목차

[I. 개요 2](#_Toc96072459)

[1. 프로젝트 개요 2](#_Toc96072460)

[2. 프로젝트 사용 도구 2](#_Toc96072461)

[3. 개발환경 2](#_Toc96072462)

[4. 외부 서비스 2](#_Toc96072463)

[5. Gitgnore 처리한 핵심 키들 3](#_Toc96072464)

[II. 빌드 3](#_Toc96072465)

[1. 환경변수 형태 3](#_Toc96072466)

[2. 빌드하기 5](#_Toc96072467)

[3. 배포하기 5](#_Toc96072468)

[4. 서비스 이용 방법 7](#_Toc96072469)

# 개요

# 프로젝트 개요

## 프로젝트 개요

## [신고자, 119, 병원을 하나로 잇는 응급통합 플랫폼]

## 회원 기능(질환, 기저질환, 보호자 연락처 관리)

## 신고자- 상담사- 응급구조대 화상서비스

## 신고자 url 수신 및 현재 위치 송신

## 신고자의 구급대원 출동 시 위치 확인

## 신고내역 자동 텍스트화 및 요약

## 시간마다 거리를 늘려가며 환자 증상을 및 이송요청을 병원에 일괄 송신

## ii.프로젝트 사용도구

1. 이슈 관리 : JIRA

1. 형상 관리 : Gitlab
2. 커뮤니케이션 : Notion, Mattermost
3. 디자인 : Figma
4. UCC : 모바비
5. CI/CD : Jenkins, Docker

## 개발환경

## VS Code : 1.64.2

## IntelliJ

## JVM : 17

## Node.js : 20

## SERVER : AWS EC2 Ubuntu 20.04.3 LTS

## DB : MariaDB 11.6, Redis

## 외부 서비스

## Google Cloud Storage :

## Google Speech to Text

## Open AI API

## Redis cloud

## T MAP

## kakaoMap

## coolsms

## 공공데이터포털 API

## GitIgnore 처리한 핵심 키들(application.properties에 기재)

## Google(Speech to Text, Cloud Storage)

## openAI

## 공공데이터포털(병원 목록, 소방서 목록, AED위치, 가용가능 응급실 목록, 의약품 목록)

## coolsms

# 빌드

## 환경변수 형태

## .env

VITE\_TMAP\_API\_KEY=xkWQ1aXWJg8ZQogcAzOhU5LfPJFMOQb488lZI3do

VITE\_KAKAOMAP\_API\_KEY=0836119f538d650272fc667d96b1a3e0

## Application.properties

spring.application.name=camference  
server.port=8080  
  
jwt.secret="V514VmypeqPoiljQ2HQlkAzh+Hsx2a3uwmHQztaEJeE="  
  
spring.coolsms.apiKey=NCSOI7OWCE13SXTP  
spring.coolsms.secretKey=I7RZORXWYWSFMPXCLGXQGVRGWW4GUMVE  
  
spring.datasource.driver-class-name= org.mariadb.jdbc.Driver  
  
  
spring.datasource.url=jdbc:mariadb://stg-yswa-kr-practice-db-master.mariadb.database.azure.com:3306/S12P11C207?characterEncoding=UTF-8&serverTimezone=UTC  
spring.datasource.username=S12P11C207  
spring.datasource.password=ddjvO0uwYS  
  
server.servlet.encoding.charset=UTF-8  
server.servlet.encoding.force=true  
  
  
  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.properties.hibernate.format\_sql=true  
spring.jpa.properties.hibernate.use\_sql\_comments=true  
spring.jpa.show-sql=true  
  
  
spring.mail.host=smtp.gmail.com  
spring.mail.port=587  
  
spring.mail.username=kmc4889@gmail.com  
spring.mail.password=mnwptibsbygljbut  
  
spring.mail.properties.mail.smtp.auth=true  
#spring.mail.properties.mail.smtp.timeout=5000  
spring.mail.properties.mail.smtp.starttls.enable=true  
  
