VLibTour Application

Eloi Besnard Joshua Randria

CSC5002 - Middleware

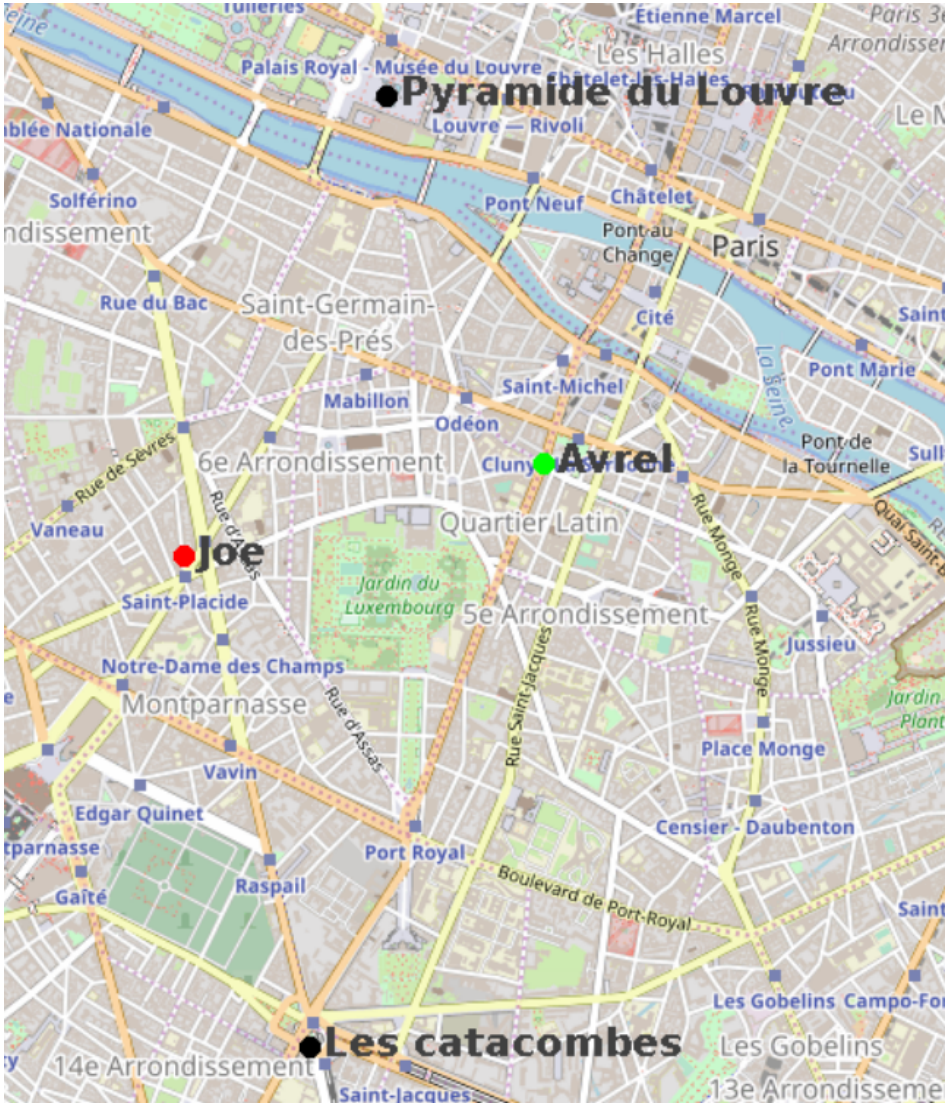
2022

Sommaire

Introduction

- JO Paris 2024
- POC Application VLibTour

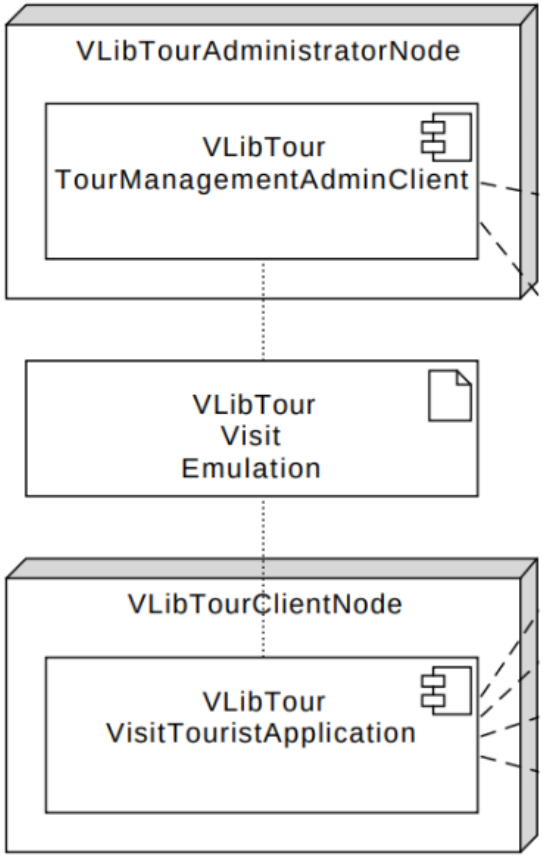
Introduction

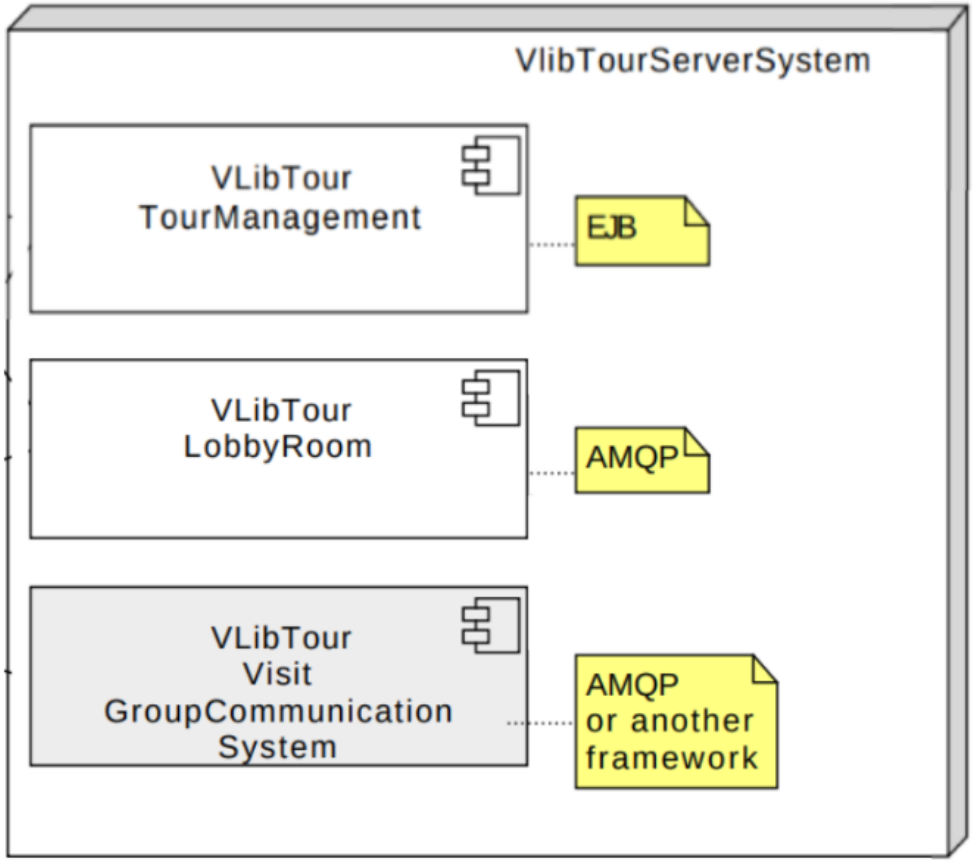


2. Architecture globale

- 2.1 Tour Management System
- 2.2 Emulation System
- 2.3 Communication System
- 2.4 Lobby Room
- 2.5 Intégration

