Invillia Backend Challenge

**Here is what was implemented:**

I’ve decided to split the application in four parts:

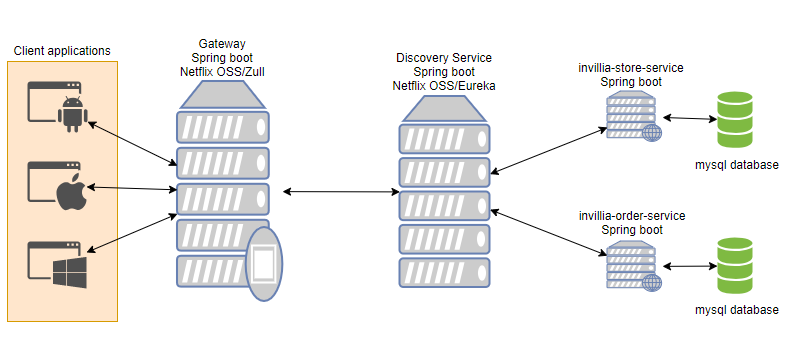
1. Invillia-service-discovery: this application aims to manage the list of microservice’s endpoints. It uses spring boot application and the eureka service;
2. Invillia-gateway-service: this application provides an entry point to those who want to consume the microservices endpoints. The clients don’t have a direct access to the microservices. They even don’t know where they are;
3. Invillia-store-service: this is the store application. It has endpoints to manage stores and its address.
4. Invillia-order-service: this is the order application. It has endpoints to manage orders and order items entities.

The endpoints operations save their data into a mysql relational database. There are two databases:

* Invillia\_store\_db: it directly used by invillia-store-service.
* Invillia\_order\_db: it directly used by invillia-order-service.

To save time, it wasn’t generated the database scripts. It is necessary to create the database schemas. To the database objects (tables), this isn’t necessary because they are created by JPA/Hibernate mechanism through “spring.jpa.hibernate.ddl-auto” directive.

Whenever an application needs to get data from another database, a call is made to the responsible service. Never through the call in a database directly.



**What wasn’t implemented:**

* Docker: It could be used in order to become the distribution more eas inside the environments beyond the fact to have similar environments (development, tests and production).
* AWS: It could be used AWS instances to provide the microservices. This approach makes easy an another way to scalable the application Using this mecanism it is possible to identify and scale only one portion of the application who needs more resources instead of scale the entire applicaton. This avoid wasting money and resources.
* Security: It could be used JWT (JSON Web Token) in the spring boot application.