Cahier de conception

Kayak

**GROUPE K**

# **Table des matières**

Introduction………………………………………………………………………………………………………………………2

Exigences de réalisation…………………………………………………………………………………………………….2

Performance………………………………....….…………………………………………………………………..2

Robustesse…………………………………....….…………………………………………………………………..2

Sécurisation.………….……………………....….……………………………………………………..…………..2

Synchronisation…….……………………....….……………………………………………………..…………..2

Analyse et conception….…………………………………………………………………………………………..……….3

Spécificités techniques…………………....….……….………………………………………………………..3

Librairies…….……………………....….……….……………………………………………………..……………..3

Interface graphique…………....….……….……………………………………………………..……………..3

Fonctionnement du système et diagrammes UML…….……………………...…………………..4

Développement….………………………………………………………………………………………..…………..……….2

Types énumérés….………………………………………………………………………………………..………..2

Les classes….…………………………………………………………………………….…………………..………..2

Attributs……………………………………………………………………….………….………..………..2

Procédures et rôles…………………………………………………………………….….…..…..…..2

Tests….…………………………………………………………………………..……………………………..…………..……….2

Cycle en V…….……………………………………………………..……………………………….………..……….2

# Introduction

Notre projet Kayak consiste en la réalisation d’une application d’agent de sortie dans Paris. Le système devra proposer successivement trois sorties : dans un bar puis un restaurant et enfin une boîte de nuit.

Ce projet devra fournir une interface avec en entrée : une date et un horaire de début de soirée, les adresses des participants, leur moyen de transport au cours de la soirée et leurs préférences alimentaires pour le choix du restaurant.

Il fournira en sortie une liste chronologique des trois adresses de sortie.

Ce cahier de conception est la deuxième phase de notre projet. Nous allons exposer nos exigences de réalisation et présenter un modèle physique de l’application en considérant toutes les contraintes imposées pour sa réalisation.

# Exigences de réalisation

**Sécurisation**

Le besoin de sécurité pour cette application est minime, il n’y aura pas de données sensibles, comme un code de carte bleue par exemple, mais uniquement des demandes anonymes pour des localisations ou des demandes spécifiques pour des restaurants. Kayak ne garde aucune donnée une fois l’opération effectuée : il n’y a donc aucun risque de vol de données.

**Performance**

L’objectif de conception est d’avoir un temps de traitement le plus faible possible. Nous allons donc limiter le nombre de réponses renvoyées par Google par un maximum afin de ne pas attendre plusieurs minutes pour trouver le meilleur résultat.

Une interface graphique fluide et facile d’utilisation sera nécessaire pour une bonne performance.

Nous avons choisi de ne pas garder les données de l’utilisateur, ce qui peut influencer sur la performance de notre application. En effet, si un utilisateur refait une recherche similaire, on aurait pu directement lui renvoyer le triplet qui a été envoyé auparavant. Mais nous avons décidé de favoriser la sécurité.

**Robustesse**

L’application doit pouvoir tenir malgré les calculs et les communications avec le serveur Google (cf performance).

**Synchronisation**

L’application doit pouvoir communiquer avec les serveurs Google et récupérer toutes les réponses.

# Analyse et conception

**Spécificités techniques**

* L’application sera développée pour un PC
* Les langages choisis sont JAVA pour le cœur de notre application et JSON pour la communication avec les serveurs Google
* L’environnement de développement se fera sur Eclipse avec un répertoire Git afin de favoriser les échanges entre les collaborateurs du projet
* Il y aura une interface graphique faite avec SWING
* Héritages entre nos classes possibles
* L’application utilisera différentes API pour communiquer avec les serveurs Google

**Librairies**

* **Communication avec Google : JSON, Google Maps Distance Matrix API, Google Maps Direction API, Google Places API Web Service**
* **Interface : javax.swing, utilisation de JApplet, JDialog, JFrame, JWindow**
* **Tests : JUnit**
* **Autre : Arraylist, Date**

**Interface graphique**

**Deux possibilités nous ont été offertes très vite : SWING ou JavaFX. JavaFX est plus fluide et permet plus de libertés. Néanmoins, l’interface graphique n’étant pas la priorité dans notre projet, nous avons donc décidé d’utiliser SWING car plus simple d’utilisation.**

