4/4/2024

Technical Design

Individual Project: FestivalConnect

Name: Lucas Jacobs
Class: S-A-RB06
PCN: 490692
Student number: 4607368
Technical teachers: Felipe Ebert, Bartosz Paszkowski
Semester coach: Gerard Elbers

Table of Contents

Introduction

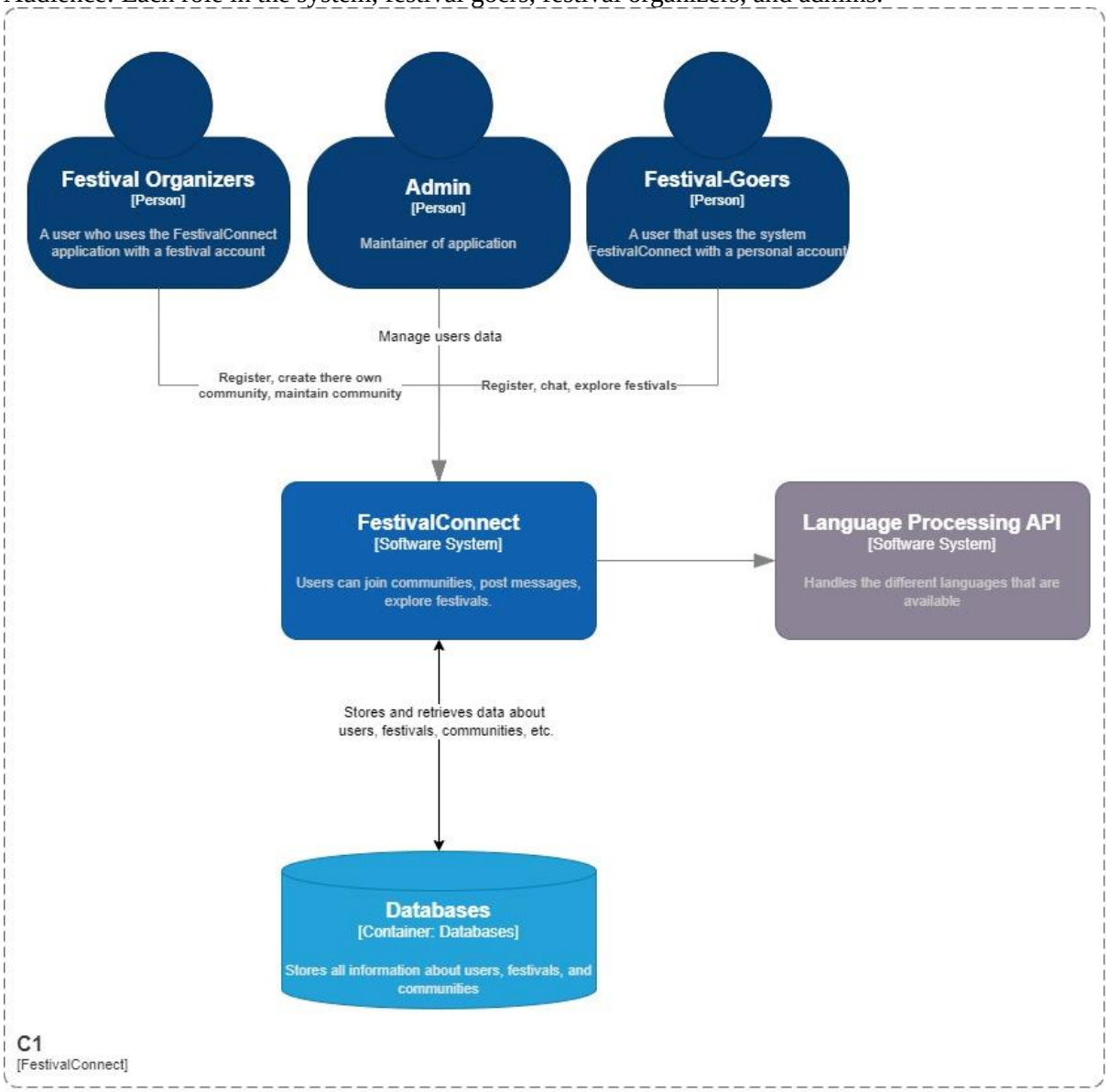
This document will show the technical details and decisions with the designs for FestivalConnect. It will be validated and say why FestivalConnect made certain choices with a showcase of how it is being designed.

