Rapport du projet CSC5002

VlibTour

Ahmed BAHRI / Sylvain MENEZ

Table des matières

1- Introduction 2
2- VlibTour use cases 2
3- Architecture 5
4- Extra-fonctionnalités 8
5- Avancement et perspectives de développement 9
Conclusion 9

1- Introduction

Le projet VlibTour vise à développer une application permettant à des touristes de visiter Paris en groupe, en suivant des tours précis composé de différents points d'intérêts, tout en utilisant le système VLib.

Le but de ce projet est d'arriver à produire une simulation où nous verrons un groupe de touristes qui utilise ce système. Ils choisiront un même tour « The unusual Paris » composé de trois points d'intérêts : « Musée Grévin », « Pyramide du Louvres » et « Les catacombes de Paris ». Chaque touriste informera de sa position et aura accès à la position des autres touristes inscrits au même tour.

Au cours de ce rapport, nous présenterons d'abord les différents use cases des différents acteurs. Puis nous définirons l'architecture de l'application. Par la suite nous évoquerons certains extra-fonctionnalités avant de conclure par un bilan de notre avancement et nos perspectives de développement.

2- VlibTour use cases

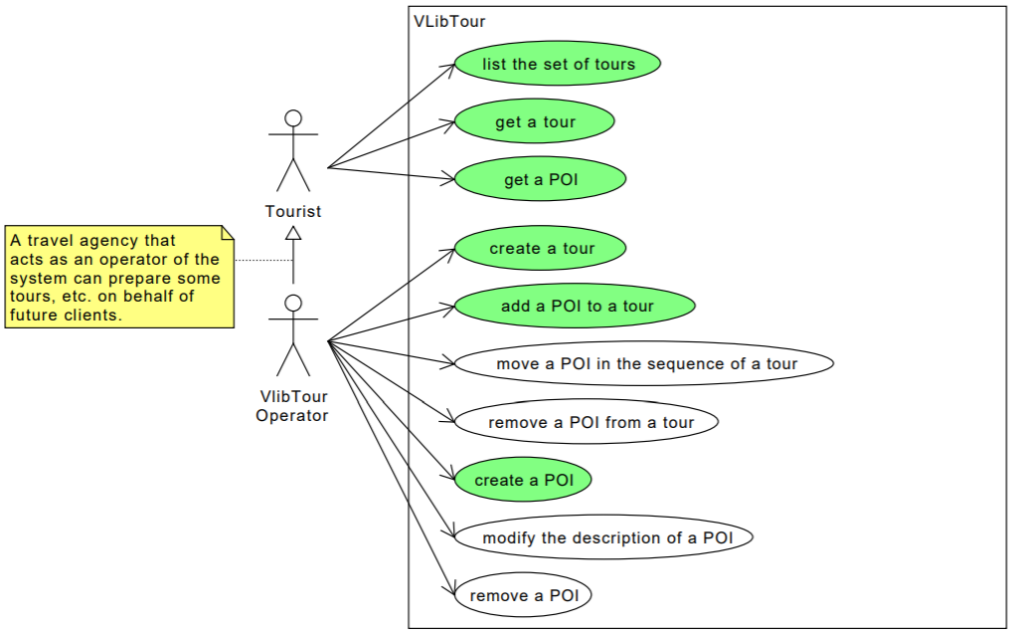
Dans la partie suivante nous décrirons l'architecture de l'application sous différents modules. Vous pourrez vous faire une première idée de cette architecture grâce aux regroupements logiques des use cases que nous allons faire ci-dessous.

a. Gestion des tours et des POI (points d'intérêts)

Dans ce groupement nous allons retrouver deux acteurs. Un premier acteur qui est un opérateur de l'application qui va s'occuper de l'administration des tours et des POI. C'est-à-dire il va créer les tours, ajouter, supprimer, changer l'ordre des POI dans les tours. Il va également créer les POI, les supprimer ou modifier leur description.

