출발선 포팅 매뉴얼

1. 개요

한눈에 보는 마라톤 일정, 간편하게 신청하고 친구들과 함께 달려보세요.
당신의 도전을 위한 서비스 "출발선"

2. 사용 도구

- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, MatterMost
- 디자인 : Figma
- CI/CD : Jenkins

3. 개발 도구

- Intellij : 2024.3.1.1
- Android Studio : 2024.2.2

4. 개발 환경

4-1. Android

항목	버전
Android Studio	최신 안정 버전
Kotlin	2.0.0
Android Gradle Plugin (AGP)	8.8.0
compileSdk	35
minSdk	26
targetSdk	34

Dependencies

라이브러리	버전
Jetpack Compose BOM	2024.04.01
Retrofit	2.9.0
OkHttp	4.9.3
Firebase BoM	33.8.0
Firebase Messaging	24.1.0
Room	2.4.3
Kakao SDK (v2-user)	2.20.6
Coil (이미지 로딩 라이브러리)	2.5.0
Coroutine	1.7.3
Security Crypto (암호화)	1.1.0-alpha06
Navigation Compose	2.8.5
Material3	1.2.1
ConstraintLayout	2.2.0

4-2. Backend (Spring Boot)

항목	버전
Java (OpenJDK)	17
Spring Boot	3.3.8
Gradle	7.x 이상
Spring Dependency Management	1.1.7

Dependencies

라이브러리	버전
Spring Boot Starter Web	최신 안정 버전
Spring Boot Starter JPA	최신 안정 버전
QueryDSL	5.0.0 (jakarta)
MySQL Connector	8.x

Spring Cloud AWS	2.2.6.RELEASE
Swagger (SpringDoc OpenAPI)	2.0.2
Google Cloud Vision API	3.34.0
Google Document AI	2.10.0
WebClient (Spring WebFlux)	최신 안정 버전
Selenium (WebDriver)	4.23.0
WebDriverManager	5.5.3
Lombok	최신 안정 버전
JUnit (테스트 프레임워크)	최신 안정 버전
Gson (JSON 파싱)	2.10.1

4-3. Wear OS

항목	버전
Android Studio	최신 안정 버전
Kotlin	2.0.0
Android Gradle Plugin (AGP)	8.8.0
compileSdk	35
minSdk	30
targetSdk	35

Dependencies

라이브러리	버전
Play Services Wearable	19.0.0
Jetpack Compose BOM	2024.04.01
Compose Material	1.2.1
Compose Foundation	1.2.1
Wear Tooling Preview	1.0.0
Activity Compose	1.10.0
Core Splashscreen	1.0.1
Tiles	1.4.0

Tiles Material	1.4.0
Tiles Tooling Preview	1.4.0
Horologist Compose Tools	0.6.17
Horologist Tiles	0.6.17
Watchface Complications	1.2.1
Navigation Runtime KTX	2.8.6
Navigation Compose	2.8.6
ViewPager2	1.1.0

Additional Dependencies

라이브러리	버전
ViewModel (Lifecycle)	2.8.7
Fragment KTX	1.8.6
Gson (JSON 파싱)	2.10.1

5. 외부 프로그램

- **FireBase**
  - RealTime Database (현재 마라톤 정보 저장)
  - Cloud Messaging
    - Firebase 콘솔 프로젝트 만들어 사용
- **카카오**
  - OAuth
    - 카카오 디벨로퍼스에서 내 앱을 만들어 사용
  - Pay
    - 카카오페이 개발자센터에서 내 앱을 만들어 사용
- **NH 오픈뱅킹**
  - 테스트 계좌 사용
    - NH 디벨로퍼스에서 내 앱을 만들어 사용

- **Chat GPT**
  - 마라톤 데이터 형태 변환 시 사용
    - GPT 키 연결하여 사용

6. 환경변수 형태

- **Backend**
  - **application.properties**

```
spring.application.name=gogoma

# MYSQL
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# JPA
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true


# ChatGPT
openai.model=gpt-3.5-turbo
openai.api.url= https://api.openai.com/v1/chat/completions

# Max File Size
spring.servlet.multipart.enabled=true
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=20MB

spring.config.import=secrets.properties
```

  - **secrets.properties**

```
spring.datasource.url=jdbc:mysql://{배포 MySQL 주소}?useSSL=f
spring.datasource.username= {MySQL UserName}
spring.datasource.password= {MySQL Password}

# Kakao Oauth
kakao.client-id= {Kakao Developers App Key}
kakao.redirect-uri= {Redirect URI}

# Kakao Pay
kakao.pay.secret-key= {KakaoPay App Key}

# S3 DB Access
cloud.aws.s3.bucket= {S3 Name}
cloud.aws.credentials.access-key= {AWS AccessKey}
cloud.aws.credentials.secret-key= {AWS SecretKey}
cloud.aws.region.static= {AWS Region}

# ChatGPT
openai.api.key= {OpenAi API Key}

# Google Document AI
google.document.ai.name= {Google App Key}

# NH Bank
nhbank.base-url= {NH BANK URL}
nhbank.iscd= {NH BANK iscd}
nhbank.access-token={NH BANK Access Token}
nhbank.fin-acno= {NH BANK FinAcno}
nhbank.acno= {NH BANK Acno}

# server ip
domain.name={Server Domain}

# App redirect URL
app.redirect-url={App Redirect URL}
```

```
# FireBase
firebase.serviceAccountKey={Firebase Service Account Key}
```

- **Android**
  - local.properties

```
sdk.dir={sdk 디렉토리 위치}
KAKAO_APP_KEY={Kakao Developers App Key}
CLIENT_ID={Kakao Developers Client ID}
REDIRECT_URI={Kakao Redirect URI}
```

