포팅 메뉴얼

1. 개요

1) 프로젝트 사용 도구

- 이슈 관리: Jira
- 형상 관리: GitLab
- 커뮤니케이션: Notion, MatterMost
- UI/UX: Pigma

2) 프로젝트 개발 환경

- Frontend

- Visual Studio Code: 1.70.0
- Node.js: 18.12.1
- React: 18.2.0
- Typescript: 4.8.4

- Backend

- Visual Studio Code: 1.70.0
- Python: 3.9
- Django: 3.2.16

- Mobile

- React: 18.1.0
- React Native: 0.70.0
- Android
    - minimum SDK : 27
    - target SDK : 31
- iOS
    - minimum Version : 12.4
    - target SDK: 15.4

- DB

- MySQL: Ver 8.0.31
- MySQL Workbench: 8.0.30

- Server

- Ubuntu: 20.04

- Infra

- Docker 20.10.20
- Docker Compose 1.25.0
- Nginx 1.18.0
- Jenkins 2.361.2

2. 프로젝트 빌드

1) 프로젝트 빌드 방법

- Frontend

```
npm i
npm run bulid
```

- Backend

> 웹 프레임워크인 Django는 작동하기 위해 웹 서버가 필요합니다.
>
> 그리고 대부분의 웹 서버는 기본적으로 Python을 사용하지 않기 때문에,
>
> 커뮤니케이션이 이루어지도록 인터페이스 WSGI인 Gunicorn을 사용하였습니다.

```
python -m venv venv
source venv/Script/activate
pip insatall -r requirements.txt
python manage.py makemigrations
python manage.py migrate
gunicorn --bind 0.0.0.0:8000 config.wsgi:application
```

- App

```
./gradlew assembleRelease
[또는]
./gradlew bundleRelease
```

2) 프로젝트 환경 변수

- Frontend

- Frontend/.env

```
REACT_APP_KAKAO =
REACT_APP_BACK_HOST =
REACT_APP_FE_HOST =
```

- Backend

- BE/.env

```
DEBUG =
SECRET_KEY =
DATABASE_URL =
SQLITE_URL =
CACHE_URL =
REDIS_URL =
client_class =
Algorithm =
MYSQL_PASSWORD =
SMSServiceId =
SMSAccessKey =
SMSSecretKey =
FromUser =
AWS_ACCESS_KEY_ID =
AWS_SECRET_ACCESS_KEY =
MYSQL_PASSWORD_2 =
```

- BE/firebase.json

```
{
  "type": ,
  "project_id": ,
  "private_key_id": ,
  "private_key": ,
  "client_email": ,
  "client_id": ,
  "auth_uri": ,
  "token_uri": ,
  "auth_provider_x509_cert_url": ,
  "client_x509_cert_url":
}
```

3) Dockerfile

- Frontend

```
FROM node:16.15.0 as build-stage
WORKDIR /var/jenkins_home/workspace/Zoa/frontend
COPY package*.json ./
RUN npm install
COPY . .
COPY ./Switch/Switch.d.ts ./node_modules/@mui/material/Switch/Switch.d.ts
RUN npm run build
FROM nginx:stable-alpine as production-stage
COPY --from=build-stage /var/jenkins_home/workspace/Zoa/frontend/build
/usr/share/nginx/html
COPY --from=build-stage
/var/jenkins_home/workspace/Zoa/frontend/nginx/default.conf
/etc/nginx/conf.d/default.conf
EXPOSE 80
CMD ["nginx", "-g","daemon off;"]
```

- Backend

```
FROM python:3.9
ENV PYTHONUNBUFFERED 1
RUN apt-get -y update
RUN apt-get -y install vim
COPY ./ /home/BE/
WORKDIR /home/BE/
RUN pip3 install --upgrade pip && pip3 install -r requirements.txt
RUN pip install gunicorn
RUN python3 manage.py test --keepdb --settings=apiserver.settings.develop
EXPOSE 8000
CMD ["bash", "-c", "python manage.py collectstatic --noinput --
settings=apiserver.settings.develop && python manage.py migrate --
settings=apiserver.settings.develop &&gunicorn apiserver.wsgi --env
DJANGO_SETTINGS_MODULE=apiserver.settings.develop --bind 0.0.0.0:8000"]
```

3. 프로젝트 배포

1) Certbot

- Certbot 설치

```
sudo apt-get update
sudo apt-get install certbot python3-certbot-nginx
```

2) SSL

- Certbot 사용

```
[SSL 설정]
sudo certbot --nginx -d k7b103.p.ssafy.io

[갱신 테스트]
sudo certbot renew --dry-run

[인증서 만료일 확인]
certbot ceritificates
```

