|  |
| --- |
|  |



|  |
| --- |
| **Newspeak** |
| **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

# 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 : TypeScript, Zustand, axios,  
   MUI(Mateial-UI), react-speech-recognition, recharts

### Backend

* 프레임워크 : Spring Boot
* 라이브러리 : JPA, Spring Security, JWT, Spring ai, Spring elasticsearch

## 4. 개발 환경

### 가) Frontend

| **Node.js** | **20.15.0** |
| --- | --- |
| React | 18.3.1 |
| TypeScript | 5.5 |
| Zustand | 4.5 |

### 나) Backend

| **Java(JVM)** | **17** |
| --- | --- |
| Spring Boot | 3.3.3 |
| Mysql | 8.0.38 |
| Spring ai | 1.0.0-M1 |
| Dependency-manager | 1.1.5 |

### 

### Infra

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

## 

## 5. 환경 변수

### 가) Spirng Boot

* application.yml

spring:

  cloud:

    gcp:

      storage:

        bucket: 'sentence\_pro'

        project-id: 'steady-citron-435112-b2'

        credentials:

          location: 5e01616d4635ae3d07be55d482fad7567834f0c2.json

  profiles:

    include: "jwt, oauth, secret"

    active: prod

  ai:

    openai:

      api-key : ${gpt.api-key}

  elasticsearch:

    uris: https://j11e103.p.ssafy.io:9200

logging:

  level:

    org.hibernate.SQL : debug

    org.hibernate.type: trace

server:

  port: 8080

gpt:

  api:

    url: https://api.openai.com/v1/chat/completions

    assistant:

      report: asst\_oY2GnaC2sJWZ0mM86obWrsGM

      conv : asst\_KeT2kzSKqpgmpAVxZ9kbOPYg

    threads: https://api.openai.com/v1/threads

  api-key: ${gpt-secret.api-key}

  prompt:

    report: 저는 이제부터 당신과 영어로 대화할 것입니다. 당신은 제가 준 뉴스 기사의 내용을 보고, 해당 내용을 주제로 적절한 응답을 한국인과의 회화 수준에서 영어로 제공하세요. 또한 당신은 해당 대화들을 모두 수집하여 마지막에 최종적으로 정해진 5개의 분야 (어휘력, 표현력, 이해력, 명확성, 문법) 에 각 20점을 만점으로 총 100점을 만점으로하는 점수를 기준으로 제공하시오. 그리고 각 분야에 대한 적절한 피드백도 일정한 기준을 가지고 제공하시오. 그리고 사용자와의 대화를 돌아보고, 특정 대화에서 바꾸었으면 좋았을 부분이나 더 나은 표현을 2~3개 정도 제시하시오.

  setting:

    model: gpt-4o-mini

    temperature: 2

    top-p: 0.6

    max-tokens: 16383

    frequency\_penalty: 0

    presence\_penalty: 0

    tool-choice: required

etri:

  api-key: ${etri-secret.api-key}

* application-prod.yml

signUpUrl: https://j11e103.p.ssafy.io/register

mainUrl: https://j11e103.p.ssafy.io

frontUrl: https://j11e103.p.ssafy.io

# [jpa property]

spring:

  jpa:

    properties:

      hibernate:

        show\_sql: true

        generate-ddl: false

        format\_sql: true

        ddl-auto: validate

        jdbc:

          lob:

            non\_contextual\_creation: true

        physical\_naming\_strategy: org.hibernate.boot.model.naming.CamelCaseToUnderscoresNamingStrategy

        dialect: org.hibernate.dialect.MySQLDialect

  datasource:

