포팅매뉴얼

버전

- Backend
    - Java: 17
    - ▼ SpringBoot: 3.4.3

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.4.3'
    id 'io.spring.dependency-management' version '1.1.7'
}

group = 'com.ssafy'
version = '0.0.1-SNAPSHOT'

java {
    toolchain {
        languageVersion = JavaLanguageVersion.of(17)
    }
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}
```

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web
    compileOnly 'org.projectlombok:lombok'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-t
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'

    // DB
    runtimeOnly 'com.mysql:mysql-connector-j'
    implementation 'org.springframework.boot:spring-boot-starter-data
//    implementation 'org.springframework.boot:spring-boot-starter-da

    // Security
    implementation 'org.springframework.boot:spring-boot-starter-secu
    implementation 'org.springframework.boot:spring-boot-starter-oaut
    testImplementation 'org.springframework.security:spring-security-t

    // JWT
    implementation 'io.jsonwebtoken:jjwt-api:0.12.3'
    runtimeOnly 'io.jsonwebtoken:jjwt-impl:0.12.3'
    runtimeOnly 'io.jsonwebtoken:jjwt-jackson:0.12.3'

    // dot env
    implementation 'me.paulschwarz:spring-dotenv:4.0.0'

    // queryDSL 설정
    implementation "com.querydsl:querydsl-jpa:5.0.0:jakarta"
    implementation "com.querydsl:querydsl-core"
    implementation "com.querydsl:querydsl-collections"
    annotationProcessor "com.querydsl:querydsl-apt:${dependencyMa

    // querydsl JPAAnnotationProcessor 사용 지정
    annotationProcessor "jakarta.annotation:jakarta.annotation-api"

    // java.lang.NoClassDefFoundError (javax.annotation.Generated) 대
    annotationProcessor "jakarta.persistence:jakarta.persistence-api"
    // java.lang.NoClassDefFoundError (javax.annotation.Entity) 대응 코
    // json
    implementation group: 'org.json', name: 'json', version: '20231013'
}

tasks.named('test') {
    useJUnitPlatform()
}

// Querydsl 설정부
def generated = 'src/main/generated'

// querydsl QClass 파일 생성 위치를 지정
tasks.withType(JavaCompile) {
    options.getGeneratedSourceOutputDirectory().set(file(generated))
}

// java source set 에 querydsl QClass 위치 추가
sourceSets {
    main.java.srcDirs += [generated]
}

// gradle clean 시에 QClass 디렉토리 삭제
clean {
    delete file(generated)
}
```

- Frontend
  - yarn

▼ package.json

```
{
  "name": "frontend",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint .",
    "preview": "vite preview"
  },
  "dependencies": {
    "@tailwindcss/vite": "^4.0.14",
    "axios": "^1.8.4",
    "date-fns": "^4.1.0",
    "lucide-react": "^0.486.0",
    "qs": "^6.14.0",
    "react": "^19.0.0",
    "react-dom": "^19.0.0",
    "react-hot-toast": "^2.5.2",
    "react-icons": "^5.5.0",
    "react-paginate": "^8.3.0",
    "react-router-dom": "^7.4.0",
    "react-time-picker": "^7.0.0",
    "styled-components": "^6.1.17",
    "tailwindcss": "^4.0.14",
    "zustand": "^5.0.3"
  },
  "devDependencies": {
    "@eslint/js": "^9.21.0",
    "@types/react": "^19.0.10",
    "@types/react-dom": "^19.0.4",
    "@vitejs/plugin-react-swc": "^3.8.0",
    "eslint": "^9.23.0",
    "eslint-config-airbnb": "^19.0.4",
    "eslint-config-prettier": "^10.1.1",
    "eslint-plugin-import": "^2.31.0",
    "eslint-plugin-jsx-a11y": "^6.10.2",
    "eslint-plugin-prettier": "^5.2.4",
    "eslint-plugin-react": "^7.37.4",
    "eslint-plugin-react-hooks": "^5.2.0",
    "eslint-plugin-react-refresh": "^0.4.19",
    "globals": "^15.15.0",
    "prettier": "^3.5.3",
    "vite": "^6.2.0"
  }
}
```

- Nginx: nginx/1.27.1.1 (Ubuntu)
- Docker: Docker version 28.0.2, build 0442a73
- IDEs
  - Java: IntelliJ
  - JSX (JavaScript XML): VScode

