|  |
| --- |
|  |



|  |
| --- |
| **새잎** |
| **포팅 메뉴얼** |
|  |
|  |
|  |
|  |
|  |

목차

[1. 프로젝트 개요 2](#_Toc96072460)

[2. 프로젝트 사용 도구 2](#_Toc96072461)

[3. 개발도구 2](#_Toc96072462)

[4. 개발환경 2](#_Toc96072463)

[5. 환경변수 3](#_Toc96072464)

[6. 배포설정 3](#_Toc96072466)

[7. 설치방법 5](#_Toc96072467)

# 개요

# 1. 프로젝트 개요

당신의 20살은 어땠나요? 친구, 부모님의 케어 속에서 든든한 삶을 살았나요? 여기 그렇지 않은 환경에 있는 소수의 청년들이 있습니다.

우리 프로젝트 ‘새잎’은 이러한 청년들에게 개발쪽 진로를 제시해주고, 라포를 형성하여 그들에게 하나의 기둥이 되어줄 수 있는 IT 멘토링 서비스입니다. 새잎과 함께라면, 같은 처지에 있는 친구들과 함께 일상을 공유하고, 앞으로 한걸음 더 나아갈 수 있습니다.

# 2. 사용 도구

* 이슈 관리: JIRA
* IDE : Visual Studio, Intellij Ultimate
* 형상 관리: Gitlab
* 커뮤니케이션: Notion, Mattermost,
* 디자인: Figma
* CI/CD: Jenkins

# 3. 개발 도구

### Frontend

* 프레임워크 : React
* 라이브러리 : Redux, axios

### Backend

* 프레임워크 : Spring Boot, Fast API
* 라이브러리 : JPA, Spring Security, JWT, Tensorflow, Numpy, Matplotlib, Seaborn,

## 4. 개발 환경

### 가) Frontend

| **Node.js** | **20.15.0** |
| --- | --- |
| React | 18.3.1 |

### 나) Backend

| **Java(JVM)** | **17** |
| --- | --- |
| Spring Boot | 3.3.1 |
| Mysql | 8.0.38 |
| Fast API | 0.111.1 |
| keras | 3.4.1 |
| numpy | 1.26.4 |
| requests | 2.32.3 |
| tensorflow | 2.17.0 |
| uvicorn | 0.30.4 |

### 

### Infra

| **Docker** | **27.1.1** |
| --- | --- |
| Nginx | nginx/1.18.0 (Ubuntu) |
| Docker-compose | v2.29.1 |
| Jenkins | 2.468 |
| AWS EC2 |  |
| AWS S3 |  |
| Ubuntu | * 1. TS |

## 

## 5. 환경 변수

### 가) Spirng Boot

* application.yml

spring:

profiles:

active: secret

jackson:

serialization:

WRITE\_DATES\_AS\_TIMESTAMPS: false

jwt:

secret: bangcutsoragodoongmeruohboksayif

server:

servlet:

encoding:

enabled: true

charset: UTF-8

force: true

port: 8080

openvidu:

url: <https://i11e107.p.ssafy.io:8443>

secret: bangcutsoragodoongmeruohboksayif

logging:

level:

org.springframework.core.env: DEBUG

* application-secret.yml

spring:

jackson:

serialization:

write-dates-as-timestamps: false

datasource:

url: jdbc:mysql://i11e107.p.ssafy.io:3306/sayif?serverTimezone=UTC

username: root

password: sayif

jpa:

hibernate:

ddl-auto: none

show\_sql: true

format\_sql: true

logging.level:

org.hibernate.SQL: debug

org.hibernate.orm.jdbc.bind: trace

CHAT-GPT-API-KEY: sk-proj-OvM4gYMb5w6UPP6BZijLT3BlbkFJQB6FB6SdNrjkzsKT1Isu

API\_URL: <https://api.openai.com/v1/chat/completions>

cloud:

aws:

region:

static: ap-northeast-2

s3:

bucket-names:

member: sayif

board: sayif-board

spt-info: sayif-spt-info

material: sayif-material

challenge: sayif-challenge

credentials:

access-key: AKIAXBVIPXM3H4J2JRFW

secret-key: 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

coolsms:

