February 01, 2024

Research Proposal

Master's Thesis in Technology (DTEK1002)

Florian Dejonckheere
fwdejo@utu.fi

Abstract

Working title: *A Framework for Writing Distributed Applications using the Modular Monolith Software Architecture*

The advent of cloud computing has fundamentally changed the way software is deployed and managed. Nowadays, developers define the software architecture in function of the underlying infrastructure, and often decompose the application into a set of smaller applications based on logical boundaries in the domain logic, which are deployed independently of each other.

This separation of concerns brings along a number of advantages such as scalability, fault isolation and independent deployment, over more traditional software development practices such as monolithic codebases. The adoption of a microservice-based architecture also introduces a number of challenges for the developers. Developers need to be aware of the distributed nature of the application, and need to take into account the additional complexity that comes with it. This affects in turn the development velocity and efficiency. The resulting software also increases in complexity, complicating maintenance and debugging.

In this thesis, we investigate the modular monolith software architecture, a hybrid model between monolithic and microservice-based architectures that aims to combine the best of both worlds. We analyze the pattern and its advantages and disadvantages, and compare it to its architectural predecessors. We identify the challenges and opportunities that arise when adopting the modular monolith architecture for dynamic languages compared to statically-typed languages. Finally, the architecture is implemented as a proof-of-concept, and evaluated in the context of a real-world use case. We discuss the results and draw conclusions about the effectiveness of the architecture.

Aims and goals

The thesis can be divided into two parts: a literary part and a more practical part. First, I analyze the origins and characteristics of the modular monolith software architecture, and compare it to similar software architectures. Second, I investigate the applicability of the architecture specifically for dynamic languages, based on existing literature for statically-typed languages. Finally, I design and implement a proof-of-concept for the architecture, and evaluate it in the context of a real-world use case.

The use case I intend to use for the proof-of-concept is called NephroFlow Link, part of the NephroFlowProductSuite.

NephroFlow Link is a middleware application written in Ruby that acts as a data broker between connected dialysis machines and the NephroFlow cloud platform. It is currently designed as a monolithic application, and is deployed as a single process on a single server. However, the application is becoming increasingly complex, and the requirements in regard to scalability are increasing rapidly, which leads me to believe that the application would benefit from a modular monolithic architecture (written as a monolith, deployed as a microservice).

Research question

Research Question 1: What defines the modular monolith architecture, and what sets it apart from monolithic and microservices architectures?

Motivation: Explore why the modular monolith architecture is becoming increasingly popular, and why it is considered a viable alternative to monolithic and microservices architectures.

Research Question 2: Which challenges and opportunities arise when considering adoption of the modular monolith architecture for dynamic languages compared to statically-typed languages?

Motivation: Investigate how modular monoliths could be used in the context of dynamic languages and which implications this has on the development process and resulting product.

Research Question 3: How effective is the modular monolith architecture in dynamic languages, and what is its impact on overall system performance and scalability?

Motivation: Evaluate the effectiveness of the modular monolith architecture for a real product.

Methodology

In order to evaluate the effectiveness of the modular monolith architecture, I will design and implement a proof-of-concept runtime for Ruby. The runtime will be used to implement a version of NephroFlow Link structured as a modular monolith. Based on existing performance tests, I will compare this version with the current version of NephroFlow Link, and evaluate the results. Non-functional requirements such as fault tolerance and flexibility will also be evaluated.

Contents

1. **Abstract**
2. **Introduction**: general introduction to the topic, and identification of challenges and obstacles related to the topic. Explanation about why and for whom the topic is relevant, and what the expected outcome of the research is. Scope of the research thesis.
3. **Background**: historical background of the topic, and theoretical background needed to understand the thesis.
4. **Related work**: relevant work and research related to the topic.
5. **Methodology**: explanation of the research methodology used in the thesis.
6. **Analysis**: analysis of the problem, and the proposed solution to the research questions. Details design decisions and implementation details.
7. **Case study**: application of the proposed solution to a real-world use case.
    1. **Background**: background information about the use case.
    2. **Evaluation**: evaluation of the proposed solution in the context of the use case.
    3. **Results**: results of the evaluation.
    4. **Discussion**: analysis and discussion of the results.
8. **Conclusion**: summary of the thesis, and answers to the research questions.
9. **Future work**: possible future work related to the topic.
10. **Bibliography**

Bibliography