드래곤 불

드래곤 불의 주인이 되자

포팅 메뉴얼

목 차

I. 개요

1. 프로젝트 개요

드래곤볼 쎄쎄쎄를 어플리케이션 `드래곤 불 천하제일 쎄쎄쎄`로 즐겨보자.

2. 프로젝트 사용 도구

이슈 관리 : JIRA
형상 관리 : Gitlab
커뮤니케이션 : Notion, Mattermost
디자인 : Figma
UCC : 모바비
CI/CD : Jenkins
가상 머신 : Docker

3. 개발환경

VS Code : 1.80.0
IntelliJ : 2023.1.3
JVM : 17
Flutter : 3.13.8
SERVER : AWS EC2 Ubuntu 20.04.3 LTS, Nginx 1.25.1
SQL : MySql 8.0.34
InMemoryDB : Redis 7.2.2

4. **Gitgnore** 처리한 핵심 키들

Spring : .idea (인텔리제이 사용자 설정 폴더), application-dev.yml (키들을 저장한 파일), service_key.json (파이어베이스를 사용하기 위한 파일)
Flutter : .env (키들을 저장한 파일)

II. 빌드

빌드는 jenkins를 통해서 진행함

1)

1. 환경변수 형태

.application-dev.yml

```
# MySQL
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url:
jdbc:mysql://[도메인이름]:3306/[db스키마이름]?serverTimezone=UTC&characterEncoding=
UTF-8
    username: [UserName]
    password: [UserName에 대한 비밀번호]



# Security OAuth
  security:
    oauth2:
      client:
        registration:
          google:
            client-id: [클라이언트 아이디]
            client-secret: [클라이언트 secret 키]

# JWT 시크릿 키
jwt:
  issuer: CHAEUM
  secret_key: [시크릿_키]

# Redis
redis:
  host: [도메인주소]
  port: [포트번호]
  password: [비밀번호]
```

2. 빌드하기

1) Back Spring - pipeline

```
        stage('BE-Build') {
            steps {
                dir("./BE/dragong") {
                    sh '''
                    cp /var/jenkins_home/workspace/docker/application-dev.yml
./src/main/resources
                    cp /var/jenkins_home/workspace/docker/service_key.json ./src/main/resources

                    chmod +x ./gradlew
                    ./gradlew clean build

                    mv ./build/libs/dragong-0.0.1-SNAPSHOT.jar
/var/jenkins_home/workspace/DragonFire-BE
                    cp /var/jenkins_home/workspace/docker/Dockerfile
/var/jenkins_home/workspace/DragonFire-BE
                    '''
                }
            }
        }
```

3. 배포하기

2) Nginx 설정

```
        events {
            worker_connections 4096;
        }

        http {
            include /etc/nginx/ip-block-list.conf;

            upstream backend {
```

```
    server 도메인 서버:8080;
}

upstream test{
    server 도메인 서버:8081;
}

upstream actuator{
    server 도메인 서버:8082;
}

server {
    listen 80;
    listen 443 ssl;

    ssl_certificate      /etc/letsencrypt/live/k9a209.p.ssafy.io/fullchain.pem;
    ssl_certificate_key  /etc/letsencrypt/live/k9a209.p.ssafy.io/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot


    server_name 내 ip 주소;

    return 444;
}

server {
    listen 80;
    server_name _;

    location / {
        deny all;
        return 444;
    }
}

server {
     listen 80;
     listen [::]:80;
```

```
        server_name 도메인 서버; # 등록한 도메인으로 변경

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

        location / {
            if ($bad_ip) {
                return 444;
            }

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

    server {
        listen 443 ssl;

        ssl_certificate /etc/letsencrypt/live/k9a209.p.ssafy.io/fullchain.pem; # managed by
Certbot
        ssl_certificate_key /etc/letsencrypt/live/k9a209.p.ssafy.io/privkey.pem; # managed by
Certbot
        include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

        server_name 도메인 서버;

        location / {
            if ($bad_ip) {
                return 444;
            }

            root   html;
            index  index.html index.htm;
        }

        location /api {
            rewrite ^/api(/.*)$ $1 break;
```

```
        proxy_pass http://backend;
        proxy_http_version 1.1;
        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 /ws {
        proxy_pass http://backend;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
    }

    location /test {
        rewrite ^/test(/.*)$ $1 break;
        proxy_pass http://test;
        proxy_http_version 1.1;
        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 /actuator {
        proxy_pass http://actuator;
        proxy_http_version 1.1;
        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;
    }


  }
}
```

3) Back Spring - jenkins pipeline

```
stage('Docker Image Build') {
        steps {
          sh '''
            docker stop dragong || true
            docker rm dragong || true
            docker stop dragong-test || true
            docker rm dragong-test || true
            docker rmi dragong:v1 || true
                                        docker build -t dragong:v1 .
                                    '''
        }
      }

      stage('Docker Container') {
        steps {
           sh '''
             docker run -d -p 8080:8080 -p 8082:8082 --name dragong dragong:v1
           '''
        }
      }
```

4) Mattermost

```
    post {
      success {
        script {
          // 빌드 성공 시 Mattermost 메시지 전송
          mattermostSend(
            color: 'good',
            message: "빌드 성공: ${currentBuild.fullDisplayName}",
            endpoint: '메타모스트 엔드 포인트',
            channel: '올리고 싶은 메타모스트 채널'
          )
        }
      }

      failure {
```

```
        script {
            // 빌드 실패 시 Mattermost 메시지 전송
            mattermostSend(
                color: 'danger',
                message: "빌드 실패: ${currentBuild.fullDisplayName}",
                endpoint: '메타모스트 엔드 포인트',
                channel: '메타모스트 채널 이름'
            )
        }
    }
}
```