ZWO API

Requirements & Design: Detailed Technical Design and Specs

Table of contents

Instruction

This document provides the up-to-date detailed technical (infrastructure) specifications and design to address the business needs.

The documents provided during the Inception phase of the project should be used as a starting point for this more detailed document.

1. Project Overview

NestJS backend API, serving the front-end on demand data, to a social media app for animal lovers.

1.1 Objective

Provide an easy-to implement, scalable and cost-effective backend API to serve both the mobile and web app of the ZWO project.

1.2 Business Case

Running a Nests app on hosted Debian VM, in conjunction with a standalone PostgreSQL Database, the backend side of the project aims to separate the different independent services. Thus, maintaining solid independent modules to insure easy scalability in the future as well as faster troubleshooting.

1.3 Risks

AWS cloud solutions cost in production phase

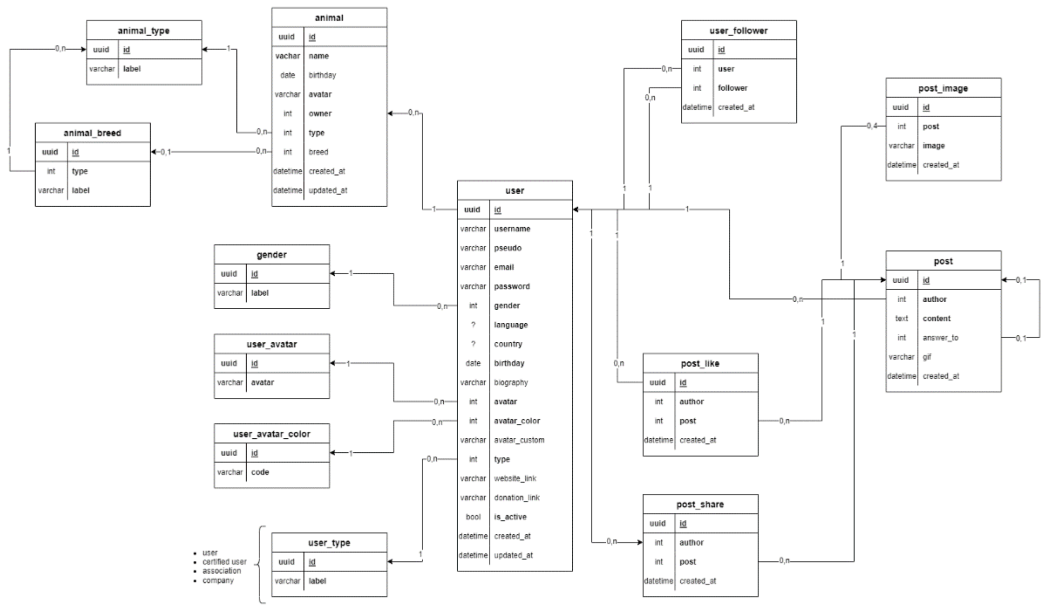
1.4 Out of Scope

Security (pushed a later phase), nonessential framework and packages, code quality.

2. DBMS UML diagram

Below Is the Database management system UML constructed by **Thibault** and **Nathan**.

You can find the full version here



3. Technical Specifications

Below are the detailed technical specifications for required infrastructure.

3.1 Servers

3.1.1 Application Servers : AWS EC2 t4g.xlarge

Specs:

- Debian 9
- 4 vCPU
- 16GB RAM
- 5Gigabit network transfer speed
- 64GB gp2 SSD
- Expected application transaction volume not yet estimated

Load Balancing:

- AWS ELB: Elastic Load Balancer

3.1.2 Database Servers

As of yet, there are two ways to go about managing the database.

- AWS RDS:
    - Scalable

        - event notification
        - Plug-and-play
        - Managed backups
        - Built-in metrics and monitoring

- PostgreSQL DB running on AWS EC2 instance:
    - Cost effective (lower initial costs)
    - Dockerized PostgreSQL container

3.2 Web App:

- Node.js 16.3
- Nest 8.0
- Typescript 4.3
- Sequelize ORM

3.3 Serverless Apps:

Although we adopted a served approach to the main API, we remain open to implementing section features and services through a serverless approach, using the lambda function and the Serverless framework.

These Features would be independent form the main App, and require no communication with the main front of the app.

Examples of use case are below, but are in no way a representation of the scope of usage

- Email and Newsletter services
- Chrono jobs: functions that run every X-amount of time
- Automated backups
- Automated documentation
- Backend cleaning
- Forgot Password service

3.4 DevOps & Sysadmin:

- Docker & docker-compose
- CI/CD: GitHub Actions
- Ansible (for migration between VMs during Dev phase)
- Monitoring:

    - CloudWatch
    - CloudTrail
    - EC2 Dashboard
    - Optional: NetAPP/SolarWinds

3.5 Databases

For a detailed look at the DB tables, refer to section 2

- PostgreSQL
- Version 14.2
- Memory requirements: N/A
- CPU requirements: N/A
- Amount of storage needed the first 12 months (in GB): 50
- Projected annual growth (% or GB): 20-50
- Expected transaction volume: N/A
- Special data preservation requirements: N/A

3.6 Storage Needs exclusive of databases

For the mass storage we are using **AWS Simple Storage Service (S3)** with the official AWS Nodejs SDK.

- The amount of storage needed for the first 12 months (in GB): 200
- Projected annual growth (% or GB): 60-90
- Data protection: SSL/TLS, MFA, user activity logging (Cloud Tail)
- Data Backup 30% of the amount of storage

4. Resource Needs

Identify the staff resources needed to successfully complete this project. Also identify the staff, technical, and other resource dependencies that should be considered.

- Staff Resources
- Staff Dependencies
- Technical Dependencies
- Other Dependencies

5. Projected Costs

Below is the costs breakdown

Service	Cost (USD) per Unit per month	Expected Number of Units
App EC2 Instances	43.5	1-N/A
DB EC2 Instances	30	1-N/A
RDS service	N/A	N/A
Lambda function **(3M request of 120ms)**	10-15	N/A

6. Assumptions

Prices mentioned in the previous section 5 are calculated based on the specifications mentioned above and are only applicable during the Production phase.

Until then all services will run on GCP services on their 3-month student free plan. Once each trial expires will migrate all services to a new account using Ansible.

Specs provided are subject to technical modifications. In those events change, project managers are to update the table at the end of this documents.

7. Concerns and Issues

Document Tracking

The following chart is used to log of all changes made to this document.

Version	Date of edit/change	Who made the edit/change	Description of edit/change
v1.0.0	22/04/2022	Ahmed Rahmouni	Initial technical doc
V1.1.0	29/04/2022	Ahmed Rahmouni	Added sections 4->7