spring.data.redis.host=i12c207.p.ssafy.io  
spring.data.redis.port=6379  
  
# openApi  
openApi.serviceKey=b6ZrIXGXaUAp6IQIlbm6wvqnoi%2BBc4sToBtp336Lb8WXZ1aWgCkST%2BMtrSfWzIBi1cRsFDL3GKs7PvEsAEY%2B0w%3D%3D  
openApi.aedServiceKey=CEZ30O1G-CEZ3-CEZ3-CEZ3-CEZ30O1G8K  
  
openApi.medicationUrl=http://apis.data.go.kr/1471000/MdcinGrnIdntfcInfoService01/getMdcinGrnIdntfcInfoList01  
openApi.dataType=json  
openApi.availableHospitalUrl=https://apis.data.go.kr/B552657/ErmctInfoInqireService/getEmrrmRltmUsefulSckbdInfoInqire  
#openApi.aedUrl=https://apis.data.go.kr/B552657/AEDInfoInqireService/getEgytAedManageInfoInqire  
openApi.aedUrl=https://safemap.go.kr/openApiService/data/getAedData.do  
openApi.fireDeptUrl=https://api.odcloud.kr/api/15048243/v1/uddi:a7630967-737e-4f06-84bc-f3e7b131f4a9  
#openApi.fireDeptUrl=https://api.odcloud.kr/api/15065056/v1/uddi:157aaef5-55cc-49b8-a318-b2605acee3ed  
openApi.hospitalUrl=https://apis.data.go.kr/B552657/ErmctInfoInqireService/getEgytListInfoInqire  
  
