



Cahier de conception

BALDE Alhassana

NGO Phong Hoàng

LASLA Youcef

LEI Siqi

Plan

- I. Introduction
- II. Conception générale
 - II.1 Cycle de développement en v
- III. Conception détaillée
 - III.1 Diagramme de classes
 - III.2 Diagramme de cas d'utilisation
 - III.3 Diagramme de séquence
- IV. Description des interfaces
 - IV.1. Scénario maquette
- V. Environnement et outils de développement
- VI. Description des tests
- VII. Pseudo code
- VIII. Contraintes
- IX. Conclusion

Table des figures

Figure I : Modèle en V

Figure II-1 : Page d'accueil

Figure II-2 : Page d'affichage de résultats

Figure III-1 : Diagramme de classes

Figure III-2 : Diagramme de cas d'utilisation

Figure III-3.1 : Diagramme de séquence : Recherche destination

Figure III-3.2 : Diagramme de séquence : Choix du trajet

I. Introduction

Après avoir tracé les grandes lignes de phase de spécification de besoins, mettons l'accent maintenant sur une phase fondamentale dans le cycle de vie d'un logiciel, la phase de conception. Cette phase a pour objectif de déduire la spécification de l'architecture de système.

Cette phase aboutira à la conception et la représentation des diagrammes de séquences et d'activités en se basant sur le langage de modélisation UML.

II. Conception générale

II.1. Cycle de développement en V

Pour la conception, le développement et la réalisation de notre application, nous avons opté pour l'application du processus de développement V qui demeure actuellement le cycle de vie le plus connu et certainement le plus convenable aux projets complexes.

Ce processus nous a accompagné depuis le début et nous accompagnera jusqu'à l'implémentation. Son principe est qu'avec toute décomposition, doit être décrite la recomposition, et que toute description d'un composant doit être accompagnée de test qui permettront de s'assurer qu'il correspond à sa description. Ceci rend explicite la préparation des dernières phases (validation-vérification) par les premières (construction de l'application) et on sait progressivement si on s'approche de ce que le client désire.

On est bien évidemment conscient que le cycle en V n'est pas toujours adapté à un développement logiciel, puisqu'il est difficile voire impossible de totalement détacher la phase de conception d'un projet de sa phase de réalisation. On se rend compte souvent au cours de la mise en œuvre que les spécifications initiales étaient incomplètes, fausses, ou irréalisables, sans compter les ajouts de nouvelles fonctionnalités par les clients. D'autres modèles permettent plus facilement des modifications (parfois radicales) de la conception initiale, donc parmi eux on a la **méthode agile**.

Le schéma ci-dessous représente les différentes phases du modèle en V :