C4 Architecture Diagram

The C4 Architecture Diagram will give a visual representation of the system for every user who is part of this project. There are different levels of deepening inside of this diagram, which gives each diagram a unique and different view of the system, with the relationships that are involved.

System Context (C1):

Stretch: FestivalConnect, Users, Dependencies
Audience: Each role in the system, festival goers, festival organizers, and admins.



System Context Diagram for FestivalConnect
The system context diagram for FestivalConnect

Figure 1: C1 Context Diagram

Description

Festival Organizers: organizations who have their own festival(s), they can create communities.
Festival Goer: The persons who are interested in the festival(s), join communities.
Admin: Maintains the application.
FestivalConnect: The whole application with all the technology to run the system.
Databases: The application that stores the needed information of the system.
Language Processing API: External service that manages multiple languages.

Containers (C2): Overall shape of architecture

Stretch: Containers and the relationships between them.
Audience: Technical Users.

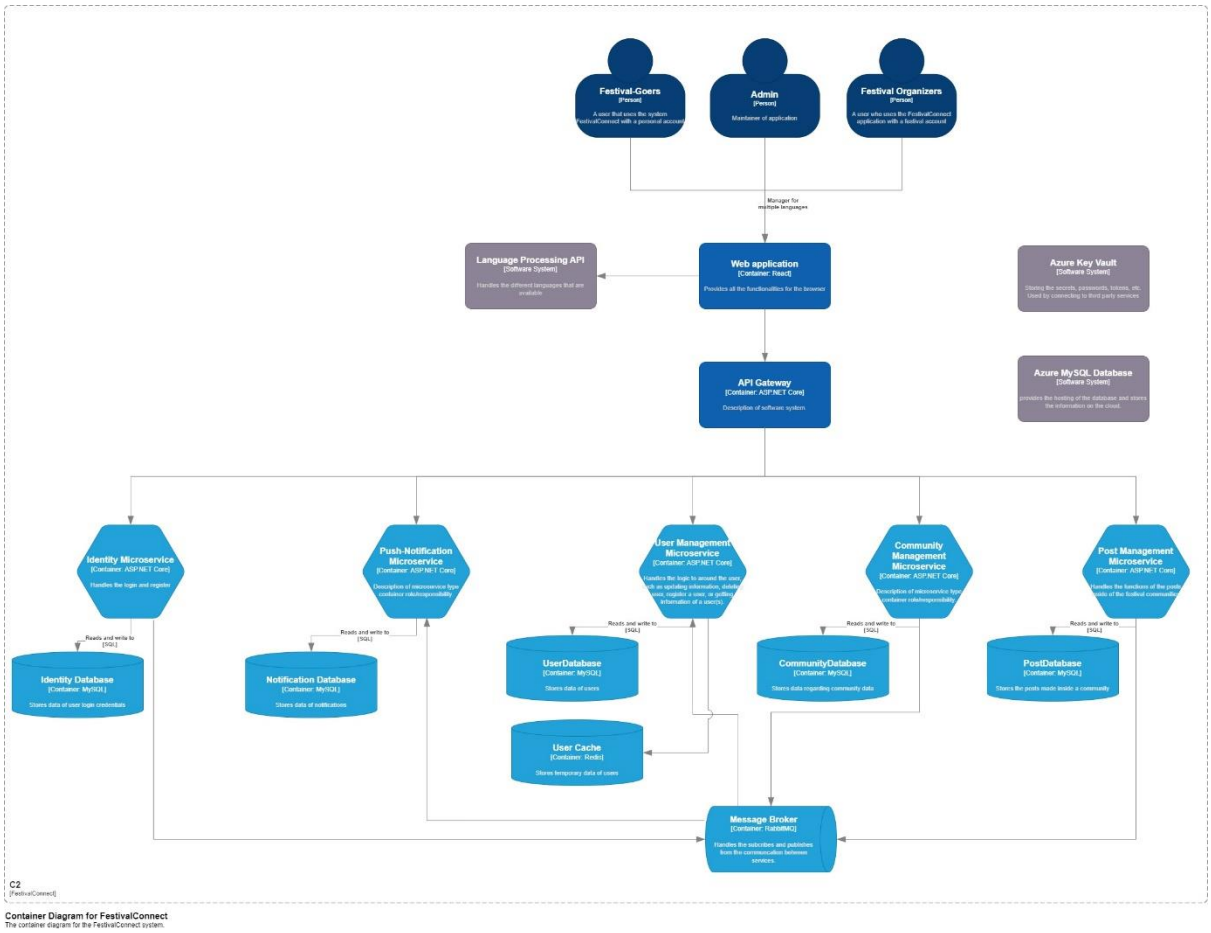


Figure 2: C2 Container Diagram

Description

Users: All the people who access the web application.
Web Application: the front end of the system, with the framework React.
API Gateway: Serves as an entry point for the front-end, based on the request goes to the correct service.
Services: Each service is responsible for one business case.
Database: Separate database for each business case, to store relevant information for each service. Improves independence, loose coupling, and privacy.
Message Broker: Communication point where services can subscribe or publish, to receive information from other services. It makes it possible for services can communicate with each other.
Azure MySQL Database: During deployments of the application, the databases will be hosted on the cloud.
Authentication flow: FestivalConnect has chosen to separate an identity service and user service for the following reasons:

- **Improving security**: By isolating the identity service, we will limit the access and permissions that each service has. The identity service will handle the authentication and authorization data, which will reduce the attack surface. Furthermore, since identity service is fully focused on security measurements, we can add multiple things such as multi-factor authentication, encryption, etc.

- **Privacy**: When the user service gets exposed by intruders, it will limit the amount of information that is exposed, which minimizes the risk of exposure. Moreover, from the conducted research on GDPR (Jacobs, 2024, Data Storage and GDPR Compliance Strategy: Practical Guide), this practice will help comply with GDPR, to have strict measures over personal data access and processing.
- **Scalability and Performance**: The service can be independently scaled based on their specific load and requirements, which can vary from each other. For performance, the services can optimize there service on their function.

Components (C3): Components with their relations.

Stretch: Components relations
Audience: Developers and Designers.

