquera les spécifications fonctionnelles et non fonctionnelles et détaillera quelques cas d’utilisation.
* Le chapitre 3: Dans Cette section on expliquera l’architecture du système puis on discutera le flux de travail de système.
* Le chapitre 4 :
* Le chapitre 5 :

**Chapitre 1: Cahier des Charges**

**1.1 Introduction:**

Dans ce chapitre, le domaine ainsi que les besoins de système seront exposés. Aussi les principales entités seront définies.

**1.2 Les parties de l’application :**

Le système d’organisation de la télévente et de la distribution à domicile est divisé en trois parties essentielles:

1. L’organisation des centres, des produits, des employés et des commandes; l’ensemble de ces besoins constitue le travail du *backoffice.*
2. L’organisation du travail d’un livreur qui assure la livraison des commandes, il possède le terminal “*YouTransactor” INGENICO I8550* disposant d’une application métier lui permettant de consulter son plan quotidien, d’enregistrer des commandes, de notifier le résultat des différents livraisons…. et d’une monétique agréée *CB5.2* [Monétique CB5.2]
3. L’organisation des livraisons. Il s’agit d’établir les tours journaliers de chaque livreur en minimisant les distances et le temps.

**Dans ce projet, nous traiterons essentiellement la 3eme partie.**

**1.3 Le domaine de l’application**

**1.3.1** ***Store***

*Store* est défini par:

* *Store name.*
* *Store type:* les types sont *Region, City, Area, SubArea, Merchant* et *Center.*
* Un seule *store*  de type *Telepaty.*
* *Telepaty* est un ensemble des *Regions.*
* *Region* est un ensemble des *Cities.*
* *City* une ensemble des *Areas.*
* *Area* est un ensemble des *SubAreas.*
* *Center* peut être le fils de quel que soit le type de parent.
* *Merchant*  est le fils de quelque soit le parent mais il faut être un *Center* parmi ses ancêtres.
* *Store status:* les statuts sont *Active* et *Inactive.*
* *Store parent:* le store qui est directement le père de ce store.
* *Store delivery period:* c’est le temps maximum pour la livraison est effectuée pour ce *store.*
* *Store latitude:* attribuer l’altitude de chaque *center* ou *merchant.*
* *Store longitude:* attribuer la longitude de chaque *center* ou *merchant.*

**1.3.2** ***Vendor***

*Vendor* est défini par:

* *Vendor name:* le nom et le prénom.
* *Vendor status:* les statuts sont *Active, Inactive* et *Fired.*
* Les informations personnelles: *birthdate, address, sex, landline number, mobile number, sex, e-mail.*
* *Recrutement date.*
* *Store id:* l’id de (*store* de type *center)* ou il est l’affecte.

**1.3.3** ***Terminal***

*Terminal* est défini par:

* *Store ID:* l’id de (*store* de type *center)* ou il est l’affecte.
* *Terminal serial:* le numéro de série de ce terminal.
* *Terminal status:* les statuts sont *Active, Inactive, Pending, Stopped.*
* *Generation code:* ce code est générée par le système et envoyé au terminal une seule fois pour l’activer.
* *Activation* *date:* la valeur est automatique en cas de l’activation du terminal.

**1.3.4** ***Product***

*Product* est défini par:

* *Product name:* le nom public du produit.
* *Product code:* le code utilisée pour spécifier un produit.
* *Category:* chaque produit fait une partie d’une catégorie ou chaque catégorie même fait partie d’un *Category type.*
* *Product status:* les statuts sont *Active* et *Inactive.*
* *Wholesale price:* le prix d’un produit en une grande quantité
* *Retail price:* le prix d’un produit en petit quantité.

**1.3.5** ***Order***

*Order* est défini par:

* *Order Serial: le* numéro de série de l’ordre.
* *Store ID: store* de type marchand.
* *Store Order Type:* le type de l’ordre, ou l’ordre peut être l’un de deux types *Order* ou *Back Ordre.*
* *Order Date:* le date de prise l’ordre.
* *Order Deadline:* la date ou l’ordre faut être délivrée.
* *Vendor ID:* le vendeur qui est prise l’ordre.
* *Delivery Vendor ID:* le vendeur qu’il faut être délivrée l’ordre.
* *Invoice ID:* l’id de la facture ou cette ordre est fait partie de.
* *Terminal ID:* l’id du terminale si l’ordre est prise par un livreur au temps de visite.
* *Total Taxes Value:* montant calculée des taxes d’après la valeur totale du montant de la facture.