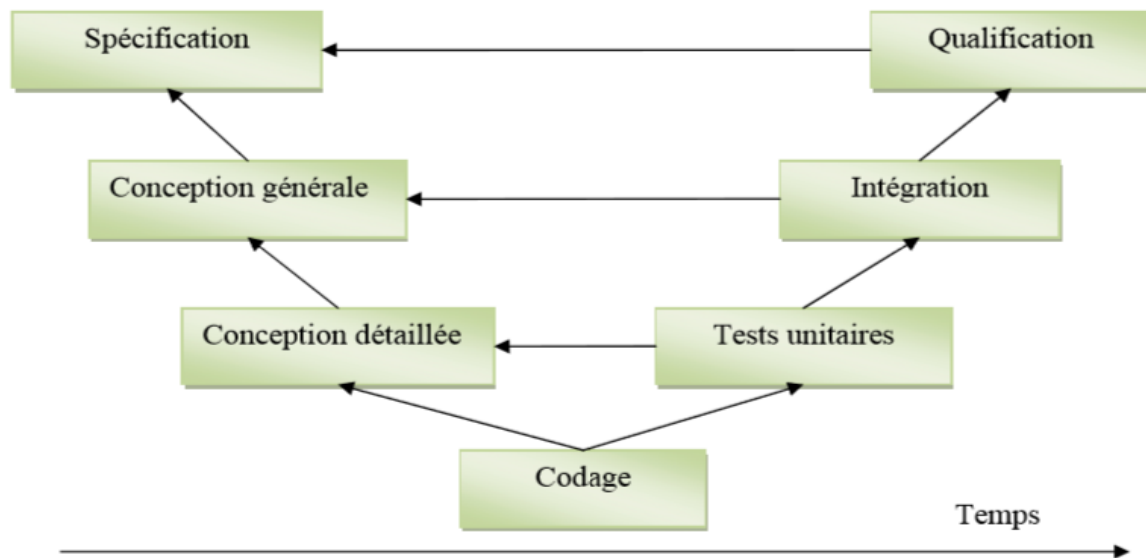


Figure I : Modèle en V

III. Conception détaillée

III.1 Diagramme de classes

Nous avons 4 classes : Interface, Adresse, AdresseSortie, GoogleAPI et enumeration.

- Interface : Classe implémentant l'interface de l'application. Donc toutes les affichages.
- Adresse : Classe implémentant la structure d'une adresse postale.
- AdresseSortie : Classe qui contiendra le triplet d'adresse généré par Google API.
- enumeration : Classe contenant les types énumérés.

Une fois toutes les informations saisies et la soumission de ces dernières faite par l'utilisateur, l'interface se connectera à GoogleAPI pour lui fournir les informations de l'utilisateur en appliquant sa politique de recherche d'adresses par la méthode chercherDes(), et GoogleAPI se servira de ses informations en tenant compte de cette politique puis générer un triplet d'adresses si toutes les conditions sont réunies.