2. Architecture globale



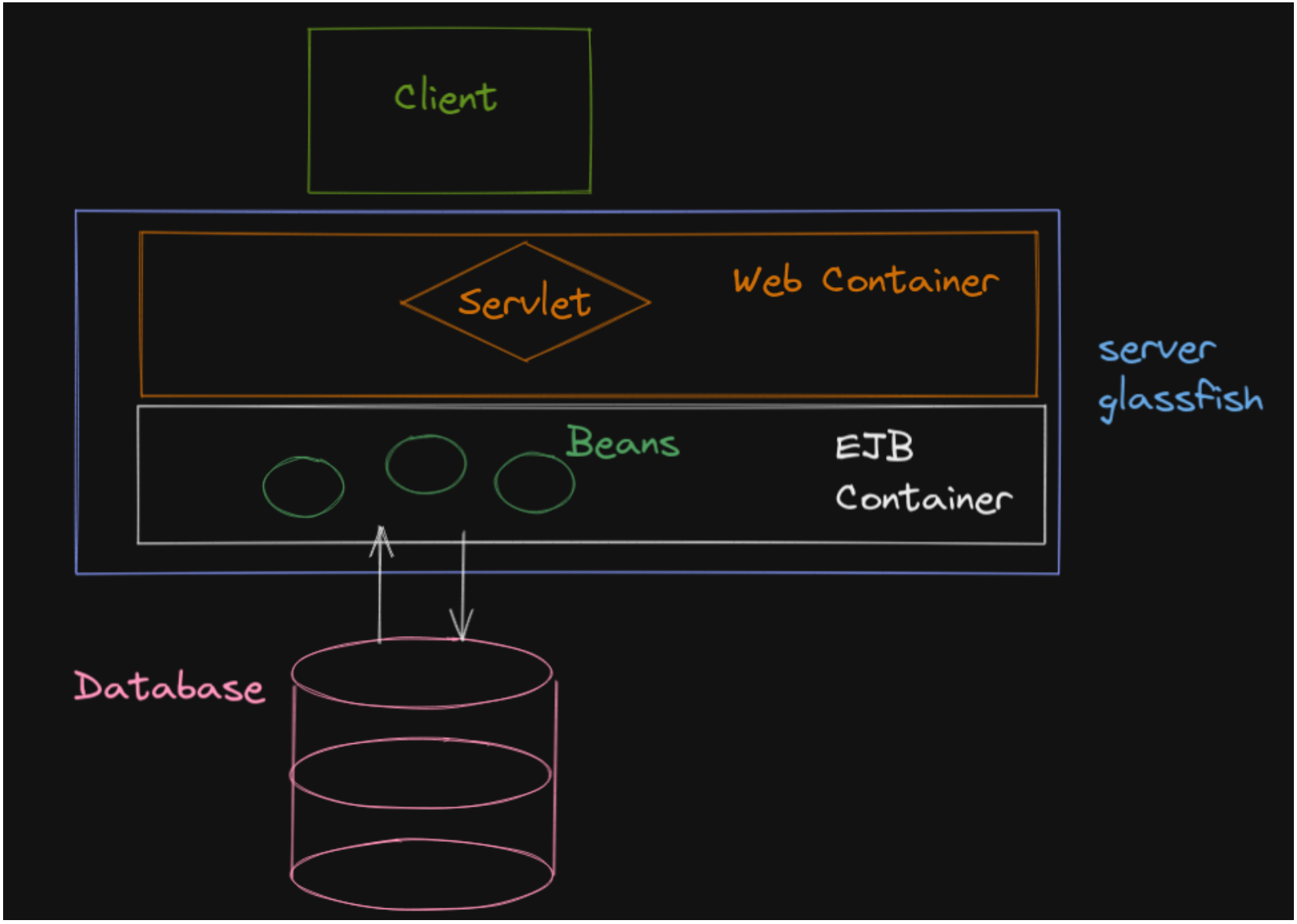


2.1 Tour Management System

Objectif: lier code orienté objet et tables BDD

1. Bean
   - avoir un constructeur vide
   - implementer Serializable
2. JPA (@*Entity*)
3. Clé primaire (@*id*)
4. persistence.xml
5. Entity Manager

