포팅 메뉴얼

📌 개발 환경

▼ Android

- IDE
  - AndroidStudio Ladybug 2024.2.1 Patch 1
- Kotlin
  - Kotilin `2.0.20`
- Compose
  - ComposeBom `2024.10.00`

▼ Backend

- IDE
  - IntelliJ `2024.3`
- Java
  - Java OpenJDK `17.0.12`
- Spring
  - Spring Boot `3.3.4`
  - Spring Security `6.3.3`
  - Spring Data JPA `3.3.4`
  - Spring Data redis `3.3.3`
  - Spring Data elasticsearch `5.3.4`
  - OAuth2.0 `6.3.3`
  - JWT `0.11.5`
  - Lombok `1.18.34`

- ELK
  - Logstash `7.17.12`
  - Elasticsearch `7.17.12`
  - Kibana `7.17.12`
- etc
  - Gradle `8.10.2`
  - Swagger `3.0.0`
- Nodejs
  - axios `1.7.7`
  - express `16.4.5`
  - node-geocoder `4.4.0`
  - socket.io `4.8.1`

AI

- python `3.11`
- vertexai `1.71.1`
- SQLAlchemy `2.0.36`
- google-cloud-aiplatform `1.71.1`
- fastapi `0.115.4`

Database

- MySQL `8.0.38`
- Redis `3.0.504`
- AWS S3

Infra

- AWS EC2
- Docker `24.0.7`
- Docker Compose `2.29.7`
- Nginx `1.27.2`

- Jenkins `2.479.1`

▼ Collaboration

- Jira
- GitLab
- Notion
- Mattermost

📌 환경 변수 설정

▼ Android

- local.properties (project root directory)
  - REST API

```
BASE_URL="{ServerURL}"
```

  - S3

```
BUCKET_NAME="{bucket_name}"
BUCKET_REGION="{bucket_region}"
AWS_ACCESS_KEY="{access_key}"
AWS_SECRET_KEY="{secret_key}"
```

  - GOOGLE OAUTH

```
GOOGLE_OAUTH_CLIENT_ID={client_id}
```

  - VERSION

```
VERSION="{version info}"
```

▼ Backend

- **application.yml**

```
spring:
  main:
    web-application-type: servlet
    allow-circular-references: true
    allow-bean-definition-overriding: true

  profiles:
    include: s3, db, security

  servlet:
    multipart:
      enabled: true
      max-file-size: 10MB
      max-request-size: 10MB

  jpa:
    hibernate:
      ddl-auto: none  # 스키마 자동 생성 전략 (update, creat
      naming:
        physical-strategy: org.hibernate.boot.model.nami
      dialect: org.hibernate.dialect.MySQLDialect  # MyS
    show-sql: true  # SQL 쿼리 로깅 여부
    properties:
      hibernate:
        format_sql: true  # SQL 쿼리 포맷팅 여부

  elasticsearch:
#    uris: localhost:9200
         uris: k11d209.p.ssafy.io:9200
#    username: elastic      # 필요한 경우
#    password: yourpassword # 필요한 경우
    data:
      elasticsearch:
        repositories:
          enabled: true
        properties:
          index:
            setting-path: classpath:elasticsearch/settir
```

```
server:
  port: 8080
  forward-headers-strategy: framework
  servlet:
    context-path: /api

logging:
  level:
    root: INFO    # 전체 애플리케이션의 기본 로그 레벨 설정
    com.d209.welight: DEBUG   # 특정 패키지에 대한 로그 레벨
    org.springframework.web: INFO   # 스프링 웹 관련 로그 러
```

- **application-db.yml**

```
spring:

    # MySQL
  datasource:
    url: jdbc:mysql://3.34.189.155:3306/WELIGHT?serverTi
    username: superuser
    password: 8q9rDD5VYqQeBYqNcKfc
    hikari:
      maximum-pool-size: 10
      idle-timeout: 300000
      minimum-idle: 5

  # Redis
  data:
    redis:
      host: 3.34.189.155
      port: 6379
      password: RnJVExwX9bqGdam8e89E
      repositories:
        enabled: false
```

- **application-s3.yml**

```
# S3
cloud:
  aws:
    s3:
      bucket: ssafy-gumi02-d209
    credentials:
      access-key: AKIAXNGUVGRET22PWP5P
      secret-key: dQFQCnJ9zKXx7o1hTGi89lYX9VcverNKfqsi7e
    region:
      static: ap-northeast-2
    stack:
      auto: false
```