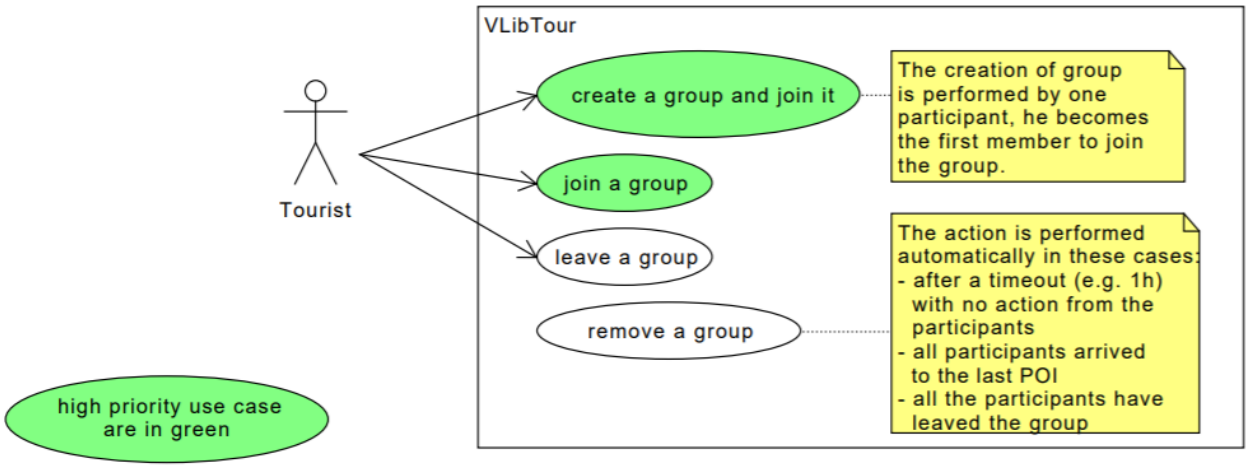
Il y a ensuite un deuxième acteur côté client qui est le touriste. Il aura accès à la liste des tours et pourra choisir un tour et avoir accès aux POI.

Vous trouverez ci-dessous le diagramme de cet use case :



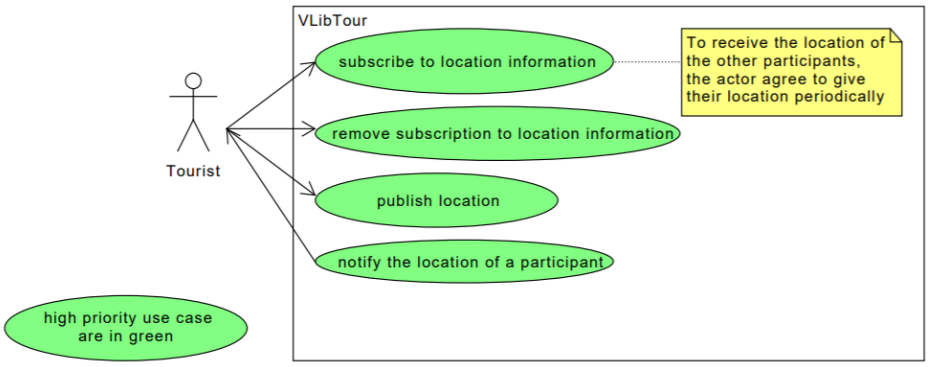
b. Gestion des groupes et des participants

Dans cette partie il y a seulement un acteur : le touriste. Chaque touriste a accès à la liste des groupes, il peut alors soit en créer un soit en rejoindre un en cours. Une fois le tour terminé les groupes sont automatiquement supprimés par le serveur, les touristes n'ont pas à s'occuper de le faire. Voici ci-dessous le diagramme use case.



