Frantic

ISAAC WÜRTH &
DAMIAN KALBERER

Inhalt

Idee

Demo

Technologien

Architektur

Requirements

Diskussion

Idee

Frantic ist ein Kartenspiel und wird ähnlich gespielt wie UNO

Viele Regeln und Aktionen, welche nachgelesen werden müssen

-> Lösung: Web-UI Cheat Sheet für Frantic

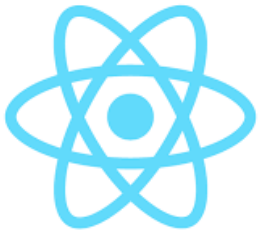
Demo

Technologien

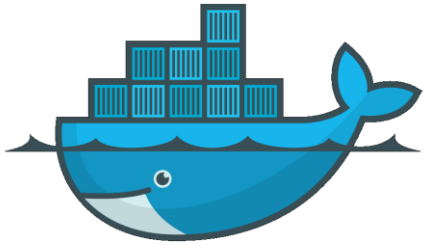
Frontend
- React
- TypeScript

Backend
- Express
- MongoDB
- TypeScript
- NodeJS

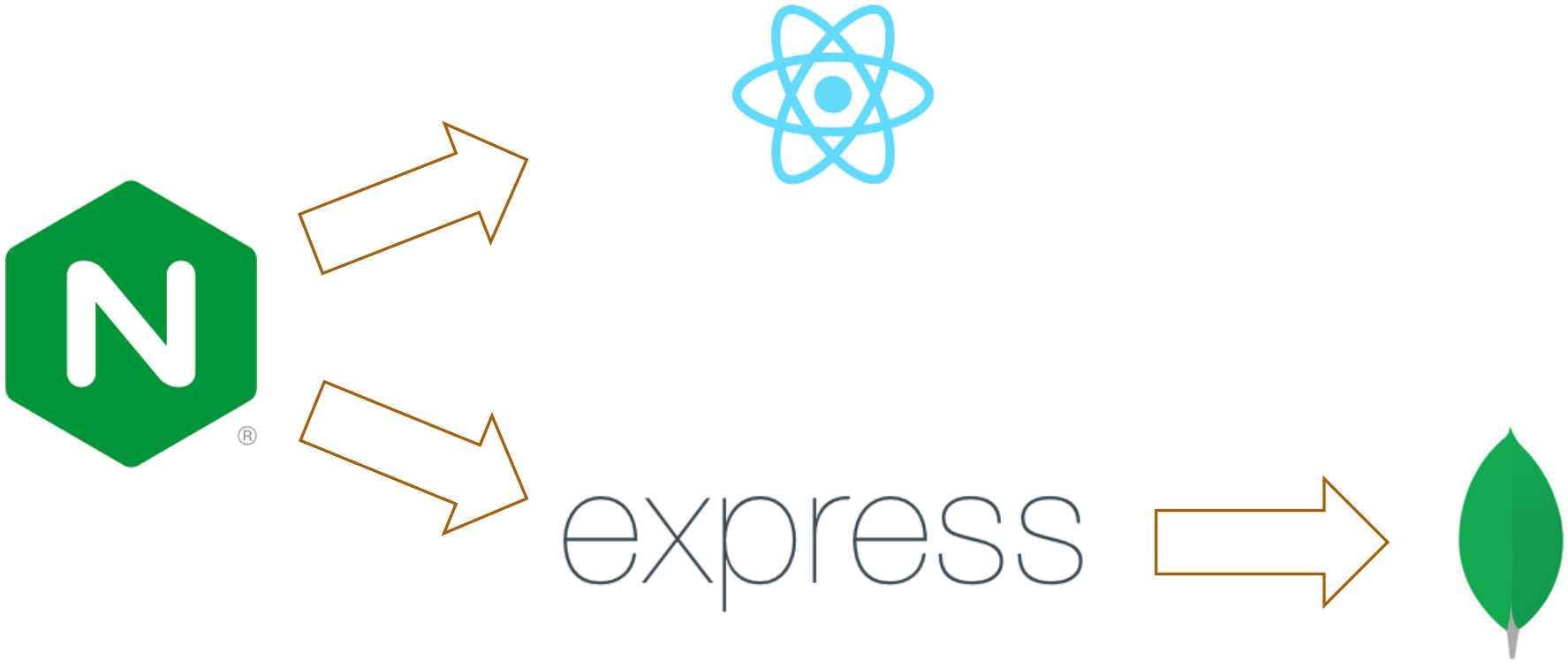
Deployment
- Nginx
- Docker
- JWT







Architektur



Implementation

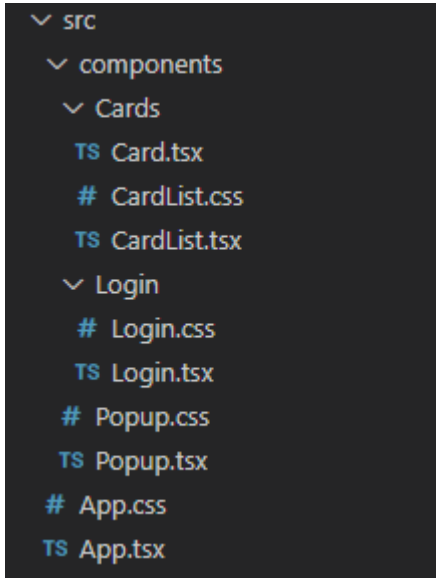
Frontend Komponenten

App
- Überprüft JWT
- Filter nach Kategorien
- Benutzt CardList Komponente

CardList
- Holt die Daten vom Backend
- Benutzt Card Komponente

Card
- Zeigt Karten an
- Benutzt Popup Komponente

Popup
- Zeigt Informationen zu den Karten



Implementation

Backend

Router
- Leitet REST Anfragen weiter

Auth
- Für die Authentisierung und JWT Generierung zuständig

Card
- Für die Karten Informationen zuständig

File
- Für die Bilder zuständig

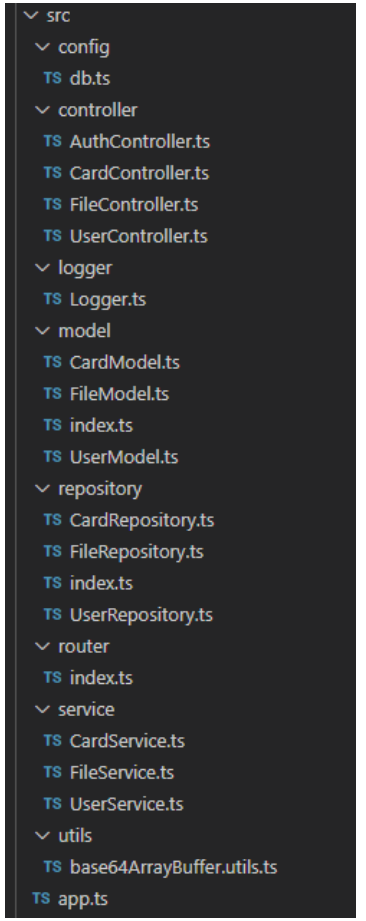
User
- Für CRUD der Benutzer zuständig

Controller
- Stellt jeweilige Funktionen vom Service zur Verfügung

Service
- Reicht Funktionsaufruf an Repository weiter

Repository
- Benutzt Funktionen vom Model

Model
- Stellt Datenbank Schema und somit Funktionen zur Verfügung



Requirements

Load balancing	Scalable service	JWT authentication	Dockerized	Use latest stable releases of chosen libraries and frameworks
• Nginx	• REST	• Npm jsonwebtoken	• Dockerfile	• Package.json

Load Balancing

default.conf

```
#/etc/nginx/conf.d/default.conf
upstream api {
    #least_conn;
    server api01;
    server api02;
    server api03;
}
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    location /api/ {
        proxy_pass http://api;
    }
    location / {
        proxy_pass http://frontend;
    }
    # You may need this to prevent return 404 recursion.
    location = /404.html {
        internal;
    }
}
```

Scalable service

Stateless -> REST

Siehe vorherige Folie “Load Balancing”

```
getData() {
    axios.get(`${process.env.REACT_APP_API_URL}/card`, {
        headers: {
            'Authorization': `Bearer ${this.getToken()}`
        }
    }).then(res => {
        let data = res.data
        this.setState({data: data})
    })
}
```

```
async function loginUser(credentials : any) {
    return fetch(`${process.env.REACT_APP_API_URL}/login`, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(credentials)
    }).then(data => data.json())
}
```