Il est à noter que le pourcentage des taxes (TVA…) est différent d’un marchand à un autre.

* *Net Total:* le montant total de l’ordre de toutes les commandes de cet ordre.
* *Total Value:* le montant total de la facture avec la calcule des taxes.

**N.B:** l’ordre est un ensemble des commandes (*Order Details)*.

**1.3.6** ***Order******Detail***

*Order detail* est défini par:

* *Order ID:* l’id de l’ordre ou cette commandes est faits partie de.
* *Product ID:* l’id du produit choisissee dans cette commande.
* *Product Quantity:* la somme requis de ce produit.
* *Product Price:* c’est une valeur généré par le système en basant sur la quantité (cette quantité fait partie d’un *wholesale* ou *retail).*

**1.3.7** ***Delivery Plan***

*Delivery Plan* est définie par:

* *Vendor ID :* l’id du vendeur de ce plan de livraison.
* *Delivery Date :* la date de livraison de ce plan de livraison.

**1.3.8** ***Visit***

*Visit* est définie par:

* *Delivery ID :* l’id du plan de livraison ou ce visite est fait partie de.
* *Visit Type :* la nature de ce type de visite, et le plus important type est la livraison d’ordre ‘*Delivery*’, les autres types comme prendre d’ordre, client endormi, …..
* *Store ID :* le marchand ou il faut visiter.

**1.4 Les acteurs et leurs rôles**

Les acteurs identifiés sont:

* Le Responsable de centre.
* Le Secrétaire.
* Le livreur (VRP)

1.4.1 **Le rôle du responsable de centre**

Le responsable de centre devra :

* Gérer les exceptions de livraison (dernière minute et replanification)
* Analyser le compte rendu de livraison.
* Organiser la liste des clients et la distribuer dans des plans de travail de chaque livreur.
* Analyser le trajet optimum.
* Envoyer des SMS au VRP de dernière minute (marchand absente…).

1.4.2 **Le rôle du livreur**

Le livreur devra :

* Saisir commande, contrôler ses commandes.
* Géocoder les clients, si cela n’est pas déjà fait.
* Renseigner sur les éventuelles anomalies de livraison.
* Saisir le mode de règlement sur son terminal.
* Recevoir d’éventuels SMS urgents du centre.
* Saisir la liste des commandes à re-livrer et les produits à réintégrer en stock (en cas de rendre une partie de l’ordre et faire une *Back Order.*

N.B: le travail de livreur se fait sur le terminal *INGENICO I8550*.

1.4.**3 Le rôle du secrétaire**

Le secrétaire devra :

* Editer les feuilles de route optimisée.
* Envoyer des SMS au VRP.
* Gérer les exceptions de livraison (dernière minute et replanification)

**1.5 Travail Demandé:**

Les étapes de réalisation du projet sont:

1. Génère une carte géographique qui simplifie la géolocalisation des *Centers, Merchants* (qui ont des ordres ou non).
2. Générer et visualiser les plus courts chemins pour les livraisons.
3. Générer les plans de livraison, et l’ordre des visites de chaque livreur.
4. Suivre le livreur sur la carte géographique en temps réel.

**Chapitre 2 : Etude Bibliographie sur les Algorithmes et leurs Implémentations**

**2.1 Introduction :**

**2.2 TSP : Travelling Salesman Problem**

Le ***problème du voyageur de commerce*** ou ***le problème du commis voyageur***consiste, étant donné un ensemble de [villes](http://fr.wikipedia.org/wiki/Ville) séparées par des [distances](http://fr.wikipedia.org/wiki/Distance) données, à trouver le plus court chemin qui relie toutes les villes et il est connu comme étant un problème [NP-complet](http://fr.wikipedia.org/wiki/NP-complet)[[1]](#footnote-1).

Enoncé du *problème du voyageur de commerce :* étant donné *n* points (des « villes ») et les distances séparant chaque point, trouver un chemin de longueur totale minimale qui passe exactement une fois par chaque point et revienne au point de départ.

Enoncé formel du *problème du voyageur de commerce :* Soit un [graphe complet](http://fr.wikipedia.org/wiki/Graphe_complet) [[2]](#footnote-2) avec un ensemble de sommets,  un ensemble d'arêtes et  une fonction de coût sur les arcs. Le problème est de trouver le plus court [cycle hamiltonien[[3]](#footnote-3)](http://fr.wikipedia.org/wiki/Cycle_hamiltonien) dans le graphe.

