포팅메뉴얼

1.개요

1-1. 프로젝트 개요

5년 가까이 싸피에서 1만명의 개발자들이 배출되었습니다. 싸피 출신 개발자들과의 커뮤니티가 어느 때보다 더욱 더 절실한 지금입니다.

SSAFYING은 이전기수와 현재기수를 통합하는 지속가능한 개발자 SNS 입니다. SSAFYING에서 당신의 고민을 털어놓고, 자유롭게 활동해보세요!

SSAFYING, 우리는 여전히 싸피중!

1-2. 개발 환경

Backend

- **Java** : Oracle Open JDK 17.0.9
- **Spring Boot** : 3.2.1
- **JPA** : hibernate-core-6.4.1
- **DB:** MySQL 8.0
- **IntelliJ** : 2023. 3

Frontend

- **Node.js** : 20.10.0
- **TypeScript** : 4.9.5
- **React** : 18.2.9
- **Redux** : 9.1.0
- **Axios** : 1.6.7
- **Vscode** : 1.85.1
- **Firebase** : 10.8.0

Server

- **AWS EC2**
  - **CPU** : 4CPU
  - **RAM** : 16.8 GB
  - **OS** : Ubuntu

Service

- **NginX :** 1.18.0
- **Jenkins** : 2.442
- **Docker** : 25.0.1

1-3. 프로젝트 사용도구

- 이슈 관리 : JIRA
- 형상 관리 : Gitlab
- 코드 리뷰: gerrit
- 커뮤니케이션 : Notion, Mattermost

- 디자인 : Figma, Figjam
- UCC : (모바비) VLLO
- CI/CD : Jenkins

1-4. 외부 서비스

- Firebase Storage
- Saramin OpenAPI
- Tesseract.js
- Tmap OpenAPI

1-5. GitIgnore 정보

- React : .env (최상단 위치)
- Spring : application.yml

---

2. 빌드

2-1. 환경변수

Spring

- application.yml

```
# 리버스 프록시 설정을 위한 context path 설정 추가, 포트 변경
server:
  servlet:
    context-path: /api
  port: 8081

spring:
  profiles:
    active: ${profile}

logging:
  level:
    org.hibernate.SQL: debug
    org.hibernate.type: trace

#jwt - 이슈 발급자, 비밀키 설정
jwt:
  issuer: ssafying@ssafy.com
  secret_key: b210-ssafying-teamjejudo
```

- application-local.yml

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/ssafying?autoReconnect=true
    username: ssafy
    password: 1234

  jpa:
    hibernate:
      ddl-auto: create
    defer-datasource-initialization: true

  sql:
    init:
      data-locations: classpath*:db/data.sql
      mode: always
      platform: mysql
```

- application-prod.yml

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: ${MYSQL_DATABASE_URL}
    username: ${MYSQL_DATABASE_USERNAME}
    password: ${MYSQL_DATABASE_PASSWORD}

  jpa:
    hibernate:
```

```
      ddl-auto: none
    defer-datasource-initialization: true
```

React

- .env : 최상단 위치

```
REACT_APP_HOME_URL=http://localhost:8081
REACT_APP_SARAMIN_BASE_URL=https://oapi.saramin.co.kr
REACT_APP_SARAMIN_API_KEY=YF34Es7BZsjwQnoKoYmIJutRLKdarXEhYnHuscdTvGTHlcH5YlPHS
REACT_APP_TMAP_API_KEY=h6gm8ietsB3dJpyfSAFJL6CbcMx0lSoM59x7MhGp
GENERATE_SOURCEMAP=false

REACT_APP_FIREBASE_API_KEY=AIzaSyCzfETN0Nd056m3x4NXiyb91tuDpRq8dzo
REACT_APP_FIREBASE_AUTH_DOMAIN=ssafying-5667d.firebaseapp.com
REACT_APP_FIREBASE_PROJECT_ID=ssafying-5667d
REACT_APP_FIREBASE_STORAGE_BUCKET=ssafying-5667d.appspot.com
REACT_APP_FIREBASE_MESSAGING_SENDER_ID=233678574280
REACT_APP_FIREBASE_APP_ID=1:233678574280:web:5da2e0aca7d5c3f3a1cbfd
REACT_APP_FIREBASE_MEASUREMENT_ID=G-GYBJ8XD448
```

2-2. 빌드하기

Frontend

- `npm i`
- `npm start` / `npm run build`

Backend

- build.gradle 실행

2-3. 외부 서비스 이용방법

- **Firebase Storage**
  1. Firebase에서 SSAFYING으로 프로젝트 생성 후, Cloud Storage 생성하기
  2. `npm install firebase` 설치하기
  3. storage를 만든 이후 제공되는 환경변수를 .env 파일에 저장하기
  4. Firebase와 storage 변수 초기화 및 Firebase 앱 객체 만들기

```
// firebase/firebase.ts

import { initializeApp } from 'firebase/app';
import { getStorage } from 'firebase/storage';

const firebaseConfig = {
  apiKey: process.env.NEXT_PUBLIC_FIREBASE_API_KEY,
  authDomain: process.env.NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN,
  projectId: process.env.NEXT_PUBLIC_FIREBASE_PROJECDT_ID,
  storageBucket: process.env.NEXT_PUBLIC_FIREBASE_STORAGE_BUCKET,
  messagingSenderId: process.env.NEXT_PUBLIC_FIREBASE_MESSAGING_SENDER_ID,
  appId: process.env.NEXT_PUBLIC_FIREBASE_APP_ID,
};

//* firebaseConfig 정보로 firebase 시작
const fbaseApp = initializeApp(firebaseConfig);

//* firebase의 storage 인스턴스를 변수에 저장
export const fstorage = getStorage(fbaseApp);
```

  5. 이미지 업로드 및 다운로드를 하는 파일에서 참조 만들기 (파일 업로드 및 삭제시 필요)
  6. 이미지 파일의 URL을 얻는 방식으로 파일 가져오기.