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

    url: jdbc:mysql://j11e103.p.ssafy.io:3306/news\_db?useSSL=false&serverTimezone=Asia/Seoul&characterEncoding=UTF-8&allowPublicKeyRetrieval=true

    username: 비밀

    password: 비밀

    hikari: # HikariCP를 사용하는 경우

      maximum-pool-size: 10 # 예시 설정, 필요에 따라 조정

  security:

    oauth2:

      client:

        registration:

          google:

            client-id: 비밀

            client-secret: 비밀

            scope: profile, email

          kakao:

            redirect-uri: http://j11e103.p.ssafy.io:8081/login/oauth2/code/kakao

            client-id: 4df9171907e240718c18e81e6c825167

            client-secret: VTSOvDA57TeBpgp4ZhrdAYzuKgiloCBT

            client-authentication-method: client\_secret\_post

            authorization-grant-type: authorization\_code

            scope: account\_email, profile\_image

            client-name: Kakao

        provider:

          kakao:

            authorization-uri: https://kauth.kakao.com/oauth/authorize

            token-uri: https://kauth.kakao.com/oauth/token

            user-info-uri: https://kapi.kakao.com/v2/user/me

            user-name-attribute: id

jwt:

  secretKey: 비밀

  access:

    expiration: 3600000 # 1시간(60분) (1000L(ms -> s) \* 60L(s -> m) \* 60L(m -> h))

    header: Authorization

  refresh:

    expiration: 1209600000 #  (1000L(ms -> s) \* 60L(s -> m) \* 60L(m -> h) \* 24L(h -> 하루) \* 14(2주))

    header: Authorization-refresh

* application-secret.yml

etri-secret:

  api-key: 비밀

gpt-secret:

  api-key: 비밀

  tts-key: 비밀

## 6. 배포 설정

### 가) AWS

* 포트 번호

| **MySQL** | **3306** | **Pipeline 미포함** |
| --- | --- | --- |
| Jenkins | 8080 | Pipeline 미포함 |
| Spring Boot | 8081 | Pipeline 포함 |
| Nginx | 80, 443 | Pipeline 미포함 |
| React | 3000 | Pipeline 포함 |
| MySql | 3306 | Pipeline 미포함 |
| MongoDB | 27017, 27018 | Pipeline 미포함 |
| Redis | 6379 | Pipeline 미포함 |
| Elastic Search | 9200 | Pipeline 미포함 |
| kibana | 5601 | Pipeline 미포함 |
| monstache | 8082 | Pipeline 미포함 |
|  |  |  |

### 나) Jenkins

pipeline {

    agent any

    environment {

        // Docker 이미지 이름

        BACKEND\_IMAGE = 'Gongman41/backend'

        PROFILES\_ACTIVE = 'prod'

        // SONAR\_TOKEN = credentials('sonacube')

    }

    triggers {

        GenericTrigger(

            genericVariables: [

                [key: 'gitlabActionType', value: '$.object\_kind'],

                [key: 'gitlabMergeRequestState', value: '$.object\_attributes.state'],

                [key: 'gitlabSourceBranch', value: '$.object\_attributes.source\_branch'],

                [key: 'gitlabTargetBranch', value: '$.object\_attributes.target\_branch']

                // [key: 'gitlabTargetBranch', value: 'dev-back']

            ],

            token: 'gitlabRepo', // Jenkins에서 발급한 Webhook 토큰

            printContributedVariables: true,

            printPostContent: true,

            causeString: 'Triggered by $gitlabActionType event'

        )

    }

    stages {

        stage('Print Webhook Variables') {

            steps {

                script {

                    echo "GitLab Event: ${env.gitlabActionType}"

                    echo "Merge Request Source Branch: ${env.gitlabSourceBranch}"

                    echo "Merge Request Target Branch: ${env.gitlabTargetBranch}"

                }

            }

        }

        stage('Check if Approved Merge Request') {

            when {

                expression {

                    return (

            //             // (env.gitlabActionType == 'MERGE' && env.gitlabMergeRequestState == 'merged' &&

            //             //   (env.gitlabTargetBranch == 'master' || env.gitlabTargetBranch == 'dev'))

                        env.gitlabTargetBranch == 'dev-back')

                }

            }

            stages {

                stage('Clone repository') {

                    steps {

                        script {

                            def branchName = env.gitlabTargetBranch

                            echo "Cloning branch: ${branchName}"

                            git branch: "${branchName}",

                                url: 'https://lab.ssafy.com/s11-bigdata-dist-sub1/S11P21E103.git',

                                credentialsId: 'lkm98401'

