CASHEW – Community as Adaptable, Scalable, High-Available and Extensible Website

Degree programme: MAS Information Technology

Notoriety is rarely an attribute that can be easily estimated in a near future when we talk about technologies or virtual communities. When a burst of popularity appears, the structure of the overall system – the software architecture – often presents a new class of problems.

Goal

The goal of this thesis is to set the fundament for a modern community website. It lists software quality requirements for common modern website, such as high-availability, scalability, security and maintainability. Then it considers a number of common architectural styles upon which the system might be based. By selecting a single style that best suits the requirements, it details its software components and relations among them. Finally, it surveys some of the outstanding software quality problems, and considers a few of the promising future extensions.

Result

Architecture style

Three architecture styles are compared: monolith pattern, microservices, serverless pattern.
To choose the best suited pattern for this project, six points of comparison have been taken to conclude to a comprehensible outcome: agility, ease of deployment, testability, performance, scalability, ease of development.
It demonstrates there is not just one single perfect architectural solution. Each of styles has its drawbacks and advantages. However, the architecture must be chosen in consideration with the project and its non-functional requirements.
Modularity, scalability and reliability have tilted the balance in favour of microservices.

Software architecture document

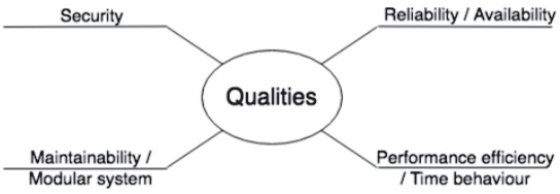
To provide a coherent description of the system's architecture views, the «4+1 architectural view model» has been used, which puts special significance of «Use case view» in the centre of the other four views. In addition to the 4+1 model, a data view as well as a security view are listed in this thesis.

Decomposition by business capability

The business logic of the system consists of backend services – called hereafter microservices. They enhance the business capability and each service has a REST API and its own private data-store. Those microservices work with a group of services, which share certain common management capability characteristics. A third division, organized around operational capabilities, has the purpose to monitor and facilitate the automation of the whole system. To reduce the dependence that one service has on another, the approach of loose coupling is used to interconnect the components in the system, so that the services depend on each other the least extent practicable. The medium used to exchange messages among services is a group of messaging channels.

Quality tree and scenarios

Each quality goal is evaluated through an evaluation scenario. The tree is composed of four qualities:



Quality tree

A special effort has been made to test the performance of the system. Three tests strategies with each two phases have been elaborated, sending constant or increasing linear load to the system under test.

Conclusion

For most of the cases, underlying technologies, such as programming language used to implement the system, are mostly irrelevant. However, in order to meet the desired system qualities, the software architecture plays a crucial role. More importantly, one can only evaluate only what can be measured; hence it is essential to set at the beginning of the project measurable non-functional requirement.

Philippe Wanner
philippe.wanner@tomylab.com