Analyse de UML

1.Description du réseau et itinéraire

### L’objectif de ce projet est de réaliser un prototype capable de guider le conducteur d’un véhicule qui souhaite se rendre d’une ville à une autre et qui peut exprimer des préférences sur le trajet à suivre (le plus court, le plus rapide…). On souhaite guider l’utilisateur durant son trajet, en le conseillant s’il se trompe de chemin. Le système n’est pas équipé d’une antenne GPS, c’est donc l’utilisateur qui indiquera le lieu où il se trouve.

### 

### Le réseau routier est formé d’un ensemble de villes et de routes qui les relie. Chaque ville est caractérisée par son nom, par le fait qu’elle possède ou non des atouts touristiques et par sa taille (grande, moyenne, petite). Chaque route a un *classement* (elle peut être départementale, nationale, autoroute, chemin de campagne…) et un nom (N7, A6..). Elle est décomposée en tronçons qui relient deux villes : sur chaque tronçon, une route à une vitesse maximum autorisée, on peut savoir si des radars y sont présents. Un tronçon possède également une taille, exprimée en kilomètres et possède ou non des attraits touristiques

### Lorsque le conducteur fait une demande d’itinéraire, il doit obligatoirement indiquer une ville de départ et une ville d’arrivée. Il peut, en outre, exprimer un certain nombre de *préférence*

### Spécifier les villes intermédiaires (en nommant ces villes) à suivre ou à éviter, ou indiquer le *type* de villes intermédiaires (grosses ville, villes touristiques…) à suivre ou à éviter ;

### Spécifier les *caractéristiques* (type de route, avec/sans radar, vitesse…) des routes à suivre ou à éviter ;

### Choisir le chemin le plus court (en kilomètres), le plus rapide (en temps), ou autre.

### Toute combinaison de ces préférences doit pouvoir être prise en compte

Sommaire

1.Description du réseau et itinéraire

2.En guise de cahier des charges

3.Cas d’utilisation :

4.Scénario

5.Diagramme de classe

6.Diagramme de séquence

En guise de cahier des charges

## Le système navigation demandent à client d’entrée d’une point actuelle et d’une point destinataire.

## 

## Aussi le client devraient saisir le type de la voiture.

## 

## choisir son préférence du trajet:

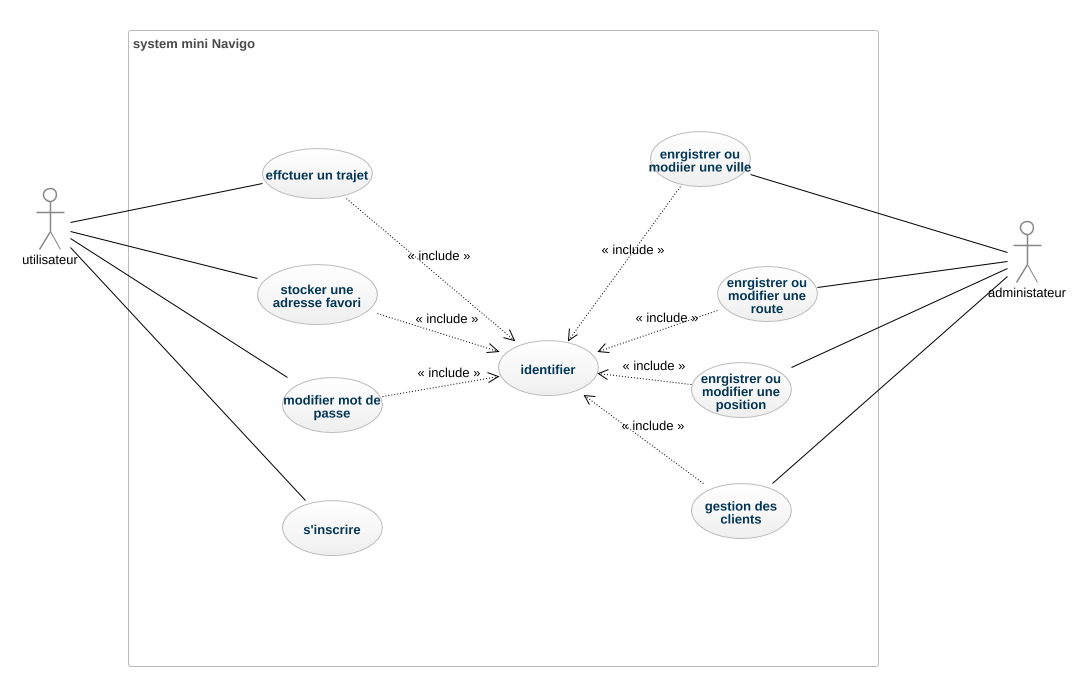
* Économiques
* Écologique :
* type de route
* nombre de radar
* telle route à suivre ou telle route à éviter
* vitesse limitée des routes
* chemin plus court
* chemin plus rapid