Dans notre cas, on considère qu’un chemin existe dans un sens mais pas dans l’autre comme il existe beaucoup des routes à sens unique. Donc notre problème du voyageur de commerce est asymétrique comme  la distance entre deux nœuds i et j d'une arête n'est pas forcément la même qu'on aille de i à j ou bien de j à i.

Le problème du voyageur de commerce est un des problèmes combinatoires classiques. Des solutions optimales avec beaucoup de nœuds sont de cas spatiaux de la nature géographique et ils travaillent en espaces métriques. Au contraire chez des solutions optimales générales une analyse complète est nécessaire des N! Tours possibles. C’est pourquoi ils ne sont pas devinés efficients.

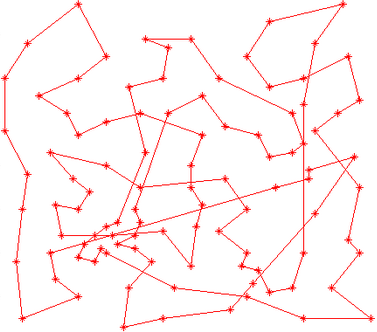


Figure 1 : Utilisation des TSP

**2.3 Algorithmes Génétiques (GA)**

De nombreuses inventions de l'homme ont été inspirées par la nature. Algorithmes génétiques (GA) est un exemple, initiés dans les années 1970 par John Holland, sont des algorithmes d’optimisation s’appuyant sur des techniques dérivées de la génétique et des mécanismes d’évolution de la nature : croisement, mutation, sélection. Recherché de GA en simulant l’évolution, à partir d'un ensemble initial de solutions ou des hypothèses, et de générer «générations» successives de solutions.

**2.3.1 Concepts**

L'algorithme génétique diffère d'autres méthodes de recherche en ce qu'il cherche parmi une population de points, et fonctionne avec un codage d’ensemble des paramètres, plutôt que les valeurs des paramètres eux-mêmes. Le régime de transition de l'algorithme génétique est probabiliste, alors que les méthodes traditionnelles d'utiliser l'information de gradient. En raison de ces caractéristiques de l'algorithme génétique, ils sont utilisés comme algorithme d’objectif d'optimisation générale. Les algorithmes génétiques fournissent des solutions aux problèmes n'ayant pas de solutions calculables en temps raisonnable de façon analytique ou algorithmique.

* Dans notre cas, une population sera un ensemble d'individus.
* Un individu sera une solution au problème donné.
* Un gène sera une partie d'une solution, donc d'un individu.
* Une génération est une itération de notre algorithme.

**2.3.2 Principe de base**

Un algorithme génétique va faire évoluer une population dans le but d'en améliorer les individus. Et c'est donc, à chaque génération, un ensemble d'individus qui sera mis en avant et non un individu particulier. Nous obtiendrons donc un ensemble de solutions pour un problème et pas une solution unique. Les solutions trouvées seront généralement différentes.

Le déroulement d'un algorithme génétique peut être découpé en cinq parties :

1. La création de la population initiale
2. L'évaluation des individus
3. La création de nouveaux individus
4. L'insertion des nouveaux individus dans la population
5. Réitération du processus

**L’organigramme de GA:**

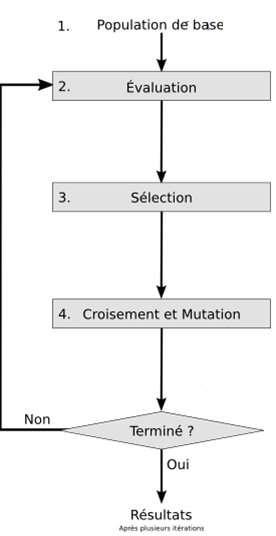


Figure 2: Organigramme de GA

1. **La création de la population :**

Pour démarrer un algorithme génétique, il faut lui fournir une population à faire évoluer. Pour cela nous avons créé une classe « Population » qui a juste une seule fonction ‘CreateRandomPopulation’ et la taille de la population initiale a été 10000.

1. **L’évolution des individus (Fitness) :**

On a créé pour cela une classe « Tour » qui a le ‘Fitness’ comme une propriété et une fonction ‘DetermineFitness’ pour pouvoir l’utiliser dans la classe « TSP ».

1. **La création d’un nouvel individu :**

Nous allons maintenant enrichir notre population en croisant des individus. Nous allons essayer de prendre des morceaux de solution de certains individus et d'autres morceaux d'autres individus pour créer des nouveaux individus qui, on l'espère, seront des solutions meilleures à notre problème.

