2019
2020

DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

A MaaS provider for a modern metropolitan area

Project for IE 2020

PART 3 of 4

1. Mobility as a Service (MaaS)

Urban mobility is changing rapidly because of environmental concerns, but also because of the impact digitalization has on the traditional ways people are able to use the transportation systems available.

Mobility as a Service is an analogy of the usual Software as a Service model popularized by the Cloud for the software industry. The idea is the same, people will use the transport network they see fitting better their needs, be it the Public Transport Operators like Metro or Buses, Taxis or the new players such as Uber or Cabify and all other innovative alternatives for personal transportation like rental bike, scooters, motorcycles, etc.

The innovation is the **seamless use of all of them** without the usual difficulties of different ticketing and payments systems, and the negative incentive for such use due to incompatibilities between cards, apps, tariffs, monthly subscriptions, etc.

The main idea of Maas is that one can take any transportation system and in the background his usage is being registered and one will pay for mobility according to the schema that best suits his needs

In the Northern Europe, there are several projects addressing similar concepts and there is a general interest for providing this advantageous system to the public, but there are many obstacles and barriers for such wide use of mobility to be implemented. Obviously, the main ones relate to the business models and the interplay of public and private companies, the percentage of government subsidies, etc. However, in spite of being easier to solve, there is a huge problem of integration between many players. Considering such a system for the large Lisbon Metropolitan area would involve far more than 20 Mobility Operators.

2. Project Objective

The basic challenge for the project is that each group define their architecture for a Maas System for a region they now well, (the majority will probably consider Lisbon, but any other area could be used as example).

The project consists in the development of the core of a MaaS operator with an architecture, scalable, open and providing seamless integration.

Context

- Users use tokens with temporal validity to have access to transportation. To do this, it will be necessary to create a blacklist mechanism to block users who are out of balance.
- Several operators accept the tokens to provide the transport service.
- Operators send events, to the MaaS operator e.g validations, taxi fare, bicycle usage.
- Events are consumed by different services that execute the core of the Maas Operator.
- The services will execute the main operational functions of the MaaS operator: User accounting, revenue consolidation, etc. (for the project, a subset of a real operator will be considered).

The services will be developed using a Microservice architecture for ensuring vertically and horizontally scalability.

Each service has its own storage and few dependencies on other services data

A performant and scalable integration system based on queuing is used for forwarding the events to MaaS

Considerer that a coordinator operator for managing users and operators already exist and should be integrated with this new core (an example in the Lisbon, area would be an operator like OTLIS responsible for the Lisboa Viva).

This coordinator executes a set of Business Processes for managing the Maas like: Client registration, services registration, revenue distribution. The main functionalities of the coordinator will be developed using a Business process methodology on top of a BPMS engine.

