RabbitMQ Infrastructure
Documentation

Paolo Bettelini
Scuola d'Arti e Mestieri di Trevano (SAMT)

Contents

1 Introduction

1.1 Abstract

The goal of this project is to make a network infrastructure which extensively uses a messaging queue system (RabbitMQ). My additional personal requirement is to use the Rust programming language as much as possible.

1.2 Information

This is a project of the Scuola Arti e Mestieri di Trevano (SAMT) under the following circumstances

- **Section**: Computer Science
- **Year:** Fourth
- **Class:** Progetti Individuali
- **Supervisor:** Geo Petrini
- **Title:** RabbitMQ prototype
- **Start date**: 2022-09-29
- **Deadline**: 2022-12-07

and the following requirements

- **Documentation**: a full documentation of the work done
- **Diary**: constant changelog for each work session
- **Source code**: working source code of the project

All the source code and documents can be found at http://gitsam.cpt.local/2022_2023_1_semestre/prototipo-rabbitmq [1].

1.3 Structure

This document is structured as such:

1. **Introduction:** General information, requirements and scope of the project
2. **Analysis:** Analysis

2 Analysis

2.1 Requirements

Req-00	
Name	Login & Register
Priority	1
Version	1.0
Notes	none
Description	The user must be able to create an account and log in.
Subrequirements	
Req-00_0	The Authentication must be kept alive by a cookie.
Req-00_1	The keep-alive cookie must contains a randomly generated token.
Req-00_2	The password must be hashed client-side.

Req-01	
Name	Functionality
Priority	1
Version	1.0
Notes	none
Description	The website must contain a file dropzone. The user must be able to upload an image which will be converted into a 200x200 px webp.
Subrequirements	
Req-01_0	During the conversion an async progress status must be display.

Req-02	
Name	Message Queues
Priority	1
Version	1.0
Notes	none
Description	Every message between WebServer and Worker must be through message queues.

Req-03	
Name	List of images
Priority	1
Version	1.0
Notes	none
Description	When the users logs in a list of the previously converted images must be display.
Subrequirements	
Req-03_0	Only the last N images are loaded. Another chunk of images is loaded if requested by the user.

Req-04	
Name	Network Structure
Priority	1
Version	1.0
Notes	none
Description	A loadbalancer (Round Robin) is the entry point for N WebServers. For each WebServer there exist a RabbitMQ server. There are M workers which access the queues of the queue servers. Each worker stores data on the same database.

Req-05	
Name	Scalability
Priority	1
Version	1.0
Notes	none
Description	The network must scale with multiple servers.

3 Planning

3.1 Initial Gantt Chart

3.2 Final Gantt Chart

4 Infrastructure

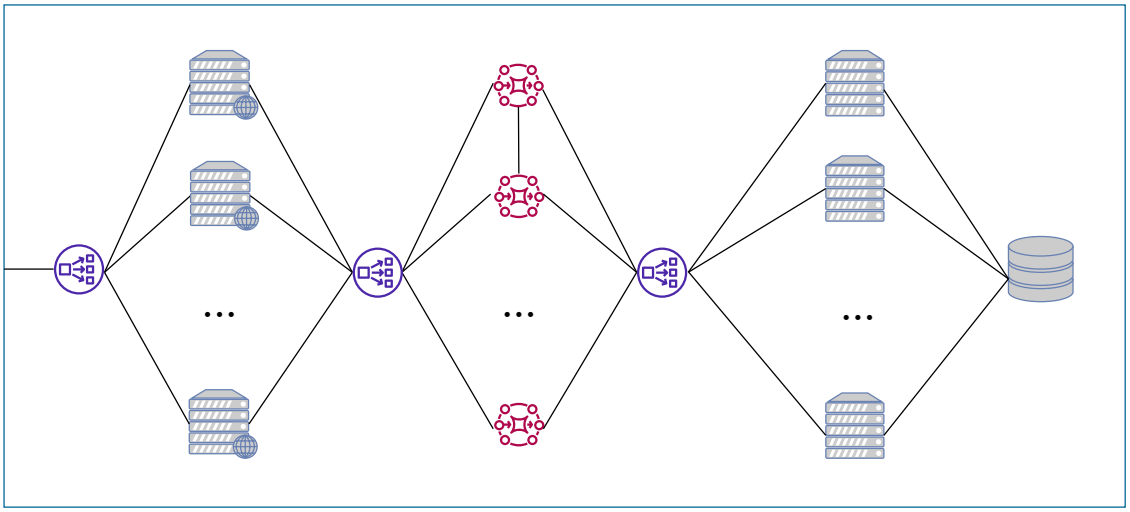

Figure 1: Network Infrastructure

5 Technologies

5.1 WebAssembly

WebAssembly (wasm) is a portable low-level language supported by all majors browsers. It can be used for a variety of things within the browser and can be mixed with HTML and JavaScript. WebAssembly does not need to be parsed by the browser since it is already in a binary format.
At the time fo writing, WebAssembly is not widely used and it is not always faster than JavaScript based applications. However, is it my opinion and hope that it will better in the future and that it will become a standard in web development.