Et pour créer le nouvel individu on applique les deux méthodes :

1. Le croisement (*Crossover*)

Les croisements permettent de simuler des reproductions d'individus dans le but d'en créer des nouveaux.

1. La mutation (*Mutation*)

Une autre solution que le croisement pour créer de nouveaux individus est de modifier ceux déjà existants. Ici, le hasard va nous être d'une grande utilité.

Et pour ce raison on a créé deux fonctions ‘*Mutate*’ et ‘*Crossover*’ dans la classe « Tour » qui retourne des nouveaux tours.

1. **L’insertion des nouveaux individus dans la population :**

Une fois que nous avons créé de nouveaux individus que ce soit par croisements ou par mutations, il nous faut sélectionner ceux qui vont continuer à participer à l'amélioration de notre population. Une fois la nouvelle population obtenue, vous pouvez recommencer le processus d'amélioration des individus pour obtenir une nouvelle population et ainsi de suite…

1. **Réitération de processus :**

Une fois le nombre maximum de générations atteint, vous obtenez une population de solutions. Mais rien ne vous dit que la solution théorique optimale aura été trouvée.

**2.4 GMap.Net**

**GMap.Net** est un super, puissant, gratuit, multiplateforme et open source de .Net contrôle. Permettre d’utiliser le routage, les géocodages, les directions et les cartes géographiques de Google, Yahoo, Bing, OpenStreetMap, ArcGIS, Pergo, SigPac, Yandex, Mapy.cz, Maps.lt, iKarte.lv, NearMap, OviMap, CloudMade, WikiMapia en Windows Forms et Presentation, soutenir la mise en cache et pouvoir fonctionner sur Windows Mobile.

Ce super contrôle est générée par :



Quelques points pour bien comprendre l’utilisation de ce contrôle qui est basée sur 2 librairies GMap.Net.Core.dll et GMap.Net.WindowsForms.dll, de même l’utilisation de System.Data.SQLite.dll qui est utilisé pour la mise en cache locale des données pour réduire la demande de la carte :

**GMap.Net.WindowsForms**

* GMapControl : C’est le contrôle qui rend la carte géographique.
* GMapOverlay : Il s’agit d’une couche au-dessus de contrôle de la carte. Il peut avoir plusieurs couches au-dessus de la carte, chaque un représente, par exemple, une route avec des points spécifiques, une liste de magasins, d’hôpital, etc…
* GMapMarker : Tels sont les points sur une couche, chacune représentant une géolocalisation précise (latitude et longitude), par exemple chaque point représente un magasin.
* GMapRoute : Il s’agit de la voie ou la direction entre deux points ou plus.
* Autres : comme ColorMatrix, GMapToolTip, GMapPolygon….

**GMap.Net.Core**

* GMap.Net.MapProviders : Qui contient un ensemble des cartes de différents fournisseurs chaque un est spécifique pour des raisons différentes, par exemple :
* Google : fournit un ensemble des cartes spécifiques pour Korea et China même si la carte groupe tous les continents et pays, et fournit GoogleSatellite, GoogleTerrain, Google, GoogleHybrid….
* ArcGIS: fournit autre ensemble des cartes comme ArcGIS\_StreetMap\_World\_2D, ArcGIS\_World\_Shaped\_Relief, ArcGIS\_Imagery\_World\_2D…
* OpenStreetMap : fournit OpenSeaMapHybrid, OpenStreetOsm,....
* Autres comme Bing, NearMap, Czech, Yahoo, Ovi, etc….
* GMap.Net.Projections

[Projection]

Comme MercatorProjection, PlateCarreeProjection, MapsLTProjection, etc…

* GMap.Net : qui offre ensemble des fonctionnalités comme le GPoint, GMaps, GDirection, PointLatLng, RenderMode, PureProjection, MapRoute, MapType, etc…
* Et autres comme GMap.Net.Internals, GMap.Net.CacheProviders, etc…

**Chapitre 3 : Méthodologie et solution proposée**

**3.1 Introduction**

Ce chapitre va présenter les spécifications fonctionnelles et non fonctionnelles du logiciel, et va développer les cas d'utilisation. L'objectif de ce chapitre est de faciliter et de clarifier les différentes fonctionnalités de l'application. Sa lecture permet une connaissance approfondie du projet, dans le but de le modifier, de l'analyser, ou de le mettre à jour. Ce document est important pour le client car il décrit précisément ce qui sera réalisé, et pour les développeurs car il est référencé, précis et non ambiguë. Nous utiliserons le formalisme UML pour la spécification du système.

