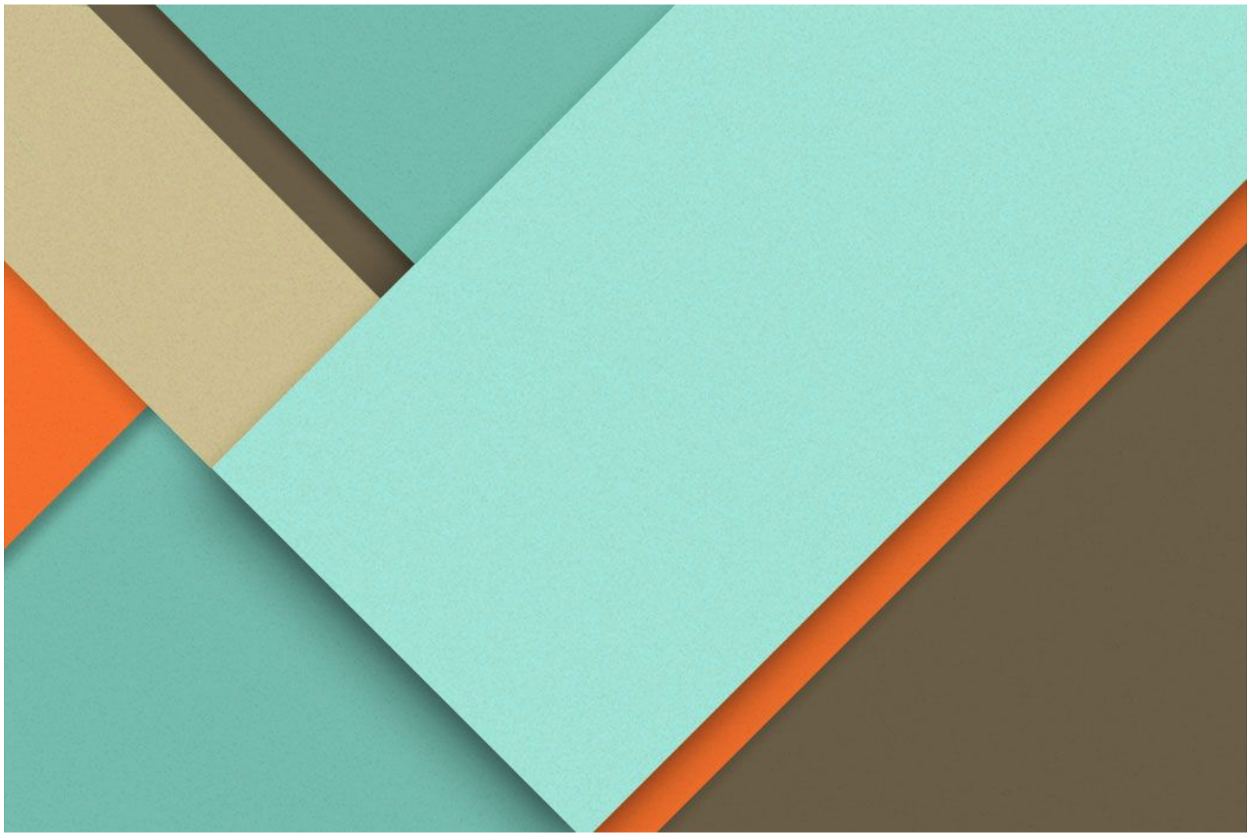


Rapport de Micro Projet VlibTour CSC5002

24.10.2019

Binôme

Farah Kilani

Nada Touil

Table of Contents:

1.Introduction:

The VlibTour is an application used by tourists for managing tours, groups and the mutual communication of their locations. This would make the tour more attractive and interactive.

In this report, we present our POC for the server side of VlibTour that acts as middleware between the clients' mobile applications and server in order to offer services of geolocalisation, group creation and bikestations' information.

We developed scalable (each new functionality could be added in project and thus tested separately) and maintainable modules that considers a huge number of users. We used technos dedicated to companies like EJB. This approach helped us acquire a vision on technologies present on the market and incorporated for the distributed systems.

