데브웨이 포팅매뉴얼 & 리드미

상태	산출물

1. 개요

1-1. 프로젝트 개요

- 개발자들이 복잡한 코드 구현에 시간을 쏟지 않고 효율적으로 쉽게 사용할 수 있는 OpenAPI를 제공하고, 그를 활용해 나만의 서비스를 쉽게 구성할 수 있도록

1-2. 개발 환경

Backend

- **Java** : Oracle Open JDK 17
- **Spring Boot** : 3.2.3
- **JPA** : hibernate-core-6.4.1
- **DB :** MySQL
- **IntelliJ :** 2023. 3

1-3. 프로젝트 사용도구

- 이슈 / 형상 관리 : Gitlab
- 코드 리뷰 : Gitlab
- 커뮤니케이션 : Notion, Mattermost
- 디자인 : Figma
- UCC : Movavi

1-4. 외부 서비스

- ChatGPT API
- Google Cloud API
    - STT
    - TTS
- Open Weather Map API
    - Current Weather Data
    - 5 Day / 3 Hour Forecast
- 네이버 검색 API
    - 블로그
    - 뉴스
    - 백과사전
    - 지역
    - 이미지
    - 오타변환
- 카카오 검색 API
    - 책

- 한국수출입은행 API
  - 환율

1-5. Gitignore 정보

루트 디렉토리 위치 : `.env`

---

2. 빌드

2-1. 환경 변수

2-1-1. 사과 서비스 Spring

- **application.yml**

```
server:
  port: 8070

spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: ${MYSQL_DATASOURCE_URL}
    username: ${MYSQL_DATASOURCE_USERNAME}
    password: ${MYSQL_DATASOURCE_PASSWORD}


  jpa:
    hibernate:
      ddl-auto: update
    show-sql: false
    database: mysql
    database-platform: org.hibernate.dialect.MySQL8Dialect
    properties:
      hibernate:
        format_sql: true

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

notification:
  mattermost:
    enabled: true
    webhook-url: ${MATTERMOST_URL}
    channel:

api:
  weather:
    key: ${API_WHETHER_KEY}

img:
  path:
    db: "https://k10b201.p.ssafy.io/static/"
    user: "/app/static/"
```

2-1-2. 오린지 서비스 Spring

```
server:
  port: 8050

spring:
  data:
    mongodb:
      uri: ${MONGO_INIT_DB_ROOT_URL}
      database: ${MONGO_INIT_DB_DATABASE}
  servlet:
    multipart:
      max-file-size: 5MB
      max-request-size: 5MB

notification:
  mattermost:
    enabled: true
    webhook-url: ${MATTERMOST_URL}
    channel:
    color: green

forward-headers-strategy: framework
```

2-1-3. 체리톡 서비스 Spring

```
server:
  port: 8060
```

```
spring:
  config:
    import: optional:file:.env[.properties]
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: ${MYSQL_DATASOURCE_URL}
    username: ${MYSQL_DATASOURCE_USERNAME}
    password: ${MYSQL_DATASOURCE_PASSWORD}
  jpa:
    database-platform: org.hibernate.dialect.MySQLDialect
    hibernate:
      ddl-auto: update
  servlet:
    multipart:
      max-file-size: 10MB
      max-request-size: 10MB
```

2-2. 빌드

React

1. **Node.js 설치**
2. **빌드 명령어 입력**

```
npm install --force
```

3. 실행

```
npm start
```

Spring

- **build.gradle 실행**

Android

1. **안드로이드 스튜디오 실행**
2. **메뉴 > Build > Build Bundle(s) / APK(s) > Build APK(s)**

---

3. 배포

- **Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1055-aws x86_64)**
- **Linux 기준**

1. **git 설치**

```
sudo apt-get install git

sudo apt install git

# 버전확인
git --version
```

2. **프로젝트 clone**

```
git clone project's repository (사과, 오린지, 체리톡, 홈페이지)
```

3. **docker 설치 - 공식문서 참조**

```
sudo apt-get update
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
```

4. **docker-compose 설치**

```
curl -L "https://github.com/docker/compose/releases/download/1.25.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
```