api:

key: NCSNKCU6GLZLGWKH

secret: OR2U6IUL04CQO2XZEAYAL3OVY1HZWL09

## 6. 배포 설정

### 가) AWS

* 포트 번호

| **MySQL** | **3306** | **Pipeline 미포함** |
| --- | --- | --- |
| Jenkins | 8080 | Pipeline 미포함 |
| Spring Boot | 7777 | Pipeline 포함 |
| Nginx | 80, 443 | Pipeline 미포함 |
| React | 3000 | Pipeline 포함 |
| Openvidu | 3478, 8442, 8443 | Pipeline 미포함 |
| Fast api | 8000 | Pipeline 포함 |

### 나) Jenkins

pipeline {

agent any

environment {

REPO\_URL = '<https://lab.ssafy.com/s11-webmobile1-sub2/S11P12E107/>'

BRANCH = 'develop'

FRONTEND\_IMAGE = 'leechanghojjang/sayif-frontend'

BACKEND\_IMAGE = 'leechanghojjang/sayif-backend'

FASTAPI\_IMAGE = 'leechanghojjang/sayif-fastapi'

DOCKERHUB\_CREDENTIALS = credentials('dockerhub-token')

GITLAB\_USER\_CREDENTIALS = credentials('gitlab-user-token')

OPENVIDU\_URL = credentials('openvidu-url')

OPENVIDU\_SECRET = credentials('openvidu-secret')

}

stages {

stage('Checkout') {

steps {

script {

echo "Checking out branch: ${BRANCH} from ${REPO\_URL}"

checkout([$class: 'GitSCM', branches: [[name: '\*/' + BRANCH]],

doGenerateSubmoduleConfigurations: false,

extensions: [[$class: 'WipeWorkspace']], // Optional: Clean workspace before checkout

userRemoteConfigs: [[url: "${REPO\_URL}", credentialsId: 'gitlab-user-token']]])

}

}

}

stage('Prepare Config') {

steps {

script {

dir('backend/sayif/src/main/resources') {

writeFile file: 'application-secret.yml', text: """

spring:

mvc:

pathmatch:

matching-strategy: ant-path-matcher

security:

user:

name: user

password: 12345

jpa:

hibernate:

ddl-auto: none

show-sql: true

datasource:

driver-class-name: com.mysql.cj.jdbc.Driver

url: jdbc:mysql://i11e107.p.ssafy.io:3306/sayif?serverTimezone=UTC

username: root

password: sayif

CHAT-GPT-API-KEY: sk-proj-OvM4gYMb5w6UPP6BZijLT3BlbkFJQB6FB6SdNrjkzsKT1Isu

API\_URL: <https://api.openai.com/v1/chat/completions>

cloud:

aws:

region:

static: ap-northeast-2

s3:

bucket-names:

member: sayif

board: sayif-board

spt-info: sayif-spt-info

material: sayif-material

challenge: sayif-challenge

credentials:

access-key: AKIAXBVIPXM3H4J2JRFW

secret-key: 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

coolsms:

api:

key: NCSNKCU6GLZLGWKH

secret: OR2U6IUL04CQO2XZEAYAL3OVY1HZWL09

"""

echo readFile('application-secret.yml')

}

}

}

}

stage('Build Backend') {

steps {

script {

dir('backend/sayif') {

sh 'echo "Building project..."'

sh 'chmod 777 gradlew'

sh './gradlew clean build -Dfile.encoding=UTF-8'

}

}

}

}

stage('Build Frontend') {

steps {

script {

dir('frontend/sayif') {

sh 'echo "Building frontend project..."'

sh 'CI=false npm install'

sh 'CI=false npm run build'

}

}

}

}

stage('Test') {

steps {

script {

dir('backend/sayif') { // 올바른 경로로 이동

sh 'echo "Running tests..."'

sh 'chmod 777 gradlew'

sh './gradlew test'

}

}

}

}

stage('Package') {

steps {

script {

dir('backend/sayif') { // 올바른 경로로 이동

sh 'echo "Packaging project..."'

sh 'chmod 777 gradlew'

sh './gradlew bootJar'

}

}

}

}

stage('Docker Login') {

steps {

script {

sh 'echo $DOCKERHUB\_CREDENTIALS\_PSW | docker login -u $DOCKERHUB\_CREDENTIALS\_USR --password-stdin'

}

}

}

// 최초 한번만 실행

// stage('Build FastAPI') {

// steps {

// script {

// dir('backend/sayifFastAPI') {

// sh '''

// echo "Building Docker image for FastAPI..."

