RESEARCH DOCUMENT

[Document subtitle]

OCTOBER 8, 2024

TONY JIANG
RB04

Contents

Introduction

Research questions

Main question:

How can I design a scalable web-based music guessing game using Golang within a microservice architecture?

Sub-questions:

1. **How should microservices be structured in Golang for optimal scalability and maintainability as the user demand increase?**
    - **Strategy: Library**
    - **Methods: Best good and bad practice – Design pattern research**

2. **How should the security of the music guessing game be validated?**
    - **Strategy: Library – Lab**
    - **Methods: Literature study - Security test**

3. **How can CI/CD pipelines be set up for deploying a microservice-based game using Golang?**
    - **Strategy: Library – Workshop**
    - **Methods: Literature study - Community research – Prototyping**

4. **What are the best practices for deploying Golang microservices using serverless cloud functions?**
    - **Strategy: Library – Workshop**
    - **Methods: Best good and bad practice – Prototyping**

5. **What tools can be used to monitor the scalability of microservices?**
    - **Strategy: Library – Lab**
    - **Methods: Literature study – Community research – Non-functional test**

6. **How should data be managed across distributed microservices to ensure consistency, reliability, and scalability?**
    - **Strategy: Library**
    - **Methods: Literature study - Best good and bad practice**

Sub-questions

1. How should microservices be structured in Golang for optimal scalability and maintainability as the user demand increase?

To start, it's important to understand what microservices are, whether Golang can be structured in a microservice architecture, and how we can scale and maintain these microservices effectively.

What are microservices?

Microservices are small, independent services within an application that each handle specific functions, collectively forming the entire application. Each service can be built, deployed, and scaled independently, making the architecture flexible and easier to manage. This approach is commonly used in growing applications and enterprise environments.

Golang can be designed in a microservice architecture. There are tools available that support Golang in creating and managing microservices, making the architecture easier to maintain and scale.

https://aws.amazon.com/microservices/

https://www.redhat.com/en/topics/microservices/what-are-microservices

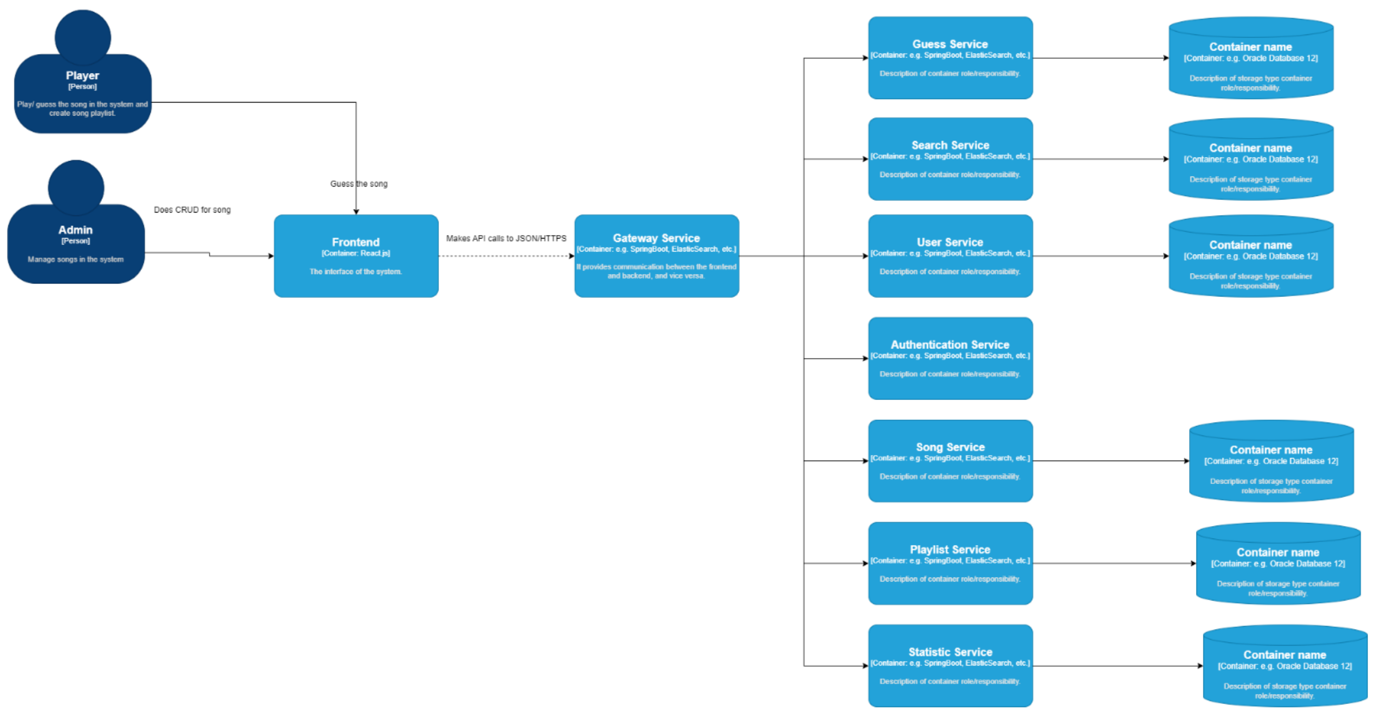
https://www.bacancytechnology.com/blog/golang-microservices-architecture