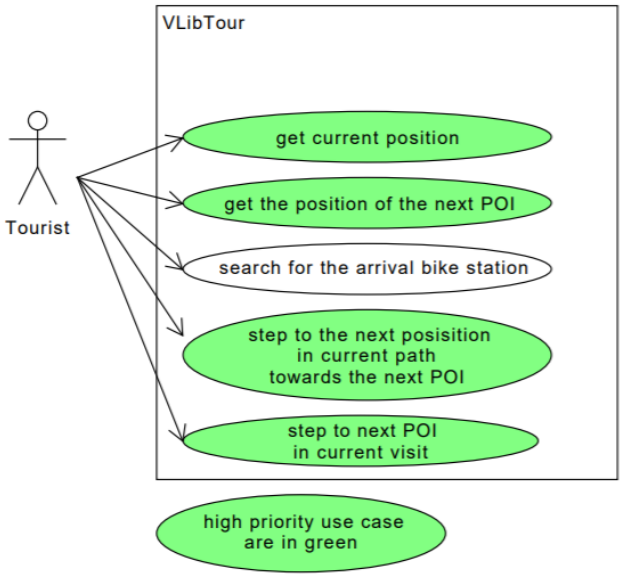
c. Gestion de la position des touristes

Cette partie concerne l'autorisation et la publication de la position de chaque touriste. Ainsi le seul acteur est le touriste, il doit donner/enlever l'accès à sa position, la publier et la signaler aux autres participants.



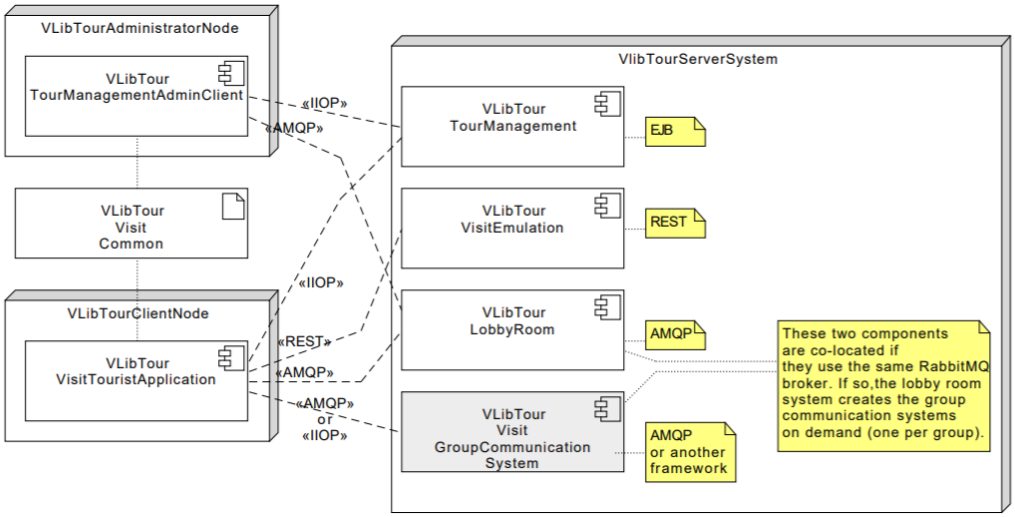
d. Gestion de l'avancement d'une visite

Finalement le dernier module concerne la gestion de l'avancement d'une visite. Ainsi à tout moment le touriste a accès à sa position, à la position du prochain POI, aux positions des différentes stations VLib ainsi que la progression dans le tour.



3- Architecture

Afin de répondre à ces différents use case nous allons utiliser l'architecture globale suivante :

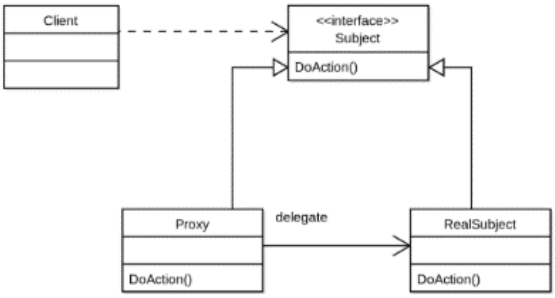


a. VLibTour VisitEmulation

Ce module utilise une API REST afin de créer une émulation d'une visite suivant le scenario présenté dans l'introduction. Chaque touriste avance pas à pas et allant de POI en POI tout en ayant accès à leur position grâce à OpenStreetMap. Ce module utilise des requêtes http avant de récupérer des données en format JSON.

