이게모야 포팅매뉴얼

목차

---

1. 프로젝트 개요

- **프로젝트 이름**: 이게모야
    - **설명**: 자연 탐험과 학습을 돕기 위한 게이미피케이션 기반의 AR/AI 동식물 탐험 애플리케이션. 사용자는 실제 공원을 탐험하며 동식물 정보를 수집하고, 챗봇 NPC와의 상호작용을 통해 재미있는 퀘스트를 수행할 수 있습니다. AR 기술을 통해 실시간으로 동식물과 상호작용하며, 수집한 동식물은 도감에 기록됩니다.
- **주요 목표 및 문제 해결 방안**
    - **목표**: 사용자가 공원에서 동식물 탐험을 즐기며 자연과 상호작용할 수 있도록 돕고, 학습과 재미를 동시에 제공

- **문제 해결 방안**
    1. **온디바이스 AI 기반 실시간 동식물 판별**
        - 클라우드 연결 없이 기기 자체에서 AI 모델을 실행하여 동식물 판별을 실시간으로 구현
        - TensorFlow Lite를 사용하여 AI 모델을 최적화하고 모바일 장치에서 실행 가능하도록 조정
        - 모델의 정확도와 반응 속도를 평가하고 최적화 진행
    2. **AR을 활용한 네비게이션**
        - 사용자의 현재 위치와 실시간 지도를 결합하여 AR 기술로 시각적인 길 안내를 제공
        - Google ARCore와 GPS 데이터를 결합하여 공원 내에서 사용자의 위치를 정확히 추적
    3. **LLM(대규모 언어 모델)을 활용한 동식물 정보 제공**
        - LLM을 사용하여 사용자 질문에 대한 정교한 동식물 정보를 생성
        - 사용자의 탐험 기록과 질문을 분석하여 맞춤형 정보 제공
        - 모델의 응답 품질을 평가하고 사용성 향상을 위한 피드백 반영
    4. **Redis를 활용한 실시간 데이터 캐싱**
        - 사용자의 실시간 위치 데이터와 탐험 중 수집된 정보를 캐싱하여 빠르고 효율적인 데이터 접근을 제공

- **주요 기능**
    - **프론트엔드**
        - **AR 탐험 기능**: 공원 내에서 AR을 통해 동식물 탐색 및 길 안내
        - **On-Device AI를 활용한 실시간 동식물 판별**: 기기 내에서 AI 모델을 실행하여 실시간으로 동식물을 판별하고 도감에 기록
        - **챗봇과 상호작용**: LLM을 활용한 NPC와의 대화형 퀘스트 시스템
    - **백엔드**
        - **위치 기반 추천 서비스**: PostGIS를 통해 사용자의 위치에 맞춘 동식물 및 공원 정보 제공
        - **캐싱 및 데이터 최적화**: Redis와 캐싱 전략을 통해 성능 향상
        - **블루-그린 배포**: Jenkins와 Docker를 통한 무중단 배포 구현

- **프로젝트 팀 구성**
  - **프론트엔드 개발**: 김성수, 여창민, 최지훈
  - **백엔드 개발**: 강미연, 김은섭, 서장원
  - **인프라 관리**: 강미연
  - **AI 개발**: 김은섭, 최지훈

2. 프로젝트 사용 도구

- **이슈 관리**: JIRA
- **형상 관리**: GitLab
- **커뮤니케이션**: Mattermost, Webex, Notion
- **디자인**: Figma
- **UCC**: LUMA Dream Machine, Suno AI, 모바비 Video Editor Plus
- **CI/CD**: Jenkins, Docker, Docker Compose, Docker Hub, Nginx, Fail2Ban, ModSecurity, Let's Encrypt
- **정적 코드 분석 도구**: SonarQube

3. 개발환경

- **프론트엔드**: Android Studio Koala, Kotlin, MVI
- **백엔드**: IntelliJ IDEA, Java 17, Spring Boot 3.3.1
- **AI**: Python 3.12, Java 17, YOLOv8 latest, Langchain4j 0.34.0, PyTorch 2.4.1+cu121, nvidia graphic driver 535.183.01, CUDA 12.2
- **IDE**: IntelliJ IDEA, Android Studio
- **서버**: Ubuntu 20.04.6
- **데이터베이스**: PostgreSQL 16.4, Redis