3) NGINX

- Port

이름	Port
Frontend	3000
MySQL	3306
Backend	8000
Jenkins	9090

- default

```
upstream backend {
   server k7b103.p.ssafy.io:8000;
}
upstream frontend {
   server k7b103.p.ssafy.io:3000;
}


server {
  listen 80;
  server_name k7b103.p.ssafy.io;
  location / {
     return 301 https://$host$request_uri;
  }
}

server {
   listen 443 ssl;
   server_name k7b103.p.ssafy.io;
   access_log /var/log/nginx/access.log;

   ssl_certificate /etc/letsencrypt/live/k7b103.p.ssafy.io/fullchain.pem;
   ssl_certificate_key /etc/letsencrypt/live/k7b103.p.ssafy.io/privkey.pem;

   ssl_protocols TLSv1 TLSv1.1 TLSv1.2 SSLv3;
   ssl_ciphers ALL;
   location / {
     client_max_body_size 20M;
```

```
    proxy_pass http://frontend;
    proxy_redirect off;
    charset utf-8;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Nginx-Proxy true;
  }
  location /api {
    client_max_body_size 20M;
    proxy_pass http://backend;
    rewrite ^/(.*)$ /$1 break;
    proxy_redirect off;
    charset utf-8;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Nginx-Proxy true;
  }
  location /api/swagger {
     auth_basic "Security";
     auth_basic_user_file /etc/nginx/.htpasswd;
     client_max_body_size 20M;
     proxy_pass http://backend;
     rewrite ^/(.*)$ /$1 break;
     proxy_redirect off;
     charset utf-8;
     proxy_http_version 1.1;
     proxy_set_header Upgrade $http_upgrade;
     proxy_set_header Host $http_host;
     proxy_set_header X-Real-IP $remote_addr;
     proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
     proxy_set_header X-Forwarded-Proto $scheme;
     proxy_set_header X-Nginx-Proxy true;
   }
  location /assets/ {
    alias /home/static/staticfiles/;
  }
}
```

4) Docker

- Docker Install

```
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io docker-compose
```

- docker-compose.yml

```
services:
    jenkins:
        image: jenkins/jenkins:lts
        container_name: jenkins
        volumes:
            - /var/run/docker.sock:/var/run/docker.sock
            - /jenkins:/var/jenkins_home
        ports:
            - "9090:8080"
        privileged: true
        user: root
```

- Jenkins Container Check

```
# 컨테이너를 생성
sudo docker-compose up -d
# Jenkins 컨테이너가 올라가 있는 것을 확인
sudo docker ps
```

- Jenkins Administrator password Check

```
docker exec <CONTAINER_NAME> cat /var/jenkins_home/secrets/initialAdminPassword
[또는]
docker logs jenkins-docker
```

5) Jenkins

- 빌드

```
docker image prune -a --force
mkdir -p /var/jenkins_home/images_tar
cp /var/jenkins_home/fe-env/.env /var/jenkins_home/workspace/Zoa/frontend/
cd /var/jenkins_home/workspace/Zoa/frontend/
docker build -t client .
docker save client > /var/jenkins_home/images_tar/client.tar

cp /var/jenkins_home/be-env/.env /var/jenkins_home/workspace/Zoa/BE/
cp /var/jenkins_home/be-env/firebase.json /var/jenkins_home/workspace/Zoa/BE/
cd /var/jenkins_home/workspace/Zoa/BE/
docker build -t server .
docker save server > /var/jenkins_home/images_tar/server.tar
```

- 빌드 후 조치

```
sudo docker load < /jenkins/images_tar/client.tar
sudo docker load < /jenkins/images_tar/server.tar

if (sudo docker ps | grep "client"); then sudo docker stop client; fi
if (sudo docker ps | grep "server"); then sudo docker stop server; fi

sudo docker run -it -d --rm -p 3000:3000 --name client client
echo "Run client"
sudo docker run -it -d --rm -p 8000:8000 -v static-
volume:/var/jenkins_home/workspace/Zoa/BE/staticfiles --name server server
echo "Run server"
```

4. DB

- Ubuntu 패키지 정보 업데이트

```
sudo apt update
```

- MySQL 설치

```
sudo apt install mysql-server
```

- MySQL 실행

```
sudo systemctl start mysql.service
```

- MySQL 접속

```
mysql -u root -p
```

- MySQL 비밀번호 설정

```
alter user 'root'@'localhost' identified with mysql_native_password by '비밀번호
설정';
FLUSH PRIVILEGES;
exit
```

- MySQL User 추가

```
mysql> CREATE USER '계정이름'@'%' IDENTIFIED BY '비밀번호';
mysql> GRANT ALL PRIVILEGES ON . TO '계정이름'@'%' WITH GRANT OPTION;
mysql> FLUSH PRIVILEGES;
```