- **Saramin OpenAPI**
  1. 사람인 open api 홈페이지에서 이용 신청 및 승인 과정을 거쳐 API key를 얻는다
  2. .env 파일에 API_KEY 를 등록한다.

- **Tesseract.js**
  1. `npm install tesseract.js` 라이브러리 설치
  2. 테스트 코드로 코드 실행해본다 (공식 사이트에서 제공) → `node imgToText.js` 로 코드 실행

```
// imgToText.js

import { createWorker } from 'tesseract.js';
```

```
const worker = await createWorker({
  logger: (m) => console.log(m),
});

(async () => {
  await worker.loadLanguage('eng'); //인식 언어
  await worker.initialize('eng'); //추출 언어
  const {
    data: { text },
  } = await worker.recognize('이미지 링크');
  console.log(text);
  await worker.terminate();
})();
```

3. 이미지 url을 인식하여 텍스트 추출 결과를 터미널에서 확인 가능하다.

- **Tmap OpenAPI**

  (수정 중)

  1. env 파일에 API 키 저장
  2. index.html에 키 작성

---

3. 배포하기

Linux기준

git 설치

```
sudo apt-get install git

sudo apt install git

# 버전확인
git --version
```

clone받기

docker 설치 - 공식문서 참조

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

3-1 도커 이미지

Frontend

1. 프론트 루트디렉토리로 이동
2. `docker build -t frontend:latest .`

Backend

1. 백엔드 루트 디렉토리로 이동
2. `docker build -t backend:latest .`

docker-compose.yml

```
version: '3'
services:
  nginx:
    image: nginx
    container_name: nginx
    ports:
      - 80:80
```

```
      - 443:443
    networks:
      - ssafying_net
    volumes:
           # 인증정보
      - /home/nginx/nginx.conf:/etc/nginx/nginx.conf
      - /home/nginx/sites-available/default:/etc/nginx/sites-available/default
      - /home/nginx/sites-available/default:/etc/nginx/sites-enabled/default
      - /var/www/i10b210.p.ssafy.io:/var/www/i10b210.p.ssafy.io
      - /var/www/ssl:/var/www/ssl
    user: root
        mysql:
        image: mysql:latest
        container_name: mysql
        ports:
          - "3306:3306"
        volumes:
          - /home/ubuntu/mysql-data:/var/lib/mysql
        environment:
          MYSQL_ROOT_PASSWORD: 1234
          MYSQL_DATABASE: ssafying
        command:
          - --character-set-server=utf8mb4
          - --collation-server=utf8mb4_unicode_ci
        networks:
          - ssafying_net
           user: root
  backend:
    image: backend:latest
    container_name: backend
    ports:
      - 8081:8081
    environment:
      TZ: Asia/Seoul
      profile: prod
      MYSQL_DATABASE_URL: ${MYSQL_DATABASE_URL}
      MYSQL_DATABASE_USERNAME: ${MYSQL_DATABASE_USERNAME}
      MYSQL_DATABASE_PASSWORD: ${MYSQL_DATABASE_PASSWORD}
  networks:
    - ssafying_net
  frontend:
    image: frontend:latest
    container_name: frontend
    ports:
      - 3000:3000
    networks:
      - ssafying_net

networks:
  ssafying_net:
    driver: bridge
```

작성 후 같은 경로에 .env 생성

```
profile: prod
MYSQL_DATABASE_URL: [db url]
MYSQL_DATABASE_USERNAME: [db username]
MYSQL_DATABASE_PASSWORD: [db password]
```

이후 `docker compose up -d` 명령어로 실행

환경변수의 적용 확인을 위해 `docker compose convert` 실행

- nginx 설정

```
server {
        listen 443 ssl default_server;
        listen [::]:443 ssl default_server;
        ssl_certificate /var/www/ssl/certificate.crt;
        ssl_certificate_key /var/www/ssl/private.key;

        root /var/www/i10b210.p.ssafy.io;

        index index.html index.htm index.nginx-debian.html;

        server_name i10b210.p.ssafy.io www.i10b210.p.ssafy.io;

        location /api {
                proxy_pass [백엔드서버]
                                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;
                # First attempt to serve request as file, then
}

        location / {
                # / 경로로 들어오는 요청을 3000번 포트로 프록시
                proxy_pass [백엔드서버]
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection 'upgrade';
                proxy_set_header Host $host;
                proxy_cache_bypass $http_upgrade;
        }

                location /ws {
                proxy_pass [웹소켓 구성한 백엔드 서버];
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection "upgrade";
                proxy_set_header Host $host;
                proxy_set_header Origin "";
        }

        location ~/.well-known/pki-validation {
                default_type "text/plain";
                index 4A6D607261FC8C902CC677CCA6D9CA6C.txt;
        }

server {
        listen 80;
        listen [::]:80;

        server_name i10b210.p.ssafy.io www.i10b210.p.ssafy.io;

        root /var/www/i10b210.p.ssafy.io;

        # --- http로 들어온 요청도 https로 매핑 ---
        return 301 https://$host$request_uri;
}
```