4. 외부 서비스

- **OAuth**: 네이버 OAuth
- **AI 서비스**: OpenAI GPT API
- **그 외 서비스**: AWS S3

5. 환경변수

- `.env` **파일에 포함된 변수 및 설정:**

```
DB_URL=
DB_NAME=
DB_USERNAME=
DB_PASSWORD=

KAKAO_CLIENT_ID=
KAKAO_CLIENT_SECRET=

NAVER_CLIENT_ID=
NAVER_CLIENT_SECRET=

REDIS_HOST=
REDIS_PORT=

JWT_SECRET=
ACCESS_TOKEN_EXPIRATION=
REFRESH_TOKEN_EXPIRATION=

AWS_ACCESSKEY=
AWS_SECRETKEY=
AWS_REGION=
AWS_BUCKET=

OPENAI_API_KEY=
```

- `application.properties`

```
spring.application.name=moya
spring.config.import=optional:file:.env[.properties]

spring.profiles.include=oauth-kakao,oauth-naver

# DB 설정
spring.datasource.url=${DB_URL}
spring.datasource.username=${DB_USERNAME}
spring.datasource.password=${DB_PASSWORD}
```

```
spring.jpa.properties.hibernate.dialect=org.hibernate.di
alect.PostgreSQLDialect
spring.jpa.database-platform=org.hibernate.spatial.diale
ct.postgis.PostgisPG95Dialect
spring.jpa.hibernate.ddl-auto=update
spring.sql.init.mode=never
spring.jpa.defer-datasource-initialization=true
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true

# Redis 설정
spring.data.redis.host=${REDIS_HOST}
spring.data.redis.port=${REDIS_PORT}

# 시간대 설정
spring.jpa.properties.hibernate.jdbc.time_zone=UTC
spring.jackson.time-zone=Asia/Seoul

# 로깅 레벨 설정
logging.level.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.type.descriptor.sql.BasicBin
der=TRACE
logging.level.org.springframework.security=DEBUG
logging.level.org.springframework.web=DEBUG
logging.level.org.springframework.oauth2=DEBUG

# AWS S3 설정
cloud.aws.credentials.accessKey=${AWS_ACCESSKEY}
cloud.aws.credentials.secretKey=${AWS_SECRETKEY}
cloud.aws.region.static=${AWS_REGION}
cloud.aws.s3.bucket=${AWS_BUCKET}

# OpenAI GPT 설정
langchain4j.open-ai.chat-model.api-key=${OPENAI_API_KEY}
langchain4j.open-ai.chat-model.model-name=gpt-4o-mini
langchain4j.open-ai.chat-model.log-requests=true
langchain4j.open-ai.chat-model.log-responses=true
```

- `application-oauth-kakao.properties`

```
spring.config.import=optional:file:.env[.properties]
oauth2.kakao.client-id=${KAKAO_CLIENT_ID}
oauth2.kakao.client-secret=${KAKAO_CLIENT_SECRET}
oauth2.kakao.user-info-uri=https://kapi.kakao.com/v2/user/me
```

- `application-oauth-naver.properties`

```
spring.config.import=optional:file:.env[.properties]
oauth2.naver.client-id=${NAVER_CLIENT_ID}
oauth2.naver.client-secret=${NAVER_CLIENT_SECRET}
oauth2.naver.user-info-uri=https://openapi.naver.com/v1/nid/me
```

6. Nginx 설치 및 SSL 인증서 발급

- **Nginx 설치**

```
sudo apt-get update
sudo apt-get install nginx
nginx -v
```

- **Let's Encrypt 설치 및 SSL 인증서 발급**

```
sudo ufw allow 80/tcp
sudo ufw reload

sudo apt-get install letsencrypt
sudo systemctl stop nginx  # Nginx 중지
sudo letsencrypt certonly --standalone -d j11d202.p.ssafy.io  # SSL 인증서 발급
sudo systemctl start nginx  # Nginx 시작

sudo ufw allow 443/tcp
sudo ufw reload
```

```
sudo nano /etc/nginx/sites-available/default
```

