



# eikona - Challenge Project

## OST — Ostschweizer Fachhochschule

Lukas Ribi, Dominik Castelberg, Pascal Christen

May 18, 2021

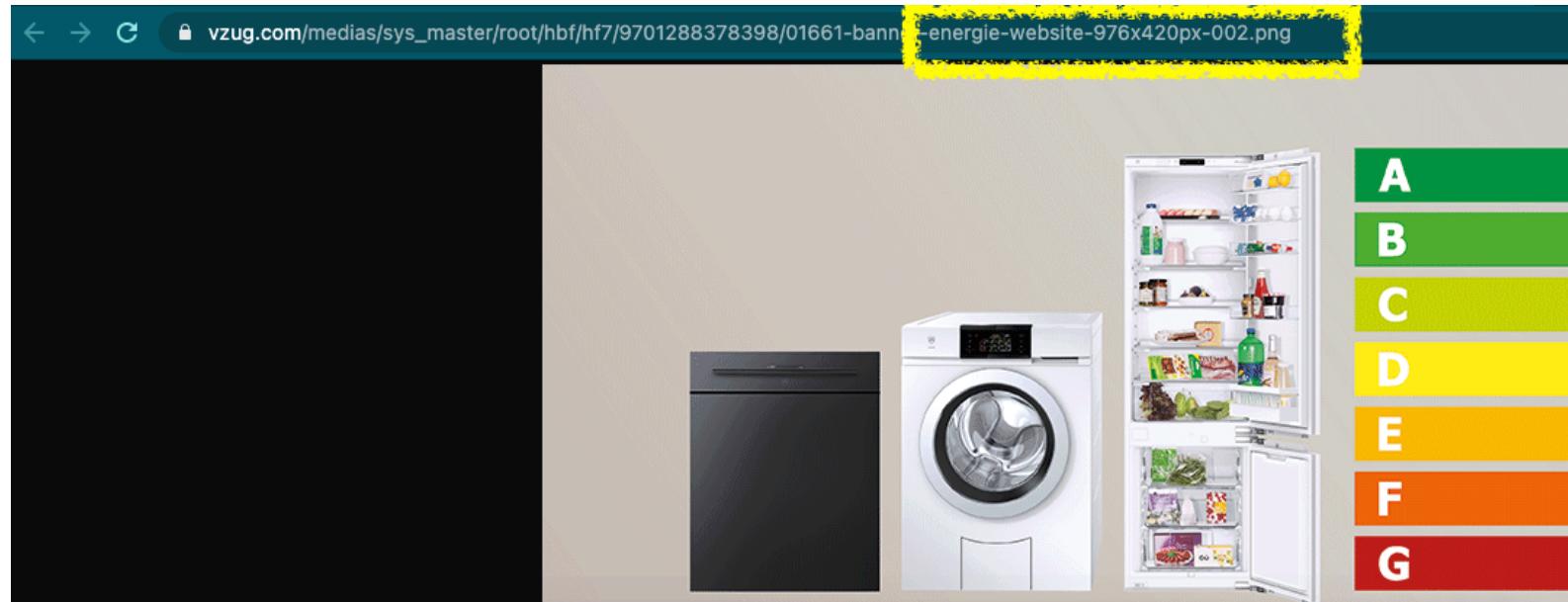
DS1 - Thomas Bocek

## 1. Introduction

# Goal

Problem: Often you need to resize, adjust quality, and convert on-demand for:

- Responsive websites
- Upload of an image



Solution: Our *eikόva* [i'kɔna] service converts the image **on-the-fly** and provides a simple *REST API*.

