Dsy_ChallengeTask2

Challenge Task FS 2023

This semester's challenge task (CT) is the design and implementation of a simple distributed system (of your choice), with at least one service with a websocket, and where a service instance can fail. The system needs to have the following components:

1. Simple Frontend (e.g., HTML, Vue, React, Svelte)
2. Loadbalancer(e.g., traefik, nginx, HAproxy, Caddy)
3. Two instances of a service (your choice), and during the challenge task presentation, one instance will be shut off.
4. One storage backend

Requirements

All requirements below must be met in order to pass this lecture.

1. Load balancing with scalable service
2. Failover of a service instance
3. Dockerized and Frontend connects via HTTP/3
4. Persistant storage (storage does not need to be scalable, but you can build it

scalable if you want)

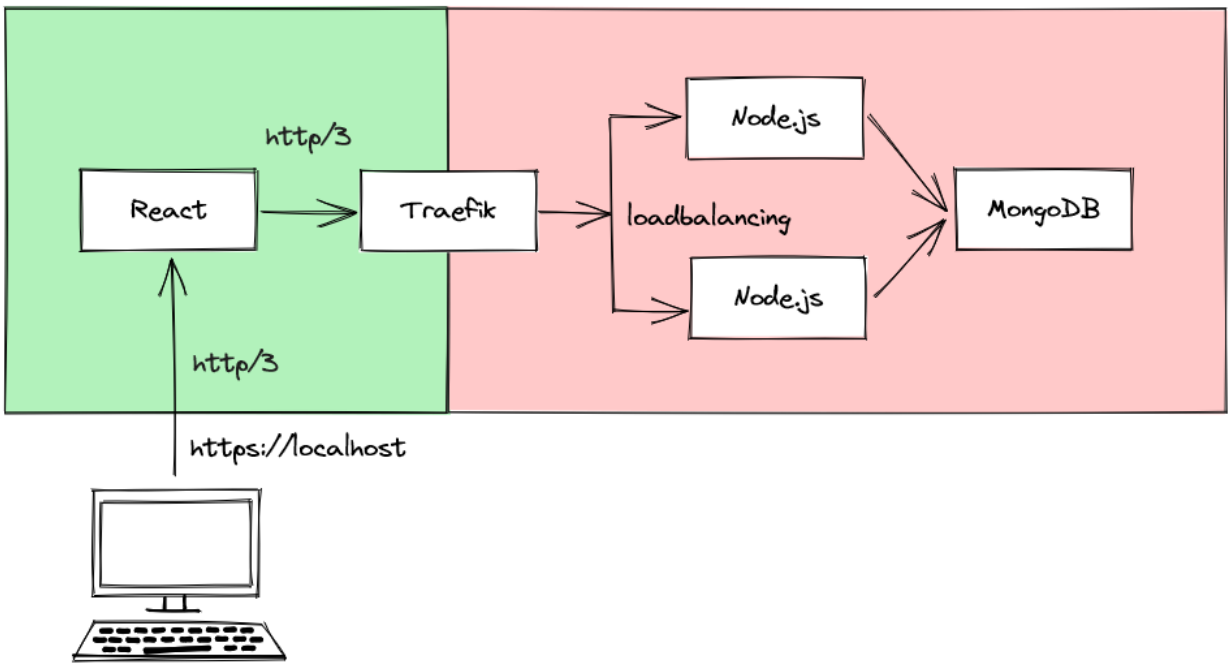
5. Use latest stable releases of chosen libraries and frameworks

Deliverables

Hand-in 1: 17.04.2023, 23:59 (CET) - initial version of your project.

Final hand-in: 22.05.2023, 23:59 (CET) (CET) - well documented infrastructure, presentation (slides) of the application, also showing the architecture and design decisions via email to thomas.ost-at-bocek.ch or via a repository invite. The code and configuration should be easy to read and/or well documented, the presentation (slides or text) should show the architecture, components, and design decisions. During the weeks 23.05./30.05., you will present and demo your solutions onsite.