- **application-security.yml**

```
cors:
  allowed-origins: 'https://k11d209.p.ssafy.io'
  # allowed-origins: 'https://welight.online:8081'
  allowed-methods: GET,POST,PUT,DELETE,OPTIONS
  allowed-headers: '*'
  max-age: 3600

jwt:
  secret-key: ssafy+gumi02+D209+welight+jwt+ssafy+gumi02
  access-token:
    expiretime: 36000000 # 10시간
  #    expiretime: 2592000000 # 30일
  refresh-token:
    expiretime: 2592000000 # 30일
```

📌 배포 환경 설정

0. 초기 세팅

▼ Deploy Script

```
#!/bin/bash

# Docker 사용자명 환경변수 설정
DOCKER_USERNAME="d209"

# 로그 파일 경로 설정
LOG_FILE="/var/log/deploy.log"

# 로그 메시지 출력 함수
log_message() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" | tee -a $LOG
}

# 배포 시작 로그
log_message "배포 프로세스 시작"

# Docker 컨테이너 중지 및 삭제
log_message "기존 Docker 컨테이너 정리 중..."
docker stop spring || true
docker rm spring || true

# 최신 이미지 가져오기
log_message "Docker 이미지 풀링 중..."
docker pull ${DOCKER_USERNAME}/welight-backend-spring:late

# 새 컨테이너 실행
log_message "새 Docker 컨테이너 실행 중..."
docker run -d \
    --name spring \
    --network welight-network \
    -p 8080:8080 \
    ${DOCKER_USERNAME}/welight-backend-spring:latest

docker network connect elastic-network spring
# 사용하지 않는 이미지 정리
log_message "미사용 Docker 이미지 정리 중..."
docker image prune -f
```

```
# 배포 완료 로그
log_message "배포 프로세스 완료"
```

1. Docker

▼ CICD

```
services:
  jenkins:
    image: jenkins/jenkins:lts
    container_name: jenkins
    restart: unless-stopped
    ports:
      - "9090:8080"
    volumes:
      - jenkins_data:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock
      - /usr/bin/docker:/usr/bin/docker
    environment:
      TZ: 'Asia/Seoul'
      JENKINS_OPTS: --prefix=/jenkins
    networks:
      - welight-network

  mysql:
    image: mysql:8.0.38
    container_name: mysql
    restart: unless-stopped
    ports:
      - "3306:3306"
    volumes:
      - mysql_data:/var/lib/mysql
      - ./mysql/init:/docker-entrypoint-initdb.d
    env_file:
      - .env
    environment:
      TZ: 'Asia/Seoul'
    command:
```

```
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci
      - --pid-file=/var/lib/mysql/mysqld.pid
    networks:
      - welight-network

  redis:
    image: redis:7.2
    container_name: redis
    restart: unless-stopped
    ports:
      - "6379:6379"
    volumes:
      - redis_data:/data
    command: >
      --requirepass ${REDIS_PASSWORD}
      --appendonly yes
    env_file:
      - .env
    environment:
      TZ: 'Asia/Seoul'
    networks:
      - welight-network

  fastapi:
    build:
      context: ./fastapi
    container_name: fastapi
    restart: unless-stopped
    ports:
      - "8000:8000"
    environment:
      TZ: 'Asia/Seoul'
    networks:
      - welight-network

  nginx:
    image: nginx:latest
```

```
    container_name: nginx
    restart: always
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx/conf/:/etc/nginx/conf.d/:rw
      - ./certbot/www:/var/www/certbot
      - ./certbot/conf:/etc/letsencrypt
    networks:
      - welight-network
      # - elastic-network

  certbot:
    image: certbot/certbot
    container_name: certbot
    volumes:
      - ./certbot/www:/var/www/certbot
      - ./certbot/conf:/etc/letsencrypt
    networks:
      - welight-network

volumes:
  jenkins_data:
  mysql_data:
  redis_data:

networks:
  welight-network:
    external: true
  # elastic-network:
  #   external: true
```