수정된 Nginx 설정 파일 예시:

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

    server_name j11d202.p.ssafy.io;

    # HTTP를 HTTPS로 리디렉션
    return 301 https://$host$request_uri;

    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;
}

# HTTPS 설정
server {
    listen 443 ssl;
    listen [::]:443 ssl;

    server_name j11d202.p.ssafy.io;

    # SSL 인증서 경로 설정
    ssl_certificate /etc/letsencrypt/live/j11d202.p.ssaf
y.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/j11d202.p.
ssafy.io/privkey.pem;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;

    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;

    location / {
```

```
            try_files $uri $uri/ =404;
    }
}
```

Nginx 설정 테스트 및 재시작

```
sudo nginx -t
sudo systemctl restart nginx
```

7. Jenkins 초기 설정 및 권한 부여

- **Jenkins 컨테이너 접근 및 Maven 설치**

```
docker exec -it --user root jenkins bash

# Maven 설치
apt-get update
apt-get install maven

# Maven 권한 설정
sudo chown -R ubuntu:ubuntu /usr/share/maven

# Maven 버전 확인
mvn --version
```

- **Jenkins와 Docker 연동**

```
# Docker 그룹에 Jenkins 사용자 추가
usermod -aG docker jenkins
```

8. Jenkins 설정

- **EC2 접속 설정**

```
# SSH 폴더 생성 및 이동
mkdir -p ~/.ssh
cd ~/.ssh
```

```
# pem 파일을 .ssh 폴더로 이동
cp [현재 pem 파일의 위치] ~/.ssh/J11D202T.pem

# config 파일 생성 및 편집
vi config

# config 파일에 아래 내용을 추가
Host ssafy
    HostName j11d202.p.ssafy.io
    User ubuntu
    IdentityFile ~/.ssh/J11D202T.pem

# 접속
ssh ssafy
```

- **초기 설정**

```
sudo apt update
sudo apt upgrade -y
sudo apt install -y build-essential

# 한국 시간대로 설정
sudo ln -sf /usr/share/zoneinfo/Asia/Seoul /etc/localtim
e

# 시간 확인
date
```

- **Docker 설치**

```
# Docker 설치 스크립트 생성
nano install-docker.sh

# install-docker.sh 내용:
sudo apt-get -y install \\
  ca-certificates \\
  curl \\
  gnupg \\
```

```
  lsb-release

chmod +x install-docker.sh
./install-docker.sh

# Docker 버전 확인
docker --version
```

- **Jenkins 설치 및 설정**

```
# Jenkins 디렉토리 생성
cd /home/ubuntu && mkdir jenkins

# Jenkins 컨테이너 실행
sudo docker run -d -p 9090:8080 -p 50000:50000 --name je
nkins jenkins/jenkins:jdk17

# Jenkins 디렉토리로 이동
cd /home/ubuntu/jenkins

# Update Center 인증서 설정
mkdir update-center-rootCAs
wget https://cdn.jsdelivr.net/gh/lework/jenkins-update-c
enter/rootCA/update-center.crt -O ./update-center-rootCA
s/update-center.crt

sudo sed -i 's#https://updates.jenkins.io/update-center.
json#https://raw.githubusercontent.com/lework/jenkins-up
date-center/master/updates/tencent/update-center.json#'
./hudson.model.UpdateCenter.xml

# Jenkins 재시작
sudo docker restart jenkins

# 추가 패키지 설치
sudo apt-get install ca-certificates curl gnupg lsb-rele
ase
```

```
# Docker GPG 키 추가
curl -fsSL https://download.docker.com/linux/ubuntu/gpg
| sudo gpg --dearmor -o /usr/share/keyrings/docker-archi
ve-keyring.gpg

# Docker 패키지 저장소 추가
echo "deb [arch=$(dpkg --print-architecture) signed-by=/
usr/share/keyrings/docker-archive-keyring.gpg] https://d
ownload.docker.com/linux/ubuntu $(lsb_release -cs) stabl
e" | sudo tee /etc/apt/sources.list.d/docker.list > /de
v/null

