HanaChain

시스템 실행 가이드

하트, 만화 영화, 창의성이(가) 표시된 사진

자동 생성된 설명만화 영화, 장난감, 동물 피규어이(가) 표시된 사진

자동 생성된 설명

광명융합기술교육원 데이터분석과 조승우

목차

[시스템 개요 3](#_Toc210999282)

[환경 변수 설정 4](#_Toc210999283)

[데이터베이스 설정 6](#_Toc210999284)

[블록체인 시스템 구동 7](#_Toc210999285)

[백엔드 서버 구동 9](#_Toc210999286)

[FDS 시스템 구동 10](#_Toc210999287)

[프론트엔드 구동 11](#_Toc210999288)

# 시스템 개요

HanaChain은 5개의 주요 컴포넌트로 구성되어 있습니다.

텍스트, 도표, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

1. 필수 소프트웨어

|  |  |  |
| --- | --- | --- |
| 소프트웨어 | 버전 | 용도 |
| Java | 17 | 백엔드(Spring Boot) |
| Node.js | 18 | 프론트엔드, 블록체인 |
| Python | 3.9 | FDS 시스템 |
| Gradle | 8.0 | 벡엔드 빌드 |
| pnpm | latest | 프론트엔드 패키지 관리 |
| Docker | latest | Oracle 데이터베이스 |

1. 외부 서비스

**Sepolia Testnet RPC**

- Infura, Alchemy 등에서 API 키 발급

**Etherscan API**

- 스마트 컨트랙트 검증용

- https://etherscan.io/apis

**포트원 (PortOne)**

- 결제 시스템 사용 시 계정 필요

- https://portone.io

# 환경 변수 설정

1. 블록체인 환경 변수

|  |
| --- |
| # Sepolia RPC URL (Infura/Alchemy)  SEPOLIA\_URL=https://sepolia.infura.io/v3/YOUR\_INFURA\_PROJECT\_ID  # 배포용 지갑 Private Key (메타마스크에서 추출)  PRIVATE\_KEY=your\_private\_key\_here  # Etherscan API Key (컨트랙트 검증용, 선택사항)  ETHERSCAN\_API\_KEY=your\_etherscan\_api\_key\_here |

* Private Key는 테스트넷 전용 지갑 사용
* 메인넷 지갑의 Private Key 사용 금지

1. 백엔드 환경 변수

|  |
| --- |
| # 블록체인 설정  SEPOLIA\_URL=https://sepolia.infura.io/v3/INFURA\_PROJECT\_ID  BLOCKCHAIN\_PRIVATE\_KEY=your\_private\_key\_here  # 플랫폼 지갑 (USDC 충전용)  PLATFORM\_WALLET\_ADDRESS=0xa757A82EDC4E1fA41E2bE0e1E0E4E847E6dd59F3  PLATFORM\_WALLET\_PRIVATE\_KEY=99eb199e5870cd7cd673e54a49b7018d31f745fd8941d8f25364c169f8603793  # 지갑 마스터 비밀번호  WALLET\_MASTER\_PASSWORD=hanachain-secure-master-password-2025  # Oracle DB 설정 (Docker 컨테이너 사용)  DB\_HOST=localhost  DB\_PORT=1521  DB\_SERVICE\_NAME=FREEPDB1  DB\_USER=system  DB\_PASSWORD=root |

1. FDS 환경 변수

|  |
| --- |
| # Oracle DB 설정  DB\_HOST=localhost  DB\_PORT=1521  DB\_SERVICE\_NAME=FREEPDB1  DB\_USER=hanachain\_user  DB\_PASSWORD=your\_password  # API 설정  API\_PORT=8000 |

* FDS에서 참조하는 DB는 백엔드와 동일

1. 프론트엔드 환경 변수

|  |
| --- |
| # 포트원 설정 (결제 기능 사용 시)  NEXT\_PUBLIC\_PORTONE\_STORE\_ID=your-store-id-here  NEXT\_PUBLIC\_PORTONE\_CHANNEL\_KEY=your-channel-key-here  PORTONE\_API\_SECRET=your-api-secret-here  # 백엔드 API URL  NEXT\_PUBLIC\_API\_URL=http://localhost:8080/api  # FDS API URL  NEXT\_PUBLIC\_FDS\_URL=http://localhost:8000 |

# 데이터베이스 설정

HanaChain은 Oracle 23c Free를 사용합니다.