▼ ELK - docker-compose.yml

```
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7
```

```
    container_name: elasticsearch
    build:
      context: .
      dockerfile: Dockerfile
    environment:
      - discovery.type=single-node
      - xpack.security.enabled=false
      - "ES_JAVA_OPTS=-Xms1g -Xmx1g"
      - bootstrap.memory_lock=true
      - "cluster.name=docker-cluster"
      - "network.host=0.0.0.0"
    ulimits:
      memlock:
        soft: -1
        hard: -1
    ports:
      - "9200:9200"
    volumes:
      - elasticsearch-data:/usr/share/elasticsearch/data
    networks:
      - elastic-network

  logstash:
    image: docker.elastic.co/logstash/logstash:7.17.12
    container_name: logstash
    volumes:
      - ./logstash/config/:/usr/share/logstash/config/
      - ./logstash/pipeline:/usr/share/logstash/pipeline
    ports:
      - "5044:5044"
      - "5000:5000/tcp"
      - "5000:5000/udp"
      - "9600:9600"
    depends_on:
      - elasticsearch
    networks:
      - elastic-network
```

```
  kibana:
    image: docker.elastic.co/kibana/kibana:7.17.12
    container_name: kibana
    ports:
      - "5601:5601"
    environment:
      ELASTICSEARCH_URL: http://elasticsearch:9200
      ELASTICSEARCH_HOSTS: http://elasticsearch:9200
    depends_on:
      - elasticsearch
    networks:
      - elastic-network

volumes:
  elasticsearch-data:

networks:
  elastic-network:
    external: true
```

▼ ELK - Dockerfile

```
FROM docker.elastic.co/elasticsearch/elasticsearch:7.17.12
```

▼ WebSocket Server

```
services:
  websocket-server:
    build:
      context: ./websocket-server
      dockerfile: Dockerfile
    container_name: websocket-server
    restart: unless-stopped
    ports:
      - "9000:9000"
    volumes:
      - ./websocket-server:/usr/src/app
    environment:
```

```
      NODE_ENV: production
    networks:
      - welight-network

  vue-app:
    build:
      context: ./vue-app
      dockerfile: Dockerfile
    container_name: vue-app
    restart: unless-stopped
    ports:
      - "3000:80"
    volumes:
      - ./vue-app:/usr/src/app
    environment:
      NODE_ENV: production
    networks:
      - welight-network

networks:
  welight-network:
    external: true
```

▼ Fast API Server

```
# Dockerfile
FROM python:3.11-slim

WORKDIR /app

# 시스템 의존성 설치
RUN apt-get update && apt-get install -y --no-install-reco
    build-essential \
    pkg-config \
    libmariadb-dev-compat \
    libmariadb-dev \
    libssl-dev \
    libffi-dev \
```

```
    libjpeg-dev \
    zlib1g-dev \
    libglib2.0-0 \
    libsm6 \
    libxext6 \
    libxrender-dev \
    && rm -rf /var/lib/apt/lists/*

# requirements.txt 먼저 복사하여 캐시 활용
COPY requirements.txt ./

# pip 업그레이드 및 Python 패키지 설치
RUN pip install --upgrade pip && \
    pip install --no-cache-dir -r requirements.txt

# requirements.txt에 명시된 모든 패키지 설치
# RUN pip install --no-cache-dir -r requirements.txt

# 애플리케이션 코드 복사
COPY . /app

# 컨테이너 실행 시 실행할 명령어 설정
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port",
```

2. Nginx

▼ welight.conf

```
server {
    listen 80;
    server_name k11d209.p.ssafy.io;

    location / {
        return 301 https://$host$request_uri;
    }

    location /.well-known/acme-challenge/ {
        root /var/www/certbot;
```

```
    }
}

server {
    listen 443 ssl;
    server_name k11d209.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/k11d209.p.ssafy.
    ssl_certificate_key /etc/letsencrypt/live/k11d209.p.ss

    location / {
        root /usr/share/nginx/html;
        index index.html index.htm;
    }

    # jenkins
    location /jenkins/ {
        proxy_pass http://jenkins:8080/jenkins/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    # backend - spring
    location /api/ {
        error_page 502 = @fallback;
        proxy_pass http://spring:8080/api/;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-Host $server_name;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location @fallback {
        return 200 "Spring 서버가 준비되지 않았습니다. 잠시 후 다
    }
```

```
    # WebSocket 서버 프록시 설정
    location /socket.io/ {
        proxy_pass http://websocket-server:9000;  # WebSoc
        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_forw
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    # Vue 애플리케이션 프록시 설정
    location /app/ {
        proxy_pass http://vue-app:80;  # Vue 앱 컨테이너 이름
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    # FastAPI 애플리케이션 프록시 설정 (AI URL)
    location /ai/ {
        proxy_pass http://fastapi:8000/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forw
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

3. Jenkins

▼ Pipeline Script