# Docker 엔진 및 관련 패키지 설치
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containe
rd.io docker-buildx-plugin docker-compose-plugin

# Docker 그룹에 사용자 추가
sudo usermod -aG docker $USER

# Docker 소켓 권한 재설정
sudo chown -R $USER:$USER /var/run/docker.sock
sudo chmod -R 660 /var/run/docker.sock

# Docker 버전 확인
docker --version
```

- **Jenkins 초기 비밀번호 확인**

```
sudo docker exec jenkins cat /var/jenkins_home/secrets/i
nitialAdminPassword
```

- **Git 설정**

```
git config --global credential.helper "cache --timeout=3
600"
```

9. Docker Compose 설치

```
sudo apt-get update
sudo apt-get install -y docker-compose
```

10. PostgreSQL 및 Redis 설치

- **PostgreSQL 설치 스크립트**

```
nano install-postgres.sh

# install-postgres.sh 내용:
if [ $# -ne 1 ]; then
  echo "Usage: $0 <POSTGRES_PASSWORD>"
  exit 1
fi

POSTGRES_PASSWORD=$1

docker run -d \
  -p 5432:5432 \
  -v /var/lib/postgres-data:/var/lib/postgresql/data \
  --name postgres \
  -e POSTGRES_PASSWORD=${POSTGRES_PASSWORD} \
  postgres:14

chmod +x install-postgres.sh
./install-postgres.sh mypassword

# Postgres 컨테이너 상태 확인
docker ps
```

- **Redis 설치 스크립트**

```
nano install-redis.sh

# install-redis.sh 내용:
docker run -d \
  -p 6379:6379 \
  -v /var/lib/redis-data:/data \
```

```
  --name redis \
  redis:7.0

chmod +x install-redis.sh
./install-redis.sh

# Redis 컨테이너 상태 확인
docker ps
```

- **PostgreSQL 초기 설정**

```
docker exec -it postgres bash

# psql 접속
psql -U postgres

# 데이터베이스 생성
CREATE DATABASE moya;

# 데이터베이스 목록 확인
\l

# 사용자 비밀번호 변경
ALTER USER postgres WITH PASSWORD '';
```

11. 배포 전략

- **블루-그린 배포(Blue-Green Deployment)**
  - **설명**: 블루-그린 배포 방식을 사용하여 무중단 배포를 구현했습니다. 이 배포 전략을 통해 배포 시 서버의 다운타임을 최소화하고, 서비스 중단 없이 애플리케이션을 업데이트할 수 있습니다. Docker Compose를 활용한 컨테이너화 및 Nginx 리버스 프록시 설정을 통해 애플리케이션의 안정적이고 효율적인 배포를 관리하고 있습니다.
- **현재 Nginx 설정 및 Docker 네트워크 구성**

  (자세한 설정 파일 및 스크립트는 이전 섹션 참조)

12. SonarQube 설정

- **SonarQube 설치 및 설정**

```
# SonarQube Docker 컨테이너 실행
docker run -d \
  -p 9000:9000 \
  --name sonarqube \
  sonarqube:latest

# SonarQube 컨테이너 상태 확인
docker ps

# SonarQube 초기 설정
# 웹 브라우저에서 http://j11d202.p.ssafy.io:9000 접속하여 초기 설정 완료
```

- **Jenkins와 SonarQube 통합**
  - **Jenkins Pipeline 설정**에서 이미 SonarQube 분석 스테이지가 포함되어 있습니다.
  - **SonarQube 서버 URL**: `http://j11d202.p.ssafy.io:9000`
  - **SonarQube 프로젝트 키**: `com.e22e:moya`
  - **SonarQube 토큰**: Jenkins Credentials에 `sonarQubeToken`으로 저장

13. API 문서화

- **Swagger 설정**

```
// Spring Boot 프로젝트에 Swagger 설정 추가
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
```

