Architecture research document

for

HeardIT

Author: Mihail Vasilev

Date: 10-04-2024

Contents

[1. Introduction 3](#_Toc151037828)

[2. Main question 3](#_Toc151037829)

[3. Sub-questions 3](#_Toc151037830)

[4. Conclusion 5](#_Toc151037831)

1. Introduction

The purpose of this document is to outline the research process taken during the architecture design process of the HeardIT application. The research is split into the following sections *– Main research question*, where the main question is defined, *Sub-questions*, where the sub-question derived from the main questions are defined and answered and *Conclusion*, where the answers to the sub-questions are combined in order to answer the main question. The final section is the *References* section where the sources of the information used during the research are presented.

1. Main question

In this section the main question will be established. In order to complete the research, the main question needs to have a concrete answer. For this reason, it is important to define the main question well. The main question will also allow us to create the sub-questions that will help us answer the main question.

The main research question is:

*What is the most suitable architecture type for the HeardIT web-application?*

Answering this question will allow us to determine the best architecture design for the HeardIT application. In the next section I will establish the sub-questions that were derived from the main question and I will formulate an answer to each of them.

1. Sub-questions
2. What type of architecture designs are suitable for the HeardIT application?

Choosing the right architecture for a music streaming web-application can be a difficult and complex process. There are many different pre-established architecture design patterns that can be used as the basis of the application. For this reason, I am going to look over my requirements and depending on them make a decision.

Let’s establish first what are the main and most widely used design patterns. According to Wayner (2021) and GfG (2021) the main software architecture designs that are most widely used for developing web-applications such as HeardIT are the following: *Layered architecture, Event-driven architecture, Microkernel architecture, Microservices architecture* and *Space-based architecture*. Some of these might suit my requirements for implementing HeardIT but first I am going to explain each of them and dive into more detail.

Here is a summary of the main capabilities of each of these architecture design patterns:

1. Layered architecture – software is organized into distinct layers – presentation, business logic, persistence, each layer represents a specific responsibility and establishes separation of concern and modular design. Layers can communicate only with their adjacent layers and follow a well structured and easy to understand, test and implement system.

As an example, a banking application that has - presentation layers for customer interfaces, business logic layers for transaction processing and data access layers for interacting with databases.

1. Event-driven architecture – decouples components and relies on events to trigger actions. This enables asynchronous communication and scalability. Events are used only by the components that need them which allows them to remain flexible. This pattern is suitable for systems with unpredictable workloads, however significant complexities can arise in the managing of the event flows and this can lead to loss of reliability.

As an example, a smart home automation system might utilize event-driven architecture to respond to sensor data, triggering actions like adjusting temperature or turning on lights.

1. Microkernel architecture – the architecture is split into a core set of essential functions and optional modules. The core provides the basic services, while modules add specific functionalities and customizations to the system. Tis pattern can be easy to maintain and upgrade but may introduce complex issues related to the communication between the modules.

As an example, the Eclipse IDE follows a microkernel architecture where core functionalities are kept minimal and additional features are added via plugins.

1. Microservices architecture – the application is structured as a collection of loosely coupled, independently deployable services, each responsible for a specific capability. Services communicate through APIs. This pattern promotes flexibility, scalability and continues delivery. It also promotes collaborative environment where multiple teams can be working on different services within the same bigger application. It fosters rapid development and fault isolation and reliability. Careful management of service boundaries is required and operation complexity can be also very high.

As an example, Netflix, Spotify and many other similar mainstream applications use microservices to handle user account management, content delivery, and recommendation algorithms.

1. Space-based architecture - different components are decoupled through the use of a shared communication space, where the data is published and distributed to the correct component. Components interact indirectly through the shared space promoting loose coupling and flexibility. Distribution systems with high concurrency and big difference between the workloads benefit the most from this kind of architecture.

As an example, financial trading systems utilize space-based architecture to distribute and process incoming market data across multiple nodes, allowing for high throughput and fault tolerance in handling trading decisions.

Now that the main architecture design patterns are established, I wanted to narrow down my choice so that I could create a more detailed overview and have better insight into which of these would be most suitable for my application.

After careful consideration, I narrowed my options to the following two design patterns: *Layered architecture* and *Microservices* architecture designs. I am going to compare them and make a decision based on my requirements. I have chosen to go more in-depth into these two because for music streaming application such as HeardIT I see these two architecture design patterns as the most suitable ones.

In the following sections I will explore more specific contexts for each architecture design pattern.

Methods used:

Literature study – plan

actual

1. Which architecture design meets the scalability and stability requirements?

Methods used: Brainstorm, Prototyping, Expert interview, System test

1. Which architecture design meets the maintainability requirements?

Methods used: Literature study, Document analysis, Expert interview

1. Which architecture design meets the requirements for cloud native applications?

Methods used: Brainstorm, Prototyping, Problem analysis, Expert interview, System test

1. Which architecture design meets the security requirements?

Methods used: Data analytics, Problem analysis, Expert interview

1. Conclusion

To conclude this research, I am going to design the HeardIT application architecture to follow the \_\_\_\_\_\_\_\_\_.

have answered all of the sub-questions above. Now that we have done this, we can answer the main question. In order to analyse the Bamboo build logs with custom regular expressions we will create a script using PowerShell scripting language, using the new regular expressions and the newly developed functions we will extract the log files from the Vanderlande servers, analyse the log files and record them in a CSV file that can be exported and further analysed in Excel.

With this we can conclude the research that was conducted during the development of the Test-Broker Engine Failure analysing script product.

**References:**

* Wayner, P. (2021, June 19). How to choose the right software architecture: The top 5 patterns. TechBeacon. <https://techbeacon.com/app-dev-testing/top-5-software-architecture-patterns-how-make-right-choic>
* GfG. (2021, October 27). Types of software architecture patterns. GeeksforGeeks. <https://www.geeksforgeeks.org/types-of-software-architecture-patterns/>