L'API est composée de 4 fonctions :

- getNextPOIPoisition qui donne la positon du prochain POI dans la visite ;
- getCurrentPosition qui donne la position actuel du client ;
- stepInCurrentPath qui donne le prochain pas dans le tour ;
- stepsInVisit qui permet de savoir lorsque l'on a atteint le dernier POI de la visite ;

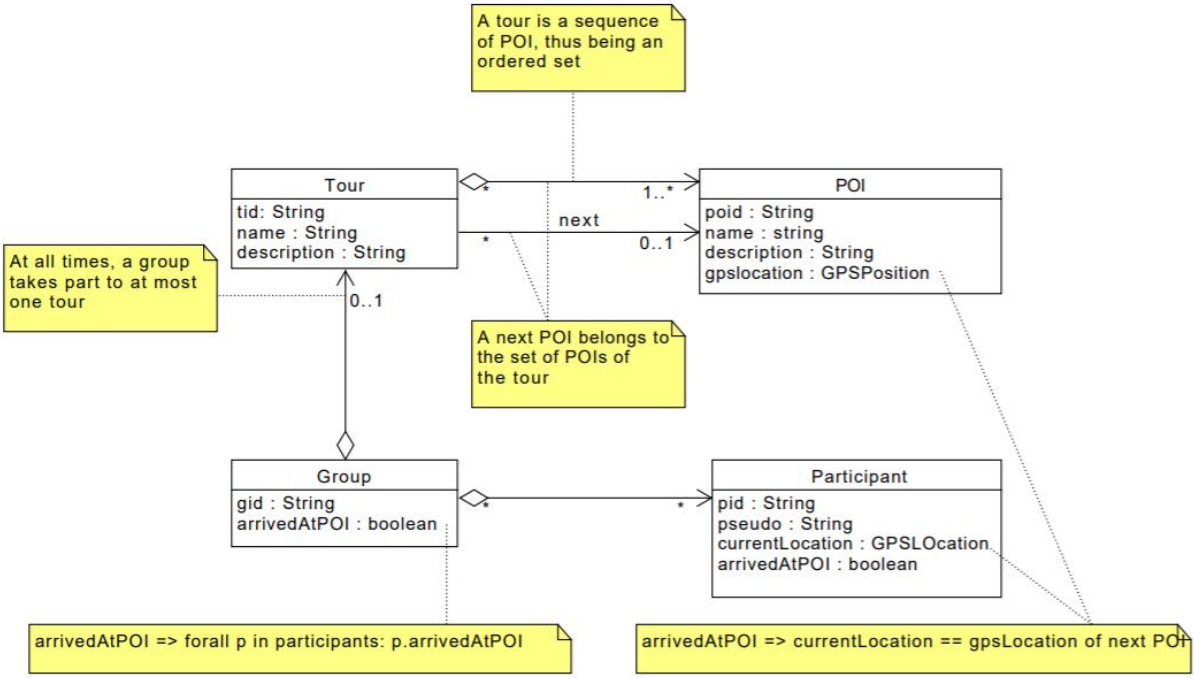


Enfin dans ce module nous avons utilisé un proxy pour le côté client afin de simplifier le développement du client.

b. VLibTour TourManagement

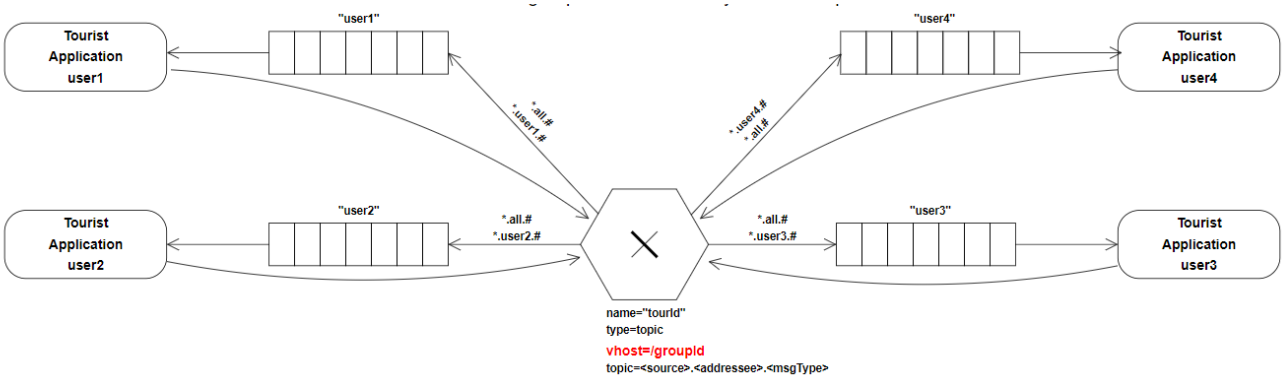
Le module VLibTour-Tour-Management contient les entités et les sessions de l'utilisateur. Cette partie garantie la persistance des objets java dans des tables SQL grâce à une application JAVA EE. Ainsi ce module contient un EJB qui modélise la session des utilisateurs et qui expose plusieurs méthodes comme l'affichage des groupes ou des tours disponibles dans la base de données sur le côté serveur. L'EJB offre aussi des méthodes à l'administrateur pour modifier, ajouter et supprimer des données dans la base de données. Ce Bean est distant, c'est à dire on peut l'utiliser via l'extérieur et non pas seulement depuis la même machine. On doit donc le déployer sur le serveur Glassfish et y accéder depuis les autres modules du système.