## 

## 

Cas d’utilisation :

Diagramme d’utilisation:



|  |  |
| --- | --- |
| Effectuer un trajet de navigation | |
| Cas principal | Et puis il peut faire une recherche de trajet correspondent à une point A (départ)et une point B(arrivé).Logiciel donne le chemin et l’indication au client. |
| Cas alternatif | L’utilisateur s'arrête son trajet, la navigation fini. |
| Cas d’exceptions | Après l’utilisateur fait son recherche,logiciel lui donne d’indication mais quand même il se perte.  L’utilisateur doit refaire une recherche de trajet par la position actuelle. |
| Remarques | lorsque l’utilisateur en train de navigation,qu’il ne peuvent pas modifier les caractéristique de trajet. |

Description textuelle d’un cas d’utilisation :

|  |  |
| --- | --- |
| Nom | Effectuer() |
| Objectif | dessiner le chemin sur la carte à propos des caractéristique que l’utilisateur choisi.  Et montrer l’indication à l’utilisateur. |
| Acteur principal | l’utilisateur |
| Acteurs secondaires | / |
| Statut | cas principal et cas alternatif |
| Étapes principales | L’utilisateur choisit les caractéristiques qu’il préfère,cliquer le bouton pour valider ,logiciel propose les chemins par les contraintes de ces caractéristique. |
| Extension |  |

|  |  |
| --- | --- |
| Nom | Stocker() |
| Objectif | Stocker une position favori |
| Acteur principal | l’utilisateur |
| Acteurs secondaires | / |
| Statut | cas principal |
| Étapes principales | L’utilisateur peuvent stocker une adresse, une ville ou une route par cliquer sur la carte. |
| Extension |  |

|  |  |
| --- | --- |
| Nom | ModifierMDP() |
| Objectif | Modifier le mot de passe d’un ID |
| Acteur principal | l’utilisateur |
| Acteurs secondaires | / |
| Statut | cas principal |
| Etapes principales | L’utilisateur peuvent stocker une adresse, une ville ou une route par cliquer sur la carte. |
| Extension |  |

|  |  |
| --- | --- |
| Nom | inscrire() |
| Objectif | inscrire un nouveau compte |
| Acteur principal | l’utilisateur |
| Acteurs secondaires | / |
| Statut | cas pancipal |
| Etapes principales | L’utilisateur cliquer sur le bouton inscription |
| Extension |  |

|  |  |
| --- | --- |
| Nom | EnregistrerVille() |
| Objectif | Enregistrer une nouvelle ville sur la carte |
| Acteur principal | admin |
| Acteurs secondaires | / |
| Statut | cas principal |
| Étapes principales | L’administrateur peuvent enregistrer une ville avec son nom, son position . |
| Extension |  |

|  |  |
| --- | --- |
| Nom | ModifieVille() |
| Objectif | Modifier la ville sur la carte |
| Acteur principal | admin |
| Acteurs secondaires | / |
| Statut | cas principal |
| Étapes principales | L’administrateur peuvent modifier une ville avec son nom, son position . |
| Extension |  |

de même pour les routes les positions…

|  |  |
| --- | --- |
| Nom | Gestion() |
| Objectif | Gestion des états des utilisateurs |
| Acteur principal | admin |
| Acteurs secondaires | / |
| Statut | cas principal |
| Étapes principales | L’administrateur peuvent consulter des états des utilisateur ex. nombre d’utilisateur |
| Extension |  |

Scénario 1

L’utilisateur doit remplir les information suivant pour inscription:nom d’utilisateur ,mot de passe (au moins 6 lettre ou chiffre ou caractère spécial),mot de passe vérifié , adresse électronique ,numéro de téléphone.

L’utilisateur n’a pas rempile tous les champs d’information,il se trompe en échec et une fenêtre va lancer :’veuillez vérifier tous les information pour vous inscrire’ .après le systèm refus la demande d’inscription. Une fois l’utilisateur clique la confirmation, tous les information rempli va être conservé.