**3.2 Spécifications fonctionnelles**

Les spécifications fonctionnelles sont:

1. Pour un ensemble de visites donné, il faut calculer le plus court chemin liant ces visites.
2. Générer le plan de livraison optimal dans lequel les visites sont ordonnées dans le sens du plus court chemin.
3. Suivre en temps réel le livreur sur la carte géographique en se basant sur la localisation du terminal porté par ce dernier.

**3.3 Spécifications non fonctionnelles**

Les spécifications non fonctionnelles sont :

**3.3.1 Le temps :**

Il faut que le temps de calcul des distances et de génération du plus court chemin pour l’ensemble des visites de chaque livreur soit acceptable est raisonnable. Il ne doit dépasser 1 minute pour un ensemble de 100 visites journalières affectées à ce dernier.

**3.3.2 Le plan de livraison (*Delivery plan)* :**

Le plan de livraisongénéré devra être lisible et facile à comprendre : les visites seront présentées par l’ordre de passage.

**3.3.3 La Solution :**

Ce projet permet d’avoir une solution du problème mais pas nécessairement la meilleure. Etant nous nous sommes basées sur les routes et les distances configurées par « *Google Map ».* Les routes et les chemins auxiliaires ne sont pas nécessairement représentés par « *Google Map »,* et donc seront ignorés par notre algorithme.

**3.4 Les cas d'utilisation**

On a utilisé le diagramme d’*Use Case* d’UML pour identifier les cas d’utilisation.

Voir Figure 1

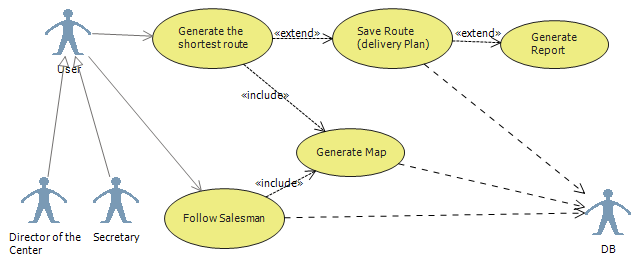


Figure 3 : Diagramme des cas d’utilisation

**Description textuelle des cas d'utilisation:**

* + 1. **Générer le plus court chemin (*Generate the shortest route)***
* **Acteurs :**
* Acteur principale : l’utilisateur du système et dans ce cas pouvoir être le directeur du centre ou le secrétaire.
* Acteur secondaire : la base de données
* **Niveau** : Stratégique.
* **Préconditions** :
* L’ensemble des livreurs proposés sont des livreurs chez le centre où nous sommes.
* L’ensemble des livreurs proposés sont actifs dans le système.
* Le livreur choisi a un ensemble des ordres à livrer où son temps de livraison est le lendemain.
* **Post-conditions** :
* représentation graphique de route où le livreur choisi faut le passer avec ordre des visites pour atteindre le plus court chemin.
* L’ensemble des marchands visités sont recolorées de bleu.
* **Scénario nominal**

1. Choisir un livreur.
2. Cliquer sur le bouton « *Generate the shortest route* ».

**3.4.2 Générer la carte géographique (*Generate Map*)**

* **Acteurs :**
* **Niveau :** Sous-fonctionnalité.
* **Préconditions :**
* Le système est connecte à l’internet pour bien accès la base de données.
* La base de données à stocker les coordonnées géographiques du centre ou nous sommes et les coordonnées géographiques des marchands qui travaille avec lui.
* **Post-conditions :** les points concernés le centre et les marchands sont représentées sur la carte géographique de couleurs différents pour bien différencier entre eux, de même avec représentation des informations de localisation géographique et les adresse de chaque un.
* **Scénario nominal :**

1. Charger la carte géographique de type ‘*Google Map’.*
2. Accéder la base de données, et identifier le centre ou nous sommes voulus.
3. Filtrer les marchands qui sont sous son direction.
4. Lire les coordonnées et les autres informations qui concernent ce centre et ses marchands puis représenter tous sur la carte.
   * 1. **Sauvegarder la route (*Save Route)***

* **Acteurs :**
* Acteur principal : l’utilisateur du système et dans ce cas pouvoir être le directeur du centre ou le secrétaire.
* Acteur secondaire : la base de données
* **Niveau** **:** Objectif utilisateur.
* **Préconditions :**
* La route est générée et le bouton « *Save »* est activé.
* La connexion à l’internet a une bonne condition.
* **Post-conditions** **:** le plan de livraison avec les visites sont sauvegardés dans la base de données.
* **Scénario nominal :**

1. L’utilisateur clique sur le bouton « *Save ».*
2. Un nouveau plan de livraison est créé à la date de livraison et reliée au livreur choisi.
3. L’ensemble des visites sont sauvegardé à ce plan de livraison dans la base de données.

