빌드 및 배포 정리

서울 6반 6조

1) 프로젝트 기술 버전(툴, 버전 O)

개발 환경

Backend

- Java : jdk11 (11.0.17)
- gradle : 7.6
- Intellij : 2022.3.2
- MySQL : 5.7.39

Fronted

- Vue.js : 3.2.45
- node.js : 18.13.0
- VS Code : 1.74.3

AWS EC2

- ubuntu : 20.04.5
- Docker : 20.10.12
- Nginx : 1.18.0

Web RTC

- openvidu : 2.25.0

OS

- Windows 10

Cooperation & Communication

- Gitlab
- Jira
- MatterMost
- Notion
- Discord

2) 빌드 특이사항

빌드 방식:

Fronted

```
/frontend
npm i
npm run serve
```

Backend

/Backend/Homezakaya

./gradlew clean bootJar 실행

java -jar -Dspring.profiles.active=local build/libs/*.jar 혹은 루트에서 ./gradlew bootRun

3) 배포 특이사항, 방법 정리

Openvidu on-promise 배포

- root 권한 설정

sudo su

- openvidu 설치 위치

cd /opt

- openvidu 설치

curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh | bash

- 설치된 openvidu 경로

$ cd openvidu

- 통신 환경 설정

$ nano .env

OpenVidu configuration

도메인 또는 퍼블릭 IP 주소

DOMAIN_OR_PUBLIC_IP=i5a608.p.ssafy.io

오픈비두 서버와 통신을 위한 시크릿

OPENVIDU_SECRET=HOMEDONG

Certificate type

CERTIFICATE_TYPE=letsencrypt

인증서 타입이 letsencrypt 일 경우 이메일 설정

LETSENCRYPT_EMAIL=user@example.com HTTP_PORT=8442 HTTPS_PORT=8443

- openvidu 실행(ctrl + c : 백그라운드 실행)

$ ./openvidu start

$ sudo su

openvidu 설치 위치

cd /opt

openvidu 설치

curl <https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh> | bash

설치된 openvidu 경로

$ cd openvidu

통신 환경 설정

$ nano .env

```
# OpenVidu configuration
# ----------------------
# 도메인 또는 퍼블릭 IP 주소
DOMAIN_OR_PUBLIC_IP=i5a608.p.ssafy.io

# 오픈비두 서버와 통신을 위한 시크릿
OPENVIDU_SECRET=HOMEDONG

# Certificate type
CERTIFICATE_TYPE=letsencrypt

# 인증서 타입이 letsencrypt 일 경우 이메일 설정
LETSENCRYPT_EMAIL=user@example.com
HTTP_PORT=8442
HTTPS_PORT=8443

# openvidu 실행(ctrl + c : 백그라운드 실행)

$ ./openvidu start
```

프론트엔드 빌드 및 배포

- Dockerfile

```
# Dockerfile

FROM nginx:stable-alpine

WORKDIR /app

RUN mkdir ./dist

ADD ./dist ./dist

RUN rm /etc/nginx/conf.d/default.conf

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

EXPOSE 3000

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

- 배포

```
# 프론트엔드 배포 /fronted

rm -rf /dist
npm i
npm run build
docker stop vue
docker rm vue
docker build -t vue .
docker run -d -p 3000:3000 --name vue vue
```

백엔드 빌드 및 배포

- Dockerfile

```
FROM openjdk:11-jdk-slim
ARG JAR_FILE=build/libs/Homezakaya-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE} app.jar
ENV JAVA_OPTS=""
ENTRYPOINT ["java","-jar","/app.jar"]
```

- 배포

```
# 프론트엔드 배포 /backend/Homezakaya
docker stop spring
docker rm spring
gradle clean build
docker build -t spring .
docker run -d -p 8081:8081 --name spring spring
```

MYSQL 배포(Docker)

```
sudo docker pull mysql:5.7.39

sudo docker images

sudo docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=root
-d -p 3306:3306 mysql:5.7.39
```

```
sudo docker exec -it mysql-container bash

mysql -u root -p

Enter password : root

create user '계정이름'@'localhost' identified by '비밀번호';

grant all privileges on *.* to '사용자'@'localhost' identified by '비밀번호';

flush privileges;

show grants for'ssafy'@'%';

#Ctrl + P + Q
```

4) ERD, properties

Gradle 의존성:

