

**E103 : DEVELOPER MAKER**

삼성SW청년아카데미 부울경캠퍼스 7기

자율프로젝트(7주: 2022.10.11 ~ 2022.11.21)

**포팅 매뉴얼**

담당 컨설턴트 : 김신일

전현우(팀장), 남한솔, 박정현, 박제학, 손효재

**목차**

1. 프로젝트 개요 -------------------------------------------- 2p

2. 프로젝트 기술 스택 -------------------------------------- 3p

3. 주요 환경 변수 ------------------------------------------- 4p

4. 도커 이미지 빌드 및 실행 ------------------------------- 5p

5. Jenkins 쉘 스크립트 -------------------------------------- 6p

6. Docker 파일 ----------------------------------------------- 7p

7. 배포 특이사항 --------------------------------------------- 8p

8. 외부 서비스 ----------------------------------------------- 11p

9. 시연 시나리오 --------------------------------------------- 11p

**1. 프로젝트 개요**

프로그래머를 준비하는 입장에서 CS, 알고리즘, 코딩테스트 등은 모두 꾸준히 준비해야 할 필요가 있습니다. 그러다 보니 숙제와 같이 지겹게 느껴지고 잘하지 못하는 경우가 많습니다.

Developer Maker는 스토리 진행을 기반으로 주인공 캐릭터를 성장시켜 CS 학습, 코딩테스트, 면접 등 취업에 필요한 역량을 강화해 나가는 취업 시뮬레이션 게임입니다.

Gamification(게임화)를 적용하여 스토리를 통해 흥미를 높이고, 스토리를 진행하기 위해 필요한 학습을 진행하는 시스템으로 좀더 쉽고 재밌게 취업 역량을 강화할 수 있도록 도와주는 것을 목적으로 하는 서비스입니다.

다양한 등장인물을 기반으로 흥미있는 스토리를 진행할 수 있으며, 웹에서 제공하는 코딩테스트 환경과 AI 면접을 통해 다양한 방법으로 취업 과정을 제공합니다. 또한, 약 250개의 CS문제와 100개의 CS개념정리를 제공하여 역량을 강화할 수 있습니다.

**2. 프로젝트 기술 스택**

가. 이슈 관리: Jira

나. 형상 관리: Gitlab

다. 커뮤니케이션: Notion, Mattermost

라. 개발 환경

1) OS: Windows 10

2) IDE

가) IntelliJ 2021.3.2

나) Visual Studio Code 1.70.1

다) UI/UX: Figma

3) Database:

가) MySQL 8.0.30

나) Redis 7.0.4

4) Server: AWS EC2 Ubuntu 20.04 LTS

5) Dev-Ops

가) Docker 20.10.18

나) Jenkins 2.60.3

마. 상세 사용

1) Frontend

가) HTML5, CSS3, JavaScript(ES6)

나) React 17.0.2, Redux 4.2.0

다) Node.js 16.14.0

라) React-wordcloud 1.2.7

2) Backend

가) Spring boot 2.7.3

나) Open JDK 8

다) Gradle 7.5

라) Querydsl 5.0

마) Spark Project core 3.3.0

바) Komoran 3.3.4

**3. 주요 환경변수**

# db  
spring.datasource.url=[DB 주소]  
spring.datasource.username=[DB 호스트명]  
spring.datasource.password=[DB 비밀번호]

spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect spring.jpa.hibernate.ddl-auto=update   
spring.jpa.properties.hibernate.format\_sql=true

logging.level.org.hibernate.SQL=debug

# jwt  
jwt.header=Authorization jwt.secret=[jwt 시크릿 키]   
jwt.token-validity-in-seconds=86400

# s3  
cloud.aws.stack.auto=false   
cloud.aws.region.static=[AWS region]  
cloud.aws.credentials.access-key=[발급받은 엑세스 키]  
cloud.aws.credentials.secret-key=[발급받은 시크릿 키]  
cloud.aws.s3.bucket=[버킷명]  
logging.level.com.amazonaws.util.EC2MetadataUtils=error

# multipartfile  
spring.servlet.multipart.max-file-size=20MB   
spring.servlet.multipart.max-request-size=25MB

server.servlet.context-path=/api   
server.error.include-stacktrace=never

# redis  
spring.redis.host=[레디스 호스트 주소]  
spring.redis.port=[레디스 포트 번호]  
spring.redis.password=[레디스 비밀번호]

# emotion token  
properties.file.luxand-token=[luxand-api 토큰]

# X-Rapid API  
properties.file.HOST\_JAVA=code-compiler.p.rapidapi.com  
properties.file.KEY\_JAVA=[rapid api 토큰]

# ssl  
security.require-ssl=true  
server.ssl.key-store=classpath:spring\_key.p12  
server.ssl.key-store-type=PKCS12  
server.ssl.key-store-password=[ssl 인증서 비밀번호]  
server.ssl.enabled=true

**4. 도커 이미지 빌드 및 실행**

**가) Docker**

$ sudo apt-get remove docker docker-engine [docker.io](http://docker.io/) containerd runc