**N.B Si nous ajoutons un nouvel ordre a ce livreur, il faut régénérer la route, écraser le vieil ensemble de visites de ce plan de livraison et sauvegarder de nouveau l’ensemble de visites créé par la génération de route.**

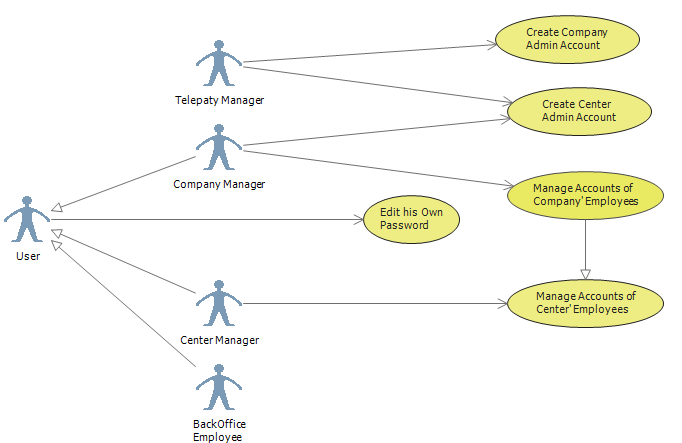
**3.4.4 Générer le Rapport (*Generate Report*)**

* **Acteurs :**
* **Niveau** **:** Objectif utilisateur.
* **Préconditions :**
* La route est sauvegardée à la base de données.
* Le bouton « *Generate Report »* est activé.
* **Post-conditions** **:** Un rapport final de visites est généré, ce rapport donne des informations complètes sur les visites de ce livreur et il faut être lisible et facile à comprendre pour un livreur.
* **Scénario nominal :**
* L’utilisateur clique sur le bouton « *Generate Report ».*

**3.4.5 Suivre le Livreur (*Follow Salesman*)**

* **Acteurs :**
* Acteur principal : l’utilisateur du système et dans ce cas pouvoir être le directeur du centre ou le secrétaire.
* Acteur secondaire : la base de données
* **Niveau** **:** Objectif utilisateur.
* **Préconditions :** le livreur suivi a un *GPS sensor* à son terminal.
* **Post-conditions** **:** pouvoir suivre le livreur pas à pas.
* **Scénario nominal :**

1. Choisir un livreur.
2. Cliquer sur le bouton «*Follow salesman* ».



- Telepaty admin:

o Creates a new company/center and an admin account for it

o Checks all clients orders, transactions, and accounts for any problem

o Monitors centers problems

- Company manager:

o Adds products and assigns them to centers

o Puts prices for products

o View orders and print them

o View merchants invoices and print them

o Adds new centers and merchants

o Activate / Deactivate centers and merchants

o Adds service center support and assigns them to centers

o Monitors centers problems

o Manages employee accounts for all employees under the company

o View merchants taxes, addresses, payments, and invoices

- Center manager:

o Adds orders and print them

o Manages the center’s employees accounts

o Manages the center’s merchants

o View available products assigned to it by the company manager

o View the center’s service support

o Manages the delivery exceptions and last minute rescheduling

o Organizes daily tours and edits delivery plans

o Sends SMS to the delivery man

o Adds taxes and addresses for merchants

o Adds payments for merchants and confirms their reception

- Back office employee:

o Answers the clients on the phone and takes their orders

o Adds payments for merchants

o Adds contact info, taxes, and addresses for merchants

o Edits his own password

- Delivery man

o Takes orders on his terminal

o Calls the service center support in case of any technical problem

o Receives emergency SMS from center

o Performs electronic banking transactions

o Prints invoices

**Chapitre 4: L’implémentation**

**4.1 Introduction :**

Ce chapitre veut fournir la structure interne de composants logiciels. Il décrit les interfaces graphiques.

**4.2 L’interface Graphique**

C’est le bouton qui génère le carte des marchands et le centre qui nous sommes faire une partie de lui.

