포팅매뉴얼

태그

LinkCare 포팅 매뉴얼

1. 개요

이 문서는 LinkCare 전체 시스템을 다른 환경으로 포팅하거나 신규 개발 환경을 구축하기 위한 가이드입니다.

LinkCare 시스템은 **백엔드 서버**, **안드로이드 모바일 앱**, **Wear OS 워치 앱**, 그리고 **AI 서버**로 구성됩니다.

---

2. 시스템 아키텍처 및 통신

- **Backend**: Spring Boot 기반의 API 서버. 데이터베이스, 캐시, 외부 서비스 연동을 관리합니다.
- **Frontend**: Android 모바일 애플리케이션. 사용자의 주 인터페이스 역할을 합니다.
- **Watch**: Wear OS 애플리케이션. 모바일 앱과 연동하여 건강 데이터 측정 등의 기능을 수행합니다.
- **AI Server**: FastAPI 기반의 Python 서버. AI 모델을 서빙합니다. (소스코드는 별도 관리)
- **Infra**: Docker Compose를 사용하여 전체 서비스의 실행 환경을 정의합니다.

2.1. 컴포넌트 간 통신 방식

- **Frontend ↔ Backend**: Retrofit 라이브러리를 통해 백엔드의 REST API를 호출하여 통신합니다.
- **Watch ↔ Frontend**: play-services-wearable 라이브러리의 Wearable Data Layer API를 사용하여 데이터를 주고받습니다.
- **Backend ↔ AI Server**: 백엔드에서 AI 서버의 API(`http://linkcare-ai:8000`)를 HTTP Client로 직접 호출하여 통신합니다.

---

3. Backend (LinkCare API Server)

3.1. 기술 스택 및 요구사항

- 언어: Java 21
- 프레임워크: Spring Boot 3.5.6
- 빌드 도구: Gradle
- 데이터베이스: MySQL 8.0
- 캐시: Redis 7

3.2. 빌드 및 실행

```
# 위치
Backend/LinkCare

# 빌드
./gradlew build

# 실행
./gradlew bootRun
# 또는 IDE에서 LinkCareApplication.java 실행
```

3.3. 환경 변수 및 설정

- **application.properties** 파일 및 **docker-compose.yml** 참조
- 필수 환경 변수:

`JWT_SECRET`, `DB_USER_PASSWORD`, `GMAIL_PASSWORD`, `GOOGLE_CLIENT_ID`, `KAKAO_CLIENT_ID`, `KAKAO_REDIRECT_URI`, `AWS_ACCESS_KEY`, `AWS_SECRET_KEY`, `GMS_KEY`

- Firebase 설정: `firebase-service-account.json` 파일이 필요하며, 실행 환경에 맞게 경로 설정
  - Docker 실행 시: `/run/secrets/firebase.json`

3.4. API 문서 (Swagger)

- `springdoc-openapi` 라이브러리 포함
- 백엔드 서버 실행 후 Swagger UI에서 API 명세 확인 가능

```
http://{서버 주소}:9090/swagger-ui.html
```

3.5. 데이터 모델

- JPA의 `hibernate.ddl-auto=update` 옵션 사용
- 데이터베이스 스키마는 `@Entity` 클래스에 의해 정의
- 데이터 구조 확인: `com.ssafy.linkcare` 하위 패키지 Entity 클래스 참조

---

4. Frontend (Android App)

4.1. 기술 스택 및 요구사항

- 플랫폼: Android
- 언어: Kotlin
- UI: Jetpack Compose
- Min SDK: 29 / Target SDK: 36
- Java Version: 11

4.2. 빌드 및 실행

```
# 위치
Frontend

# Android Studio에서 프로젝트 열고 에뮬레이터 또는 실제 기기 실행
```

4.3. 로컬 설정

- `local.properties` 파일 설정 필요

  ```
  GOOGLE_WEB_CLIENT_ID: Google 로그인에 사용
  KAKAO_NATIVE_APP_KEY: Kakao 로그인에 사용
  ```

4.4. 주요 모듈

- `:app`: 메인 애플리케이션 모듈, 백엔드 및 워치 앱과 통신
- `:llama`: `llama.cpp` 기반 온디바이스 AI 모델 JNI 래퍼 라이브러리 모듈

---

5. Watch (Wear OS App)

5.1. 기술 스택 및 요구사항

- 플랫폼: Wear OS
- 언어: Kotlin

- UI: Jetpack Compose for Wear OS
- Min SDK: 30 / Target SDK: 36
- Java Version: 17

5.2. 빌드 및 실행

```
# 위치
Watch

# Android Studio에서 프로젝트 열고 Wear OS 에뮬레이터 또는 실제 기기 실행
```

---

6. 인프라 및 전체 서비스 실행 (Docker)

- Infra 디렉토리의 `docker-compose.yml` 파일 사용하여 전체 서비스 실행

6.1. 위치

- Infra

6.2. 사전 준비

- Backend, AI 서버 Docker 이미지 준비
- `.env` 파일 생성 또는 쉘 환경에 모든 환경 변수 설정
- Firebase 인증 파일(`firebase-service-account.json`) 지정 경로에 위치
- AI 모델 파일 지정 경로에 위치

6.3. 실행 및 종료

```
# 실행
docker-compose up -d

# 종료
docker-compose down
```

---

7. 개발 및 배포 워크플로우 (CI/CD)

- 프로젝트 루트의 `Jenkinsfile` 사용
- 백엔드 애플리케이션 빌드 및 배포 자동화

7.1. 파이프라인 단계

1. **변경 감지 (Check Backend Changes)**
   - 최근 커밋에서 `Backend/` 또는 `backend/` 디렉토리 변경 확인
   - 변경 없으면 파이프라인 중단
2. **JAR 파일 빌드 (Build JAR)**

```
cd Backend/LinkCare
./gradlew clean build -x test
```

3. **Docker 이미지 빌드 및 푸시 (Build & Push Docker Image)**

- Dockerfile 사용하여 이미지 빌드, 태그는 Jenkins `BUILD_NUMBER` 사용
- Docker Hub 레지스트리에 푸시

4. **서버 배포 (Deploy to Server)**

- Jenkins 인증 정보 사용하여 SSH 접속

- `.env` 파일 생성 및 배포 서버로 전송
- 컨테이너 실행: `docker-compose up -d`
- 오래된 이미지 정리

5. **배포 확인 (Verify Deployment)**

```
docker ps
# 실행 중 컨테이너 목록 확인
```

- Git 푸시 시 트리거되며, 백엔드 변경 감지 → 빌드 → 패키징 → 배포 전체 과정 자동 수행