The content of our report is organized as follows:

We start by specifying application needs and presenting the global architecture of our solution. Later, we will detail each developed module in chronological order of their realisation.

Then, we analyse some extra-functionalities that are provided in our implementation such as scalability and security and finally suggest that we would add to the implemented elements.

2. Needs specification

After analysing the different use cases described in the micro project subject, we come to the conclusions above.

- The tour management service needs to be secure and scalable.
- The visit emulation and bike stations servers needs to be always available in order to respond to tourist's queries.
- The Lobby room service needs to be secure from external attacks and interception to ensure communication confidentiality .

In a nutshell, we need a scalable and available application that responds rapidly to client queries .

3. Architecture of our POC

3.1 Structural Architecture

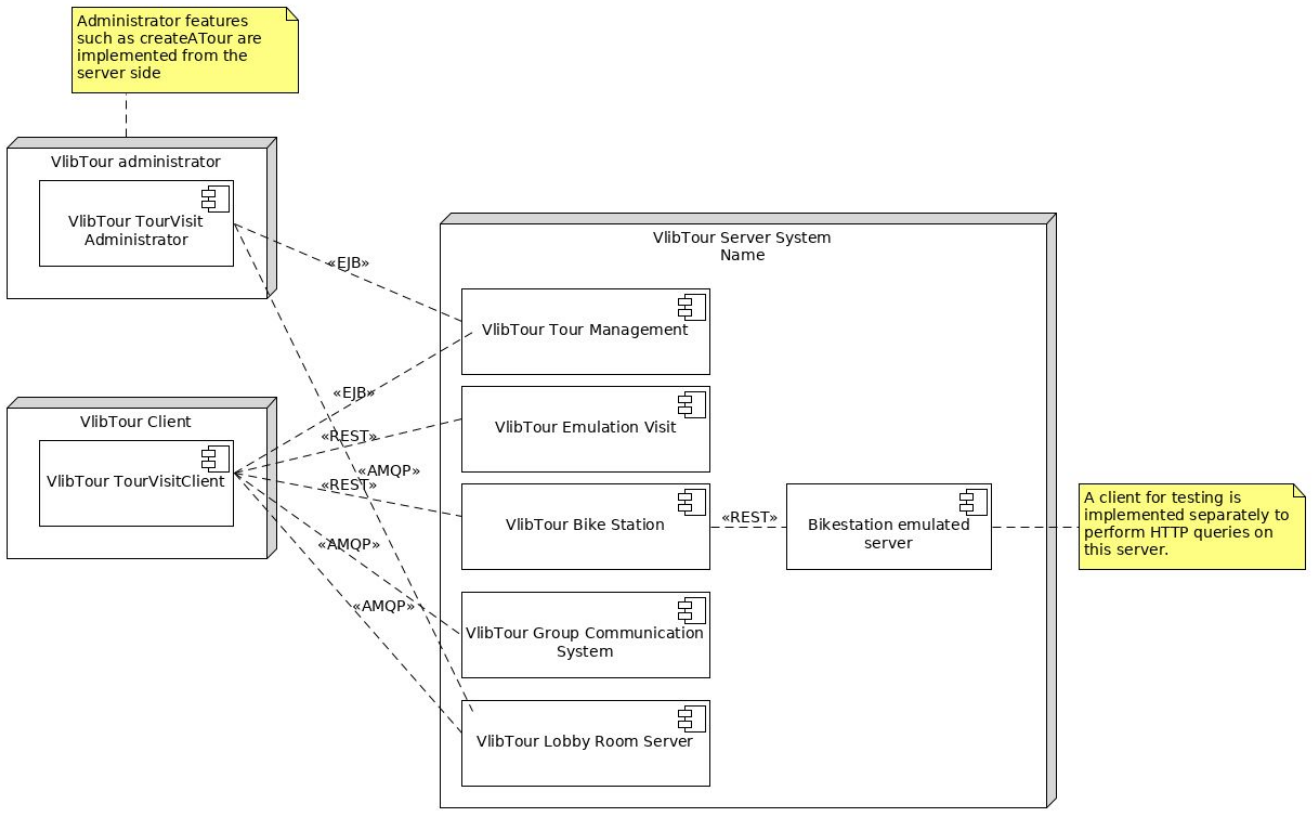

Administrator features such as createATour are implemented from the server side
VlibTour administrator
VlibTour TourVisit Administrator
VlibTour Client
VlibTour TourVisitClient
VlibTour Server System Name
VlibTour Tour Management
VlibTour Emulation Visit
VlibTour Bike Station
VlibTour Group Communication System
VlibTour Lobby Room Server
Bikestation emulated server
A client for testing is implemented separately to perform HTTP queries on this server.
«EJB»
«EJB»
«REST»
«REST»
«REST»
«AMQP»
«AMQP»
«AMQP»