2.1.1 Tour Management System



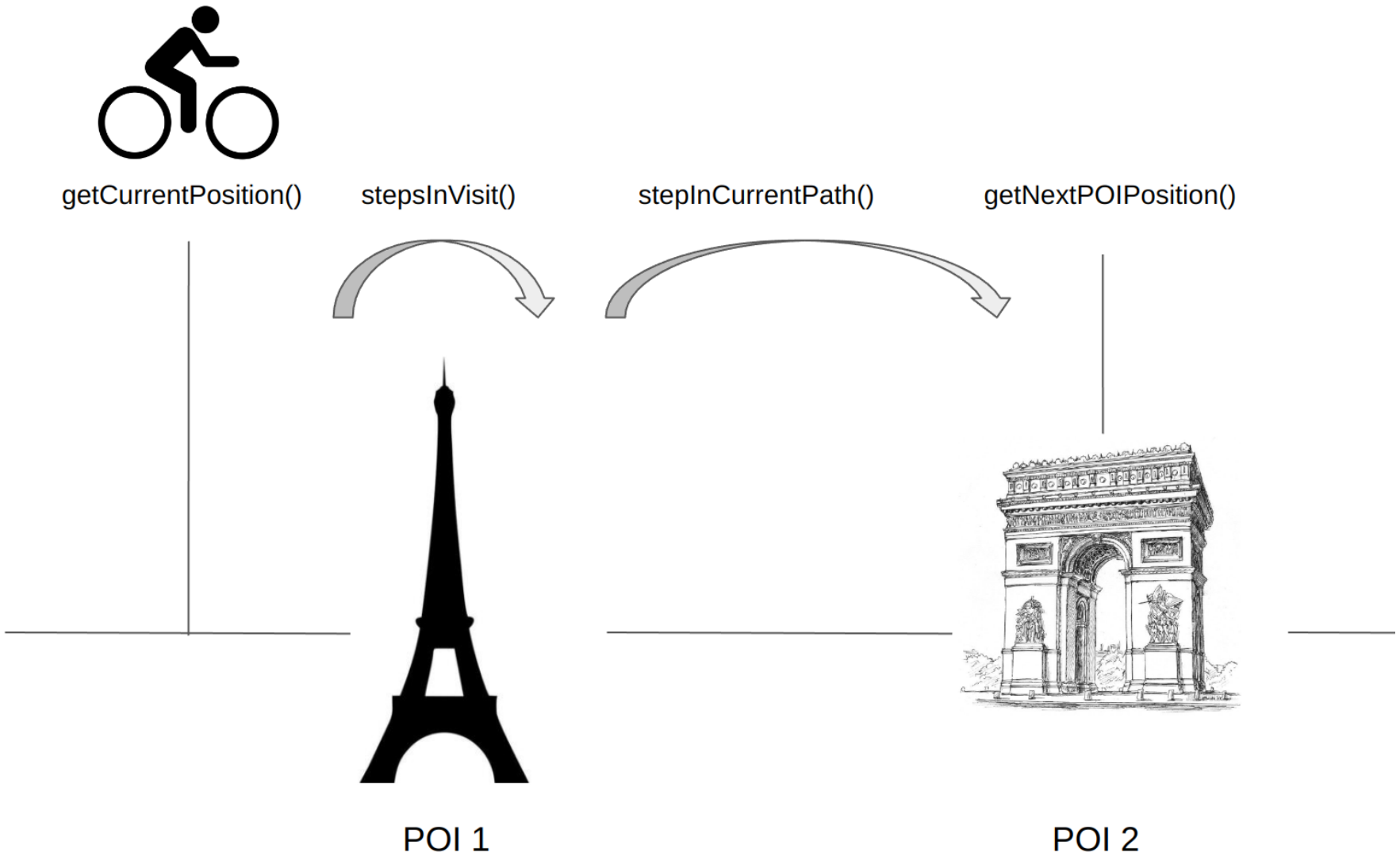
2.1.2 Tour Management System

Relation:

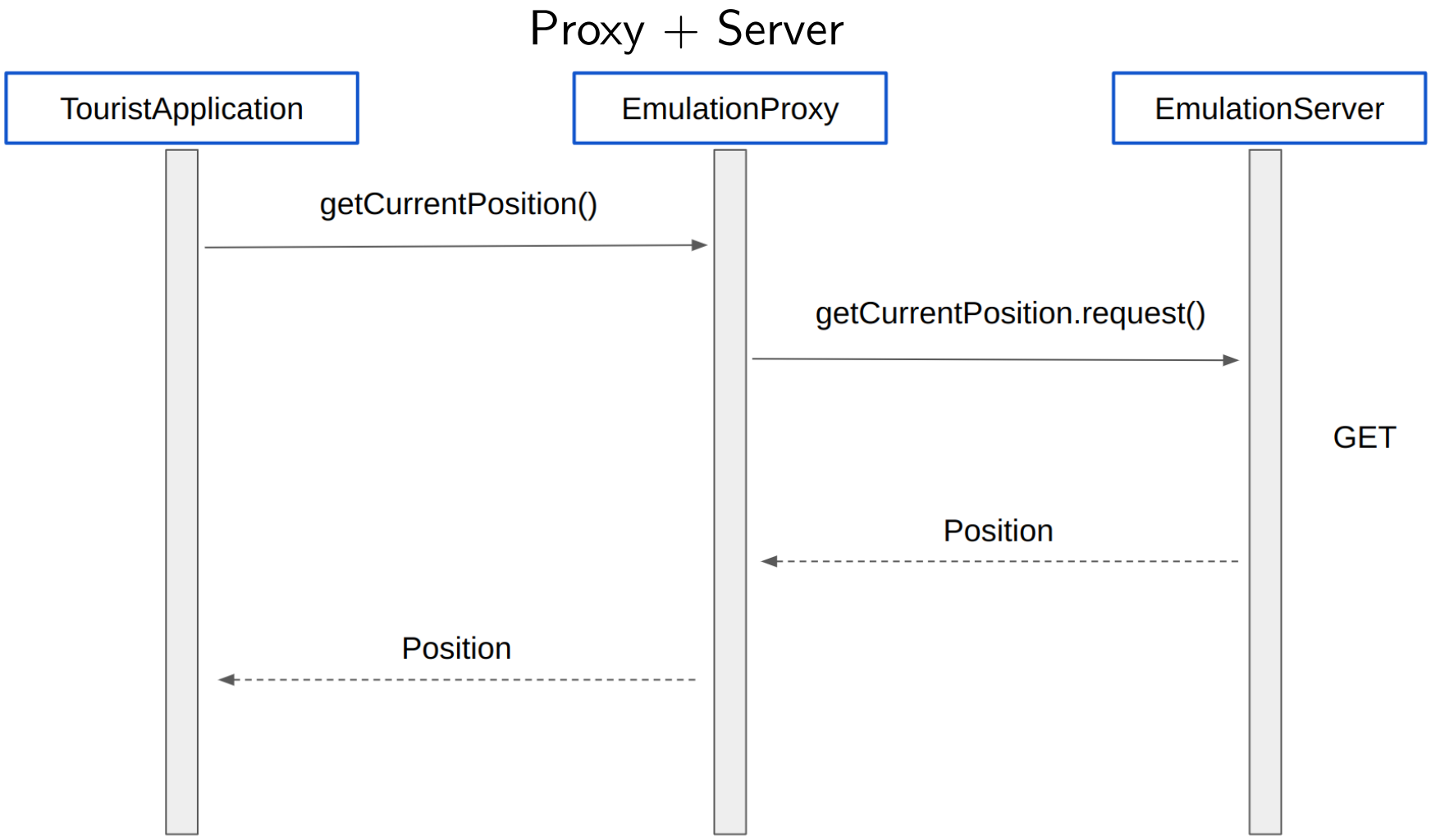
- @ManyToMany entre *POI* et *Tour*
- cascade

```
1  public class Tour implements Serializable
2  {
3      @ManyToMany(cascade = ALL, mappedBy = "tour")
4      public Collection<POI> getPOIs() {
5          return pois;
6      }
7
8  }
```