  
# OPENVIDU\_URL: https://13.124.195.234:443/  
OPENVIDU\_URL: https://i12c207.p.ssafy.io/  
OPENVIDU\_SECRET: 1234  
  
google.cloud.credentials.json={"type": "service\_account", "project\_id": "camference", "private\_key\_id": "ca3aff6a80d632e7353d8e5e8478df1a7ff0f99e", "private\_key": "-----BEGIN PRIVATE KEY-----\\nMIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQC3ia28K7weWb32\\nJvCbcBXRgVZ0UdzZL94C6X36m1ZfpDNSZou6p27EAcQVHUvwM2FgxDsduYjOhZUt\\nBM0VVEaPC8nBg96hhjB1mC6Kxta50hTLSWGzgsk+BVV33lofdBxJwzQekOxiuUFr\\n8maKDtHv7Z+pOZZZIqkCPwrZzyFiImxZTL+SSNEww6N6yCl3qFjv0pOLEcdB7yE2\\n0RJ5ga7rOYwFLPpMb/bQ5zfdQLKm7GfWgzO/YlJFx4WJH+aBA1uQ50Xd2tWfdZbc\\nrOLfXsr5KkY8IcCcnkBZaNpopal5DvFy0Zxk1xGK++u8Z+fRBfn6M42i3Mx158NQ\\nW0nbm/KvAgMBAAECggEADWLxVQQrmFsbS9TTP8MRFFLGp31BCmY/Xx8AnUKvLhKf\\n26a3Ekob/hfS4Wddnotg0tn9JdHedJGxq7Lp8mBPNSe6825W0pki97C91ElkZ/E2\\nqw0THzCu1OM2tKN3F0w0s/yU1JXlZ93zM5heejtSfJlsR793g9DPWuaWOASvfqS8\\nZ4GlYoykaBHpDGuzq71jGVS93W+gqA4MmAFLSOCOMWVvuIeQyu1+qIVogjefsRei\\ne/fWpsvGVaZloTiN5kZtbql0UgdjO2fAaEXmw+WGDLV4p98Cfm6clnv6zFn7Tk5r\\nEhIX4AUKKKiPLZ3OyUQ2XPADJTwMnEHet8yD3i4i4QKBgQDomWZOQZJJqq5c4iHL\\nbYAKaABckpez7NHgDHBsjbOAuFFf42mPHVmnNGFFjUVSAi2BQYySLoDhZpBFjylU\\n1hU3ZpoHU09d45/uJVG+O9xfvW8tOiw0HhV+zk0SyAvtRI/eBCyxfxn4L74+72io\\nHeeYOYC4ramwr4WC99UAdcFvdQKBgQDKALNovFwulPRwdbDcCNDAI5JrIAdaQh4T\\no74Ws6/PaA4H0om4maYvlXNA9BJpY3FfpAyOW9Y9mHtpjoTctZ/AEDnN4tJPGG9E\\n4rhiBndjnahjDtSlRGt1phISKQVLgb1Xh6V0vuovHdOkVFXGSHzgFZ552FKAOwEM\\nMxN6vV1ZEwKBgD+QnGrKdKX5JQlxKvcjB5jQbzjdPuQG1hw6igbAesu0k1h3F4Pq\\n63NMV0DgqI8O1Uy7kyTn3IS8uLSpuQvcLZQkT/eshsYLzMsSG9K3HTR02osrmKTt\\ni8ZJrrpxPQEseXyc+3HOYFR5B66/UoSPF5l8LXdKzlLgxMXT4Lvtcl15AoGAVSdk\\nehNlXLGoNbw3ORpyPX0BXS1cTB0xJqBRIqrOw0UfHQcQzYPf/ZEny8rX9BtakLHM\\nJTrrnn2oEMBmm8/orwNoCrXCaFtplIVqqtAFBjZQ5Kk81Qp5rle6qHMoa0Me+UYS\\nZpuQhxowvMZcfkDSe3hBnSWm6a117JmfbY9kSYcCgYEApP1yzVTqQ0fYtHKZdQNK\\ngVMRzrhrUzKe/G0AX0CDSJshETCQw9Et4XMqXzLj0tYYw7vPvUiEzbrphZvmgY5e\\nwQDSwOH28xUyWQVKIy0r+hLPUsP43oAeDTmV+FtXf9W6glre/n1CC2kroeWVkxKw\\nG6oGVR00Liaj1V790R5Edoo=\\n-----END PRIVATE KEY-----\\n", "client\_email": "camference@camference.iam.gserviceaccount.com", "client\_id": "111434812704155452948", "auth\_uri": "https://accounts.google.com/o/oauth2/auth", "token\_uri": "https://oauth2.googleapis.com/token", "auth\_provider\_x509\_cert\_url": "https://www.googleapis.com/oauth2/v1/certs","client\_x509\_cert\_url": "https://www.googleapis.com/robot/v1/metadata/x509/camference%40camference.iam.gserviceaccount.com","universe\_domain": "googleapis.com"}  
  
openai.api.key=sk-proj--m4YnQCz-avbxFePXJKTdWXl\_xtWPlTwx5qxFU9\_87EG1cs0EFonB78HaT-InErrRnEtDTZuAFT3BlbkFJOwD56Bns2tqeTkcW5qV\_QMDzXMNmZ8sAQgbcZ8gMthesnm8WJ1srVRypnJCqvuQQrkJ0LEna4A  
  
# spring.profiles.active= dev  
spring.profiles.active= prod  
  
spring.servlet.multipart.max-file-size=50MB  
spring.servlet.multipart.max-request-size=50MB  
  
kakao.apiUrl=https://dapi.kakao.com/v2/local/geo/coord2address.json  
kakao.apiKey=dd59c6093ad33b4d66efe744bd4076a4