Figure 1: Architecture implemented in the POC

In this architecture, each component represents a functionality that uses the best fitting techno to meet the provided service needs.

3.2 Modules Description

VlibTour VisitEmulation

This component is based on REST architecture allowing clients to perform http queries related to their geolocalisation. The client issues its queries through a proxy which implements the methods of the API. It is used in the tourist application to get the next step in the tourist visit or the next POI's position while moving in its path.

The proxy is not implemented in the conventional manner of proxy pattern but the way we implemented it makes the code more readable.

REST architecture presents many advantages that enforces application scalability.

It's stateless, which means that the server does not store any state about the client session on the server side . As a consequence, if our application is in strong demand (increased number of queries), we can add other servers and load-balance their work by spreading requests.

VlibTour Bike Station

We have developed in VlibTour bike station emulation, an HTTP server that reacts in the same way as JCDeaux. Since the existing JCDeaux manipulates dynamically the data of bike stations located in Lyon, we have chosen to work with an emulated server that uses a static data (Paris.json) related to bike stations in Paris. We used JAX-RS specification . We have implemented a test client that interacts with this server but due to lack of time, we didn't integrate it in VlibTour Visit Tourist Application. It's implementation is similar to the one we have realized in VisitEmulation module.

VlibTour TourManagement

In this section, we instantiated a glassfish server which manages the beans sessions. The server bean, implementing the API interface, offers a variety of services applied on tours and POIs such as adding a new POI or organising a tour that could be consulted later by the clients.

We also used relational database to create and persist the entities as SQL tables via JavaPersistence API (JPA). We've opted for such a choice because

Vlib Tour LobbyRoom

This component manages the lobby room to which a tourist connects to create a group or to join an existing one. To perform the RPC calls with RabbitMO, we use AMQP architecture between the tourist visit application and the lobby room server.

The tourist application incorporates an RPC client that send their RPC calls through an exchange dedicated to accept new clients requests. Then, the tourist gets a link calculated from the sent parameters as an access to the group communication system which permits the sharing of geolocalisation.

We isolated the group communications in separate vhosts to ensure a more secure message exchange between same group members.

3.3 Design patterns

Proxy pattern

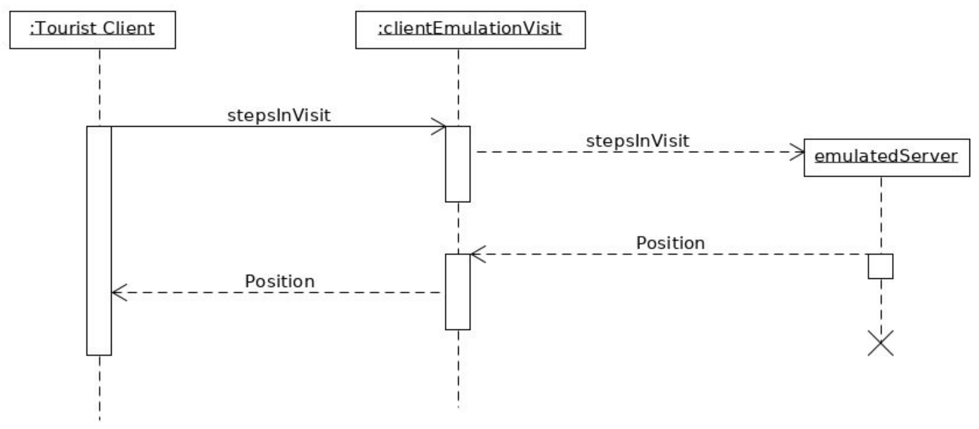


Figure 2: Sequence diagram of “stepsInVisit” use case

The Proxy is providing a barrier between the client and the server.