5. **Dockerfile-nginx**

```
# Use the official Nginx image
FROM nginx:alpine

# Remove the default Nginx configuration file
RUN rm /etc/nginx/conf.d/default.conf

# Copy a new configuration file from your project to the container
COPY nginx.conf /etc/nginx/nginx.conf
#COPY nginx.conf /devway/nginx.conf

# Expose port 80
```

```
#EXPOSE 80 443

# Start Nginx when the container has provisioned
CMD ["nginx", "-g", "daemon off;"]
```

6. **Dockerfile** (spring)

```
# Use an official OpenJDK runtime as a parent image
FROM openjdk:17

RUN microdnf install findutils

# Set the working directory in the container
WORKDIR /app

# Copy the entire project directory (including gradlew and gradle/) into the working directory
COPY . .

# Ensure the Gradle wrapper script is executable
RUN chmod +x ./gradlew

# Run the Gradle build
RUN ./gradlew build && ls -la build/libs/

# Remove any unwanted jars
RUN rm build/libs/*-plain.jar

# Copy the JAR file from build output to a simpler path
RUN cp build/libs/*SNAPSHOT.jar app.jar

# Run the JAR file
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app/app.jar"]
```

7. **Dockerfile** (React)

```
FROM node:16-alpine3.11

WORKDIR /app

COPY package*.json ./

RUN npm install --force

COPY . .

EXPOSE 3000

CMD [ "npm", "start" ]
```

8. **nginx.conf**

```
user  nginx;
worker_processes  1;
error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;
events {
    worker_connections  1024;
}
http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;
    client_max_body_size 10M;  # 전체 http 블록에 적용

    server {
                listen 80;
                server_name k10b201.p.ssafy.io; # 발급한 도메인 주소
                server_tokens off;

                location /.well-known/acme-challenge/ {
                    root /var/www/certbot; # Certbot을 통해 Let's Encrypt 인증서를 발급받을 때 사용하는 경로
                }

            location /oringe/api {
                        proxy_pass http://app:8050;
                        proxy_redirect off;
                        proxy_set_header Host $host;
                        proxy_set_header X-Real-IP $remote_addr;
                        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            }

            location /cherry/api {
                        proxy_pass http://app_cherry:8060;
                        proxy_redirect off;
                        proxy_set_header Host $host;
                        proxy_set_header X-Real-IP $remote_addr;
```

```
                        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }

        location /sagwa/api {
                        proxy_pass http://app_sagwa:8070;
                        proxy_redirect off;
                        proxy_set_header Host $host;
                        proxy_set_header X-Real-IP $remote_addr;
                        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }

        location /static {
                        alias /usr/share/nginx/html/static;
        }
    }

    server {
        listen 443 ssl;
        server_name k10b201.p.ssafy.io;

        ssl_certificate /etc/letsencrypt/live/k10b201.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/k10b201.p.ssafy.io/privkey.pem;

        location /oringe/api {
            proxy_pass http://app:8050;
            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;
        }

        location /cherry/api {
            proxy_pass http://app_cherry:8060;
            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;
        }

        location /sagwa/api {
                                proxy_pass http://app_sagwa:8070;
                                proxy_redirect off;
                                proxy_set_header Host $host;
                                proxy_set_header X-Real-IP $remote_addr;
                                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            }

            location /static {
                alias /usr/share/nginx/html/static;
            }
    }

    server {
        listen 80;
        server_name www.devway.kr;

        location /.well-known/acme-challenge/ {
                            root /var/www/certbot; # Certbot을 통해 Let's Encrypt 인증서를 발급받을 때 사용하는 경로
                        }

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

    server {
        listen 443 ssl;
        server_name www.devway.kr;

        ssl_certificate /etc/letsencrypt/live/www.devway.kr/fullchain.pem; # Ensure you have these certificates
        ssl_certificate_key /etc/letsencrypt/live/www.devway.kr/privkey.pem;

        location / {
            proxy_pass http://app_devway:3000;  # Adjust if you have a specific app or service for this server
            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;
        }

        # Define other locations if needed
    }

    sendfile        on;
    keepalive_timeout  65;
```

```
    include /etc/nginx/conf.d/*.conf;
}
```