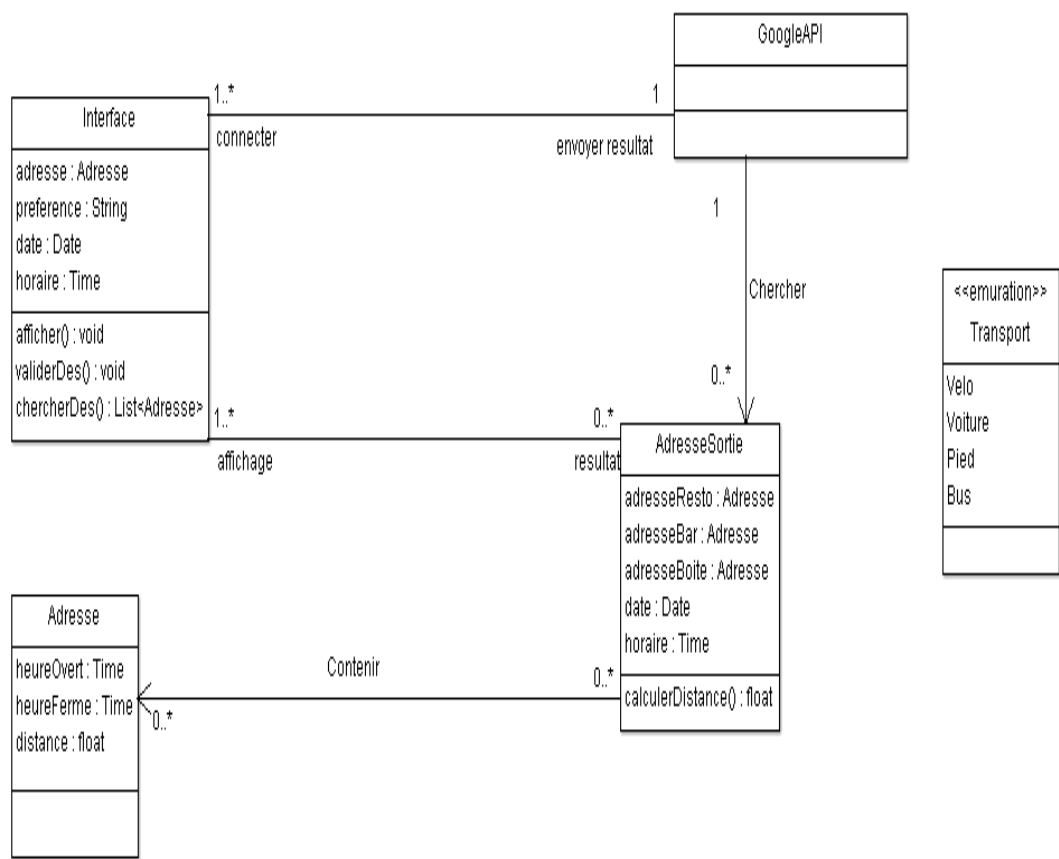


Figure III-1 : Diagramme de classes

III.2 Diagramme de cas d'utilisation

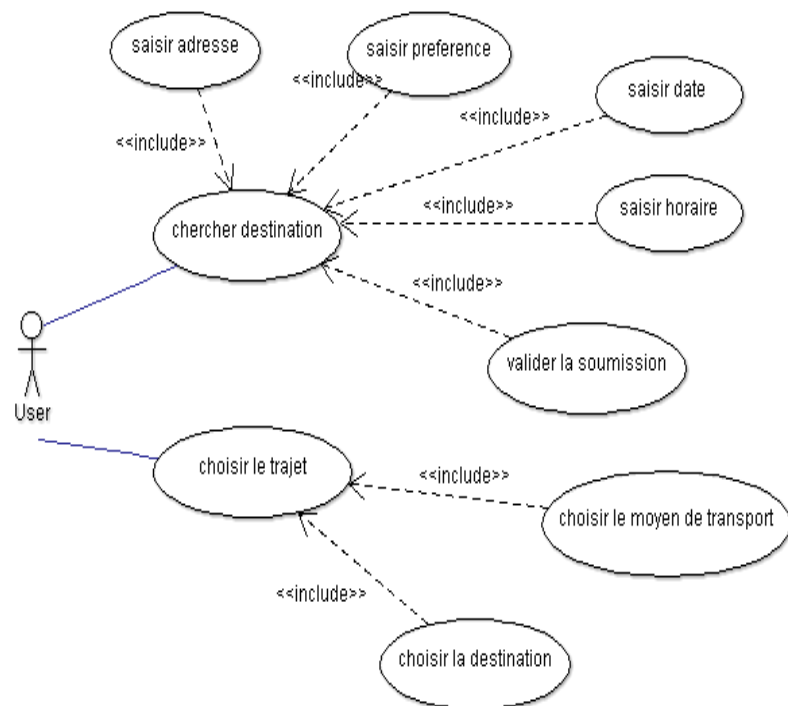


Figure III-2 : Diagramme de cas d'utilisation

Identification des acteurs

Acteurs	Rôles
Utilisateur	<ul style="list-style-type: none">- Chercher les destinations :<ul style="list-style-type: none">+ Saisir les informations.+ Valider la soumission.- Choisir le trajet :<ul style="list-style-type: none">+ Choisir les destinations.+ Choisir le moyen de transport.

GoogleAPI	Fournir un triplet d'adresse
-----------	------------------------------