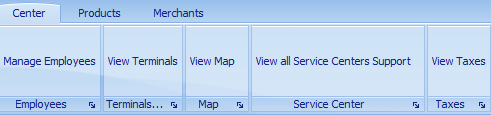


Figure 4 : Menu Principale

Donc nous sommes identifiés comme un employé au “Nantes Center” et nous voulons générer la carte de Nantes Center

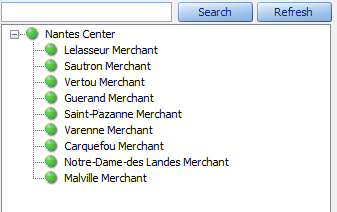


Figure 5 : *Tree* qui représente le centre et ses marchands

C’est la carte des Marchands de « Centre Nantes » ou tous les marchands qui sont géocoder sont présents sur la carte.

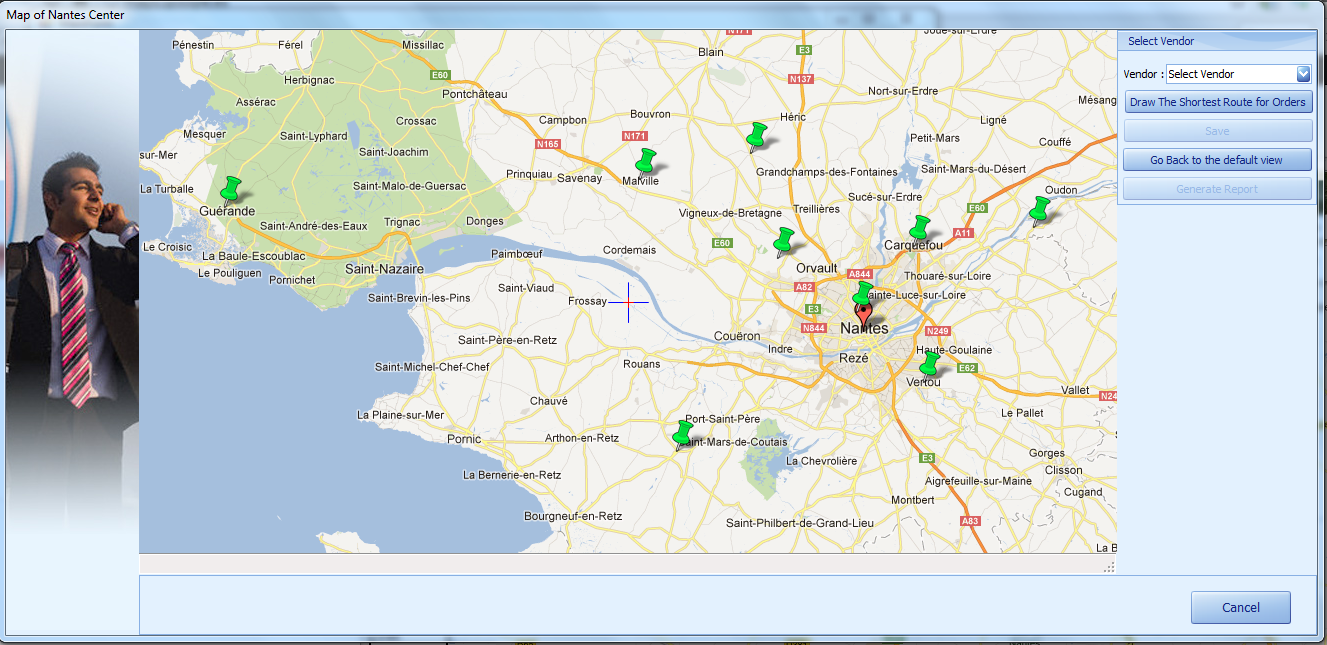


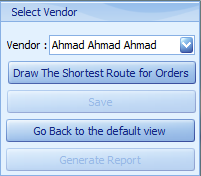
Figure 6 : La carte générée



Figure 7 : Les informations du centre et des marchands

A la droite de page on trouve un menu pour sélectionner le vendeur que nous voulons générer son plan de livraison.

On sélectionne un vendeur pour generer son plan de livraison du demain ou il prend tous les marchands qui ont des ordres prêts et le temps de livraison est demain, et dans ce *comboBox* on montre seulement les vendeurs actives.



Il suffit de cliquer sur ce bouton pour générer la route du vendeur

Figure 8 : Menu qui continue le fonctionnement su la carte

Le route du vendeur sélectionner est numérotée selon l’ordre des visites pour atteindre le plus court chemin ou il va du centre et revient ou centre et devient de couleur bleu tous les centres qui ont des ordres et confirment avec tous les conditions.