3-1. 수동 배포

- **docker-compose.yml의 디렉토리에서 아래 명령어를 실행한다.**

```
sudo docker-compose up -d
```

3-1-1. 사과 서비스 docker-compose.yml

```
version: '3.7'

services:
  app_sagwa:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "8070:8070"
    environment:
      - TZ=Asia/Seoul
    volumes:
      - ./src/main/resources/static:/app/static
    networks:
      - devway_network

  mysql_sagwa:
    image: mysql
    environment:
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      TZ: Asia/Seoul
    volumes:
      - mysql_data_sagwa:/var/lib/mysql
    ports:
      - "3307:3306"
    networks:
      - devway_network

volumes:
  mysql_data_sagwa:

networks:
  devway_network:
    external: true
```

3-1-2. 오린지 서비스 docker-compose.yml

```
version: '3.7'

services:
  app:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "8050:8050"
    environment:
      - TZ=Asia/Seoul
      - GOOGLE_APPLICATION_CREDENTIALS=/credentials/MyGC.json
    volumes:
      - /home/ubuntu/oringe/devway/src/main/resources/static:/app/static
      - /home/ubuntu/MyGC.json:/credentials/MyGC.json
    networks:
      - devway_network
    depends_on:
      - mongo

  nginx:
    build:
      context: .
      dockerfile: Dockerfile-nginx
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
```

```
      - /home/ubuntu/apple/devway/src/main/resources/static:/usr/share/nginx/html/sagwa_static
      - /home/ubuntu/oringe/devway/src/main/resources/static:/usr/share/nginx/html/oringe_static
      - /home/ubuntu/certbot/www:/var/www/certbot
      - /home/ubuntu/certbot/conf:/etc/letsencrypt
    depends_on:
      - app
    restart: unless-stopped
    environment:
      - TZ=Asia/Seoul
    networks:
      - devway_network

  certbot:
    image: certbot/certbot
    volumes:
      - /home/ubuntu/certbot/www:/var/www/certbot
      - /home/ubuntu/certbot/conf:/etc/letsencrypt
    entrypoint: "/bin/sh -c 'trap exit TERM; while :; do certbot renew; sleep 12h & wait $${!}; done;'"
    depends_on:
      - nginx
    networks:
      - devway_network

  mongo:
    image: mongo:latest
    container_name: mongodb
    ports:
      - "27017:27017"
    volumes:
      - mongo_data:/data/db
    environment:
      MONGO_INITDB_ROOT_USERNAME: ${MONGO_INITDB_ROOT_USERNAME}
      MONGO_INITDB_ROOT_PASSWORD: ${MONGO_INITDB_ROOT_PASSWORD}
      MONGO_INITDB_DATABASE: ${MONGO_INITDB_ROOT_DATABASE}
      TZ: Asia/Seoul
    restart: unless-stopped
    networks:
      - devway_network

  jenkins:
    image: jenkins/jenkins:lts
    container_name: jenkins
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /jenkins:/var/jenkins_home
      - /home:/home # 호스트의 /home 디렉토리를 컨테이너의 /home에 마운트
    ports:
      - 8080:8080
    privileged: true
    user: root
    environment:
      - TZ=Asia/Seoul
    networks:
      - devway_network

volumes:
  mongo_data:

networks:
  devway_network:
    external: true
```

3-1-3. 체리톡 서비스 docker-compose.yml

```
version: '3.7'

services:
  app_cherry:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "8060:8060"
    environment:
      - TZ=Asia/Seoul
      - GOOGLE_APPLICATION_CREDENTIALS=/app/MyGC.json
    networks:
      - devway_network
    volumes:
      - /home/ubuntu/MyGC.json:/app/MyGC.json

  mysql:
    image: mysql
    environment:
```

```
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    volumes:
      - mysql_data:/var/lib/mysql
    ports:
      - "3306:3306"
    networks:
      - devway_network

volumes:
  mysql_data:

networks:
  devway_network:
    external: true
```

3-1-4. devway 홈페이지 docker-compose.yml

```
version: '3.7'

services:
  app_devway:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "3000:3000"
    environment:
      - TZ=Asia/Seoul
    networks:
      - devway_network

networks:
  devway_network:
    external: true
```