```
plugins { id 'java' id 'org.springframework.boot' version '2.7.8' id 'io.spring.dependency-management' version '1.0.15.RELEASE'}group = 'com.ssafy'version = '0.0.1-SNAPSHOT'sourceCompatibility = '11'configurations { compileOnly { extendsFrom annotationProcessor }}repositories { mavenCentral()}dependencies { implementation 'org.springframework.boot:spring-boot-starter-thymeleaf' implementation 'org.springframework.boot:spring-boot-starter-web' implementation 'org.mybatis.spring.boot:mybatis-spring-boot-starter:2.2.0' compileOnly 'org.projectlombok:lombok' developmentOnly 'org.springframework.boot:spring-boot-devtools' annotationProcessor 'org.projectlombok:lombok' testImplementation 'org.springframework.boot:spring-boot-starter-test' runtimeOnly 'mysql:mysql-connector-java:8.0.27' // jwt implementation 'io.jsonwebtoken:jjwt:0.9.1' // JWT implementation 'com.auth0:java-jwt:4.2.0' // com.sun.xml.bind implementation 'com.sun.xml.bind:jaxb-impl:4.0.1' implementation 'com.sun.xml.bind:jaxb-core:4.0.1' // javax.xml.bind implementation 'javax.xml.bind:jaxb-api:2.4.0-b180830.0359' // email 인증// https://mvnrepository.com/artifact/com.sun.activation/javax.activation implementation group: 'com.sun.activation', name: 'javax.activation', version: '1.2.0' implementation 'org.springframework.boot:spring-boot-starter-mail:2.7.0'}tasks.named('test') { useJUnitPlatform()}tasks { processResources { duplicatesStrategy = org.gradle.api.file.DuplicatesStrategy.INCLUDE }}
```

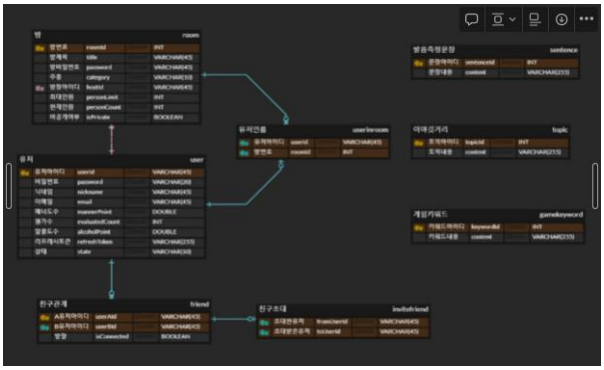
Spring boot Application properties:

```
spring.mvc.view.prefix=/WEB-
INF/views/spring.mvc.view.suffix=.jspspring.datasource.url=jdbc:mysql://i8a606.p.ssafy.io:3306/homezakaya?serverTimezone=UTCspring.datasource.username=ssafyspring.datasource.password=ssafym
ybatis.type-aliases-package=com.ssafy.Homezakaya.model.dtomybatis.mapper-locations=classpath:mapper/*.xmlspring.mvc.pathmatch.matching-strategy =
ANT_PATH_MATCHERserver.port=8081logging.level.xyz.quietjun=trace #
emailspring.mail.host=smtp.gmail.comspring.mail.port=465spring.mail.username=jsj.project.testspring.mail.password=nhidzjqjqbcungvmspring.mail.properties.debug=truespring.mail.properties.mail.s
mtp.auth=truespring.mail.properties.mail.smtp.ssl.enable= truespring.mail.properties.mail.smtp.starttls.enable=truespring.mail.properties.mail.smtp.ssl.trust=smtp.gmail.comserver.ssl.key-
store=classpath:keystore.p12server.ssl.key-store-type=PKCS12server.ssl.key-store-password=ssafy
```

Vue package.json:

```
{ "name": "front", "version": "0.0.0", "private": true, "scripts": { "dev": "vite", "build": "vite build", "preview": "vite preview" }, "dependencies": { "axios": "^1.2.6", "element-plus": "^2.2.28", "face-api.js": "^0.22.2", "jwt-decode": "^3.1.2", "nvm": "0.0.4", "openvidu-browser": "^2.25.0", "vue": "^3.2.45", "vue-router": "^4.1.6", "vuex": "^4.1.0", "vuex-persistedstate": "^4.1.0" }, "devDependencies": { "@vitejs/plugin-vue": "^4.0.0", "unplugin-auto-import": "^0.12.1", "unplugin-element-plus": "^0.4.1", "unplugin-vue-components": "^0.22.12", "vite": "^4.0.0" } }
```

ERD



API 명세서

/api/users

category	기능	URI	RequestParam	dao	service	method	request-body	상태코드	response
회원관리	회원 정보 등록	/api/users		insertUser	createUser	POST	{ "userId" : 유저 id(string), "password" : 비밀번호(string), "nickname" : 닉네임(string), "name" : 이름(string), "email" : 이메일(string), "alcoholPoint":알콜도수(double) }	201 : 성공	{ "message": "success" }
								400 : 중복된 id	{ "message": "중복된 id입니다." }
	아이디 중복 확인	/api/users/id/{userId}				GET		200 : 성공	{ "message": "success" }
								409 : 중복된 id	{ "message": "중복된 id입니다." }
	닉네임 중복 확인	/api/users/nickname/{nickname}				GET		200 : 성공	{ "message": "success" }
								400 : 중복된 id	{ "message": "중복된 닉네임입니다." }
	회원 정보 수정	/api/users		updateUser	modifyUser	PUT	{ "userId" : 아이디(string), "password" : 비밀번호(string), "nickname" : 닉네임(string), "email" : 이메일(string), "birthDate" : 생년월일(string), "alcoholPoint" : 알콜도수(double) }	200 : 성공	{ "message": "success" }

와이어프레임