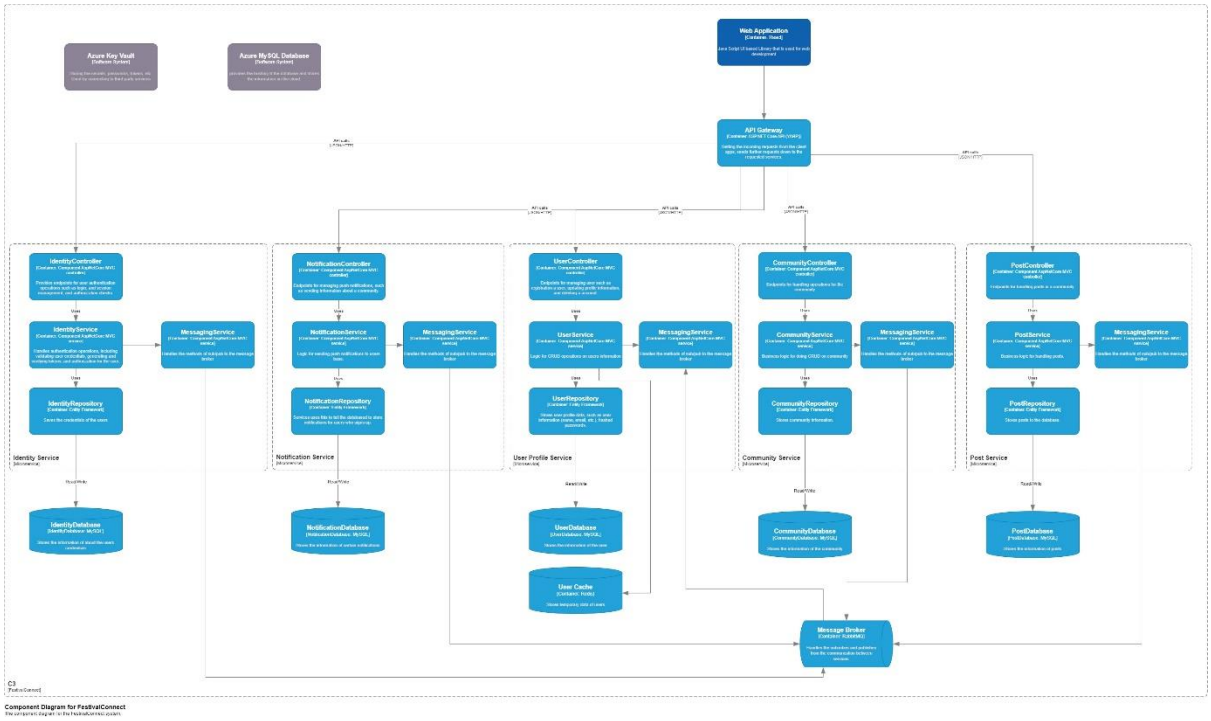


Figure 3: C3 Component Diagram

Separation of Components

Each component serves for its own purpose. This diagram dives further into having separate components in the services and how they communicate with each other. Every layer is responsible for one thing only, making sure that the single responsibility principle in SOLID is guaranteed.

Service Flow

When a service gets a request from the API Gateway, it will be an API call with JSON/HTTP, which will mainly be around CRUD. To check for certain validation, it will use the service layer to check for validation and verify for correct data. With the repository layer, the service layer will read/write to the database that is being used.

Code (C4): Code structure services

Stretch: Structure of Code

Audience: Technical Users and Designers (UI/UX)

The following structure will show how each service structure is set up, this applies for each microservice that uses a database.