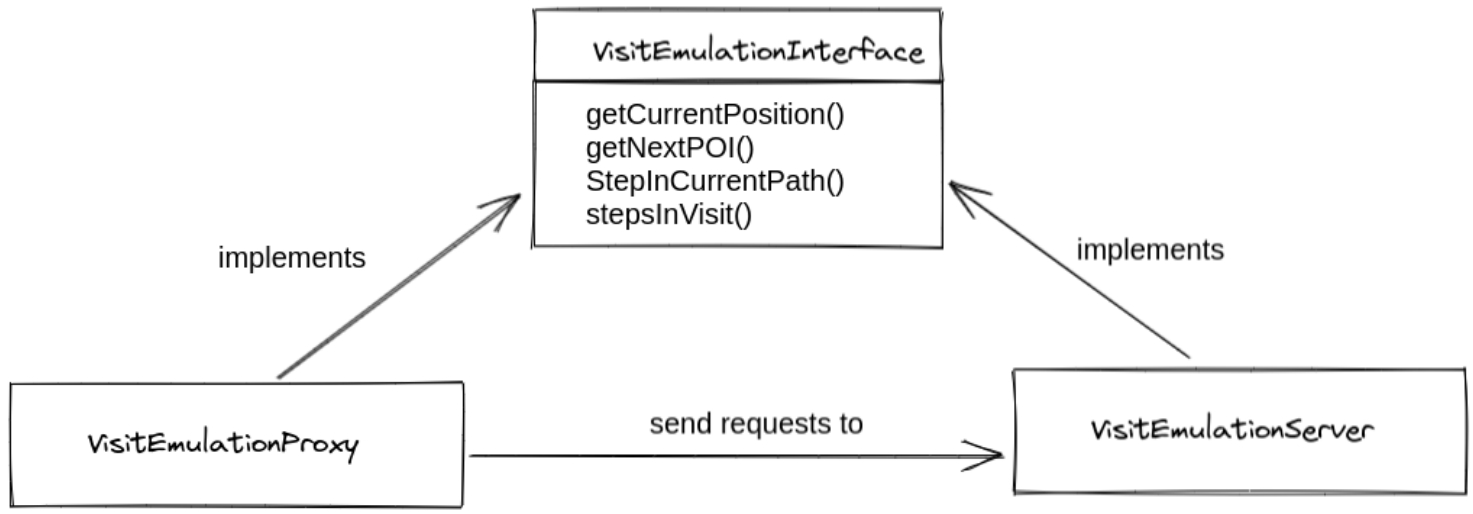
2.2 Emulation System



2.2 Emulation System



2.2 Emulation System



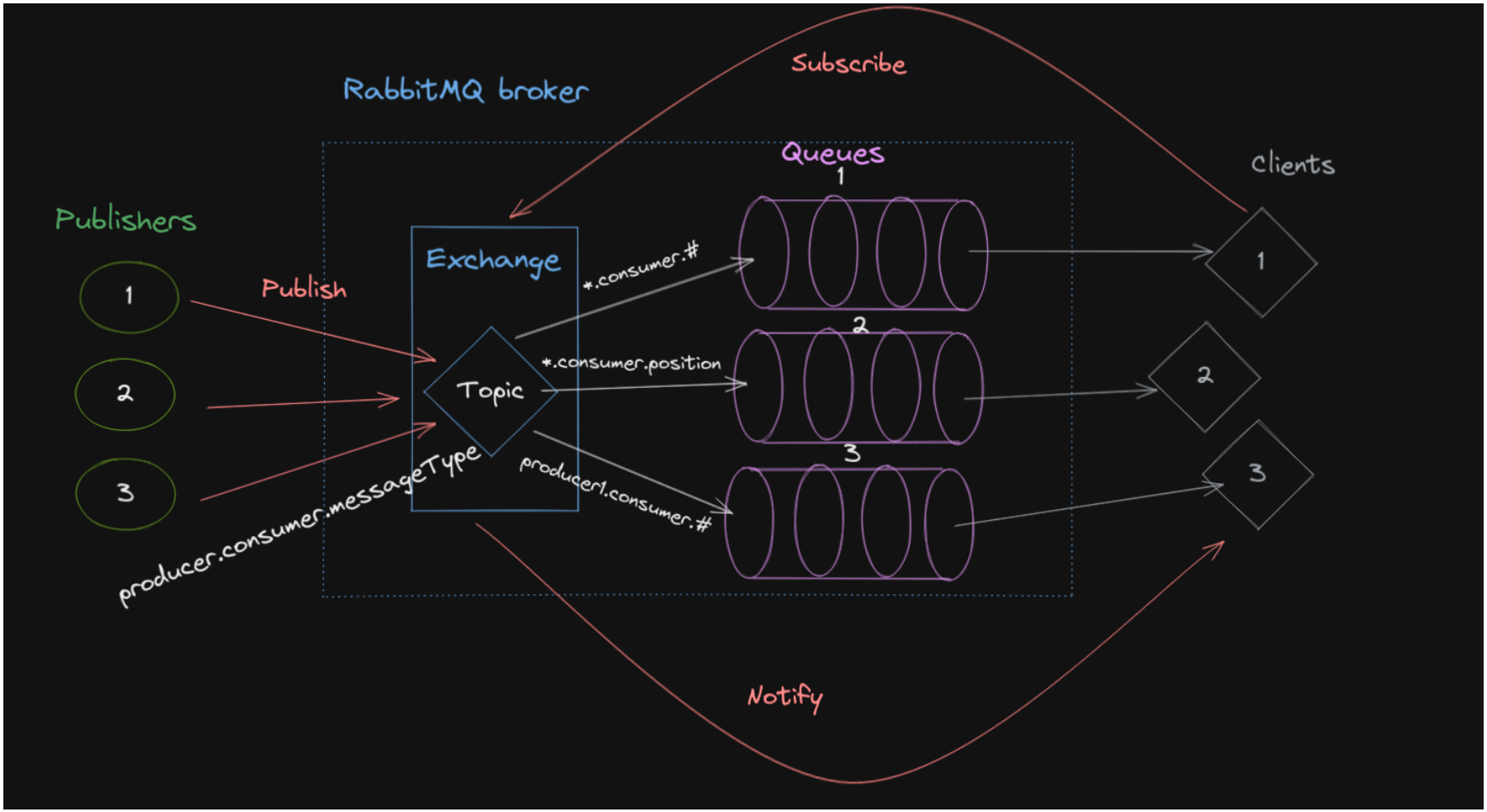
REST SERVICE

```
position = service.path("visitemulation/getCurrentPosition/" + user).request()
```

