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 [**gitrepo**].

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-00_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

5 Technologies

5.1 WebAssembly

WebAssebly (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

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`.

```sql
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
);
```

```sql
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

RabbitMQ is a popoular message broker implementing many messaging protocol.

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