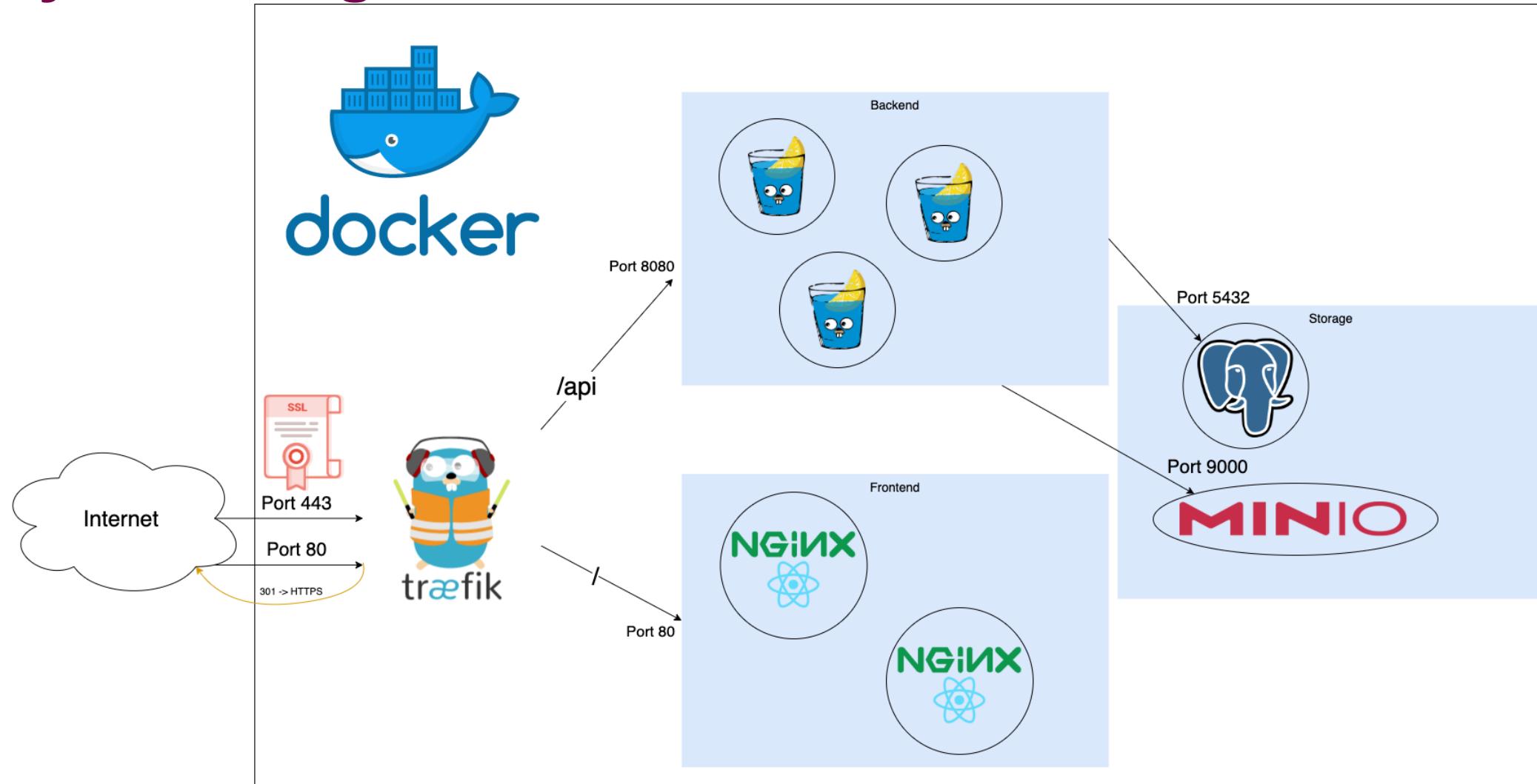
## 2. Technology

# Setup

- Backend
  - `go` (1.16.4)
    - `gin` (1.7.1) with `Ristretto` (0.0.3)
    - JWT
- Frontend
  - `ReactJS` (17.0.2)
    - Material-UI
    - Served by `Nginx` (1.20.0)
- Loadbalancer
  - `Traefik` (2.4.8)
- Storage
  - `PostgreSQL` (13.2)
  - `MinIO - Object Storage` (RELEASE.2021-03-26)
- Presentation
  - `LaTeX`