- MySQL 외부 접속 설정
    - cd /etc/mysql/mysql.conf.d

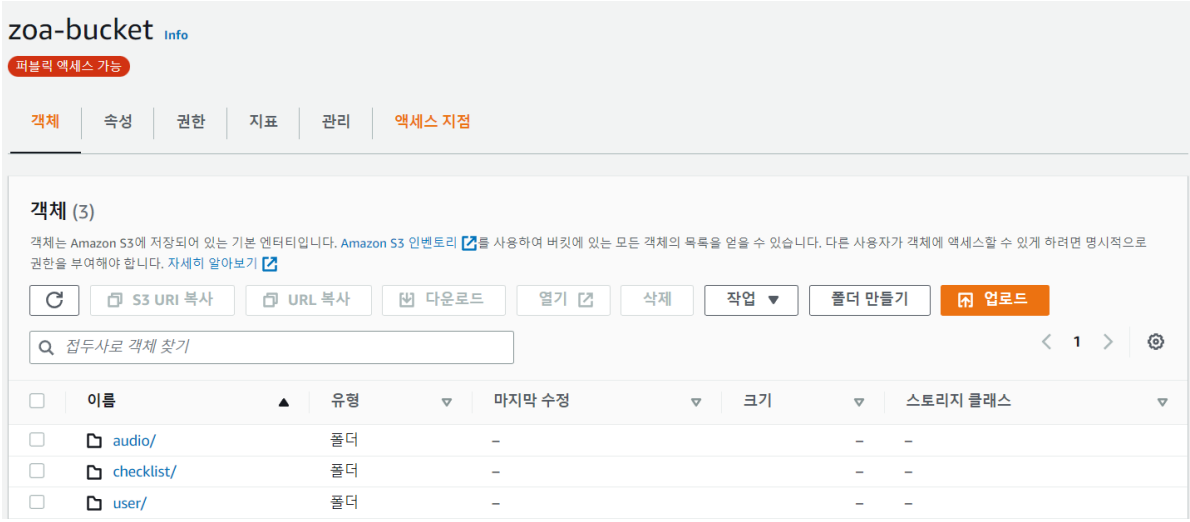
```
bind-address = 0.0.0.0 으로 수정
```

- MySQL 재실행

```
service mysql restart
```

5. 외부 서비스

1) Amazon S3



- S3 설정

```
AWS_ACCESS_KEY_ID = env('AWS_ACCESS_KEY_ID')
AWS_SECRET_ACCESS_KEY = env('AWS_SECRET_ACCESS_KEY')
AWS_REGION = 'ap-northeast-2'

AWS_STORAGE_BUCKET_NAME = 'zoa-bucket' # 설정한 버킷 이름
AWS_S3_CUSTOM_DOMAIN = '%s.s3.%s.amazonaws.com' %
(AWS_STORAGE_BUCKET_NAME,AWS_REGION)
AWS_S3_OBJECT_PARAMETERS = {
    'CacheControl': 'max-age=86400',
}
DEFAULT_FILE_STORAGE = 'storages.backends.s3boto3.S3Boto3Storage'
```

2) KAKAO Login

- KAKAO Login API URL

```
GET /oauth/authorize?
client_id=${REST_API_KEY}&redirect_uri=${REDIRECT_URI}&response_type=code
```

3) FCM(Firebase Cloud Messaging)

> Mobile push alarm에 사용하였습니다.

- Firebase 설정

```
import json
from django.core.exceptions import ImproperlyConfigured

secret_json = os.path.join(BASE_DIR,'firebase.json')
with open(secret_json) as f :
    secrets = json.loads(f.read())

def get_secret(setting,secrets=secrets) :
    try :
        return secrets[setting]
    except KeyError :
        error_msg = f"Set the {setting} environment variable"
        raise ImproperlyConfigured(error_msg)
service_account_key = {
    "type" : get_secret('type'),
    "project_id" : get_secret('project_id'),
    "private_key_id" : get_secret('private_key_id'),
    "private_key" : get_secret('private_key'),
    "client_email" : get_secret('client_email'),
    "client_id" : get_secret('client_id'),
    "auth_uri" : get_secret('auth_uri'),
    "token_uri" : get_secret('token_uri'),
    "auth_provider_x509_cert_url" : get_secret('auth_provider_x509_cert_url'),
    "client_x509_cert_url" : get_secret('client_x509_cert_url'),

}
```

4) Naver Cloud SMS

> 회원 가입 시 핸드폰 인증에 사용하였습니다.

- API URL

```
https://sens.apigw.ntruss.com/sms/v2
```

- 요청 URL

```
GET https://sens.apigw.ntruss.com/sms/v2/services/{serviceId}/messages?
requestId=
```

- SMS 설정

```
SMS_SECRET = {
    'SMSServiceId' : env('SMSServiceId'),
    'SMSAccessKey' : env('SMSAccessKey'),
    'SMSSecretKey' : env('SMSSecretKey'),
    'FromUser' : env('FromUser'),
}
```