환경변수 및 외부 서비스

- Java Properties Files
  - MariaDB, MongoDB 정보 포함
  - GoogleLogin 정보 포함
  - ▼ application.yml

```
server:
  forward-headers-strategy: native
  tomcat:
    remote-ip-header: x-forwarded-for
    protocol-header: x-forwarded-proto

spring:
  config:
    import: optional:file:.env[.properties]

  datasource:
    url: ${DB_URL}
    username: ${DB_USER}
    password: ${DB_PASSWORD}
    driver-class-name: com.mysql.cj.jdbc.Driver

  #  data:
```

```
  #    mongodb:
  #      uri: ${NOSQL_URI}

  jpa:
    hibernate:
      ddl-auto: update
    show-sql: true
    properties:
      hibernate:
        format_sql: true
  #    defer-datasource-initialization: true
  datasource.hikari:
    connection-init-sql: SET NAMES utf8mb4

  #  sql:
  #    init:
  #      mode: always

  security:
    oauth2:
      client:
        registration:
          google:
            client-id: ${GOOGLE_CLIENT_ID}
            client-secret: ${GOOGLE_CLIENT_SECRET}
#             redirect-uri: https://j12a408.p.ssafy.io/login/oauth2/code/goo
            scope:
              - profile
              - email
        provider:
          google:
            authorization-uri: https://accounts.google.com/o/oauth2/auth
            token-uri: https://oauth2.googleapis.com/token
            user-info-uri: https://www.googleapis.com/oauth2/v3/userinfo
            user-name-attribute: email

logging.level:
  show_sql: debug
  org.hibernate.type: trace
```

▼ .env

```
# Maria DB
DB_URL=jdbc:mysql://stg-yswa-kr-practice-db-master.mariadb.datab
DB_USER=S12P22A408@stg-yswa-kr-practice-db-master
DB_PASSWORD=t9hurA0fqX
NOSQL_URI=mongodb+srv://S12P22A408:wr4KWJ8ypg@ssafy.ngivl.

# Google OAuth2 환경 변수
GOOGLE_CLIENT_ID=914951230029-aliiq3khj5vlcdejd84543rbh516un
GOOGLE_CLIENT_SECRET=GOCSPX-6ayvglkl0WkwB_mNmO5ADO43
```

- FastAPI

▼ .env

```
SECRET_KEY = 5cBwDJ67eufBN1ZOeClQwYMcSwUe6WGKU8nBzYk\
```

CI/CD 구성

설치

- 서버에 docker 설치
- 서버에 Nginx 설치

실행

- EC2 서버에 Docker와 Jenkins를 직접 설치
- 시스템 서비스(systemd)로 구동
- Jenkins Pipeline을 구성해 Gitlab 소스를 감지하고 Docker Compose로 서비스 자동 배포

Frontend

▼ docker-compose.yml

```
version: '3'
services:
  frontend:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: frontend
    ports:
      - "3000:80"
    restart: unless-stopped
    networks:
      - proxy_network

networks:
  proxy_network:
    external: true
```

▼ Dockerfile

```
FROM node:20-alpine as builder
WORKDIR /app
COPY package*.json ./
COPY yarn.lock ./

RUN yarn install
COPY . .
RUN yarn build

# nginx을 베이스 이미지로 사용
FROM nginx:stable-alpine AS production
COPY --from=builder /app/dist /usr/share/nginx/html
EXPOSE 80
```

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

Backend

▼ docker-compose.yml

```
version: "3.8"
services:
  backend:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: backend
    ports:
      - "8080:8080"
    env_file:
      - .env
    environment:
      SPRING_PROFILES_ACTIVE: dev
      TZ: Asia/Seoul
    restart: always
    networks:
      - proxy_network
networks:
  proxy_network:
    external: true
```

▼ Dockerfile

```
# 1. Build Stage (프로젝트에 포함된 Wrapper 사용)
FROM gradle:jdk17-alpine As builder
WORKDIR /build
COPY . .
RUN ./gradlew bootJar
```

```
# 2. Run Stage
FROM openjdk:17-jdk-slim
WORKDIR /app

# 3. 타임존 설정
ENV TZ=Asia/Seoul
RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/tin

# 4. 빌드된 JAR 복사
COPY --from=builder /build/build/libs/*.jar app.jar

# 5. 컨테이너 포트 공개
EXPOSE 8080

# 6. 실행 명령어
ENTRYPOINT ["java", "-jar", "-Duser.timezone=Asia/Seoul", "/app/app.jar"
```

AI

▼ docker-compose.yml

```
version: "3.8"

services:
  ai-service:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: ai
    image: ai-service # 명시 안 해도 됨
    ports:
      - "8000:8000"
    restart: unless-stopped
    volumes:
      - ./google_keys.json:/app/google_keys.json:ro
      - ./env/.env:/app/.env:ro
    networks:
```

```
   - proxy_network

networks:
  proxy_network:
    external: true
# test
```

▼ Dockerfile

```
FROM python:3.10-slim

# 작업 디렉토리 생성
WORKDIR /app

# 기본 패키지 설치
RUN apt-get update && \
    apt-get install -y --no-install-recommends \
        tzdata \
        g++ \
        git \
        curl \
        default-jdk \
        default-jre \
        make \
        sudo \
        unzip \
    && apt-get clean

# locale 설정 (한글 데이터 인코딩 오류 방지)
ENV LANG=C.UTF-8
ENV LANGUAGE=C.UTF-8
ENV LC_ALL=C.UTF-8

# requirements 복사 및 설치
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# mecab start
```

```
RUN curl -s https://raw.githubusercontent.com/konlpy/konlpy/master/scrip
# mecab end

# 전체 소스 복사
COPY . .

# uvicorn 실행
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```