5.2 Rust

Rust is a generic compiled programming language. The code is compiled using LLVM to machine code and its speed is comparable to C and C++. Rust is the first programming language to guarantee memory safety; memory is not manually freed nor garbage collected. It is not possible to dereference a null pointer, cause memory segfaults, core dumps and memory leaks. Code that could cause undefined behavior can still be written, but it is strictly bounded in blocks where the compiler is relaxed. This relaxation implies that the language is also low-level. Another key feature to the performance of Rust is zero cost abstraction, which means that generic types and function abstractions are resolved at compile-time. Conditional compilation and compile-time computations are also extensively used.
There are also many features concerning the programming experience, such as advanced metaprogramming and code generation using macros, intelligent compiler, dependency system (Cargo), modern syntax and many tools to ease development.

5.3 RabbitMQ

RabbitMQ is a popoular message broker implementing many messaging protocols. Message brokers such as RabbitMQ can make an infrastructure to route messages, validate them and transform them. Messages queue are used to store messages. Multiple consumers may consume messages from a queue.
RabbitMQ servers can also form a cluster. `todo`

6 Implementation

6.1 Frontend

6.1.1 Yew

Rust can be compiled into wasm, this allows frameworks such as Yew to exist. Yew is a Rust framework to build web application using WebAssembly.

6.2 Database

The databased is an instance of `MariaDB`.

```
CREATE TABLE user (
    id INT PRIMARY KEY AUTO_INCREMENT,
    mail VARCHAR(50) NOT NULL,
    username VARCHAR(25) NOT NULL,
    password BINARY(32) NOT NULL,
    created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

```
CREATE TABLE image (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    uploaded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    data BLOB NOT NULL,
    FOREIGN KEY (user_id)
        REFERENCES user(id)
            ON UPDATE CASCADE
            ON DELETE CASCADE
);
```

6.2.1 Diesel

`diesel` is an ORM library for the Rust programming language. It supports MySQL, Postgres and SQLite.

6.3 Load Balancer

6.4 Backend

6.5 Messaging

6.5.1 RabbitMQ

6.5.2 Messages

RabbitMessage (enum)

Field	Content	Description
LoginRequest	LoginRequestData	Login request packet
LoginResponse	LoginResponseData	Login response packet
RegisterRequest	RegisterRequestData	Register request packet
RegisterResponse	RegisterResponseData	Register response packet
GetImage	GetImageData	Get image data packet
ShrinkAndUpload	ShrinkAndUploadData	Shrink and upload image packet
GetTotalImages	GetTotalImagesData	Get total images packet
GetTotalImagesResponse	GetTotalImagesResponseData	Get total images response
ErrorResponse	ErrorResponseData	Error packet

LoginRequestData (struct)

Field	Type	Description
mail	String	The mail
username	String	The username
password	Vec<u8>	The password

LoginResponseData

Field	Content	Description
Ok	LoginResponseDataOk	Positive login response
Err	LoginResponseDataErr	Negative login response

LoginResponseDataOk

Field	Type	Description
token	Vec<u8>	The authentication token

LoginResponseDataErr

Field	Content	Description
NotFound	()	User was not not
WrongPassword	()	Password was incorrect

TODO...

7 Structure

7.1 mandate

The `mandate` folder contains all the documents regarding the project (documentation + diary).

7.2 common

`common/` is a collection of Rust libraries.

7.2.1 config

7.2.2 database

7.2.3 messaging

7.3 worker

`worker/` is the Rust project for the backend service.

7.3.1 Usage

7.4 webserver

`webserver/` is TODO.

7.4.1 Usage

8 Testing

8.1 Test protocol

8.2 Test results

ID	Result	Note
Test-00	Failed	Someting
Test-01	Failed	Someting
Test-02	Failed	Someting
Test-03	Failed	Someting
Test-03	Failed	Someting

9 Conclusion

10 References

Sitography

[1] Paolo Bettelini. *rs-rabbitmq-infrastructure*. 2022. URL: https://github.com/paolobettelini/rs-rabbitmq-infrastructure.