III.3 Diagramme de séquence

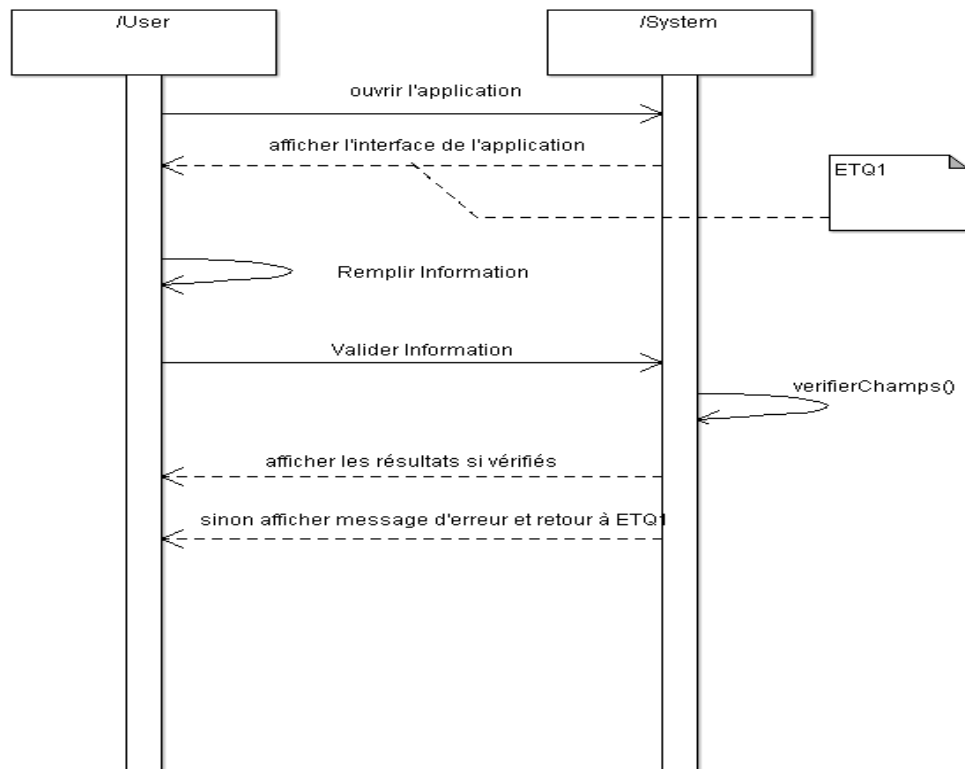


Figure III-3.1 : Diagramme de séquence : Recherche destination

Par système ici on parle de l'interface.

- L'utilisateur ouvre l'application
- Affichage de l'interface d'accueil
- Remplissage des informations par l'utilisateur et leur validation par ce dernier
- Vérification des champs de saisie
- Si tout est ok, affichage des résultats
- Sinon, affiche d'un message d'erreur et retour aux champs de saisie