REST API

```
@GET
@Path("/getCurrentPosition/{user}")
public Position getCurrentPosition(@PathParam("user") ...)
```

2.3 Communication System



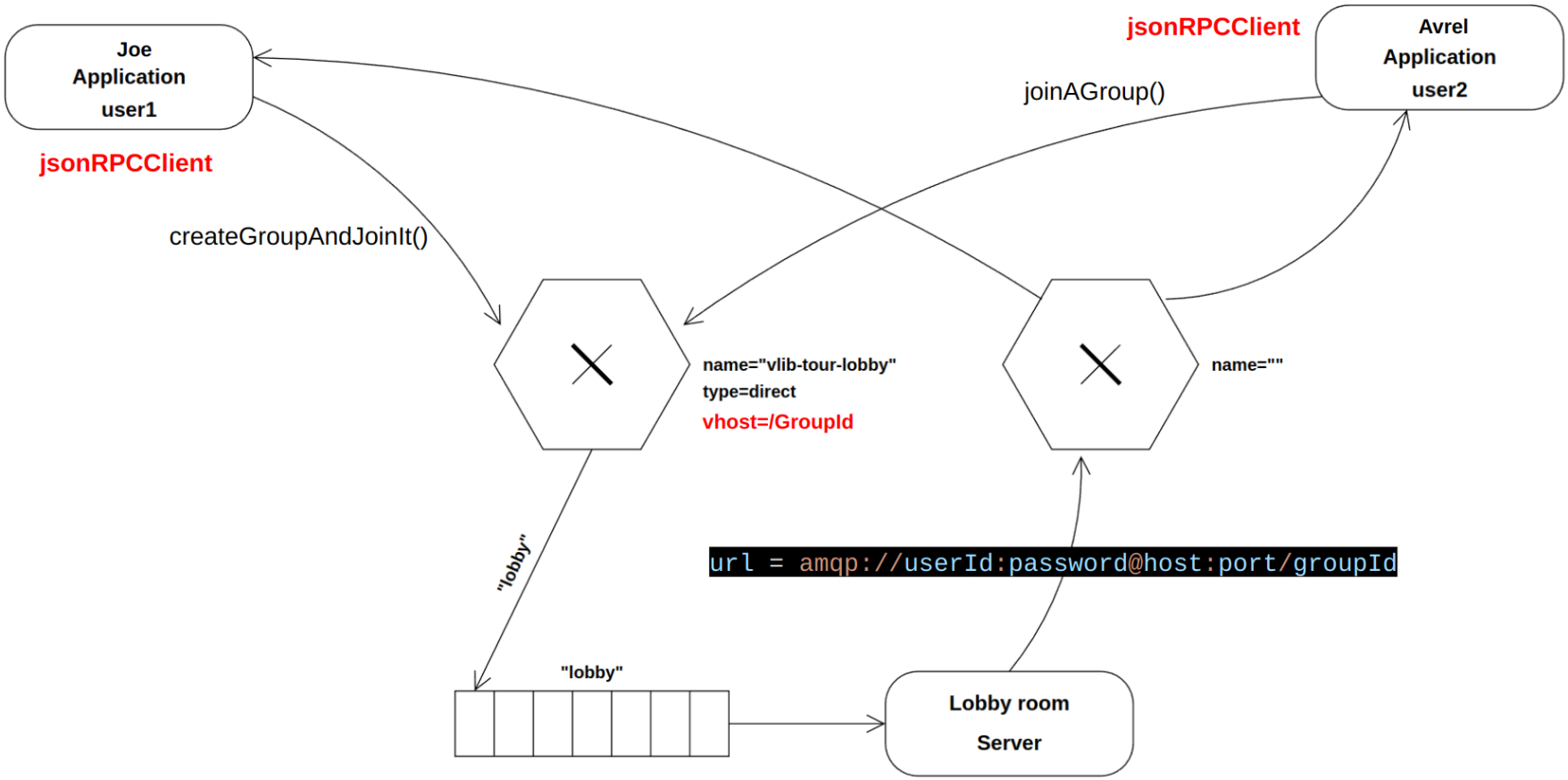
2.3.1 Communication System

RabbitMQ permet :

- Découplage consommateurs / producteurs
- Mise à l'échelle
