LOGIT ONE™

Logit One Architecture

Modelling with Microservices

Version: 7.02.2020

Modelling with Microservices

Introduction

Microservices as a concept existed for very long time. The idea of loose coupling between separate modules of a software system and to be able to encapsulate specific core responsibility to a module also existed for a very long time. Purely from the solution design point of view, this idea is very relevant for microservices architecture. It is essential to have modularization as concept incorporated into the DNA of the design activity. It cannot be inserted at the implementation phase.

Conceptual solution building for complex problems is not easy. It involves breaking down the complexity into smaller parts so that each part can be more easily overseen - with minimal interference between them. This process is quite challenging as each individual part is quite abstract to start with and only after certain iterations of design the concrete definitions start to emerge. It needs persistent effort to make sure that the individual parts together are able to completely solve the original problem.

The traceability to the original requirement is therefore important. In order to have solid traceability, it is key to have a complete problem statement to start with. This needs to be validated with all the stake holders involved to make sure that it is complete and consistent. For this purpose we employ user story scenarios.

Breaking down complexity

How do we come up with the individual microservices? In our view there is no exact science involved here. Knowledge about the current and future product, systematic organization and creativity go hand in hand to be able to break down higher complexity into multiple smaller complexity items. There are certain criteria that can be employed to make sure that the individual services are at the right level of granularity:

- Each service has a well-defined purpose.
- Each service should hold a central piece of information that it deals with.
- Typically, if a BPMN (Business Process Model and Notation) is made for each of the services, then it should result in “Seven, Plus or Minus Two” activities to make sure that the service becomes neither too complex nor too simple.
- Collaboration becomes a problem if there are too many services while handling complexity becomes a problem if there are too few services.
- Each different integration point can become a separate service.

We brought up BPMN in the previous paragraph and that leads us to documenting the services. One very effective way is to build a high level BPMN identifying each of the micro-service as a high-level activity. Care should be taken to define the collaboration contract between each of the micro-services. This contract of communication between the individual services should use a well-defined conceptual data model.

Documenting a design

UML is quite handy documenting the conceptual design. A popular misconception is that the UML is employed as a design technique but that is not true. UML is merely used in documentation. A good UML document does not necessarily imply a good design, but representing the design as UML artifacts is important: Having a visual representation enhances one’s ability to improvise the design, collaborate on it, and also hold a record of it for further maintenance.

The UML artifacts together will form the model of your application, and we use it for our model driven development. The model would eventually become code that is executed live in production systems. This model based approach gives us benefits in terms of development efficiency, quality and maintainability.

Conclusion

It is a natural human tendency to under-estimate the complexity of the system to be built. The model driven development helps us with this situation by simulating the user scenarios even before the system is built so that the gaps are found before the code is written. Even if the problem looks less complex to start with, it is always advisable to apply model driven development as most of the complexity comes to surface only once we start to analyse and build the model.