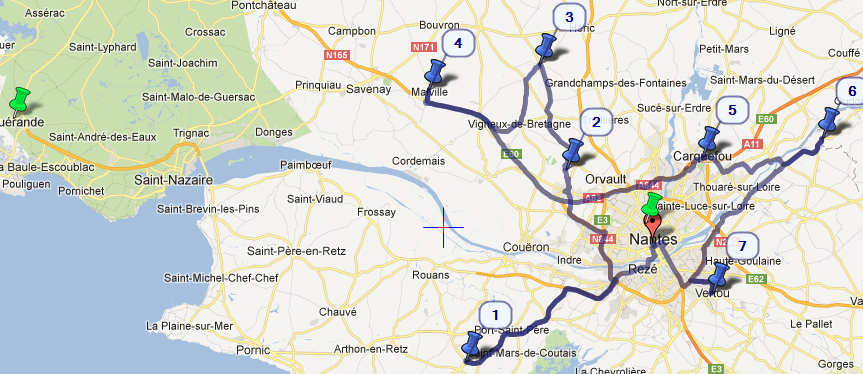
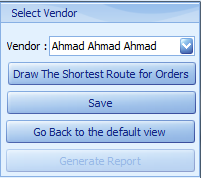


Figure 9 : Graphe qui représente la plus court chemin d'un vendeur (livreur)



Autant ou nous sauvegardons le plan et les visites le bouton de génération du rapport du plan est active.

A la fin de tracer le route de vendeur le bouton *Save est* active pour sauvegarder ce plan de livraison et ces visites.

Figure 10 : Activation et désactivation des boutons

Un rapport de visites est généré ou il représente l’ordre de visite, la ville de départ, la ville de retour, la longueur du route et à la fin le type de visite qui peut être une visite pour délivrer ordres ou juste une visite pour prendre un ordre. Le dernier record est non pas une visite réelle mais justement le cas de retour au centre.

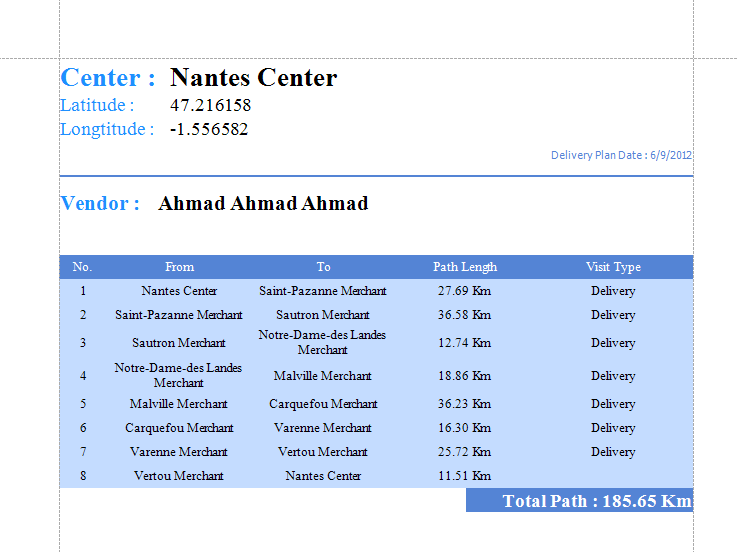


Figure 11 : Le rapport qui représente les visites d'un vendeur (livreur)

En tous cas on peut déplacer sur la carte et faire *Zoom in*  et *Zoom out.*

1. En pratique, savoir que le problème sur lequel on travaille est NP-complet est une indication du fait que le problème est difficile à résoudre, donc qu'il vaut mieux chercher des solutions approchées en utilisant des [algorithmes d'approximation](http://fr.wikipedia.org/wiki/Algorithme_d%27approximation). [↑](#footnote-ref-1)
2. En [théorie des graphes](http://fr.wikipedia.org/wiki/Th%C3%A9orie_des_graphes), le graphe complet Kn est l'unique graphe à isomorphisme près possédant n sommets tous reliés deux à deux par une arête, donc pour n sommets on a n (n-1)/2 arêtes. [↑](#footnote-ref-2)
3. En [théorie des graphes](http://fr.wikipedia.org/wiki/Th%C3%A9orie_des_graphes), un graphe hamiltonien est un graphe possédant au moins un cycle passant par tous les sommets une et une seule fois. [↑](#footnote-ref-3)