Vous trouverez ci-dessous le diagramme des classes de notre base de données :

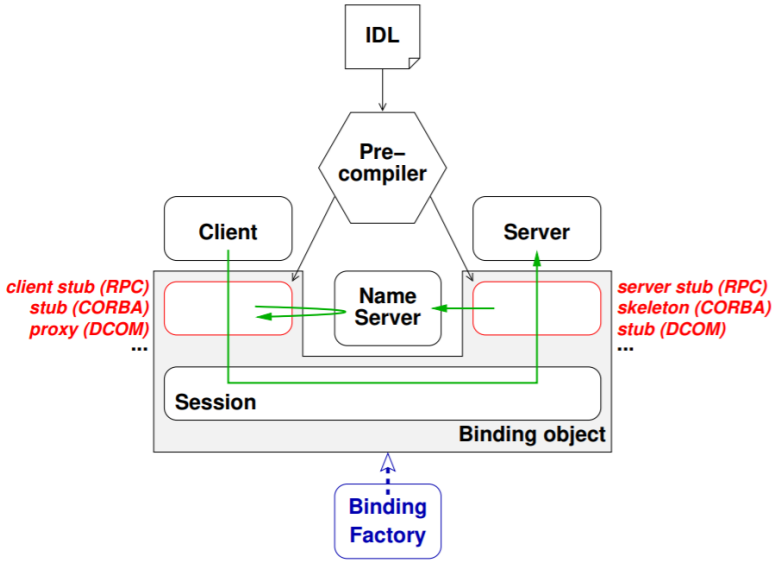


c. VLibTour Visit GroupCommunication System & LobbyRoom

Pour les modules LobbyRoom et GroupCommunication on établit tout d'abord une connexion distante avec les EJB. Ensuite on utilise un serveur RabbitMQ et le protocole AMQP afin de gérer l'échange des informations entre clients et serveur. Pour chaque groupe d'utilisateurs on crée des hôtes virtuels 'vhost' en utilisant un proxy pour le client afin de faciliter l'échange des messages entre clients.



Le module LobbyRoom utilise un design pattern de client/serveur qui utilisent des requêtes RPC. En effet, le serveur attend des requêtes des clients afin de rejoindre ou créer des groupes en utilisant une instance GroupCommunication qui crée la structure pour les échanges de données.



Pour la partie LobbyRoom nous avons mis en place le pattern singleton. C'est-à-dire qu'il existe une seule instance LobbyRoom qui est mise à jour sur laquelle on opère régulièrement des « get ».

LobbyRoom
- lobbyroom : LobbyRoom
- Lobbyroom() + getInstance() : LobbyRoom

d. VLibTour VisitTouristApplication

Ce module est celui qui fait lien entre tous les modules. C'est ici que l'on réalise notre scénario de test. Ainsi, nous avons créer un scenario avec 2 utilisateurs qui réalisent un même tour, se suivent au début puis suivent un chemin différent pour le dernier POI.

