Service Orchestration - Smart City

January 15, 2019

Problem representation

Service orchestration is the coordination and arrangement of multiple services exposed as a single aggregate service. Developers utilize service orchestration to support the automation of business procceses by loosely coupling services across different applications and enterprises and creating composite applications. In other words, service orchestration is the combination of service interactions to create higher-level business services. This works through the exchange of messages in the domain layer of enterprise applications. Since individual services are not programmed to cummunicate with other services, message must be exchanged according to a predetermined business logs and execution order so that the composite service or application can run as it is demanded by the end-user.

By 2050, 66% of the world's population is expected to live in urban areas. The challenge will be to supply these populations with basic resources like safe food, clean water and sufficient energy, while also ensuring overall economic, social and environmental sustainability. More than ever before, many different organizations will need to collaborate to help make cities smarter; technology integration is a special challenge that requires broad cooperation in a systems approach.

Given a scenario that shows an urban environment with a growing demand for efficiency and resources, public administrations have to consider an evolution in the management models of cities. To do this, the use of information and communication technologies (ICT) is essential. This has been translated as the Smart City concept which, with its services, is moving forward to what

has now become known as the Internet of things and itself the Internet of the future.

We define a Smart City as a city which uses information and communication technologies so that its critical infrastructure as well as its components and public services provided are more interactive, efficient and so that citizens can be made more aware of them.

State of the art

Managing increasingly complex enterprise computing environments is one of the hardest challenges that most organizations have to face in the era of cloud computing, big data and IoT. Advanced automation and orchestration systems are the most valuable solutions helping IT staff to handle large-scale cloud data centers.

Having functionality loosely coupled is every efficient team's goal, and that was pretty much obtained with the widespread use of containers given their ease of management.

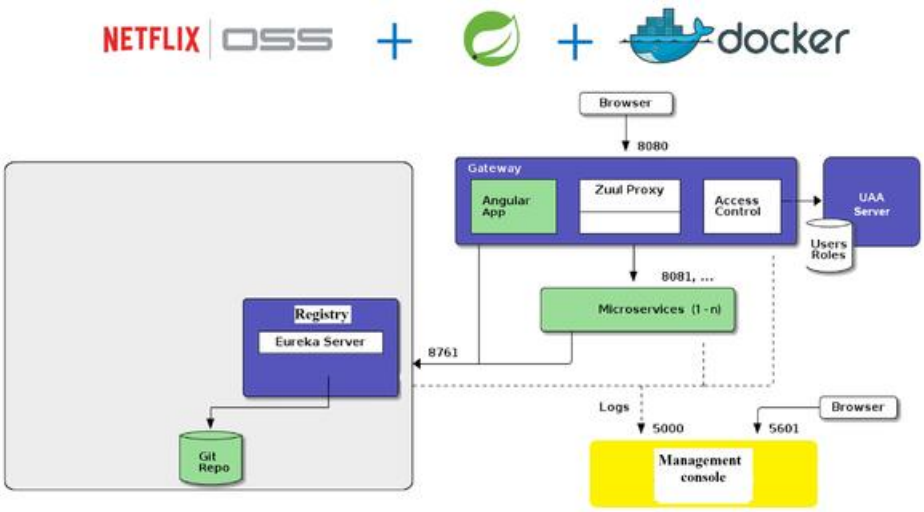
With Self Registration, service instances that are registered within a registry are discovered on startup and unregistered on shutdown. A service instance is responsible for registering itself with the service registry, which results in a service instance that is aware of it's own state, is coupled to the registry and, most importantly, will unregister itself if unavailable.