3-2. 자동 배포 (CI/CD)

- **Jenkins 파일은 각 프로젝트 최상단에 위치한다**
- **각 프로젝트의 release에 push hook 발생 시 CI/CD 작업 진행**
- **CI 작업을 위해 아래와 같은 설정을 진행한다.**

```
// Jenkins 컨테이너 접근
sudo docker exec -it jenkins /bin/bash
or
sudo docker-compose exec jenkins /bin/bash

// Jenkins git safe directory 추가
git config --global --add safe.directory project경로
```

3-2-1. 사과 서비스 JenkinsFile

```
pipeline {
    agent any
    environment {
        DOCKER_COMPOSE_VERSION = '1.25.0' // 사용할 Docker Compose의 버전
        GITLAB_TOKEN = credentials('wns1915_sagwa') // Jenkins에 저장된 GitLab Token의 ID
    }
    stages {
        stage('Checkout') {
            steps {
                git branch: 'release', credentialsId: 'wns1915_sagwa', url: 'https://lab.ssafy.com/ztjdwnz/apple.git' // GitLab 리포지토리
            }
        }
        stage('Update Local Repository') {
            steps {
                script {
                        withCredentials([usernamePassword(credentialsId: 'wns1915_sagwa', usernameVariable: 'GIT_USERNAME', passwordVariable: 'GIT_PASSWORD')
                            sh '''
                            ENCODED_USERNAME=$(echo $GIT_USERNAME | sed 's/@/%40/g')
                            cd /home/ubuntu/apple
                            git pull https://$ENCODED_USERNAME:$GIT_PASSWORD@lab.ssafy.com/ztjdwnz/apple.git release
                            '''
                        }
                }
            }
        }
        stage('Build Docker Images') {
            steps {
                script {
```

```
                    sh 'docker-compose -f /home/ubuntu/apple/devway/docker-compose.yml build --no-cache app_sagwa'
                    sh 'docker-compose -f /home/ubuntu/apple/devway/docker-compose.yml up -d app_sagwa'
                }
            }
        }
        stage('Deploy') {
            steps {
                script {
                    sh 'docker-compose -f /home/ubuntu/oringe/devway/docker-compose.yml build --no-cache nginx '
                    sh 'docker-compose -f /home/ubuntu/oringe/devway/docker-compose.yml up -d nginx '
                    sh 'docker-compose -f /home/ubuntu/oringe/devway/docker-compose.yml up -d certbot'
                }
            }
        }
    }
}
```

3-2-2. 오린지 서비스 JenkinsFile

```
pipeline {
    agent any
    environment {
        DOCKER_COMPOSE_VERSION = '1.25.0' // 사용할 Docker Compose의 버전
        GITLAB_TOKEN = credentials('wns1915') // Jenkins에 저장된 GitLab Token의 ID
    }
    stages {
        stage('Checkout') {
            steps {
                git branch: 'release', credentialsId: 'wns1915', url: 'https://lab.ssafy.com/wns1915/oringe.git' // GitLab 리포지토리
            }
        }
        stage('Update Local Repository') {
            steps {
                script {
                        withCredentials([usernamePassword(credentialsId: 'wns1915', usernameVariable: 'GIT_USERNAME', passwordVariable: 'GIT_PASSWORD')]) {
                            sh '''
                            ENCODED_USERNAME=$(echo $GIT_USERNAME | sed 's/@/%40/g')
                            cd /home/ubuntu/oringe
                            git pull https://$ENCODED_USERNAME:$GIT_PASSWORD@lab.ssafy.com/wns1915/oringe.git release
                            '''
                        }
                }
            }
        }
        stage('Build Docker Images') {
            steps {
                script {
                    sh 'docker-compose -f /home/ubuntu/oringe/devway/docker-compose.yml build --no-cache app'
                    sh 'docker-compose -f /home/ubuntu/oringe/devway/docker-compose.yml up -d app'
                }
            }
        }
        stage('Deploy') {
            steps {
                script {
                    sh 'docker-compose -f /home/ubuntu/oringe/devway/docker-compose.yml build --no-cache nginx '
                    sh 'docker-compose -f /home/ubuntu/oringe/devway/docker-compose.yml up -d nginx '
                    sh 'docker-compose -f /home/ubuntu/oringe/devway/docker-compose.yml up -d certbot'
                }
            }
        }
    }
}
```