Scénario 2

lorsque l’utilisateur faire sa incription et modifient son mot de passe ,il faut respecter les règles suivant :

1.tous les noms d’utilisateur sont unique.

le nom d’un utilisateur doit être différent avec les autre

(si c’est la même lancée la fenêtre ‘le nom d’utilisateur déjà existé, veuillez entrer un autre’ )

2.le nouveau mot de passe doit être différent avec la mot de passe actuel.

(si c’est le même ,lancer la fenêtre ‘nouvel mot de passe doit être différent avec le mot de passe actuel’)

3.mot de passe et mot de passe vérifié doit etre le meme, sinon

l’inscription se trompe en échec ,lancer la fenêtre ‘veuillez vérifier votre mot de passe et mot de passe vérifié est le même’

Scénario 3

Lorsqu'un utilisateur recherche un itinéraire,  
Vous devez entrer deux adresses. Ces deux adresses doivent exister dans la carte. Dans le cas contraire, la fenêtre contextuelle s'affiche. "L'adresse n'existe pas. Entrez l'adresse correcte."

Une adresse correcte doit être composée de chiffres, noms de rues, noms de villes

Scénario 4

Lorsque l'utilisateur recherche un itinéraire, le système trace un itinéraire sur la carte pour l'indiquer. Pour que l'utilisateur puisse atteindre le lieu où il souhaite arriver, si les deux adresses entrées par l'utilisateur se trouvent dans deux villes différentes et doivent transiter par la ville, chaque ville de transit est une phase de navigation et le système le sera à chaque phase. Demandez à l'utilisateur s'il a réussi à atteindre l'emplacement prédéterminé. Dans l'affirmative, il passera à l'étape suivante de la navigation jusqu'à ce que le point final soit atteint. Si ce n'est pas le cas, l'utilisateur doit naviguer à nouveau.

Scénario 5

L'utilisateur peut stocker une adresse en tant qu'adresse favorite, mais cette adresse doit exister sur la carte. Si l'adresse n'existe pas sur la carte, la fenêtre contextuelle s'affiche: "L'adresse n'existe pas, veuillez entrer l'adresse à nouveau". Une fois l'adresse enregistrée, elle sera enregistrée. En forme de coeur sur la carte

Scénario 6

L'administrateur peut consulter le nombre d'utilisateurs et gérer le compte et le mot de passe de l'utilisateur, mais ne peut pas accéder aux données privées de l'utilisateur, telles que les adresses favorites.

Scénario 7

L'administrateur peut modifier les données de ville, de route et d'adresse sur la carte ou ajouter de nouvelles données de ville, de route et d'adresse à la carte.（l’addresse par la forme : numéro de la route, nom de la route, nom de la ville）

Diagramme de classe

# Image de diagramme de classes

Donc c’est clair qu’on a totalement six classes ici. Une classe d'Utilisateur, une classe de Trajet, une classe de Transport, une classe de Route, une classe de Position et une classe de Ville.

# Signification des classes

On utilise le classe d'Utilisateur pour représenter les utilsateurs de cette application.Et le classe de Trajet pour définir les plans différents pour guider les conducteurs d’un véhicule. Et dans ces trajets différents, on désigne des classe Route, Position et Ville pour présenter ses débuts, les fins et les chemins qu’on a besoins dans les trajets.

Dans ces tableaux, on va expliquer les sens des classes et les relations entre ces classes.

## Classe Utilisateur

|  |  |  |
| --- | --- | --- |
| **Attributs** | **Définition d’attribut** | **Type des attributs** |
| IdUser | Identifiant d’utilisateur | Integer |
| Email | L’adresse email de l'utilisateur pour une inscription et l’identifier | String |
| ListFavori | Une liste des trajets que l'utilisateur suive souvent | List<Trajet> |
| TypeUser | Type des utilisateurs pour les distinguer | String |

|  |  |  |
| --- | --- | --- |
| **Operation** | **Paramètres de Fonction** | **Type de fonction** |
| DemanderTrajet | Position  position | Trajet |
| ChoisirTrajet | Rien | Trajet |
| ChoisirMéthodesTrans | Rien | Transport |

|  |  |  |
| --- | --- | --- |
| **Titre de Relation** | **Classes concernantes de Relation** | **Relation scalaire** |
| Choisir | Utilisateur -> Trajet | 0..1 |
| Offrir | Trajet -> Utilisateur | 1..\* |
| Choisir | Utilisateur ->Transport | 0..1 |
| Être choisi | Transport -> Utilisateur | 1..\* |