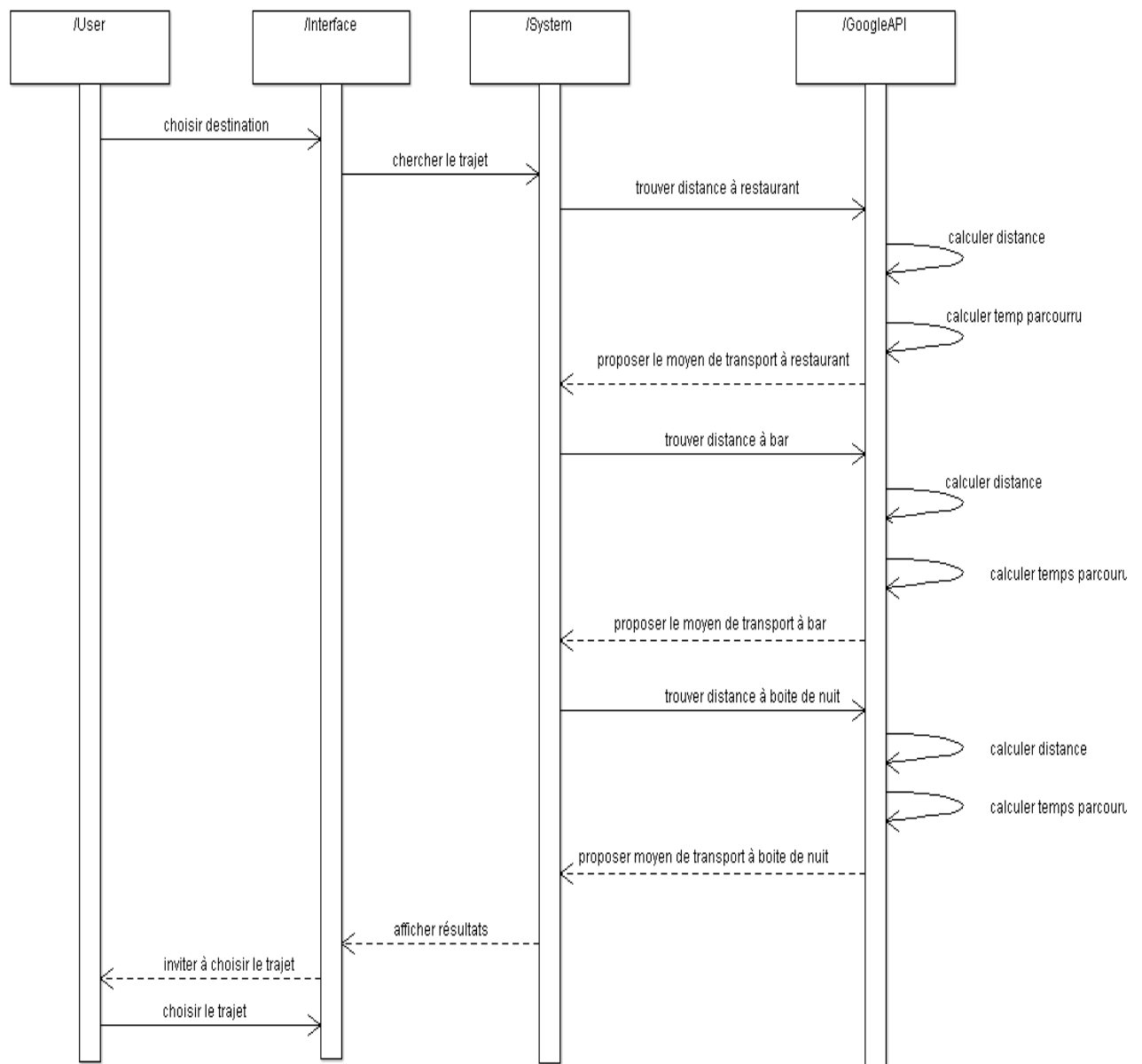


Figure III-3.2 : Diagramme de séquence : Choix du trajet

Une fois le triplet d'adresse fournit par Google API et son affichage sur l'interface, l'utilisateur peut maintenant voir les différentes adresses ainsi les moyens de s'y rendre.

Il verra également en plus des adresses les horaires d'ouverture et fermeture de chaque lieu, ce qui lui permettra de bien gérer son temps durant la sortie.

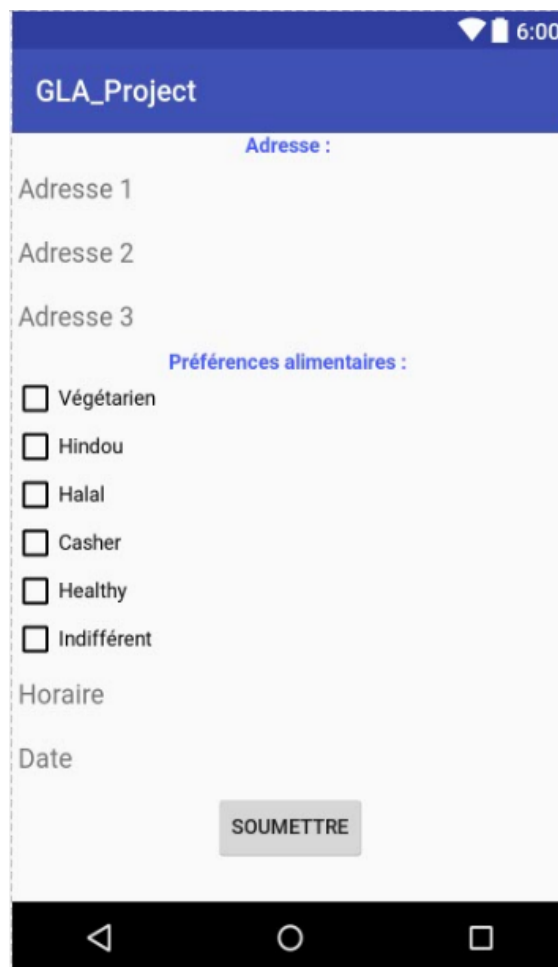
IV. Description des interfaces

IV.1. Scénario maquette

Lors de lancement de notre application, une interface de chargement s'affiche contenant son nom.

Après quelques secondes le menu principal s'affiche, disposant d'une page d'accueil de 5 champs : Adresses, Préférences alimentaires, Horaire (heure à laquelle l'utilisateur veut sortir), Date (la date de sortie), Soumettre (bouton de soumission).

Pour le champs Préférences, l'utilisateur peut faire un ou plusieurs choix. Donc le choix du resto dépendra non seulement de l'adresse de ou des utilisateurs, mais aussi des préférences alimentaires.



The screenshot displays the home screen of a mobile application titled "GLA_Project". The interface is organized into several sections. At the top, there is a blue header bar with the text "GLA_Project" and a status bar showing a Wi-Fi icon, a battery icon, and the time "6:00". Below the header, the section "Adresse :" is highlighted in blue. It contains three text input fields labeled "Adresse 1", "Adresse 2", and "Adresse 3". Following this, the "Préférences alimentaires :" section is also highlighted in blue. It lists six dietary preferences, each with an unchecked checkbox: "Végétarien", "Hindou", "Halal", "Casher", "Healthy", and "Indifférent". Below the preferences, there are two more text input fields labeled "Horaire" and "Date". At the bottom of the form area, there is a grey button labeled "SOUMETTRE". The entire screen is framed by a black border at the bottom, which contains three white navigation icons: a back arrow, a circle, and a square.