## 빌드 및 배포하기

## # Jenkins pipeline

## pipeline {

## agent any

## // tools {

## // nodejs "nodejs-20.18.2"

## // }

## environment {

## // Docker Hub 자격 증명

## //DOCKERHUB\_CREDENTIALS = credentials('dockerHubAccessAccount')

## // Docker 이미지 정보

## IMAGE\_NAME\_BACKEND = 'camferencebackend'

## IMAGE\_NAME\_FRONTEND = 'camferencefrontend'

## IMAGE\_TAG = 'latest'

## // 배포 서버 정보

## DEPLOY\_SERVER = 'ubuntu@i12c207.p.ssafy.io'

## SSH\_CREDENTIALS = 'sshKeyForEC2'

## // Docker 레지스트리 (Docker Hub)

## DOCKER\_REGISTRY = 'docker.io'

## DOCKER = credentials('dockerHubAccessAccount')

## 

## REACT\_ENV = credentials('REACT\_ENV\_VARIABLES')

## }

## 

## 

## stages {

## stage('Initialize') {

## steps {

## script {

## def dockerHome = tool 'my\_docker'

## env.PATH = "${dockerHome}/bin:${env.PATH}"

## echo "Docker Home Path: ${dockerHome}"

## echo "Updated PATH: ${env.PATH}"

## }

## }

## }

## 

## stage('Git Clone'){

## steps {

## git branch: 'develop', credentialsId: '85d77b74-37fc-4074-90e3-c1cec3c1a518', url: 'https://lab.ssafy.com/s12-webmobile1-sub1/S12P11C207.git'

## 

## }

## }

## 

## 

## stage('Test Deploy') {

## steps {

## script {

## sshagent(credentials: [SSH\_CREDENTIALS]) {

## sh """

## ssh -v -o StrictHostKeyChecking=no ${DEPLOY\_SERVER} <<EOF

## 

## echo "원격 서버에 성공적으로 접속했습니다."

## # backend/src/main/resources 디렉토리 생성 및 권한 설정

## 

## <<EOF

## """

## }

## }

## }

## }

## 

## stage('setting application.properties') {

## steps {

## 

## withCredentials([file(credentialsId: 'application-properties-credential', variable: 'applicationProperties')])

## {

## script {

## sh """

## mkdir -p backend/src/main/resources

## chmod 755 backend/src/main/resources/

## cp -f ${applicationProperties} backend/src/main/resources/

## 

## # cp ${applicationProperties} backend/src/main/resources/application.properties

## """

## 

## }

## }

## }

## }

## 

## stage('setting .env file') {

## steps {

## withCredentials([file(credentialsId: 'REACT\_ENV\_VARIABLES', variable: 'reactEnvFile')]) {

## script {

## sh """

## # React 프로젝트의 루트 디렉토리에 .env 파일 복사

## cp -f ${reactEnvFile} frontend/.env

## """

## }

## }

## }

## }

## 

## stage('Build Backend') {

## steps {

## dir('backend') {

## script {

## // 현재 디렉토리 위치 출력

## echo "현재 디렉토리 위치:"

## sh 'pwd'

## // gradlew 파일 존재 여부 확인

## echo "gradlew 파일 존재 여부 확인:"

## sh '''

## if [ -f ./gradlew ]; then

## echo "gradlew 파일이 존재합니다."

## else

## echo "gradlew 파일이 존재하지 않습니다."

## exit 1

## fi

## '''

## }

## sh 'chmod +x gradlew' // 클론 후 실행 권한 부여

## sh './gradlew build'

## }

## }

## }

## 

## stage('Build Frontend') {

## steps {

## dir('frontend') {

## // nodejs(nodeJSInstallationName: 'nodejs-20.18.2') { // 위의 설정에서 지정한 node 이름

## // sh 'npm install && npm run build'

## // }

## script{

## // Node.js가 포함된 Docker 이미지 사용

## docker.image('node:20').inside('-v $WORKSPACE/frontend:/app -w /app') {

## // sh '''

## // touch .env

## // echo "VITE\_KAKAOMAP\_API\_KEY=0836119f538d650272fc667d96b1a3e0" > .env

## // echo "VITE\_BASE\_URL=http://i12c207.p.ssafy.io:8080" > .env

## // '''

## // sh 'echo "$REACT\_ENV" > .env'

## sh 'pwd'

## sh 'npm install'

## // sh 'npm run build'

## sh 'npm run build:ci'

## }

## }

## }

## }

## }

## 

## stage('Prepare BackEnd Dockerfile') {

## steps {

## script {

## def dockerfilePath = 'backend/Dockerfile'

## // JAR 파일 확인

## //def jarName = sh(script: 'ls -1 backend/build/libs/\*.jar | head -n 1 | xargs basename', returnStdout: true).trim()

## def jarName = sh(script: 'ls -1 backend/build/libs/\*SNAPSHOT.jar | head -n 1 | xargs basename', returnStdout: true).trim()