// docker build -t ${FASTAPI\_IMAGE}:latest .

// echo "Pushing Docker image for FastAPI..."

// docker push ${FASTAPI\_IMAGE}:latest

// '''

// }

// }

// }

// }

stage('Docker Build and Push Frontend') {

steps {

script {

dir('frontend/sayif') {

sh '''

echo "Building Docker image for frontend..."

docker build -t ${FRONTEND\_IMAGE}:latest .

echo "Pushing Docker image for frontend..."

docker push ${FRONTEND\_IMAGE}:latest

'''

}

}

}

}

stage('Docker Build and Push Backend') {

steps {

script {

dir('backend/sayif') {

sh '''

echo "Building Docker image for backend..."

cp src/main/resources/application-secret.yml .

docker build -t ${BACKEND\_IMAGE}:latest .

echo "Pushing Docker image for backend..."

docker push ${BACKEND\_IMAGE}:latest

'''

}

}

}

}

stage('Deploy') {

steps {

script {

sh '''

echo "Setting environment variables for deployment..."

export OPENVIDU\_URL=${OPENVIDU\_URL}

export OPENVIDU\_SECRET=${OPENVIDU\_SECRET}

echo "Deploying application..."

docker-compose stop backend frontend

docker-compose rm -f backend frontend

docker-compose pull backend frontend

docker-compose up -d backend frontend

'''

}

}

}

}

}

### 다) Nginx

server {

listen 443 ssl; # HTTPS 포트

server\_name i11e107.p.ssafy.io;

ssl\_certificate /etc/letsencrypt/live/i11e107.p.ssafy.io/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/i11e107.p.ssafy.io/privkey.pem;

include /etc/letsencrypt/options-ssl-nginx.conf;

ssl\_dhparam /etc/letsencrypt/ssl-dhparams.pem;

location / {

proxy\_pass <http://127.0.0.1:3000>; # 프론트엔드 서버 (포트 3000)

proxy\_set\_header X-Forwarded-Proto $scheme;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection "upgrade";

}

location /api {

proxy\_pass <http://127.0.0.1:7777/api>; # 백엔드 API 서버 (포트 7777)

proxy\_set\_header X-Forwarded-Proto $scheme;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection "upgrade";

add\_header 'Access-Control-Allow-Origin' '\*';

add\_header 'Access-Control-Allow-Credentials' 'true';

add\_header 'Access-Control-Allow-Methods' 'GET, POST, DELETE, PATCH, OPTIONS';

add\_header 'Access-Control-Allow-Headers' 'Content-Type, Authorization, token';

proxy\_connect\_timeout 300s; # 백엔드 서버로의 연결 타임아웃

proxy\_read\_timeout 300s; # 백엔드 서버의 응답을 기다리는 타임아웃

proxy\_send\_timeout 300s; # 클라이언트로 응답을 보내는 타임아웃

send\_timeout 300s; # 클라이언트와의 연결에서 데이터 전송 타

}

location /api/ws/ {

proxy\_pass <http://127.0.0.1:7777>;

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;

add\_header Access-Control-Allow-Origin \*;

add\_header Access-Control-Allow-Methods 'GET, POST, OPTIONS';

add\_header Access-Control-Allow-Headers 'Origin, Content-Type, Accept, Authorization';

proxy\_buffering off;

}

}

server {

listen 80;

server\_name i11e107.p.ssafy.io;

if ($host = i11e107.p.ssafy.io) {

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

}

}

### dockerfile

* Frontend

FROM node:20

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npm run build

EXPOSE 3000

CMD ["npm", "start"]