3-2-3. 체리톡 서비스 JenkinsFile

```
pipeline {
    agent any
    environment {
        DOCKER_COMPOSE_VERSION = '1.25.0' // 사용할 Docker Compose의 버전
        GITLAB_TOKEN = credentials('wns1915_cherry') // Jenkins에 저장된 GitLab Token의 ID
    }
    stages {
        stage('Checkout') {
            steps {
                git branch: 'release', credentialsId: 'wns1915_cherry', url: 'https://lab.ssafy.com/2_yewon/chelitalk.git'
            }
        }
        stage('Update Local Repository') {
            steps {
                script {
                        withCredentials([usernamePassword(credentialsId: 'wns1915_cherry', usernameVariable: 'GIT_USERNAME', passwordVariable: 'GIT_PASSWORD'
                            sh '''
```

```
                        ENCODED_USERNAME=$(echo $GIT_USERNAME | sed 's/@/%40/g')
                        cd /home/ubuntu/chelitalk
                        git pull https://$ENCODED_USERNAME:$GIT_PASSWORD@lab.ssafy.com/2_yewon/chelitalk.git release
                        '''
                    }
            }
        }
        stage('Build Docker Images') {
            steps {
                script {
                    sh 'docker-compose -f /home/ubuntu/chelitalk/Backend/docker-compose.yml build --no-cache app_cherry'
                    sh 'docker-compose -f /home/ubuntu/chelitalk/Backend/docker-compose.yml up -d app_cherry'
                    sh 'docker-compose -f /home/ubuntu/oringe/devway/docker-compose.yml build nginx'
                    sh 'docker-compose -f /home/ubuntu/oringe/devway/docker-compose.yml build certbot'
                }
            }
        }
        stage('Deploy') {
            steps {
                script {
                    sh 'docker-compose -f /home/ubuntu/oringe/devway/docker-compose.yml up -d nginx'
                    sh 'docker-compose -f /home/ubuntu/oringe/devway/docker-compose.yml up -d certbot'
                }
            }
        }
    }
}
```

3-2-4. Devway 홈페이지 JenkinsFile

```
pipeline {
    agent any
    environment {
        DOCKER_COMPOSE_VERSION = '1.25.0' // 사용할 Docker Compose의 버전
        GITLAB_TOKEN = credentials('wns1915_devway') // Jenkins에 저장된 GitLab Token의 ID
    }
    stages {
        stage('Checkout') {
            steps {
                git branch: 'release', credentialsId: 'wns1915_devway', url: 'https://lab.ssafy.com/judy3504/devway.git' // GitLab 리포지토리
            }
        }
        stage('Update Local Repository') {
            steps {
                script {
                        withCredentials([usernamePassword(credentialsId: 'wns1915_devway', usernameVariable: 'GIT_USERNAME', passwordVariable: 'GIT_PASSWORD'
                            sh '''
                            ENCODED_USERNAME=$(echo $GIT_USERNAME | sed 's/@/%40/g')
                            cd /home/ubuntu/devway
                            git pull https://$ENCODED_USERNAME:$GIT_PASSWORD@lab.ssafy.com/judy3504/devway.git release
                            '''
                        }
                }
            }
        }
        stage('Build Docker Images') {
            steps {
                script {
                    sh 'docker-compose -f /home/ubuntu/devway/devway/docker-compose.yml build app_devway'
                    sh 'docker-compose -f /home/ubuntu/devway/devway/docker-compose.yml up -d app_devway'
                }
            }
        }
        stage('Deploy') {
            steps {
                script {
                    sh 'docker-compose -f /home/ubuntu/oringe/devway/docker-compose.yml build --no-cache nginx '
                    sh 'docker-compose -f /home/ubuntu/oringe/devway/docker-compose.yml up -d nginx '
                    sh 'docker-compose -f /home/ubuntu/oringe/devway/docker-compose.yml up -d certbot'
                }
            }
        }
    }
}
```