```
@Configuration
public class SwaggerConfig {
    @Bean
    public Docket api() {
        return new Docket(DocumentationType.OAS_30)
                .select()
                .apis(RequestHandlerSelectors.basePackag
e("com.e22e.moya"))
                .paths(PathSelectors.any())
                .build();
    }
}
```

- **Swagger UI 접근**
  - `http://j11d202.p.ssafy.io/swagger-ui/` 에서 API 문서 확인 가능

14. 추가 설치 및 설정 스크립트

- **Fail2Ban 설치 및 설정**

```
# Fail2Ban 설치
sudo apt-get update
sudo apt-get install fail2ban -y

# Fail2Ban 설정 파일 생성
sudo nano /etc/fail2ban/jail.local

# jail.local 파일에 다음 내용 추가
[nginx-req-limit]
enabled = true
filter = nginx-req-limit
action = iptables-multiport[name=ReqLimit, port="http,ht
tps", protocol=tcp]
logpath = /var/log/nginx/error.log
findtime = 600
bantime = 7200
maxretry = 10

# 필터 설정 파일 생성
```

```
sudo nano /etc/fail2ban/filter.d/nginx-req-limit.conf

# nginx-req-limit.conf 파일에 다음 내용 추가
[Definition]
failregex = limiting requests, excess:.* by zone

# Fail2Ban 재시작
sudo systemctl restart fail2ban

# Fail2Ban 상태 확인
sudo fail2ban-client status
sudo fail2ban-client status nginx-req-limit
```

- **ModSecurity 웹 방화벽 설치 및 구성**

```
# ModSecurity 설치
sudo apt-get install libnginx-mod-security -y

# ModSecurity 설정 파일 편집
sudo nano /etc/nginx/modsec/modsecurity.conf

# modsecurity.conf 파일에 다음 내용 추가
SecRuleEngine On

# OWASP ModSecurity Core Rule Set(CRS) 설치
sudo apt-get install modsecurity-crs -y

# CRS 설정 파일을 Nginx에 추가
sudo cp /usr/share/modsecurity-crs/crs-setup.conf.exampl
e /etc/nginx/modsec/crs-setup.conf
sudo ln -s /usr/share/modsecurity-crs/rules /etc/nginx/m
odsec/rules

# modsecurity.conf 파일에 CRS 설정 파일 포함
echo 'Include /etc/nginx/modsec/crs-setup.conf' | sudo t
ee -a /etc/nginx/modsec/modsecurity.conf
echo 'Include /etc/nginx/modsec/rules/*.conf' | sudo tee
-a /etc/nginx/modsec/modsecurity.conf
```

```
# Nginx 설정 테스트 및 재시작
sudo nginx -t
sudo systemctl reload nginx
```

15. 기타 설정 및 명령어

- **GitLab API 토큰 및 WebHooks 설정**

```
# Snippet Generator 예시
checkout scmGit(branches: [[name: '*/backend']], extensi
ons: [], userRemoteConfigs: [[credentialsId: 'gitlab', u
rl: 'https://lab.ssafy.com/s11-ai-image-sub1/S11P21D202.
git']])
```

- **Maven 권한 설정**

```
sudo chown -R ubuntu:ubuntu /usr/share/maven
```

- **PostGIS 설치 및 활성화**

```
docker exec -it postgres bash
apt-get update
apt-get install postgis postgresql-14-postgis-3

psql -U postgres -d moya
CREATE EXTENSION postgis;
```

16. 추가 참고 자료

- **Jenkins 컨테이너 재시작**

```
docker restart jenkins
```

- **Spring Boot 앱 컨테이너 접근 및 설정 확인**

```
docker exec -it backend_springboot-app_1 /bin/sh
ls -la /path/to/.env  # .env 파일이 있는지 확인
```

- **Spring Boot 앱 컨테이너 실행 예시**

```
docker run -it --name backend_springboot-app_1 -p 8080:8
080 openjdk:17-jdk-alpine /bin/sh

apk update && apk add git

mkdir /moya
cd /moya

git clone -b develop https://lab.ssafy.com/s11-ai-image-
sub1/S11P21D202.git .

docker run -d \
  -p 8080:8080 \
  -v /home/ubuntu/app/logs:/logs \
  --env-file=/home/ubuntu/.env \
  --name springboot-app \
  backend_springboot-app
```