1. 사용한 JVM, 웹 서버, WAS, IDE

Frontend

* React.js → 17.0.2
* Redux → 4.1.2
* TypeScript → 4.6.2
* Mui → 5.5.2

Backend

* Python (3.7.9)
* Django (3.2.13)
  + PORT : 8443
* Server
  + Computer : AWS EC2 → Ubuntu 20.04 LTS
  + DB : MySQL → 8.0.28
  + WAS : Nginx → 1.18.0
* Game
  + Unity : 2020.3.30f1
  + Photon : Photon PUN Cloud(100CCU)

1. 빌드시 환경변수

* React - .env.production

VUE\_APP\_SERVER\_URL=https://k6e204.p.ssafy.io/

* React - .env.local

VUE\_APP\_SERVER\_URL=https://k6e204.p.ssafy.io/

[http://localhost:3000/](http://localhost:8081/)

* Frontend - Dockerfile

FROM node:16-alpine as build-stage

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npm run build

FROM nginx:stable-alpine as production-stage

COPY ./nginx.conf /etc/nginx/conf.d/default.conf

COPY --from=build-stage /app/build /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

* Backend – Dockerfile

FROM python:3.7.9

WORKDIR /usr/src/app

RUN mkdir wave\_file

RUN apt-get update

RUN apt-get --yes install libsndfile1

COPY requirements.txt ./

RUN pip3 install -r requirements.txt

COPY . .

EXPOSE 8433

CMD ["python3", "manage.py", "runserver", "0.0.0.0:8443"]

* Nginx : nginx.conf

server {

listen 80;

listen [::]:80;

# server\_name 도메인;

server\_name k6e204.p.ssafy.io;

access\_log /var/log/nginx/access.log;

error\_log /var/log/nginx/error.log;

location / {

return 301 https://$server\_name$request\_uri; # http 접속 시 https 로 자동 접속

}

}

server {

listen 443 ssl;

listen [::]:443 ssl;

client\_max\_body\_size 0;

# server\_name 도메인

server\_name k6e204.p.ssafy.io;

root /usr/share/nginx/html;

index index.html index.htm;

access\_log /var/log/nginx/access.log;

error\_log /var/log/nginx/error.log;

ssl\_certificate /var/www/html/fullchain.pem;

ssl\_certificate\_key /var/www/html/privkey.pem;

location / {

try\_files $uri $uri/ /index.html;

}

location /api {

proxy\_http\_version 1.1;

proxy\_pass http://k6e204.p.ssafy.io:8443;

charset utf-8;

}

location /admin {

proxy\_http\_version 1.1;

proxy\_pass http://k6e204.p.ssafy.io:8443;

charset utf-8;

}

location /google {

proxy\_http\_version 1.1;

proxy\_pass http://k6e204.p.ssafy.io:8443;

charset utf-8;

}

}

* Docker-compose.yml

version: "3"

services:

frontend:

image: frontend

build:

context: ./front/

dockerfile: Dockerfile

ports:

- '80:80' # - 띄어쓰기

- '443:443'

volumes:

- /home/ubuntu/docker-volume/ssl/:/var/www/html

container\_name: frontend

server:

image: backend

build:

context: ./server/

dockerfile: Dockerfile

ports:

- '8443:8433'

container\_name: backend

3. 배포시 특이 사항

* 유니티 배포는 유니티에서 WebGL로 빌드 후에 생긴 Build 파일들을 Frontend / public / build에 넣어준 후 빌드해야 한다.
* sndfiles 라이브러리는 ubuntu 환경에서 설치 불가능 하기 때문에 dokcer 컨테이너를 이용하여 설치한다.

1. DB 접속 정보

#datasource

* driver-class-name: com.mysql.cj.jdbc.Driver
* url: jdbc:mysql://k6E204.p.ssafy.io:3306/unique?seUnicode=true&characterEncoding=utf8&serverTimezone=Asia/Seoul&zeroDateTimeBehavior=convertToNull&rewriteBatchedStatements=trueusername=root
* username: root
* password: forest