```
pipeline {
    agent any
```

```
    environment {
        DOCKER_CREDENTIALS = credentials('DOCKER_USER')
        DOCKER_PROJECT = 'welight-backend-spring'
        EC2_SERVER_IP = credentials('EC2_SERVER_IP')
    }

    stages {
        stage('Checkout') {
            steps {
                echo "Starting Checkout Stage..."

                git branch: 'develop',
                credentialsId: 'gitlab',
                url: 'https://lab.ssafy.com/s11-final/S11P

                echo "Checkout completed."
            }
        }

        stage('Copy') {
            steps {
                echo "Starting Copy Stage..."

                sh '''
                    chmod -R u+w /var/jenkins_home/workspa
                '''

                withCredentials([file(credentialsId: 'APPF
                    script {
                        sh 'rm /var/jenkins_home/workspace
                        sh 'cp $APPFILE /var/jenkins_home/
                    }
                }

                withCredentials([file(credentialsId: 'DBFI
                    script {
                        sh 'cp $DBFILE /var/jenkins_home/w
                    }
```

```
                }

                withCredentials([file(credentialsId: 'S3FI
                    script {
                        sh 'cp $S3FILE /var/jenkins_home/w
                    }
                }

                withCredentials([file(credentialsId: 'SECU
                    script {
                        sh 'cp $SECURITYFILE /var/jenkins_
                    }
                }


                echo "Copy completed."
            }
        }

        stage('Build') {
            steps {
                echo "Starting Build Stage..."

                dir('/var/jenkins_home/workspace/test-pipe
                    sh 'pwd'
                    sh 'ls -al'
                    sh 'chmod +x ./gradlew'
                    sh 'chmod +x ./gradlew.bat'
                    sh 'java --version'
                    sh './gradlew clean build'
                }

                echo "Build completed."
            }
        }

        stage('Test') {
            steps {
```

```
                echo "Starting Test Stage..."
                echo "Test completed."
            }
        }

        stage('Deploy') {
            steps {
                echo "Starting Deploy Stage..."

                sh '''
                    echo $DOCKER_CREDENTIALS_PSW | docker
                '''

                sh """
                    cd ./Backend
                    docker build -t ${DOCKER_CREDENTIALS_U
                    docker push ${DOCKER_CREDENTIALS_USR}/
                """

                sshagent(['SSH_KEY']) {
                    sh '''
                        chmod 600 ~/.ssh/id_rsa
                        ssh -o StrictHostKeyChecking=no ub
                    '''
                }

                echo "Deploy completed."
            }
        }
    }
    post {
        always {
            script {
                def Author_ID = sh(script: "git show -s --
                def Author_Name = sh(script: "git show -s
                def Commit_Message = sh(script: "git log -
                def Build_Status = currentBuild.result ?:
                def Status_Color = Build_Status == 'SUCCES
```

```
                def Status_Text = Build_Status == 'SUCCESS
                def branchName = sh(script: "git rev-parse
                def previousCommit = env.GIT_PREVIOUS_SUCC
                def allCommits = sh(script: "git log --pre
                def formattedCommits = allCommits.split('\
                    def escapedLine = line.replaceAll("([\
                    "• ${escapedLine}"
                }.join('\\n')
                def message = """
                    #### BE $Status_Text
                    **빌드 번호:** $env.JOB_NAME #$env.BUILD
                    **브랜치:** $branchName
                    **작성자:** $Author_ID ($Author_Name)
                    **빌드 URL:** [Details]($env.BUILD_URL)
                    **포함된 커밋:**
                    $formattedCommits
                """.stripIndent()
                mattermostSend(
                    color: Status_Color,
                    message: message,
                    endpoint: 'https://meeting.ssafy.com/h
                    channel: 'D209-Jenkins-BOT',
                )
            }
        }
    }
}
```

📌 DB 덤프

DBdump_welight.sql

📌 프로젝트에 사용된 외부 서비스

외부 저장소

- S3
  - 사용자 프로필 사진 저장
  - 디스플레이 썸네일, 이미지 저장