Tout d'abord nous créons un client EmulationVisit, puis un client LobbyRoom qui va créer ou rejoindre un groupe. Enfin une instance GroupCommunication System est créé et les deux touristes vont s'échanger leurs positions et leurs noms à l'aide de messages cryptés. En effet, à chaque fois qu'un client a une nouvelle position, il partage sa position et son nom à tous les membres du groupe grâce à la Binding Key *.all.*. Ces derniers vont consommer le message et afficher la position avec un délai de trois secondes.

4- Extra-fonctionnalités

Afin de répondre à l'extra-fonctionnalité de la scalabilité nous avons mis en place dans le module LobbyRoom des hôtes virtuels. Ainsi chaque 'vhost' peut être déployé sur une machine virtuelle ce qui permet d'augmenter de façon conséquente le nombre d'utilisateurs en évitant d'augmenter le nombre de ressources dans le cas où on utiliserait uniquement une machine. La scalabilité de notre scénario est possible même si dans notre exemple on réalise un scénario avec 2 touristes. En effet, nous pouvons utiliser une liste/map qui contient les noms et positions des autres clients au lieu d'un seul. De plus, dans le module TourManagement on associe un Bean pour chaque client. Le Bean étant sans état, pour chaque requête on ne maintient pas l'état de la discussion entre le client et le Bean. Cela permet une grande scalabilité puisqu'on a besoin d'un nombre réduit d'instanciation de Bean et que chaque Bean peut traiter plusieurs clients.

Afin de répondre à l'extra-fonctionnalité de la sécurité, il faut déterminer quels sont les points critiques du système. Tout d'abord, il est essentiel de chiffrer tout message passant d'un module à un autre, ou bien entre un client et le serveur. Pour cela nous avons crypté les messages en utilisant un cryptage AES. Ensuite, l'utilisation du 'vhost' assure une fonctionnalité de sécurité puisque chaque client doit avoir un mot de passe pour y accéder. De plus, ce 'vhost' permet d'empêcher un client d'envoyer un message aux autres groupes.

5- Avancement et perspectives de développement

Dans les contraintes de temps imposés nous avons réussis à implémenter chaque module indépendamment et créer un scénario qui permet de tous les utiliser. Pour cela, on commence par lancer le serveur avec le module TourManagement, l'AdministratorNode, on initie le LobbyRoom et GroupCommunication System, puis vient le module VisitTourEmulation avant de finir par lancer la VisitTouristApplication qui lance la visualisation des tours. Nous avons réalisé un scénario avec 2 clients mais il est possible de faire avec plus de clients grâce à une utilisation de liste de clients.

Nous avons eu pour idée de développement futur de configurer des privilèges aux différents types de client suivant qu'il s'agit d'un simple touriste ou d'un administrateur. Également, nous avions envisager de mettre en place un load Balancer sur les serveurs REST et RabbitMq afin d'améliorer la scalabilité et la disponibilité de l'application.

De plus nous avions envisagé de rajouter des modules permettant la gestion des Vlib disponibles dans les stations, synchroniser leur usage avec les tours, et de limiter les places par groupe en fonction des Vlib disponibles par exemple.

Nous avons rencontré quelques difficultés au cours de ce projet. La principale est la contrainte de temps. Sur le plan technique nous avons commencé à rencontrer des difficultés à partir de la partie EJB. En effet, il fallait précisément comprendre et implémenter le diagramme des classes en ajoutant les relations nécessaires. Ensuite sur le module GroupCommunication System, il a fallu bien comprendre les architectures AMQP, les implémenter et trouver les tests appropriés afin de vérifier la bonne implémentation de ce module. Sur le module LobbyRoom, la principale difficulté fut l'utilisation d'instance GroupCommunication System lié aux requêtes RPC.

Conclusion

Dans ce projet, nous avons appris plusieurs nouveaux concepts et technologies. Il nous a également donné une idée de comment concevoir une application dédiée aux entreprises. Il ne s'agit pas simplement de faire des fonctionnalités pour un nombre réduit d'utilisateur, mais plutôt concevoir un système en ayant l'idée de traiter des milliers d'utilisateurs à la fois.

De plus, au cours de ce projet nous avons pu développer nos compétences personnelles et travailler des méthodes de développement et d'organisation en binôme.

Pour conclure, ce fut une expérience enrichissante personnellement et techniquement.