JWT authentication

https://www.npmjs.com/package/jsonwebtoken

import jwt from "jsonwebtoken"

```
private generateAccessToken(username : object) {
    return jwt.sign(username, this.TOKEN_SECRET, { expiresIn: '900000000s' });
}
```

```
async login(req: Request, res: Response) {
    const username = req.body.username
    const password = req.body.password
    const user = await this.userRepository.getFirst({username, password})
    if (user != null) {
        const token = this.generateAccessToken({ username });
        res.cookie('jwt', token); // Should be httpOnly
        res.json({token});
    } else {
        res.statusCode  = 401;
        res.json();
    }
}
```

```
private extractToken (req : Request) {
    if (req.headers.authorization && req.headers.authorization.split(' ')[0] === 'Bearer') {
        return req.headers.authorization.split(' ')[1];
    } else if (req.query && req.query.token) {
        return req.query.token;
    }
    return null;
}
```

```
async isAuthenticated(req: Request, res : Response, next : NextFunction) {
    if(process.env.AUTHENTICATION === 'false') return next()
    if(req.cookies?.jwt === undefined && req.header('Authorization') === undefined) return this.unauthorized(req, res)
    const token = req.cookies.jwt || this.extractToken(req)
    jwt.verify(token, this.TOKEN_SECRET, (err : any, user : any) => {
        if (err) return this.unauthorized(req, res)
        // @ts-ignore
        req.user = user
        return next()
    })
}
```

Dockerized

Frontend

```
# STAGE 1
FROM node:current-alpine as build
WORKDIR /app
COPY package.json ./
RUN npm install
COPY . .
RUN npm run build

# STAGE 2
FROM nginx:stable-alpine
LABEL maintainer = "Isaac Würth, Damian Kalberer"
WORKDIR /usr/share/nginx/html
COPY --from=build /app/build .
COPY --from=build /app/.env ./.env
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Backend

```
FROM node:15.10.0-alpine3.12

LABEL maintainer = "Isaac Würth, Damian Kalberer"
RUN mkdir /app
RUN addgroup -S app
RUN adduser -S app -G app

WORKDIR /app
COPY ./package*.json /app/
RUN npm install
COPY ./ /app/
RUN npm run build
RUN rm -rf /app/.git
RUN mkdir /app/uploads
RUN chown -R app:app /app

USER app
ENTRYPOINT npm start
```

Use latest stable releases of chosen libraries and frameworks

Frontend

```
"dependencies": {
  "axios": "latest",
  "lodash": "latest",
  "react": "latest",
  "react-card-flip": "latest",
  "react-dom": "latest",
  "react-lazyload": "latest",
  "react-native-web": "latest",
  "react-scripts": "latest",
  "reactjs-popup": "latest"
},
"devDependencies": {
  "@types/axios": "latest",
  "@types/lodash": "latest",
  "@types/react": "latest",
  "@types/react-dom": "latest",
  "@types/react-lazyload": "latest",
  "@types/react-native": "latest",
  "typescript": "latest"
}
```

Backend

```
"devDependencies": {
  "@types/cookie-parser": "latest",
  "@types/cors": "latest",
  "@types/express": "latest",
  "@types/jsonwebtoken": "latest",
  "@types/morgan": "latest",
  "@types/multer": "latest",
  "@types/node": "latest",
  "@types/recursive-readdir": "latest",
  "copyfiles": "latest",
  "nodemon": "latest",
  "ts-node": "latest",
  "typescript": "latest"
}
```

```
"dependencies": {
  "axios": "latest",
  "body-parser": "latest",
  "cookie-parser": "latest",
  "cors": "latest",
  "dotenv": "latest",
  "express": "latest",
  "form-data": "latest",
  "jsonwebtoken": "latest",
  "lodash": "latest",
  "mongodb": "latest",
  "mongoose": "latest",
  "mongoose-to-swagger": "latest",
  "morgan": "latest",
  "multer": "latest",
  "pine": "latest",
  "react-lazy-load-image-component": "latest",
  "react-lazyload": "latest",
  "recursive-readdir": "latest"
},
```

Diskussion