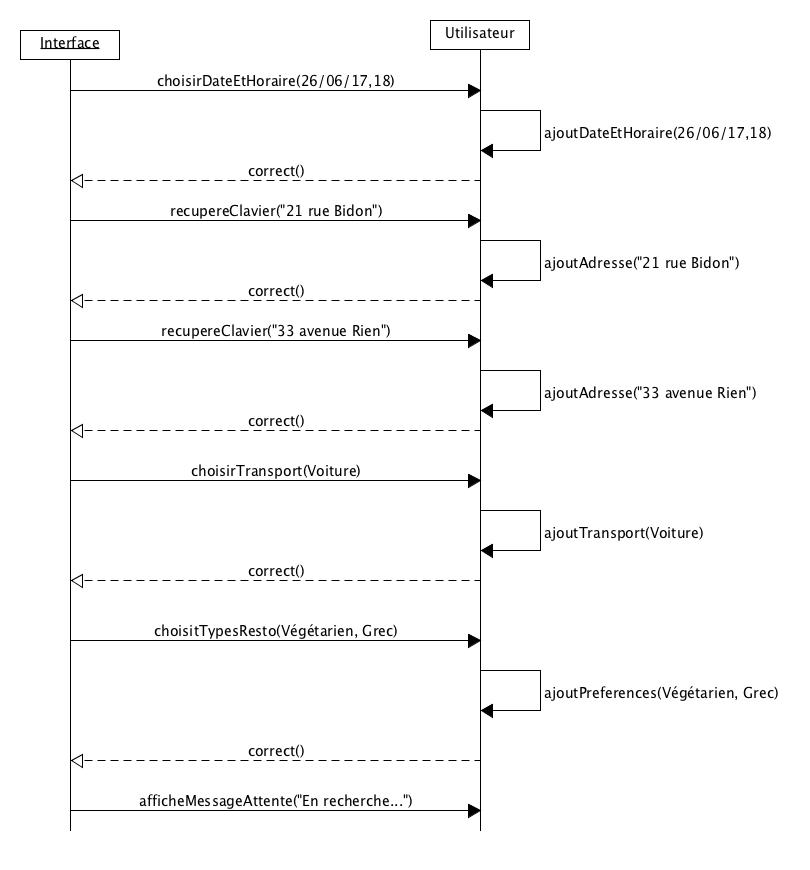
**Fonctionnement du système et diagrammes UML**

Détails des interactions

**L’utilisateur interagira avec l’application via l’interface de la manière suivante :**

* **Il entre une date et choisit dans un menu déroulant un horaire de début de sortie**
* **Il entre son adresse ainsi que celles de ses amis participant à la sortie**
* **Il choisit un moyen de transport dans un menu déroulant que tout le monde va utiliser pendant la soirée**
* **Il choisit maintenant les préférences alimentaires et les types de restaurant dans lequel ils veulent manger (par exemple : chinois, végétarien, grec)**
* **Il valide ces informations en cliquant sur un bouton**
* **Pendant le calcul, une fenêtre de chargement s’affiche pour signaler que la demande est en cours de traitement** 
  + **Voir le diagramme de séquence 1 illustré plus loin**
* **Une fois un établissement trouvé, l’utilisateur aura le choix de l’accepter ou de le refuser (le processus est décrit plus loin)**
* **(d’abord un bar, puis un restaurant, enfin une boîte de nuit), l’utilisateur aura le choix de l’accepter ou de le refuser. Si l’utilisateur accepte, l’application cherche le lieu suivant. Sinon, elle recherche un autre établissement (par exemple : si l’application venait de proposer un bar et que l’utilisateur n’est pas satisfait alors elle va choisir le deuxième bar dans la liste que le serveur Google avait retourné)**
* **À l’écran s’affiche enfin le triplet « Bar-Restaurant-Boîte » avec leur adresse respective**

Diagramme de séquence 1



**Le rôle des classes et des méthodes sera détaillé plus loin dans ce cahier (cf Développement). Sur ce diagramme, « Utilisateur » est une classe de notre projet et ne représente pas l’acteur, c’est-à-dire la personne qui utilise l’interface. Cette classe enregistre les données de l’utilisateur seulement, et constitue un objet représentant celui qui utilise l’application.**

Détails du noyau du système

Le noyau du système repose sur la gestion des données que l’utilisateur a renseigné et l’interaction entre l’application et les différentes API Google.