How to divide your application into microservices?

To do this, you first need to define the functionalities of your application and group these functions by their similarities. Then, assign a service name to each group of functions. A microservice can also consist of just one function. For a scalable application, microservices should be decoupled so that the services don't depend on each other. However, in some cases, they may be tightly coupled due to specific requirements, which can lead to configuration challenges. Additionally, each microservice should have its own database to avoid depending on other services for data retrieval, resulting in lower response times.

Result

As a result of what I understand, here is the first draft of the microservices architecture design I created for a web-based music guessing game.


Player
[Person]
Play/ guess the song in the system and create song playlist.
Admin
[Person]
Manage songs in the system
Guess the song
Does CRUD for song
Frontend
[Container: React.js]
The interface of the system.
Makes API calls to JSON/HTTPS
Gateway Service
[Container: e.g. SpringBoot, ElasticSearch, etc.]
It provides communication between the frontend and backend, and vice versa.
Guess Service
[Container: e.g. SpringBoot, ElasticSearch, etc.]
Description of container role/responsibility.
Search Service
[Container: e.g. SpringBoot, ElasticSearch, etc.]
Description of container role/responsibility.
User Service
[Container: e.g. SpringBoot, ElasticSearch, etc.]
Description of container role/responsibility.
Authentication Service
[Container: e.g. SpringBoot, ElasticSearch, etc.]
Description of container role/responsibility.
Song Service
[Container: e.g. SpringBoot, ElasticSearch, etc.]
Description of container role/responsibility.
Playlist Service
[Container: e.g. SpringBoot, ElasticSearch, etc.]
Description of container role/responsibility.
Statistic Service
[Container: e.g. SpringBoot, ElasticSearch, etc.]
Description of container role/responsibility.
Container name
[Container: e.g. Oracle Database 12]
Description of storage type container role/responsibility.

https://martinfowler.com/articles/break-monolith-into-microservices.html

https://learn.microsoft.com/en-us/azure/architecture/microservices/model/domain-analysis

How would the microservice communicate with each other?

There are various way of how to microservice communicating with each other. It depends on the systems requirement like performance, consistency or scalability.

The communication options are:

- Synchronous Communication (Request-Response)
- Asynchronous Communication (Event-Driven or Message-Based)
- Hybrid Communication

2. How should the security of the music guessing game be validated?

3. How can CI/CD pipelines be set up for deploying a microservice-based game using Golang?

4. What are the best practices for deploying Golang microservices using serverless cloud functions?

5. What tools can be used to monitor the scalability of microservices?

6. How should data be managed across distributed microservices to ensure consistency, reliability, and scalability?

Firstly, I need to identify what type of database my system needs, so for that I use the CAP theorem to determine what type of database my system needs.