- interopérabilité (AMQP 0.9.1)
- Performance

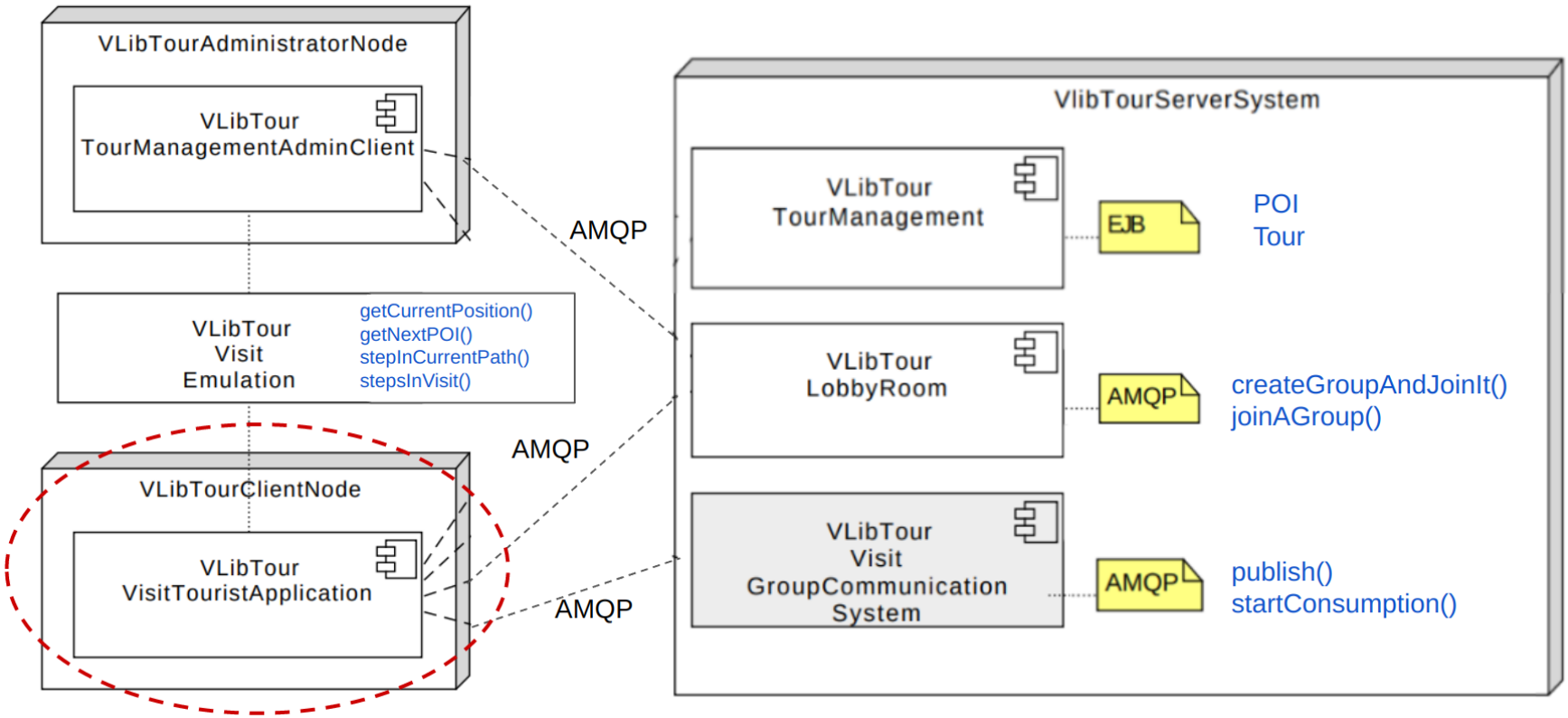
2.4 Lobby Room

Proxy + Server




```
ProcessBuilder("rabbitmqctl","add_vhost", groupId)
ProcessBuilder("rabbitmqctl","add_user", userId, password)
ProcessBuilder("rabbitmqctl","set_permissions", "-p", groupId, userId)
```