## echo "빌드된 JAR 파일: ${jarName}"

## 

## echo "Dockerfile 생성 -->"

## writeFile file: dockerfilePath, text: """

## FROM openjdk:17-jdk

## VOLUME /tmp

## COPY build/libs/${jarName} /app.jar

## ENTRYPOINT ["java", "-jar", "/app.jar"]

## EXPOSE 8080

## """

## echo "기본 Dockerfile이 생성되었습니다."

## 

## }

## }

## }

## 

## stage('Prepare Frontend Dockerfile') {

## steps {

## //dir('frontend') {

## script {

## 

## def dockerfilePath = 'frontend/Dockerfile'

## echo "Creating Frontend Dockerfile at ${dockerfilePath} ..."

## writeFile file: dockerfilePath, text: """

## FROM node:20-alpine

## WORKDIR /app

## COPY dist ./dist

## RUN npm install -g serve

## CMD [ "serve", "-s", "dist", "-l", "80" ]

## EXPOSE 80

## """

## echo "Frontend Dockerfile created."

## }

## //}

## }

## }

## 

## stage('Build Docker Image') {

## steps {

## script {

## 

## echo "Docker Build 전에 frontend 디렉토리 파일 확인"

## sh "ls -l frontend/"

## 

## // Docker 백엔드 이미지 빌드

## sh 'docker build -t tjdwns2243/camferencebackend:latest -f backend/Dockerfile ./backend/'

## 

## // Docker 프론트엔드 이미지 빌드

## sh "docker build -t tjdwns2243/camferencefrontend:${IMAGE\_TAG} -f frontend/Dockerfile ./frontend/"

## }

## }

## }

## 

## 

## 

## stage('Push') {

## steps {

## script {

## sh "docker login -u $DOCKER\_USR -p $DOCKER\_PSW"

## sh "docker push ${DOCKER\_USR}/${IMAGE\_NAME\_BACKEND}:${IMAGE\_TAG}" // 백

## sh "docker push ${DOCKER\_USR}/${IMAGE\_NAME\_FRONTEND}:${IMAGE\_TAG}" // 프론트

## 

## }

## }

## }

## 

## stage('Deploy') {

## steps {

## script {

## sh "docker pull ${DOCKER\_USR}/${IMAGE\_NAME\_BACKEND}:${IMAGE\_TAG}"

## // 이미 해당 이미지가 존재한다면 중지후 삭제.(백엔드)

## sh 'docker stop camferencebackend || true'

## sh 'docker rm camferencebackend || true'

## sh "docker run -d --name camferencebackend -p 8080:8080 ${DOCKER\_USR}/${IMAGE\_NAME\_BACKEND}:${IMAGE\_TAG}"

## 

## 

## sh "docker pull ${DOCKER\_USR}/${IMAGE\_NAME\_FRONTEND}:${IMAGE\_TAG}"

## // 이미 해당 이미지가 존재한다면 중지후 삭제.(프론트엔드)

## sh 'docker stop camferencefrontend || true'

## sh 'docker rm camferencefrontend || true'

## sh "docker run -d --name camferencefrontend -p 8000:80 ${DOCKER\_USR}/${IMAGE\_NAME\_FRONTEND}:${IMAGE\_TAG}"

## 

## }

## }

## }

## 

## 

## }

## }

## # nginx 설정파일

## server {

## server\_name i12c207.p.ssafy.io;

## # 1. React 프론트엔드 (예: 기본 경로)

## location / {

## proxy\_pass http://127.0.0.1:8000;

## 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;