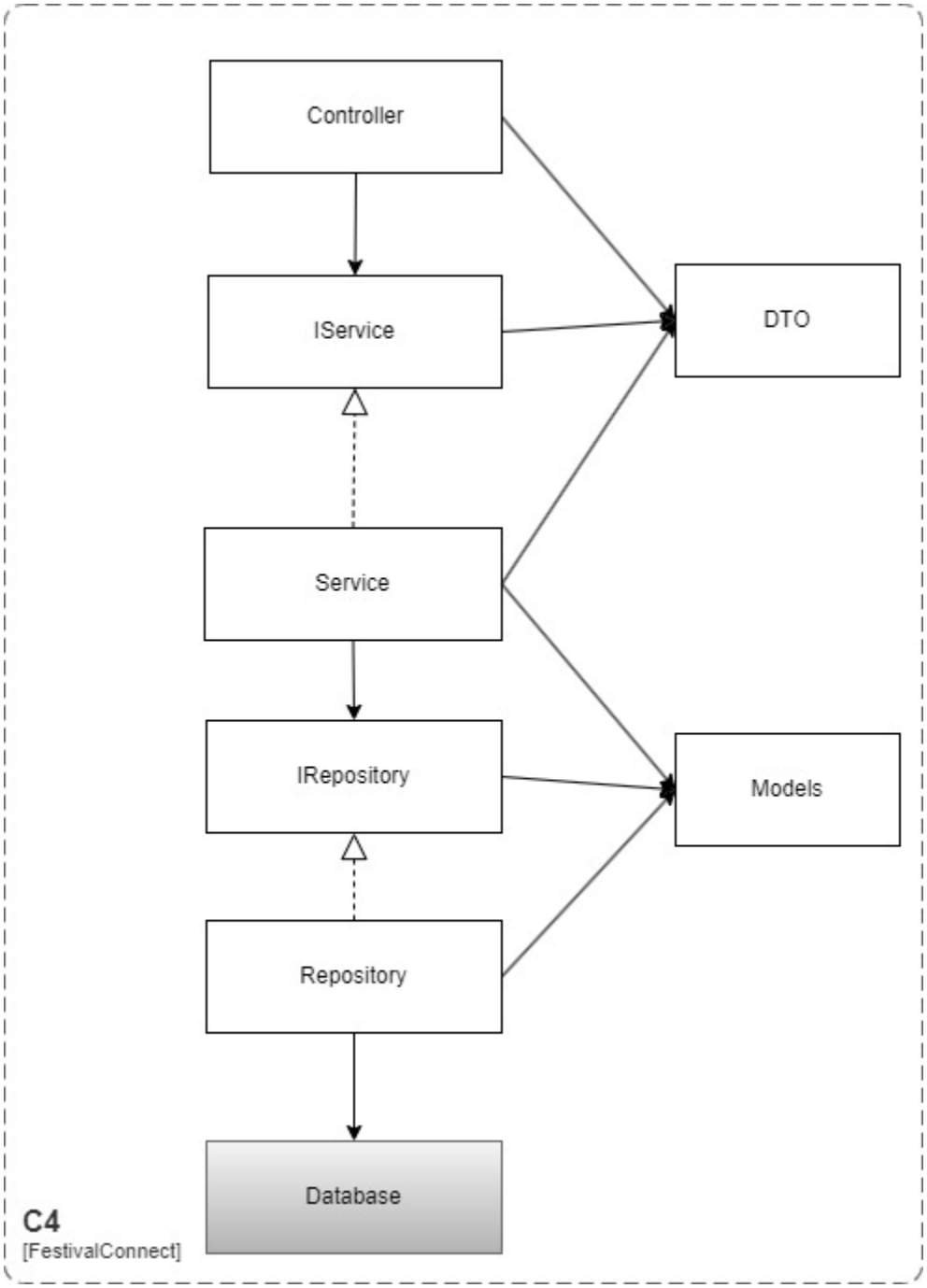


Figure 4: C4 Code Structure Diagram

Message broker

With a message broker, FestivalConnect can make it possible for services can communicate with each other. It will manage the routing, translation, and delivery of the message. A message broker serves as a middleman so that services can exchange data in a decoupled way and in an async manner. (What is a Message Broker?, n.d.)

A Message Broker is a key component inside of the microservices architecture, it will enable various benefits such as:

- **Decoupling Service:** Microservices interact with each other without knowing of each other. This makes it easy to modify, replace, and add new services.
- **Scalability:** Services can scale independently. The message broker will take care of the load balancing and make sure that the services are evenly appropriate amount of work.
- **Reliability:** When a service fails, the message broker can still be retrieved once the service is back up and working.
- **Event-Driven Architecture**: Enables real-time communication and responsive systems.
- **Communication Patterns**: It supports various communication which allows FestivalConnect to be flexible in choosing the correct pattern for each use case.

For the message broker, FestivalConnect looked into various options and found that RabbitMQ is a reliable source, high in performance, and has enough sources to correctly implement a message broker. (JackyNote, 2023)

See the following sequence diagram to have an idea of the flow of how a message broker works between two services. Note that one service subscribes (Consumes) and one publishes (Produce) a message.