$ sudo apt-get update

$ sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

$ sudo apt-key fingerprint 0EBFCD88

$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

$ sudo apt-get update

$ sudo apt-get install docker-ce docker-ce-cli containerd.io

$ sudo docker --version

**나) mysql**

$ sudo docker pull mysql

$ sudo docker images

$ sudo ufw allow 3306

$ sudo docker run -d --name mysql -e MYSQL\_ROOT\_PASSWORD=패스워드 -p 3306:3306 mysql

$ sudo docker ps

**다) Jenkins**

$ sudo docker pull jenkins/jenkins:lts

$ sudo docker

$ sudo ufw allow

$ sudo docker run --name jenkins -d -p 8080:8080 -p 50000:50000 -v /home/jenkins:/var/jenkins\_home -v /var/run/docker.sock:/var/run/docker.sock -e TZ=Asia/Seoul -u root jenkins/jenkins:lts

$ sudo docker ps

$ sudo docker logs jenkins

**5. Jenkins 쉘 스크립트**

**가) backend**

$ cd backend

$ docker build -t backend .

$ docker ps -q --filter "name=backend" | grep -q . && docker stop backend && docker rm backend | true

$ docker run -p 8081:8080 -d -e TZ=Asia/Seoul --name=backend backend

$ docker rmi -f $(docker images -f "dangling=true" -q) || true

**나) frontend**

$ cd frontend

$ docker build -t frontend .

$ docker ps -q --filter "name=frontend" | grep -q . && docker stop frontend && docker rm frontend | true docker run -d -p 80:80 -p 443:443 -v /home/ubuntu/certbot/conf:/etc/letsencrypt/ -v /home/ubuntu/certbot/www:/var/www/certbot --name frontend frontend

$ docker rmi -f $(docker images -f "dangling=true" -q) || true

**6. Docker 파일**

**가) backend**

FROM openjdk:8-jdk-slim as builder

COPY gradlew .  
COPY gradle gradle  
COPY build.gradle .  
COPY settings.gradle .  
COPY src src  
RUN chmod +x ./gradlew  
RUN ./gradlew bootJar

FROM openjdk:8-jdk-slim  
COPY --from=builder build/libs/\*.jar app.jar  
ENTRYPOINT ["java","-jar","-Dspring.profiles.active=gcp","/app.jar"]  
EXPOSE 8081

**나) frontend**

# build stage  
FROM node:lts-alpine as build-stage  
WORKDIR /app  
COPY package\*.json ./  
RUN yarn install  
COPY . .  
RUN npm run build

# production stage   
FROM nginx:stable-alpine as production-stage   
COPY --from=build-stage /app/build /usr/share/nginx/html   
EXPOSE 80   
CMD ["nginx", "-g", "daemon off;"]

**7. 배포 특이사항**

**가) Spring boot에 SSL 적용**

1) Certbot container 생성 및 인증서 발급

|  |
| --- |
| sudo mkdir certbotcd certbotsudo mkdir conf www logs​sudo docker pull certbot/certbotsudo docker run -it --rm --name certbot -p 80:80 \-v "/home/ubuntu/certbot/conf:/etc/letsencrypt" \-v "/home/ubuntu/certbot/log:/var/log/letsencrypt" \-v "/home/ubuntu/certbot/www:/var/www/certbot" \certbot/certbot certonly |

2) SSL인증서를 spring boot에서 필요한 형식(PKCS12)로 변환

openssl pkcs12 -export -in fullchain.pem -inkey privkey.pem-out keystore.p12 -name tomcat -CAfile chain.pem -caname root

3) keystore p.12 파일을 /src/main/resources에 이동

**나) nginx SSL 설정**

1) /home/ubuntu/nginx/conf/default.conf

|  |
| --- |
| server {listen 80;server\_name k7e103.p.ssafy.io;location / {return 301 https://$host$request\_uri;}}server {listen 443 ssl;server\_name k7e103.p.ssafy.io;access\_log /var/log/nginx/access.log;error\_log /var/log/nginx/error.log;ssl\_certificate /etc/letsencrypt/live/k7e103.p.ssafy.io/fullchain.pem;ssl\_certificate\_key /etc/letsencrypt/live/k7e103.p.ssafy.io/privkey.pem;ssl\_protocols TLSv1 TLSv1.1 TLSv1.2 SSLv3;ssl\_ciphers ALL;location / {root /usr/share/nginx/html;index index.html index.htmproxy\_redirect off;charset utf-8;try\_files $uri $uri/ /index.html;proxy\_http\_version 1.1;proxy\_set\_header Upgrade $http\_upgrade;proxy\_set\_header Connection "upgrade";proxy\_set\_header Host $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;} } |

**8. 외부 서비스**

가) [카카오 로그인 기능](https://developers.kakao.com/docs/latest/ko/kakaologin/common)

나) [네이버 로그인 기능](https://developers.naver.com/products/login/api/api.md)

다) [AWS S3](https://aws.amazon.com/ko/s3/)

라) [카카오 맵](https://apis.map.kakao.com/)

**9. 시연 시나리오**