Figure II-1 : Page d'accueil

Nous avons aussi décidé de ne pas rendre obligatoire la saisie de tous les champs adresses en tenant compte du cas où le nombre de personnes à sortir n'atteint pas forcément trois. Mais néanmoins il faut saisir au moins une adresse afin de pouvoir déterminer le triplet d'adresses (Bar, Resto, Boite) dont l'utilisateur a besoin pour s'y rendre.

Donc après remplissage des champs de cette page et soumission des données, on traite ces dernières et afficherons le résultat de traitement sur une nouvelle page.

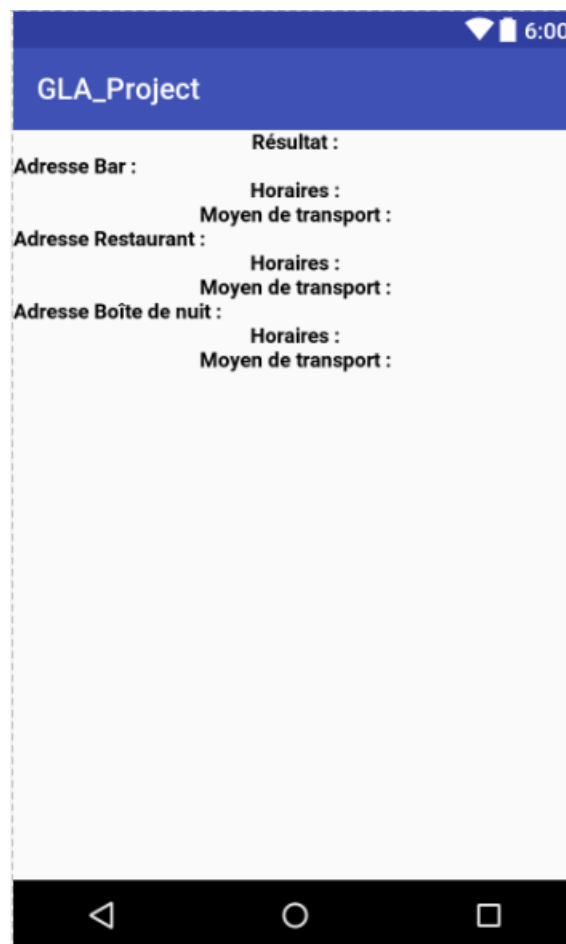


Figure II-2 : Page d'affichage de résultats

V. Environnement et outils de développement

Nous allons utiliser comme matériel de développement nos propres machines et d'une quantité suffisante de RAM (minimum 4Go) pour avoir une qualité de développement acceptable, en partie en raison des nombreux services à exécuter notamment l'IDE.

On utilisera **Eclipse** comme outil principale de développement et aussi **Android Studio** comme environnement de développement pour notre application android.

Le langage utilisé sera donc Java.

On utilisera donc des API Google Maps pour offrir aux utilisateurs bon moyen de localisation, ajouter une carte utilisant des données Google Maps à notre application en fonction des adresses proposées à l'utilisateur.

VI. Description des tests

Les tests servent à déterminer le comportement de l'application en fonction de ce qu'on a saisi et ensuite corriger les erreurs si ces dernières existent.

Le premier test consiste à vérifier le bon fonctionnement de l'application en saisissant les données attendues par l'utilisateur, ce test devra donner un triplet d'adresse.

Le deuxième test ressemble au premier sauf que l'on tombera sur un résultat inattendu, autrement dit l'utilisateur a bien saisi ce que lui est demandé mais n'obtient pas ce qu'il voulait, dans ce cas il s'agit sûrement d'une erreur indéfinie, on affichera donc un message signalant à l'utilisateur de saisir à nouveau ses informations mais cette fois ci en changeant certaines informations telles que l'horaire de sortie ou la date de sortie.

Le troisième test provoque un dysfonctionnement à cause d'un manque d'une entrée attendue par l'application, s'il s'agit d'une absence d'adresse elle va en demander une, vu qu'il est nécessaire d'avoir au moins une adresse en entrée pour fonctionner, s'il s'agit d'une absence de préférences alimentaires qui sera sous forme de plusieurs cases à cocher l'application va considérer que l'utilisateur n'a pas de choix précis et lui donnera quand même l'adresse d'un resto, s'il s'agit d'un manque de date ou d'horaire, l'application va choisir la date d'aujourd'hui et l'heure de la saisie.