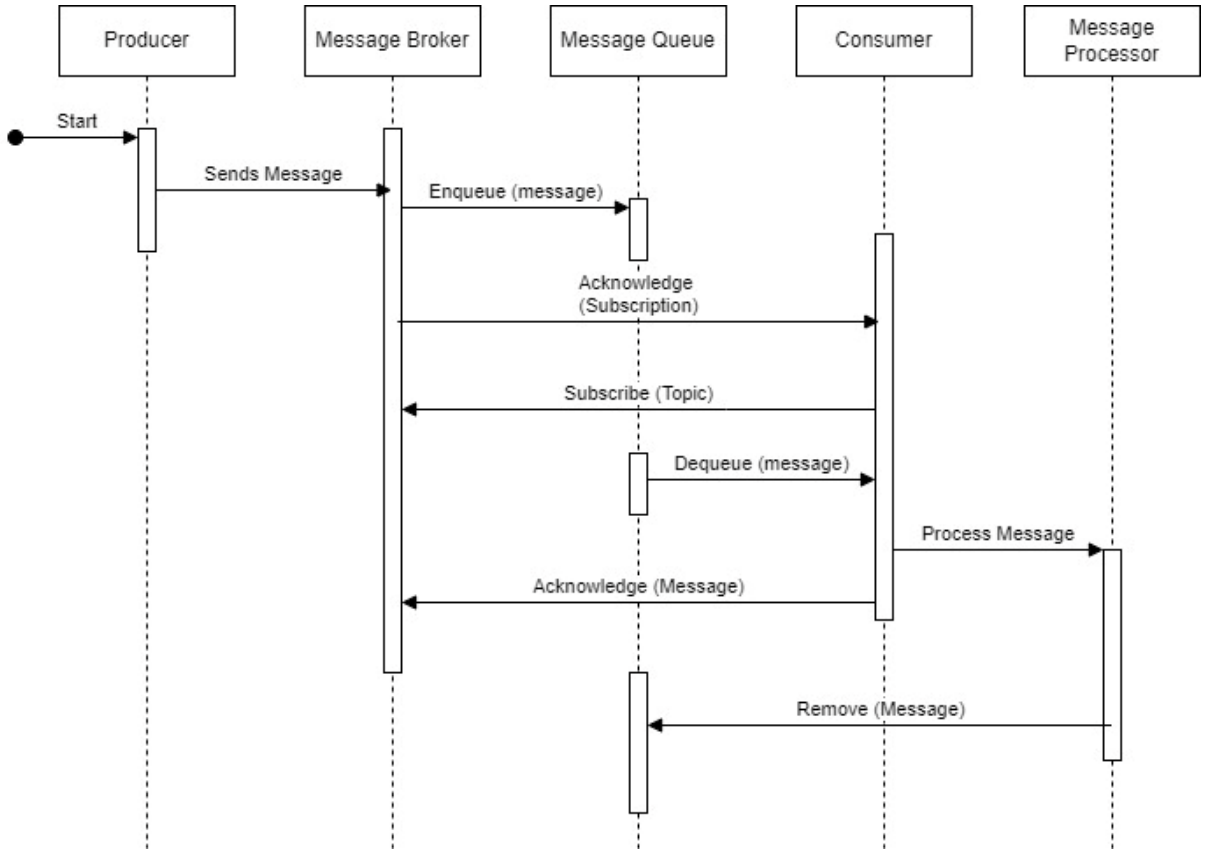


Figure 5: Sequence Diagram Message Broker

Load balancing

This method can be used so that we distribute the network's traffic, over an equal number of resources. Furthermore, FestivalConnect needs to handle a lot of traffic, we will have many servers with duplicate data, where the load balancer will evenly the traffic. This will improve the following:

- **Availability**: When a server is down, the load balancer will increase the fault tolerance, by also providing auto-detecting server problems and redirecting requests to available servers.
- **Scalability**: FestivalConnect can scale horizontally because traffic can be spread among multiple servers.
- **Security**: Load balancing will introduce an extra layer of security. It can deal with DDoS attacks, but also it provides monitoring traffic and automatically redirects attack traffic to specific backend servers which will have minimal impact on the system.
- **Performance**: Distributing the load will improve the response time and reduce the latency.

FestivalConnect can choose different algorithms to effectively use a load balancer, such as static load balancing, where you have limited rules to the load, such as round-robin or weighted round-robin method. Moreover, you can also choose to have a dynamic load balancing, which will look at the current state of traffic for the servers, which will make it possible to choose from servers that have the least traffic at the moment, etc. (What is Load Balancing?, n.d.)

Rate limiting

With rate limiting we can make it possible to set a maximum of requests for the client which will prevent overload, and ensure there is stability and security. It will improve the performance since it can protect FestivalConnect against threats such as DoS attacks. (Microservices Rate Limiting, 2023)

References

JackyNote. (2023, 10 14). *Message Brokers: Pros, Cons, and Their Crucial Role in Microservice*. Retrieved from jackynote.medium: https://jackynote.medium.com/message-brokers-pros-cons-and-their-crucial-role-in-microservice-3dc6c0df2e53

Microservices Rate Limiting. (2023, 09 27). Retrieved from appmaster: https://appmaster.io/glossary/microservices-rate-limiting

What is a Message Broker? (n.d.). Retrieved from vmware: https://www.vmware.com/topics/glossary/content/message-brokers.html

What is Load Balancing? (n.d.). Retrieved from aws.amazon: https://aws.amazon.com/what-is/load-balancing/#:~:text=Load%20balancers%20improve%20application%20performance,closer%20server%20to%20reduce%20latency

Jacobs, L. (2024). Data Storage and GDPR Compliance Strategy: Practical Guide (Unpublished manuscript), FontysICT.