                        }

                    }

                }

                stage('Build and Deploy Backend') {

                    stages {

                        stage('Build Backend') {

                            steps {

                                script {

                                    dir('backend') {

                                        withCredentials([

                                            file(credentialsId: 'application-secret', variable: 'APPLICATION\_SECRET\_YML')

                                        ]) {

                                            sh 'cp -f $APPLICATION\_SECRET\_YML src/main/resources/application-secret.yml'

                                        }

                                        withCredentials([

                                            file(credentialsId: 'application-prod', variable: 'APPLICATION\_PROD\_YML')

                                        ]) {

                                            sh 'cp -f $APPLICATION\_PROD\_YML src/main/resources/application-prod.yml'

                                            sh 'chmod 777 gradlew'

                                            sh './gradlew clean build -x test'

                                        }

                                    }

                                }

                            }

                        }

                        stage('Docker Build Backend') {

                            steps {

                                script {

                                    sh """

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

                                    """

                                }

                            }

                        }

                        stage('Deploy Backend') {

                            steps {

                                script {

                                    // 컨테이너가 이미 존재하면 삭제

                                    sh """

                                    docker stop backend || true

                                    docker rm backend || true

                                    """

                                    // 새 컨테이너 시작

                                    sh """

                                    docker run -d --name backend \

                                    -p 8081:8080 \

                                    -e SPRING\_PROFILES\_ACTIVE=${PROFILES\_ACTIVE} \

                                    ${BACKEND\_IMAGE}:latest

                                    """

                                }

                            }

                        }

                    }

                }

            }

        }

    }

    post {

        success {

            echo 'Backend Build and Deployment succeeded!'

        }

        failure {

            echo 'Backend Build or Deployment failed!'

        }

    }

}

### 다) Nginx

server {

listen 80;

server\_name j11e103.p.ssafy.io;

location / {

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

}

}

server {

listen 443 ssl;

listen [::]:443 ssl;

server\_name j11e103.p.ssafy.io;

ssl\_certificate /etc/letsencrypt/live/j11e103.p.ssafy.io/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/j11e103.p.ssafy.io/privkey.pem;

ssl\_session\_cache shared:SSL:1m;

ssl\_session\_timeout 10m;

ssl\_ciphers 'ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:!aNULL:!MD5:!3DES';

ssl\_prefer\_server\_ciphers on;

root /var/www/html;

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

location / {

proxy\_pass http://localhost:3000;

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 Upgrade $http\_upgrade;

proxy\_set\_header Connection "upgrade";

proxy\_redirect off;

}

location /api/v1/ {

proxy\_pass http://localhost:8081;

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\_pass\_header Authorization;

proxy\_set\_header Authorization $http\_authorization;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection "upgrade";

proxy\_redirect off;

# CORS 설정

# add\_header 'Access-Control-Allow-Origin' 'https://localhost:5173' always;

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

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

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

}

}

### dockerfile

* Frontend

FROM node:18-alpine AS build

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npm run build

FROM node:18-alpine

WORKDIR /app

RUN npm install -g serve

COPY --from=build /app/dist ./dist

EXPOSE 3000

CMD ["serve", "-s", "dist", "-l", "3000"]

* Backend

FROM openjdk:17-jdk-alpine

WORKDIR /app

COPY build/libs/newSpeak-0.0.1-SNAPSHOT.jar app.jar

EXPOSE 8080

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

* docker-compose

version: '3.8'

services:

backend:

build:

context: ./backend

ports:

- "8081:8080"

environment:

SPRING\_PROFILES\_ACTIVE: prod

networks:

- app-network

frontend:

build:

context: ./NewSpeak

ports:

- "3000:3000"

networks:

- app-network

networks:

app-network:

driver: bridge

## 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 --name mysql-db --network bridge -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=1234 mysql:8.0

### 다) Jenkins

# Jenkins 실행

docker run -d --name jenkins --user root -v jenkins\_home:/var/jenkins\_home -v /var/run/docker.sock:/var/run/docker.sock -p 8080:8080 -p 50000:50000 jenkins/jenkins:lts-jdk17

# Jenkins 컨테이너 내부 진입

docker exec -it jenkins /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) 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/conf.d /default.conf

# Nginx 재시작 sudo systemctl restart nginx