Architecture Overview

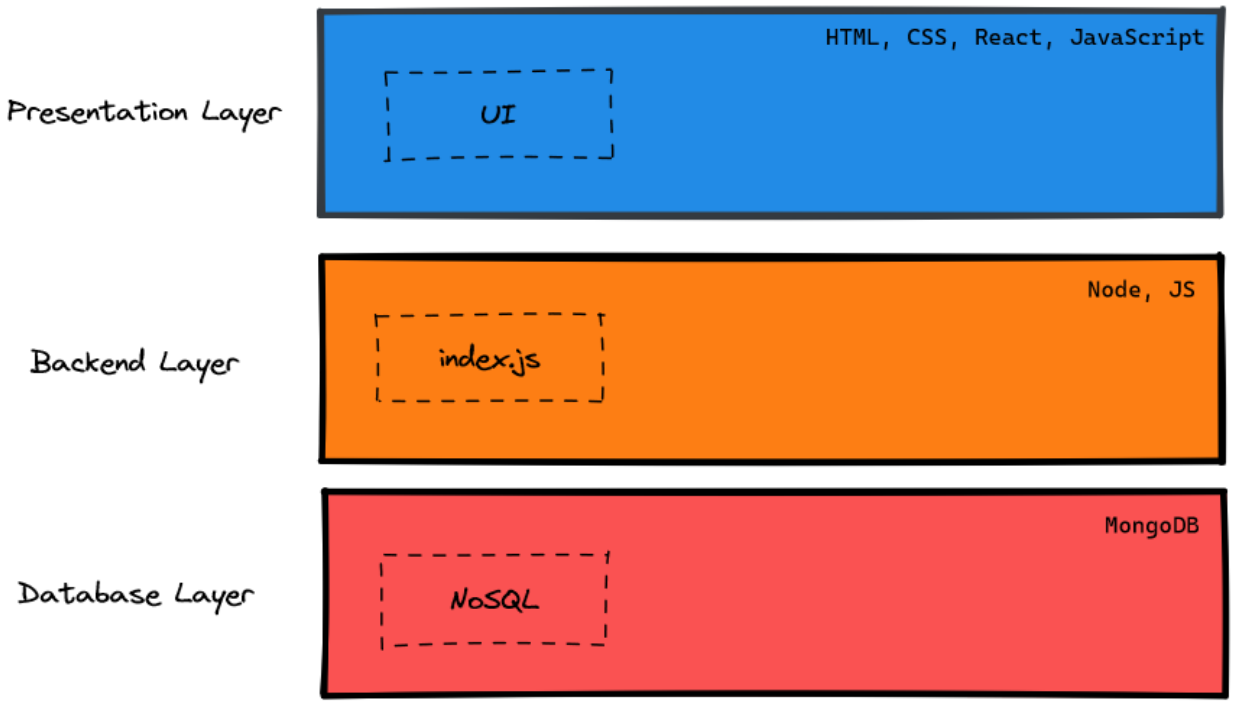


There are 4 components:

- React
- Traefik
  - Listens to all traffic on port 80 (http) which redirects 443 (https)
- Nodejs

- MongoDB

Layer Architecture



Frontend/Presentation layer

Technology stack:

- HTML
- CSS
- Javascript
- React.js

To write our frontend service we decided to use React. React is the most popular frontend framework and therefore provides a lot of great documentation. React was another new technology for us. We had previous experiences with javascript and npm.

Our presentation Layer consists of the user interface component. The user can interact with the application over a web browser. This layer communicates over the backend layer to retrieve data and notify for changes.

Loadbalancer

As a loadbalancer for our application we decided to use traefik. In the beginning we used traefik, but HTTP3 was not always visible, therefore we tried it with Caddy and it worked. But after some thinking, we decided to give traefik another try and it worked. Our thinking process was the following: we have some experience with traefik so we should stick with it. Another reason is the excellent documentation and of course it is used in our course. Traefik is open source and designed to be used with virtualisation technologies.

Backend/Backend layer

Technology stack:

- Node.js
- JavaScript
- Express
- Docker

We had some experience with most of the technologies, but never really in detail.

The backend layer is the foundation of the application and defines services and logical implementations for our webapplication. Every service works through the backend layer and the backend layer communicates to the presentation layer if any changes have occurred.

Usage with Docker

```
docker-compose up --build
```

This will take a second so please have little patience. After that open your browser at https://localhost