Voici le procédé de recherche des 3 adresses de sorties :

* L’application enregistre les données de l’utilisateur (dans la classe Utilisateur) afin qu’elle puisse les utiliser par la suite (par exemple, les adresses des participants seront enregistrées dans une ArrayList)
* Elle convertit ces données en JSON afin de pouvoir les envoyer au serveur Google
* À l’aide de règles mathématiques simples, l’application trouve le point à égale distance de toutes les adresses des participants pour décider d’une localisation de départ pour la recherche d’un bar
* Elle choisit un périmètre maximum dans lequel va se trouver les trois établissements de sortie
* L’application demande à l’API Google Places de nous trouver une liste de bar les plus proches de la localisation de départ
* Par défaut, notre application choisira le premier bar de la liste afin de le renvoyer à l’utilisateur
* L’utilisateur accepte ou refuse l’établissement proposé par notre application. S’il refuse, on propose le deuxième bar de la liste et ainsi de suite.   
  S’il accepte, notre application va faire le même processus pour le restaurant puis la boite de nuit. Elle renvoie le temps de trajet entre l’établissement d’avant ou le lieu de départ et celui renvoyé, dans le moyen de transport que l’utilisateur a choisi, calculé par l’API Google Matrix
* Pour le restaurant, l’application renseigne les préférences des participants au serveur Google
* Lorsque l’utilisateur aura accepté la boite de nuit, l’application lui renvoie le triplet final « Bar-Restaurant-Boite » avec les adresses respectives et peut accepter définitivement ce triplet
* En quittant l’application, les données de l’utilisateur ne sont pas enregistrées

Le cas d’utilisation de l’application avec les différents acteurs :

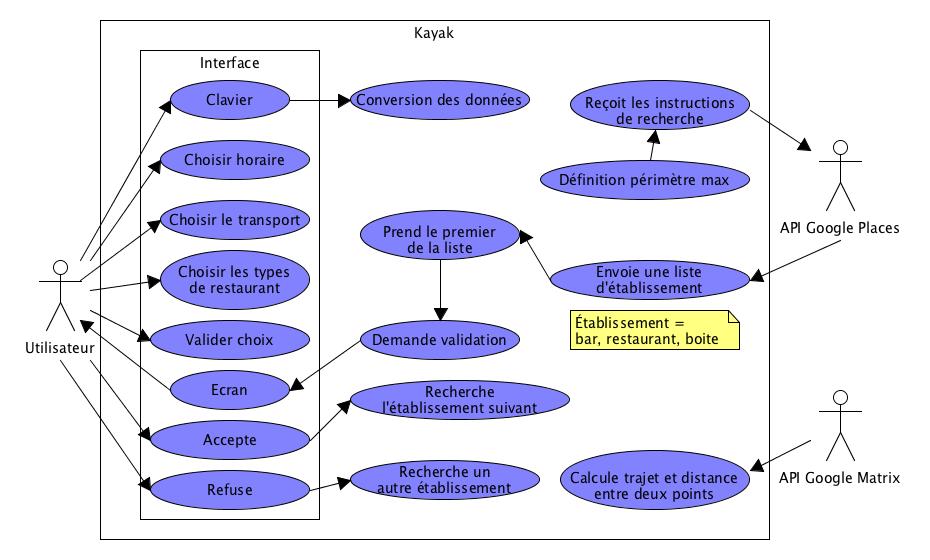


Diagramme de cas d'utilisation

* Le clavier se charge de récupérer les informations renseignées par l’utilisateur telles que les adresses des participants.
* L’écran communique à l’utilisateur les informations renvoyées par l’application.
* La conversion des données se charge de convertir les données de l’utilisateur en JSON, langage dans lequel les API Google traitent les informations.
* Valider choix sera un bouton de validation final du triplet « Bar-Restaurant-Boite »

Deux diagrammes de classe :

* Diagramme de classe interface-utilisateur :