## 2. Technology

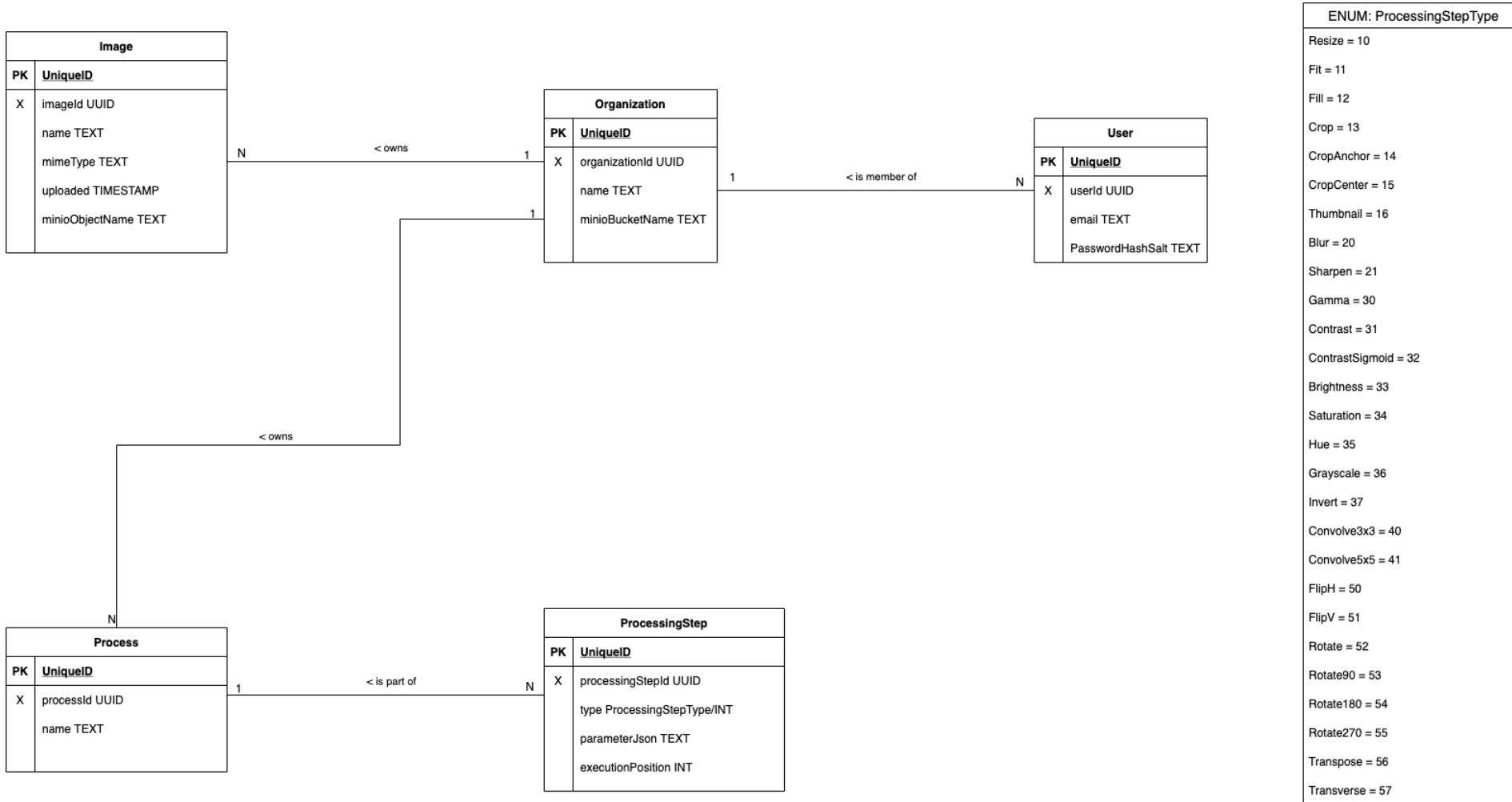
# System Diagram



### 3. Development

# Data Model

imgProcessing Entity Relations



### 3. Development

## Github Actions

- Docker Images - builded and versioned by Github Actions
  - Frontend
  - Backend
- Images hosted by GitHub Packages ([ghcr.io](https://ghcr.io))
- Security by scanning for [CVE](#)

📦 2 packages

 <b>frontend</b> latest	Published 8 days ago by imgProcessing in imgProcessing/backend	 276
 <b>backend</b> latest	Published 8 days ago by imgProcessing in imgProcessing/backend	 270

### 3. Development

## docker-compose

- Development environment:
  - Debugging with `delve`
  - Live reloading with `air`
  - Local certificates with `mkcert`
- Production environment:
  - Only exposing Loadbalancer network (Port 443/80)
  - Using multiple networks
  - Working with environments