The main goal of this implementation is to expose a different interface in order to lighten the code.

Singleton pattern

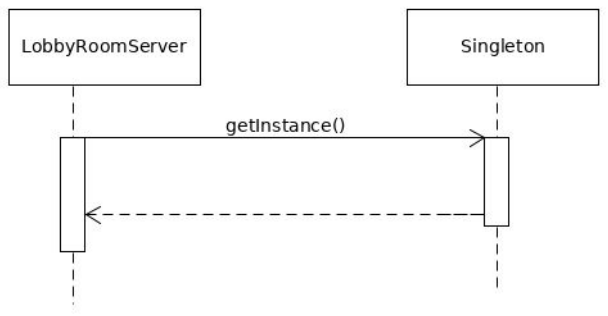


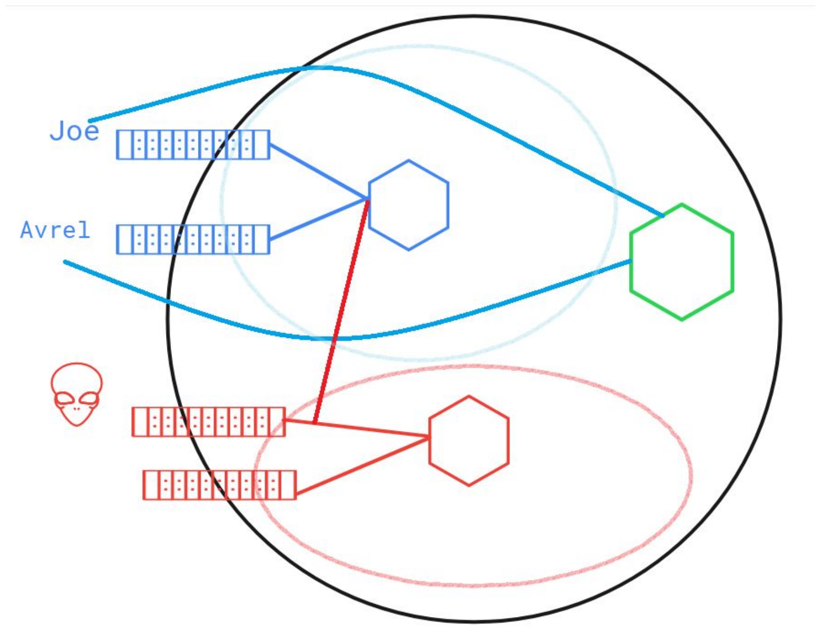
Figure3: Sequence diagram of Singleton

In our solution, we need to implement a unique LobbyRoomServer.Such Pattern Implementation may avoid Man in the Middle attack. Therefore, we just provide one point of access to create an instance of the Singleton class. The constructor is kept private, giving the getInstance() method the responsibility of providing access to the Singleton.

4. Extra-functional Requirements

1.Security

In the LobbyRoom module, we have designed our system in a way that treats each group apart in a virtual host. Such an implementation protects the system from any external attack. In the figure above, if any external hacker intercepts the URL of Joe or Avrel (normal users belonging to the same group), he will be able to receive messages exchanged between them.



2.Scalability

Sessions Beans used in the tour management are stateless. REST server is stateless as well. Such characteristic enforces the system scalability Any server can handle any request because there is no session related dependency.

5. Perspectives

The time constraint did not allow us to implement several other features, however, we would like to present some ideas we could do if we will continue to work on the project.

In **VlibTourVisitEmulation Module,** we imagine having multiple clients. Therefore, a recurrent and frequent polling (Example: stepsInVisit) will overload the server. We suggest slight changes such as implementing an availability replica of the server and ensure the load balance .

We suggest as well, adding a cache to reduce reading from the disc process and as a consequence, speed up operations and ensure availability.

Such a suggestion is possible because we are using REST architecture that enables

caching resources and spread requests across multiple servers. This thanks to the absence of states.

In group communication System, we would have preferred creating two proxies for which we delegate respectively the consuming and publishing task.