7. CI/CD 구축

기본 설정

0. 기본적으로 위의 세팅을 참고하여 필요한 패키지, 프로그램을 EC에 다운로드 받았다는 가정 하에 진행

```
- Docker, Nginx, MySQL, Java 등
```

1. Docker Compose 설치

```
# 버전 지정 및 설치(2.32.4), uname -s: os이름 출력, uname -m: 하드웨어
sudo curl -L "https://github.com/docker/compose/releases/download/v

# 실행 권한 부여
sudo chmod +x /usr/local/bin/docker-compose
```

2. docker-compose.yml 구성, Docker file 디렉토리 구성

- docker-compose.yml

```
services:
  jenkins:
```

```
    image: jenkins/jenkins:lts
    container_name: jenkins
    ports:
      - "9090:8080"
      - "50000:50000"
    environment:
      TZ: "Asia/Seoul"
    volumes:
      - jenkins_data:/var/jenkins_home
      - /var/run/docker.sock:/var/run/docker.sock

    springboot:
    build:
      context: /home/ubuntu/backend   # jar 파일과 Dockerfile이
      dockerfile: Dockerfile
    container_name: springboot-app
    ports:
      - "8080:8080"
    environment:
        TZ: "Asia/Seoul"

  mysql:
    image: mysql:8.0
    container_name: mysql-server
    environment:
      MYSQL_ROOT_PASSWORD: {SQL Root Password}
      MYSQL_DATABASE: {SQL Database Name}
      MYSQL_USER: {SQL User Name}
      MYSQL_PASSWORD: {SQL User Password}
      TZ: "Asia/Seoul"
    volumes:
      - mysql_data:/var/lib/mysql
    ports:
      - "3306:3306"

volumes:
  jenkins_data:
  mysql_data:
```

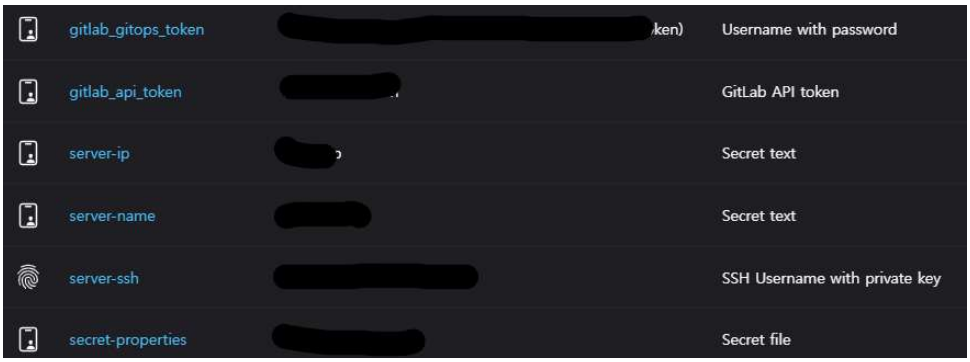
- Docker file

```
cd /home/ubuntu/backend
sudo nano Dockerfile
```

```
FROM openjdk:17-jdk-slim

WORKDIR /app

COPY gogoma-0.0.1-SNAPSHOT.jar app.jar

ENTRYPOINT ["java", "-jar", "app.jar"]
```

3. Jenkins 설치 및 구성

- Jenkins에 secrets.properties 및 env 관련된 key내용 저장(Dashboard - Jenkins 관리 - Credentials)
  - Add credentials클릭시 유형 선택 가능
    - 보통은 secret text
    - key.pem과 서버 user등록은 SSH Username with private key
    - secrets.properties는 secret file형태로 file 업로드



- gitlab과 Jenkins 연결
  - gitlab에서 Jenkins에 등록할 Access Token발급(프로필-prefernece - Access tokens - Add new token)
  - Jenkins 관리 - system에 gitlab관련 내용 Key 작성
  - Jenkins 파이프라인 설정(Dashboard - item 생성)

        - 프로젝트 주소 작성
        - push event source 위치 설정
        - Credentials 등록
    - Jenkins에서 플러그인 설치 및 재시작
        - GitLab, Docker pipeline, Docker API, Genrice Webhook Trigger, SSH Agent
    - gitlab에 Webhook설정
        - Jenkins에서 발급된 jenkins trigger secret token 등록
        - Trigger: push event 발생할 branch 설정

4. nginx 세팅

```
server {
    listen 443 ssl;
    server_name i12a808.p.ssafy.io;

    ssl_certificate /etc/letsencrypt/live/i12a808.p.ssafy.io/fullch
    ssl_certificate_key /etc/letsencrypt/live/i12a808.p.ssafy.io/pr

    # Certbot 인증을 위한 .well-known 경로 추가
    location /.well-known/acme-challenge/ {
        root /var/www/html;
        allow all;
    }

    # Swagger 요청 프록시
    location /swagger-ui/ {
        proxy_pass http://localhost:8080/swagger-ui/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for
    }

    # API 요청 프록시
    location /api/ {
        proxy_pass http://localhost:8080;
        proxy_set_header Host $host;
```

```
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for
        # CORS 설정
        add_header 'Access-Control-Allow-Origin' '*' always;
        add_header 'Access-Control-Allow-Methods' 'GET, POST, PUT,
        add_header 'Access-Control-Allow-Headers' 'Authorization, C
        add_header 'Access-Control-Allow-Credentials' 'true' always
    }

    # `/v3/api-docs`를 Spring Boot로 프록시하여 요청이 정상 처리되도록
    location /v3/api-docs {
        proxy_pass http://localhost:8080/v3/api-docs;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for
        add_header Access-Control-Allow-Origin *;
    }
}

server {
    listen 80;
    server_name i12a808.p.ssafy.io;
    return 301 https://$host$request_uri;
}
```