### 3. Development

## Loadbalancer

### Traefik:

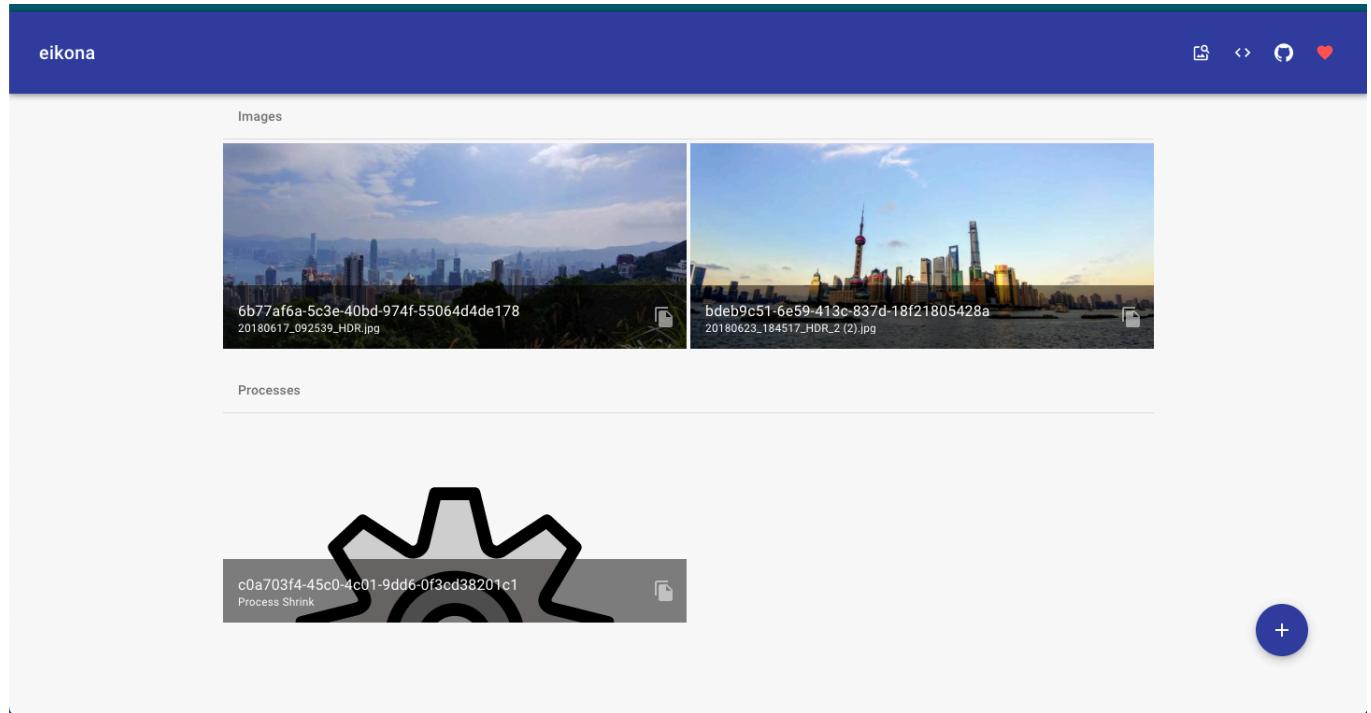
- Docker Socket mounted:
  - + Autodiscover
  - - Security
- HTTP -> HTTPS Middleware
- Healthchecks

### 3. Development

## Frontend

ReactJS with Material-UI:

- Responsive design
- Simple error handling
- Functions:
  - Dashboard
  - Upload Image
  - Create Process
  - API Docs



### 3. Development

## Backend

Golang:

- Image filtering using [gift](#):
  - [gift vs. imaging](#)
  - Advanced image processing capabilities and filter chaining
  - 25 implemented operations right now (expanding...)
- REST API using [gin](#):
  - JWT authentication using middleware
  - API Documentation using [swagger](#)
- Quick win: Simple caching using [Ristretto](#) at the single process level
  - Caching based on request (parameters / image id & pipeline combination)
  - Enhancement: Use varnish, nginx or Traefik enterprise in front of the application to handle image caching

### 3. Development

## Storage

### MinIO:

- High Performance, Kubernetes Native Object Storage
- Ready for distributed usage
- Compatible with Amazon S3
- Isolated buckets per organization with source images

The screenshot shows the MinIO Browser interface. At the top, it displays the bucket name: `a3b5f05f-0fb3-4a4b-93b5-8079c93534fe`. Below this, it shows the storage usage: `Used: 1.79 MB`. The main area is a table listing objects within the bucket:

Name	Size	Last Modified	Actions
<code>1016a384-0386-4cb1-8995-89d35c76f0c5</code>	1.31 MB	Apr 27, 2021 5:51 PM	...
<code>a0bc40db-840f-4c10-bbd2-a3b896b89ba8</code>	491.06 KB	Apr 27, 2021 4:55 PM	...

#### 4. Legal

## Fulfillment of Requirements

- Dockerized: Used (images hosted on Github)
- Load balancing: Traefik
- Scalable service: Backend and Frontend
- JWT Authentication: Used
- Latest FOSS: Used

# Demo

<https://eikona.pesc.xyz>

# End

Source Code