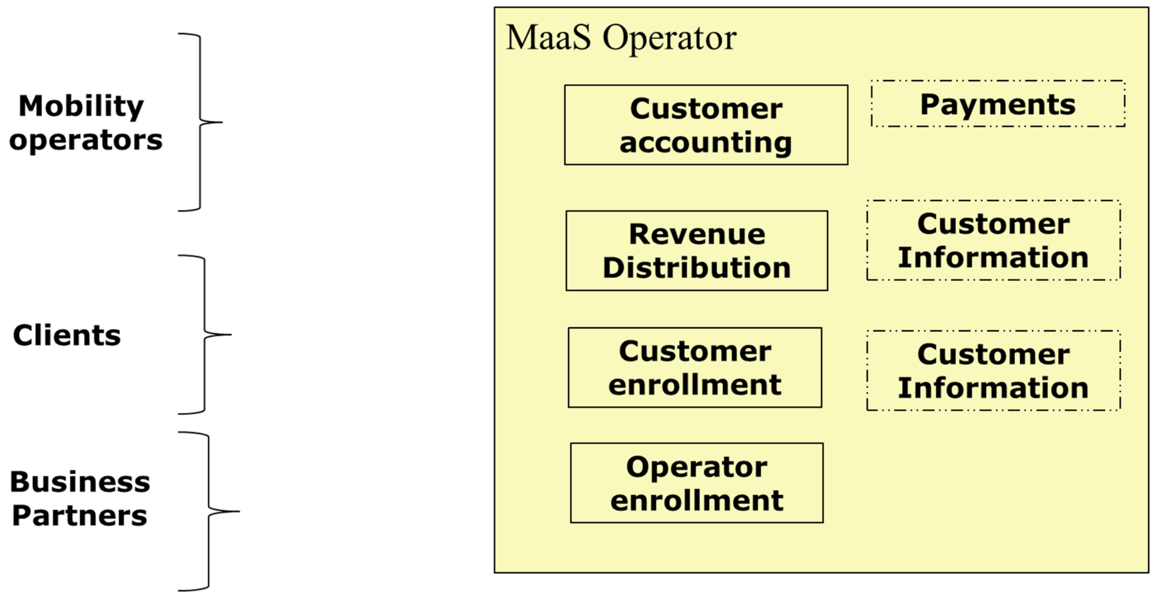

FIGURE 1 MAAS OPERATOR

1ST SPRINT

- Definition of the mobility operators and respective messages
- Definition of the event queueing integration: topics partition
- Definition of the fault tolerance requirements for Kafka
- Kafka installation, parametrization
- Test of the integration using applications for event generation

2ND SPRINT

In the previous sprint, the groups implemented the basic communication channels between transport operators and the MaaS operator

The objective of the second sprint is to define the architecture of the MaaS operator and to start the implementation

A strategic decision was taken that the architecture of the new operator should follow a microService approach. Thus, the group high-level objectives are:

1. **Definition of the microServices needed for the MaaS functionality**. This implies a careful analysis of the data stores and functions that each module will have.
   Deliverable: representation of the microServices and their integration with the Kafka broker. A BPMN representation is encouraged (e.g., using Camunda Modeller). Moreover, BPMN clarifies the MaaS business processes involved.
2. From objective 1, the groups must *choose* **two** microService to implement. These microServices should be representative of the MaaS main functions particularly the customer accounting and usage of transports, and the relation of the usage with the cost defined by the operators and described in their respective catalogue of services.
   Deliverable: textual description stating the rationale of the decisions.
3. Functional definition of the two microServices **Interfaces**, these microServices process events from Topics and are invoked by APi's for other synchronous services and should stick to the basic tenant of encapsulating all functionality and data to provide an independent set of services.
   Deliverable: description (XML and JSON) of the input and output expected for each microservice.
4. **Functional integration of the two microServices with the previous Kafka topics** (ingestion of events and eventually production of new events)
   Deliverable: configuration of the new topics in the Kafka cluster from sprint 1.
5. **For each microService**, **definition of their functionality and the respective data stores schemas.**
   Deliverable: description of the behavior for each microservice using pseudo-code.
6. **Implementation of the two microServices.**
   Deliverable: source code and configurations from both environments: AWS λ and JAVA javax.jws.WebService library should be used – one for each microservice. Moreover, the AWS RDS should be used for database, and thus all the configurations and SQL scripts should be included.
7. **Functional testing** using the applications that generate the events that should exercise the functions within the microService and provided evidence of their accurate executions in the persistent state of the microServices
   Deliverable: textual description, and analytics, of the obtained test results.

Note: the other interfaces of the microservices can be implemented now but their test and integration will be the objective of the next sprint.

3ND SPRINT

The sprint has two objectives:

- To document and deploy the API´s to the Maas Operator
- To design, implement and integrate the executable business processes that support the MaaS business.

The technological integration between API consumers and microservices should be implemented using the Kong platform.

The technological architecture for the BPM execution is based on the Camunda engine as the entry point for the provisioning of all the stakeholders, e.g., customers, providers. Then

All the executable BPMN tasks reuse the microservices developed during the 2nd sprint. In addition, if during the 3rd sprint development, you need more microservices functionalities you are free to develop them and hosted them in EC2 (using jws library) or AWS Lambda platform.

Thus, the group high-level objectives are:

1. Define all interfaces of the microservices and document then using Swagger. This should be a revision of the definitions done in 2nd sprint- objective 3.

Deliverable: Open API specifications files

2. Configuration of the Kong hq endpoints to serve each microservice

Deliverable: command file with all the configurations required

3. Modelling of all the MaaS business processes using Camunda modeller

Deliverable: BPMN models

4. Development of one executable business processes regarding the process provisioning of customers or providers, using Camunda modeller and deployed in Camunda engine

Deliverable: source code deployed in Camunda engine

5. Development of one executable business processes regarding the dunning process[1] of customers, using Camunda modeller and deployed in Camunda engine

Deliverable: source code deployed in Camunda engine

6. Development of the required databases, microservices, kafka topics, etc.

Deliverable: command file with all the configurations/developments required

7. Functional testing using the applications that generate the events that should exercise the functions within the microservice and provided evidence of their accurate executions in the microservices frameworks, Kong and Camunda engine.

date: 15 May 2020, 23h59

[1] https://en.wikipedia.org/wiki/Dunning_(process)