Les invariants informels

* Un utilisateur a une identifiant par une adresse mail
* Un utilisateur a une liste des trajets favoris
* Un utilisateur a une seule type
* Un utilisateur peut demander beaucoup des trajets avec deux différentes positions dans chaque fois de guide
* Un utilisateur peut choisir un trajet dans la liste des trajets et la suivre
* Un utilisateur peut choisir une seule méthode de transport dans chaque trajet

## Classe Trajet

|  |  |  |
| --- | --- | --- |
| **Attributs** | **Définition d’attribut** | **Type des attributs** |
| Début | Le début de chaque trajet dans la guide | Position |
| Fin | La fin de chaque trajet dans la guide | Position |
| Durée | La durée totale nécessaire pour chaque trajet de guide | Time |
| Longueur | Le longueur totale calculé de chaque trajet | Double |

|  |  |  |
| --- | --- | --- |
| **Operation** | **Paramètres de Fonction** | **Type de fonction** |
| CalculLongueur | Route.Longueur | Double |
| CalculDuree | Rien | Double |
| CalculConsoCarbon | Transport.ConsoCarbon | Double |
| RenouvellerTrajet | Rien | Trajet |
| CalculTarif | Rien | Double |

|  |  |  |
| --- | --- | --- |
| **Titre de Relation** | **Classes concernantes de Relation** | **Relation scalaire** |
| Contenir | Trajet -> Route | 1..\* |
| Composer | Route -> Trajet | 0..1 |
| Choisir | Utilisateur -> Trajet | 0..1 |
| Offrir | Trajet -> Utilisateur | 1..\* |

Les invariants informels

* Un trajet a juste une durée
* Un trajet a juste une consommation de charbon à cause de type de transport
* Un trajet a juste un longueur comme la somme des chemins choisis dans le trajet
* Un trajet peut être renouvelée par un autre trajet
* Un trajet a juste une tarif pour les cas où il y a des auto-route dans le trajet
* Un trajet est choisi par un utilisateur ou pas d’utilisateur le choisit
* Un trajet contient au moins un chemin

## Classe Transport

|  |  |  |
| --- | --- | --- |
| **Attributs** | **Définition d’attribut** | **Type des attributs** |
| TypeTrans | Type de méthode de transport que l’utilisateur choisit dans chaque trajet à suivre | String |
| ConsoCarbon | La consommation de carbon à cause de cette méthode de transport dans chaque trajet | Double |

|  |  |  |
| --- | --- | --- |
| **Titre de Relation** | **Classes concernantes de Relation** | **Relation scalaire** |
| Choisir | Utilisateur ->Transport | 0..1 |
| Être choisi | Transport -> Utilisateur | 1..\* |
| Marcher | Route -> Transport | \* |
| Permettre | Transport -> Route | 1..\* |

Les invariants informels

* Une méthode de transport doit être choisi par un seul utilisateur
* Une méthode de transport doit permettre au moins une type de route à marcher
* Une méthode de transport doit avoir une somme de consommation de carbon qui est définie

## Classe Route

|  |  |  |
| --- | --- | --- |
| **Attributs** | **Définition d’attribut** | **Type des attributs** |
| IdRoute | L’identifiant de chaque route | Integer |
| Titre | La titre de route | String |
| Longueur | Le longueur de chaque route | Double |
| AvoirRadar | S’il y a un radar dans la route | Boolean |
| TypeRoute | Le type de route | String |
| TypeTrans | Les types des méthodes de transport qui sont permets à marcher dans la route | Transport |
| TitreVille | Le titre de ville ou la route se situe | Ville |
| LimieVitesse | La limite de vitesse pour les marches dans la route | Double |

|  |  |  |
| --- | --- | --- |
| **Titre de Relation** | **Classes concernantes de Relation** | **Relation scalaire** |
| Marcher | Route -> Transport | \* |
| Permettre | Transport -> Route | 1..\* |
| Contenir | Trajet -> Route | 1..\* |
| Composer | Route -> Trajet | 0..1 |
| Situer | Position -> Route | 1 |
| Contenir | Route -> Position | \* |
| Situer | Route -> Ville | 1 |
| Contenir | Ville -> Route | \* |