2.5 Integration



2.5 Integration

Constructor

```
public VLibTourVisitTouristApplication(tourId, groupId, userId){
       emulationVisitProxy = new VisitEmulationProxy();

// LobbyRoom Part
       lobbyRoomProxy = new VLibTourLobbyRoomProxy(groupId.get(), tourId, userId);
                                                  (tourId, userId);
       lobbyRoomProxy.createGroupAndJoinIt(groupId.get(), userId);
                     .joinAGroup(groupId.get(), userId);
```

2.5 Integration

Constructor

```
public VLibTourVisitTouristApplication(tourId, groupId, userId){
        emulationVisitProxy = new VisitEmulationProxy();
// LobbyRoom Part
// GroupCommProxy Part
        groupCommProxy = new VLibTourGroupCommunicationSystemProxy(url);

        consumer = new DefaultConsumer(groupCommProxy.getChannel()) {
            @Override
            public void handleDelivery(consumerTag, envelope, properties, body) {
                String message = new String(body, StandardCharsets.UTF_8);
                Position position = Position.GSON.fromJson(...);
                String sender = envelope.getRoutingKey().split("\\.")[0];
                if (sender.equals(ExampleOfAVisitWithTwoTourists.USER_ID_AVREL)) {
                    MapHelper.moveTouristOnMap(mapDotAvrel, position);
                } else {
                    MapHelper.moveTouristOnMap(mapDotJoe, position);
                }
            }
        };
```

2.5 Integration

Main Loop

```
while (!(positionOfClient.equals(nextPositionOfClient)))
            positionOfClient = nextPositionOfClient;
            client.emulationVisitProxy.stepsInVisit(userId);
            nextPositionOfClient = client.emulationVisitProxy.stepInCurrentPath(userId);

            client.map.get().repaint();

            if (userId.equals(ExampleOfAVisitWithTwoTourists.USER_ID_JOE)) {
                MapHelper.moveTouristOnMap(mapDotJoe, nextPositionOfClient);
            } else {
                MapHelper.moveTouristOnMap(mapDotAvrel, nextPositionOfClient);
            }

            // share Position
            String jsonPos = Position.GSON.toJson(positionOfClient, Position.class);
            client.groupCommProxy.publish(userId + ".all.#", jsonPos);

            client.map.get().repaint();
        }
```

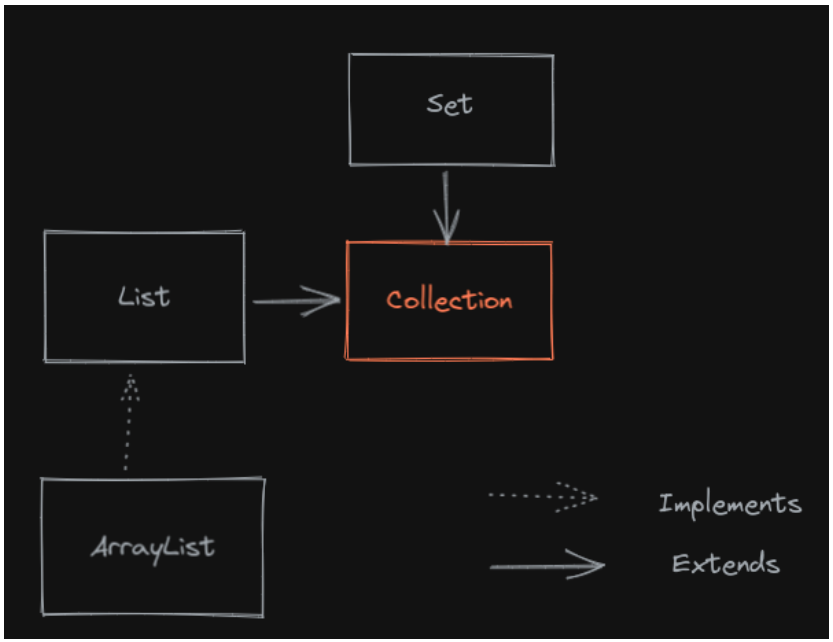
3. Exigences extrafonctionnelles

- 3.1 Interopérabilité
- 3.2 Mise à l'échelle

3.1 Interopérabilité

Java

- modèle de conception Top-Down
- Utilisation type généraux
  - Collection, Map à la place de List ou Array
- Attention aux valeurs Nulles
  - *Optional < groupId >*



3.1 Interopérabilité

Advanced Message Queuing Protocol

- Standard de communication
- "wire-level", format de données
- peu importe le langage
- *producer.consumer.messageType*

RabbitMQ

- OS ou langages différents

3.2 Mise à l'échelle

Beans

- appel à la BDD
- cache

GlassFish

- load balancer
- clustering

4. Démonstration

5. Conclusion et sources

- Nombreuses technologies
- Intégration délicate
- Meilleure compréhension des applications distribués & middleware

[1] glassfish
https://glassfish.org/docs/5.1.0/performance-tuning-guide.pdf
[2] oracle
https://docs.oracle.com/javase/tutorial/collections/interoperability/index