## Enterprise Integration Story

An enterprise is a huge distributed system composed of agents, that are either human or machines, working together to fulfil the business purposes. When we are analysing an enterprise two aspects need to be differentiated: the business aspect related with the modelling of business purposes, *i.e.*, the clear understanding of the enterprise' goals and its representation in an abstract language to be understood by all the stakeholders; and the implementation aspect encompassing the software applications, the integration between applications, and all the other technology related details, *e.g.*, networks, servers, development cycles, databases, *etc.* Both aspects are intertwined, it is not possible to steer an enterprise without considering the traceability between them. There is no modelling without implementation, and no implementation without modelling.

For instance, reflect upon the following questions. What is the application stack that support the execution of a product ordering business process? How are the customer facing applications of one enterprise integrated with the back-office applications of other enterprise? Why, at a certain point in time, is a software application bottlenecked at some performance level as perceived by one enterprise? Deep knowledge in the engineering of the enterprise systems is needed to formalize informed answers to these crucial questions. To better exemplify the software application stack of a modern enterprise, and the challenges that integration is facing let us consider a living example. At the same time, we can identify the core competencies delivered by this Enterprise Integration course.

Mobility as a Service (MaaS) domain is an analogy with the usual Software as a Service model popularized by the cloud for the software industry. The idea is similar, customers use a transport network according with their immediate needs, public transport operators like metro or bus, private transport operators like taxi or uber, or other light personal transportation like rental bikes, scooters, motorcycles, *etc.* Moreover, MaaS offers a seamless use of all forms of mobility without the usual difficulties of different ticketing, apps, tariffs, subscriptions, or payments systems. A single subscription gives access to all offers and the payment is done from only one previously provisioned account.

Every time a passenger starts any form of transportation, an event is generated (*e.g.*, on a Kafka cluster) to characterize the properties of the customer, station, date, time, location, passenger mood…*etc.* Moreover, during, and in the end of the transportation, multiple events are continuously generated to keep the trace of what is happening. Those events are asynchronous messages, produced at scale, that grounds the behaviour of contemporary event-driven systems. They are based on the principle of decoupling between data producing and data consuming. No centralized approach is followed, the event processing occurs concomitantly, anywhere, and relies on physical or virtual (*e.g.*, provided by AWS EC2) availability of consumers. The Kafka consumers process streams of events and store the result on others Kafka messaging systems, in cloud databases (*e.g.*, provided by AWS RDS) or simple cloud storage (*e.g.*, provided by AWS S3).

Due to uncertainty about customer's behaviour, to financial constraints or to socio-economic aspects, we do not know ahead the amount of computational power that will be required. It is not a matter of buying dozens of computers, connect them in a network, and that's it. Our infrastructure is ready. That is incorrect. Contemporary solutions offer elasticity to scale accordingly with demand. Serverless functions as provided by AWS Lambda can perform accordingly. The core idea is to allocate the exact amount of power, when needed, and only if needed, by request. An enterprise only pays the computational power that is really required instead of allocating a fixed amount. This is especially relevant when microservices framework are used (*e.g.*, provided by Quarkus). A microservice is an autonomous unit of business, that could be realized by a well-determined business function, a business data entity, *etc.*, and that provides an Application Programming Interface (API) to other enterprise' components.

In MaaS, a REST API provided by Quarkus offers core business functionalities as a transport provider creation, a customer account creation, *etc.* The landscape of REST APIs inside an enterprise could be a comprehensive collection and hard to document. In that situation, swagger and the openAPI specification, are the catalogues and executor facilitators. Moreover, API gateways (e.g., Kong or AWS API Gateway) are solutions to fine grain control the execution of the available APIs. API gateways can trace the pairs request/response, collect analytics, filter requests, keep logging, enforce security mechanisms, and extend their capabilities using plug-in extensions. Today, API gateways are located as mediation components between the top and bottom architectural layers. In the past, the API gateways were made available inside the Enterprise Service Bus (ESB) platforms.

On the top of this, MaaS' customers, with the help of MaaS' personnel, need to manage their subscription services, *e.g.,* activate, deactivate, change, pay, consult, *etc.* If a customer activates his/her account then a set of personal data need to be filled in the MaaS mobile app, the payment details need to be validated, and finally, a welcome message needs to be sent back to the customer. This is an example of a business process, the concept allowing the higher-level control of the business. A business process model is a representation of the desired behaviour (*e.g.*, using BPMN 2.0 specification language) for all the involved actors intending the production of a new service or product, while a business process instance is a factual execution of a business process model (*e.g.*, supported by Camunda Engine).

Finally, The HyperAutomation capabilities are the facilitator for managing all this complexity. Some are old fashioned solutions, while others are now emerging in the industry, namely, virtualized environments (docker and dockerhub), infrastructure as code (terraform), scripting capabilities (bash), source code build tools (maven), configurations change management (git and github), testing (JUnit), among others.

Welcome to Enterprise Integration course, the challenge released to you is to understand all these concepts, and then to integrate all this technology using a hands-on approach!