Les invariants informels

* Une route a un seul identifiant
* Une route a un seul titre
* Une route a un seul longueur
* Une route a une seule type
* Une route permet des différents types de méthode de transport
* Une route se situe dans une seule ville
* Une route a ses limits de vitesse
* Une route a un radar ou pas

## Classe Position

|  |  |  |
| --- | --- | --- |
| **Attributs** | **Définition d’attribut** | **Type des attributs** |
| TitrePosition | Le titre de position | String |
| VillePosition | La ville ou la position choisie se situe | Ville |
| RoutePosition | La route ou la position choisi se situe | Route |
| NumRoute | Le numéro de la route ou la position choisi se situe | Integer |

|  |  |  |
| --- | --- | --- |
| **Titre de Relation** | **Classes concernantes de Relation** | **Relation scalaire** |
| Situer | Position -> Route | 1 |
| Contenir | Route -> Position | \* |

Les invariants informels

* Une position a un seul titre
* Une position se situe dans une seule ville
* Une position se situe dans une seule route
* Une position a son numero unique dans la route

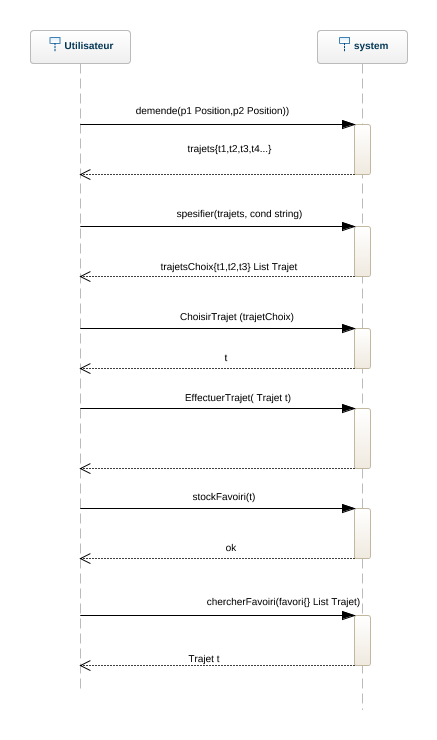
## Classe Ville

|  |  |  |
| --- | --- | --- |
| **Attributs** | **Définition d’attribut** | **Type des attributs** |
| CodePostal | La code postale de chaque ville | Integer |
| NomVille | Le titre de chaque ville | String |
| TypeVille | Le type de chaque ville | String |

|  |  |  |
| --- | --- | --- |
| **Titre de Relation** | **Classes concernantes de Relation** | **Relation scalaire** |
| Situer | Route -> Ville | 1 |
| Contenir | Ville -> Route | \* |

Les invariants informels

* Chaque ville a une seule code postale
* Chaque ville a un seul titre
* Chaque ville a une seule type
* Chaque ville peut contenir plusieurs routes

2.2 Analyse de diagramme de séquence 

-

tout début l'utilisateur entre deux position pour demander chercher des trajets possible. le programme retourne un table de trajet. un trajet c’est une suite des positions. et après l’utilisateur peut spécifier des trajets par la consommation , la distance , le temps. l’utilisateur peut trouver facilement le trajet souhaité. pour plus partique l’utilisateur peut enregistrer le trajet sélectionné dans une liste de favori. la prochain fois, utilisateur peut trouver les trajets dans la liste de favori.

pour le cas perdu on recommence au début.

|  |  |  |  |
| --- | --- | --- | --- |
| call message | paramètre | retourne | description |
| demande | p1 Position,p2 Position | trajets{t1,t2,t3,t4...} | utilisateur demander au class trajet des trajets |
| tousTrajetsPossibles | p1 Position,p2 Position | trajets{t1,t2,, t3,t4...} | la class trajet rappel la class position pour créer des trajets possible. |
| spécifier | trajets, cond string | trajetsChoix{t1,t2,t3} List Trajet | métrer des trajets en ordre selon le condition (plus court, moins) |
| ChoisirTrajet | trajetChoix | Trajet t | choisir un trajet qui est dans la liste trajetChoix |
| EffectuerTrajet | Trajet t | retourne vais si l’utilisateur arrive  faux sinon | effectuer un trajet sélectionné si retourne faux on recherche un trajet |
| stockFavoiri | Trajet t | retourne vais si le trajet bien stocké dans la liste favori  faux sinon | stocker un trajet dans la liste favori |
| chercherFavoiri() | favori{} List Trajet | Trajet t | chercher un trajet dans la liste favori |