* backend

FROM openjdk:17

ARG JAR\_FILE=build/libs/\*.jar

COPY ${JAR\_FILE} app.jar

COPY src/main/resources/application-secret.yml /app/config/application-secret.yml

ENV SPRING\_CONFIG\_LOCATION=/app/config/application-secret.yml

ENV SPRING\_JWT\_SECRET=bangcutsoragodoongmeruohboksayif

ENTRYPOINT ["java", "-jar", "/app.jar","--spring.config.additional-location=/app/config/application-secret.yml"]

* docker-compose

version: '3'

services:

backend:

image: leechanghojjang/sayif-backend:latest

build:

context: ./backend/sayif

dockerfile: Dockerfile

ports:

- "7777:8080"

environment:

- SPRING\_PROFILES\_ACTIVE=secret

- openvidu.url=${OPENVIDU\_URL}

- openvidu.secret=${OPENVIDU\_SECRET}

frontend:

image: leechanghojjang/sayif-frontend:latest

build:

context: ./frontend/sayif

dockerfile: Dockerfile

ports:

- "3000:3000"

## 7. 설치방법

### 가) Docker

# 기존의 docker 관련 engine 제거

sudo apt-get remove docker docker-engine docker.io containerd runc

# 패키지 설치

sudo apt-get update

sudo apt-get install ca-certificates curl gnupg lsb-release

# Docker 공식 GPG 키 추가

sudo mkdir -p /etc/apt/keyrings

curl -fsSL <https://download.docker.com/linux/ubuntu/gpg> | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

# Docker 저장소 설정

echo \\

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] <https://download.docker.com/linux/ubuntu> \\

$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Docker 설치

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

# Docker 루트 권한 없이 실행

sudo usermod -aG docker $USER

newgrp docker

# Docker 시작

sudo systemctl start docker

sudo systemctl enable docker

### 나) ****Mysql****

docker run -d --network sayif\_default --name sayif-mysql -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=sayif mysql:8.0

### 다) Jenkins

# Jenkins 실행

docker run -d --name jenkins-server --user root --network sayif\_default -v jenkins\_home:/var/jenkins\_home -v /var/run/docker.sock:/var/run/docker.sock -p 8080:8080 -p 50000:50000 jenkins/jenkins:latest

# Jenkins 컨테이너 내부 진입

docker exec -it jenkins-server /bin/bash

# Jenkins 내부에서 node.js / docker-compose 설치

curl -fsSL <https://deb.nodesource.com/setup\_18.x> | bash -

apt-get install -y nodejs

curl -L "<https://github.com/docker/compose/releases/download/v2.20.2/docker-compose-$>(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

# 버전확인

docker-compose --version

node –v

### 라) Openvidu

# 기존 openvidu 관련 docker container, image 삭제

docker rm <container ID>

docker rmi <image ID>

# 루트 권한

sudo su

# Openvidu 설치 경로

cd /opt

# Openvidu on premise 설치

$ curl <https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install\_openvidu\_latest.sh> | bash

# Openvidu로 이동

cd openvidu

# 설정파일 들어가서 설정 변경 (기존 nginx 쓸 때에 80, 443 외 다른 포트 지정)

nano .env

DOMAIN\_OR\_PUBLIC\_IP=${Address} // 도메인 주소 OPENVIDU\_SECRET=${Secret} // 본인 지정 CERTIFICATE\_TYPE=letsencrypt LETSENCRYPT\_EMAIL=${Email} // 이메일 무조건 입력 HTTP\_PORT = 8442 HTTPS\_PORT = 8443

# Openvidu 실행

./openvidu start // /opt/openvidu

### 마) Nginx

# 시스템 패키지 업데이트 sudo apt-get update

# Nginx 설치 sudo apt-get install nginx -y

# Nginx 시작 및 부팅 시 자동 시작 설정 sudo systemctl start nginx sudo systemctl enable nginx

# SSL 인증서 설정 sudo apt-get install certbot python3-certbot-nginx –y

# SSL 인증서 발급 및 Nginx 설정 자동화 sudo certbot --nginx

# Nginx 설정 확인 sudo nano /etc/nginx/sites-available/default

# Nginx 재시작 sudo systemctl restart nginx