Le dernier test s'assure de l'affichage des trois adresses, le cas où l'application ne trouve pas de résultat, un message d'erreur sera affiché en fonction de la variable qui a pour but de stocker lors du fonctionnement de l'algorithme la cause (elle pourrait être à cause : d'une mauvaise date ou horaire, des préférences alimentaires, etc.) qui l'a empêché de ne pas trouver d'adresses, ou tout simplement les informations saisies ne permettent de fournir aucun résultat.

VII. Pseudo code

Proposition de certaines méthodes en pseudo code pour les classes interface et adresse pour expliquer leur fonctionnement.

Choisir_bar(){

```
Si ((horaire + temps_deplacement_bar >= heure_ouverture_bar )
et (heure_fermeture > horraire + temps_deplacement_bar + N_bar))
return adresse_bar ;
else {
    chercher_bar() ;
}
```

Choisir_resto() {

```
si (( horraire + temps_deplacement_bar+temps_deplacement_resto >= heure_ouverture_bar )
et ( heure_fermeture_resto>( horraire +temps_deplacement_bar+temps_deplacement_resto +
N_resto))

return adresse_resto ;
else {
    chercher_resto() ;
}
```

Choisir_boite_nuit(){

```
si (( horraire +temps_deplacement_boite+temps_deplacement_resto+  
temps_deplacement_bar+N_bar+N_resto >= heure_ouverture_bar )
```

```
et ( heure_fermeture_resto> horraire +temps_deplacement_boite+temps_deplacement_resto+  
temps_deplacement_bar+N_bar+N_resto +N_boite))
```

```
return adresse_boite ;
```

```
else {
```

```
chercher_boite() ;
```

```
}
```

VIII. Contraintes

1. Saisie de date et horaires :

Seule les dates et horaires ultérieures à celles du moment de saisie sont autorisées ; Il faut tenir compte de la comparaison entre 23 :59 du jour même et 00 :00 du jour d'avant. Adresse : une adresse valide doit être composée d'un numéro de la voie, d'un nom de rue, d'un code postal et d'un nom de commune, et un nom du pays.

2. La position actuelle de l'utilisateur (la saisie de l'adresse ne suffit pas)

3. Calcul de prévision :

Le temps que prend une sortie = distance entre l'adresse de saisie et l'adresse d'arrivée / vitesse de déplacement selon le transport proposé ; Le transport est proposé par rapport à la distance à parcourir, > 8 km = voiture, > 2,5 km et < 8 km = vélo, < 2,5km = à pied ; Il faut calculer trois temps de prévision séparément, un avant d'arriver au bar, le deuxième entre le bar et le restaurant, le troisième entre le restaurant et la boîte de nuit.

4. Préférences alimentaires :

Doit être sélectionné dans les choix proposés.

5. Validation :

Seule l'interface où tous les champs nécessaires remplis sont autorisés à être validée.

6. Affichage des adresses de résultats dans l'ordre chronologiques en indiquant les horaires d'ouvertures

IX. Conclusion

Le cahier de conception nous permet, dans la suite du cahier de charge, de décrire plus précisément les technologies employées et la manière dont les approches algorithmiques sont abordées. Pour ce fait, nous avons présenté le modèle en V, puis les fonctionnalités que nous envisageons de développer sous forme des différents diagrammes dont l'interface, diagramme de classes, diagrammes de cas d'utilisation, et enfin diagramme de séquence. Le cahier de conception agit comme un document nécessaire au maintien du développement logiciel. Afin de remplir cet objectif, nous avons décrit en détail les scénarios possibles afin que les utilisateurs puissent appréhender les scènes d'utilisation à prévoir. Ce document permet aux futurs développeurs d'aborder un projet en marche, sans toucher l'implémentation. Les architecture technique et fonctionnelle tel que les choix d'IDE, de langages et librairies et enfin le fonctionnement des tests, permet aux programmeurs de mieux identifier les contraintes et failles possibles en s'en servant de références, lors du futur maintenance et d'éventuellement amélioration.