1. Oracle 컨테이너 새로 생성

|  |
| --- |
| docker run -d \  --name orcl23 \  -p 1521:1521 \  -p 5500:5500 \  -e ORACLE\_PWD=root \  -v oradata23:/opt/oracle/oradata \  container-registry.oracle.com/database/free:latest |

1. 하나체인 사용자 생성 및 세션 변경

|  |
| --- |
| CREATE USER hanachain\_user IDENTIFIED BY hanachain\_password;  GRANT CONNECT, RESOURCE TO hanachain\_user;  GRANT UNLIMITED TABLESPACE TO hanachain\_user;  ALTER SESSION SET CONTAINER = FREEPDB1; |

# 블록체인 시스템 구동

1. 의존성 설치

|  |
| --- |
| cd hanachain-blockchain  npm install |

1. 스마트 컨트랙트 컴파일

|  |
| --- |
| # 컨트랙트 컴파일  npx hardhat compile  # 컴파일 성공 확인  ls -la artifacts/contracts/ |

1. 환경 검증

|  |
| --- |
| # 네트워크 연결 테스트  npm run test-network  # 지갑 확인  npm run check-wallet  # 전체 환경 검증  npm run verify-setup |

1. 스마트 컨트랙트 배포 정보

HanaChain 스마트 컨트랙트는 이미 Sepolia 테스트넷에 배포되어 있습니다.

|  |
| --- |
| # Mock USDC Token Contract  USDC\_ADDRESS=0x5eA3DE89105aE1be35219eaf91b86A3Ba8607D59  # Campaign Factory Contract  CAMPAIGN\_FACTORY\_ADDRESS=0xC4e6726Cd8083eD88b0e521e9a8EF3a8701FBB32 |

Etherscan에서 본 컨트랙트를 확인 가능합니다.

* Mock USDC Token Contract
  + <https://sepolia.etherscan.io/address/0x5eA3DE89105aE1be35219eaf91b86A3Ba8607D59>
* Campaign Factory Contract
  + <https://sepolia.etherscan.io/address/0xC4e6726Cd8083eD88b0e521e9a8EF3a8701FBB32>

1. 백엔드 설정 확인

application.properties에 이미 배포된 주소가 설정되어 있는지 확인합니다.

|  |
| --- |
| ...  blockchain.contracts.usdc-address=0x5eA3DE89105aE1be35219eaf91b86A3Ba8607D59  blockchain.contracts.campaign-address=0xC4e6726Cd8083eD88b0e521e9a8EF3a8701FBB32  ... |

# 백엔드 서버 구동

1. 의존성 설치 및 빌드

|  |
| --- |
| cd ../hanachain-backend  # Gradle 빌드  ./gradlew clean build -x test  # 빌드 성공 확인  ls -la build/libs/ |

1. 데이터베이스 마이그레이션

* 백엔드 첫 실행 시 Flyway를 통해 자동으로 스키마를 생성합니다.

1. 어플리케이션 실행

|  |
| --- |
| # Gradle로 실행 (개발 환경)  ./gradlew bootRun |

# FDS 시스템 구동

1. Python 가상 환경 설정

|  |
| --- |
| cd ../hanachain-fds  # 가상 환경 생성  python3.9 -m venv .venv  # 가상 환경 활성화  source .venv/bin/activate # Linux/macOS  # 또는  .venv\Scripts\activate # Windows |

1. 의존성 설치

|  |
| --- |
| # 배포용 의존성 설치  pip install -r deploy/requirements.txt |

1. 모델 확인

|  |
| --- |
| # 학습된 모델 파일 확인  ls -la data/models/ |

* 1. 모델 재학습 필요 시

|  |
| --- |
| # 모델 재학습  python src/trainer.py |

* + - * 재학습 시 15분 가량 소요됨

1. FDS API 서버 실행

|  |
| --- |
| cd deploy  # API 서버 시작  python api\_server.py  # 백그라운드 실행  nohup python api\_server.py > fds.log 2>&1 & |

# 프론트엔드 구동

1. 의존성 설치

|  |
| --- |
| cd ../hanachain-frontend  # pnpm으로 패키지 설치  pnpm install  # 설치 확인  ls -la node\_modules/ |

1. 서버 실행

|  |
| --- |
| # 개발 모드로 실행  pnpm dev |