In a Future Internet context, service orchestrations will deal with very important numbers of services. Therefore, the orchestration languages and engines need the cope with the high distribution requirement, the multi-partnership, and the scalability of services. [**?**]

Solution

Application or service orchestration is the process of integrating two or more applications and/or services together to automate a process, or synchronize data in real-time. Our solution ensures:

- A high-performance and robust Java stack on the server side with Spring Boot
- Front-end created with Angular framework
- A microservice architecture with a Registry page, Netflix OSS and Docker



The architecture of our application is described in the previous diagram. Basically, our application is divided into three major components. The first component is represented by the services offered by the application. The application comes with a standard set of 5 services. Each service has its own database to be as atomic and as reusable as possible. Each microservice uses the Springboot framework and exposes a REST API to communicate with the outside. Each microservice has a simple and well-defined functionality. The second component is the Registry component, where services are automatically logged as long as they have a particular configuration and know the physical

address of the Registry server (using the Netflix, Netflix Eureka solution). The third component is the API Gateway, which organizes the services, making them work as a whole. At this level, users authenticate and authorize using JWT.

Results

Comparison with other solutions

Open api is the most relevant idea regarding the central them of the project. It represents a platform that agregates multiple code bases and orchestrates them behind the scenes thus it appeas to the client as one central service. The platform exposes to the client a couple of business orientes APIs such as: metrics a bout local business in romania (such as VTA, earned money in a year) exchange rate for different curriencies (relevant in Romania) validation of ID numbers, bank account numbers For an user to use the platform, he needs to register to the platforms database and generate an unique number (called token). Then based on the unique token generated by a customer, he can query the platform for different information.

For example: curl header x-api-key:[cheia-ta]

-s https://api.openapi.ro/api/companies/13548146 The above query will generate an output containing information about the company with the registration number 13548146. The way the project works on its backend is that it has a number of microservices that gaters information regarding one single field (like exchange rate) from specialised sources (like a banks website). The services ar connected inside by a central services that dispatches the requests from various clients.

The services offered by OpenApi are more oriented towards the economic side of a city (the ability to see companies looking for, currency exchange). This makes OpenApi not so popular among regular users. In addition, OpenApi comes with a usage tariff that could discourage ordinary users from using it.

Unlike OpenAPI, Smart City is open source and totally free. Our application is oriented on another vertical, and offers very different and different services such as: weather check, traffic check, iasi events, locations in Iasi.

The simplicity of services offered makes SmartCity the ideal recipe for an application: it is oriented both on the entertainment side and on the business side. On the entertainment side, ordinary users can use the services they offer as they are of interest. For example, tourists visiting the city of Iasi can use the services offered by the application to plan their trip to the city of the 7 hills. The simplicity of services offered makes SmartCity the ideal recipe for an application: it is oriented both on the entertainment side

and on the business side. On the entertainment side, ordinary users can use the services they offer as they are of interest. For example, tourists visiting the city of Iasi can use the services offered by the application to plan their trip to the city of the 7 hills. However, the application does not currently have a graphical interface in which this information is displayed, but we hope that there will still be some geeks that will use the application.

On the business side, our application can serve a very important start in a project. This already comes with the services offered, and above them can be easily added new services defined by the programmers. All the registration of microservices is done, as well as their orchestration. This makes it very plausible for a start-up to win some advance.

Romania Open Source Data (https://github.com/romania/) is a national initiative that aims to ease the work of developers who are implementing applications to help Romanian citizens. It provides data on postal codes in Romania, a list of romanian localities, and a list of bus stations.
A great disadvantage of this initiative is that it only provides us with some sql files that can be inserted into a database. It does not offer a modern, adequate solution. Modifications such as adding a new bus station need to be made manually by those who maintain the open-source project. However, it is a useful tool in building applications.

Future work

Our architercute has been created in the way that we could easily add new microservices to our service discovery server. Since the client registers automatically to the gateway, we just need to add the dependency of eureka and add the bootstrap.yml file to connect to the server. Also, we will have to create the client app (mobile or web) to consume all the services

Conclusions

Smart City is an app designed for helping people from Iasi and tourists that want to explore the city. It will help us fight the hard trafic from the city, find different information about the weather from the city, find interesting events and check for cultural places (ideal for a tourist that visits the city for the first time in his life).

References

[1] Linagora: Orchestration State of the Art
`https://research.linagora.com/display/aggregation/Orchestration+State+of+the+Art`

[2] Service Orchestration and SOA
`https://www.mulesoft.com/resources/esb/service-orchestration-and-soa`