## }

## # 2. Spring Boot 백엔드 (예: /api 로 시작하는 요청)

## location /api/ {

## proxy\_pass http://127.0.0.1:8080/;

## 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;

## # SSE를 위한 설정 추가

## proxy\_set\_header Connection ''; # Keep-alive 연결을 위해 필요

## proxy\_http\_version 1.1; # HTTP/1.1 사용

## proxy\_buffering off; # 버퍼링 비활성화

## proxy\_cache off;

## # 추가해야 할 설정

## proxy\_set\_header X-Accel-Buffering no; # Nginx 버퍼링 비활성화

## chunked\_transfer\_encoding off; # chunked transfer 비활성화

## # Keepalive 설정

## keepalive\_timeout 300s; # keepalive 타임아웃 설정

## keepalive\_requests 1000; # keepalive 요청 수 설정

## # 응답 대기하는데 걸리는 시간

## proxy\_read\_timeout 120s;

## }

## # 3. OpenVidu (예: /openvidu 로 시작하는 요청)

## location /openvidu/api/ {

## proxy\_pass https://127.0.0.1:8443; # OpenVidu 서버의 포트에 맞게 변경

## 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 Authorization $http\_authorization; # Basic 인증 헤더 추가

## proxy\_pass\_header Authorization; # Authorization 헤더 통과

## }

## listen 443 ssl; # managed by Certbot

## ssl\_certificate /etc/letsencrypt/live/i12c207.p.ssafy.io/fullchain.pem; # managed by Certbot

## ssl\_certificate\_key /etc/letsencrypt/live/i12c207.p.ssafy.io/privkey.pem; # managed by Certbot

## include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot

## ssl\_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

## }

## server {

## if ($host = i12c207.p.ssafy.io) {

## return 301 https://$host$request\_uri;

## } # managed by Certbot

## listen 80;

## server\_name i12c207.p.ssafy.io;

## return 404; # managed by Certbot

## }

## EC2 포트번호

## 443 : nginx

## 9090 : Jenkins

## 6379 : Redis

## 8080 : backend

## 8000 : frontend

## 22, 80, 3478, 40000–57000, 57001–65535 : Openvidu v2

1. **사용자 시나리오**

시나리오

<신고자 = 환자인 경우>

1. 신고자 신고 전화

2. 상황실에서

2-1. 신고자와의 짧은 대화를 통해 영상통화 생성 여부 판단

2-2. 영상통화가 필요한 경우 URL 전송하고, 신고자에게 전송 안내하기

2-3. 신고자에게 URL 접속 방법 안내하기

3. 신고자가 URL을 통해 영상통화 방 입장

<신고자 = 회원인 경우>

3-1. 등록된 정보 불러옴.

3-2. 현재 상황 대화를 통해 자세히 파악하기

3-3. AI 요약

3-4. 정보 저장 -> 출동 지령 페이지로 자동 이동

3-5. 출동이 필요한 경우, 인근 소방서로 출동 지령내리기

3-6. 구급대원이 영상통화방에 입장하면 통화 종료하기 -> 보호자 알림 전송

4. 구급대원이 출동 지령을 받은 상황

4-1. 영상통화 입장 및 환자 정보 페이지 이동

4-2. 대화를 통해 출동하고 있다는 정보와 함께 필요시 정보 파악하기

4-3. 필요한 대화 종료 후 통화 종료

4-4. 현장 도착

4-5. 현장 처치 후 환자 상태 및 처치 내용 입력 -> pre-KTAS 사용

4-6. 환자 구급차로 이동 후 이송 신청하기

5. 병원이 이송 요청 받은 경우

5-1. 이송 알림을 통해 이송 상세 이동

5-2. 이송 수락 여부 결정

6. 병원이 이송 수락한 경우

6-1. 이송 수락 알림이 오면 해당 병원으로 이송

6-2. 이송 완료 후 보호자 알림 전송 및 이송 종료하기