**📊 포트폴리오 통합 관리 시스템 (Portfolio Management System)**

**📋 프로젝트 개요**

**프로젝트 정보**

* **프로젝트명**: 포트폴리오 통합 관리 시스템 (PMS - Portfolio Management System)
* **버전**: v1.0.0
* **개발 기간**: 2025.06 ~ 2025.08 (3개월)
* **개발자**: [개발자명]
* **프로젝트 타입**: 웹 애플리케이션

**프로젝트 목적**

주식과 암호화폐 자산을 하나의 플랫폼에서 통합 관리하고, 실시간 포트폴리오 현황과 수익률을 시각적으로 제공하는 웹 서비스

**🎯 주요 기능 (Features)**

**1. 사용자 관리**

* ✅ 소셜 로그인 (구글, 카카오, 네이버)
* ✅ OAuth 2.0 기반 인증
* ✅ JWT 토큰 관리
* ✅ 프로필 관리

**2. 자산 연동**

* ✅ 암호화폐 거래소 API 연동 (업비트, 빗썸)
* ✅ 증권사 API 연동 (키움증권, 한국투자증권)
* ✅ API 키 암호화 저장
* ✅ 수동 자산 입력

**3. 포트폴리오 관리**

* ✅ 실시간 자산 현황 조회
* ✅ 수익률 계산 (실현/미실현)
* ✅ 자산별 분류 관리
* ✅ 평균 매수가 계산

**4. 데이터 시각화**

* ✅ 자산 분배 파이차트
* ✅ 수익률 추이 라인차트
* ✅ 종목별 수익/손실 바차트
* ✅ 실시간 가격 업데이트

**5. 실시간 기능**

* ✅ WebSocket 기반 실시간 가격 업데이트
* ✅ 포트폴리오 가치 실시간 갱신
* ✅ 가격 알림 기능

**🛠️ 기술 스택 (Technology Stack)**

**Backend**

| **기술** | **버전** | **용도** |
| --- | --- | --- |
| Java | 17+ | 메인 개발 언어 |
| Spring Boot | 3.2.5 | 웹 프레임워크 |
| Spring Security | 6.2+ | 인증/보안 |
| Spring Data JPA | 3.2+ | ORM |
| Spring WebSocket | 6.1+ | 실시간 통신 |
| MySQL | 8.0+ | 데이터베이스 |
| Redis | 7.0+ | 캐싱 |
| Gradle | 8.0+ | 빌드 도구 |

**Frontend**

| **기술** | **버전** | **용도** |
| --- | --- | --- |
| HTML5 | - | 마크업 |
| CSS3 | - | 스타일링 |
| JavaScript | ES6+ | 클라이언트 로직 |
| Chart.js | 4.0+ | 차트 라이브러리 |
| Bootstrap | 5.3+ | UI 프레임워크 |
| Axios | 1.6+ | HTTP 클라이언트 |

**External APIs**

| **API** | **용도** |
| --- | --- |
| 업비트 Open API | 암호화폐 데이터 |
| 빗썸 API | 암호화폐 데이터 |
| 키움 Open API+ | 주식 데이터 |
| 한국투자증권 API | 주식 데이터 |

**🏗️ 시스템 아키텍처**

**전체 구조**

┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐

│ Frontend │ │ Backend │ │ Database │

│ (HTML/JS) │◄──►│ (Spring Boot) │◄──►│ (MySQL) │

│ │ │ │ │ │

│ - Chart.js │ │ - REST API │ │ - Users │

│ - Bootstrap │ │ - WebSocket │ │ - Holdings │

│ - Axios │ │ - JWT Auth │ │ - API Keys │

└─────────────────┘ └─────────────────┘ └─────────────────┘

│

▼

┌─────────────────┐

│ External APIs │

│ │

│ - 업비트 API │

│ - 키움증권 API │

│ - 가격 API │

└─────────────────┘

**패키지 구조**

com.portfolio.pms/

├── config/ # 설정 클래스

├── controller/ # REST 컨트롤러

├── service/ # 비즈니스 로직

├── repository/ # 데이터 접근

├── entity/ # JPA 엔티티

├── dto/ # 데이터 전송 객체

├── security/ # 보안 관련

├── external/ # 외부 API 클라이언트

└── util/ # 유틸리티 클래스

**📊 데이터베이스 설계**

**ERD (Entity Relationship Diagram)**

┌─────────────┐ ┌─────────────┐ ┌─────────────┐

│ Users │ │ Api\_Keys │ │ Holdings │

├─────────────┤ ├─────────────┤ ├─────────────┤

│ id (PK) │◄───┤ user\_id(FK) │ │ user\_id(FK) │◄──┐

│ email │ │ exchange │ │ symbol │ │

│ password │ │ access\_key │ │ quantity │ │

│ name │ │ secret\_key │ │ avg\_price │ │

│ created\_at │ │ is\_active │ │ asset\_type │ │

└─────────────┘ └─────────────┘ └─────────────┘ │

│

┌─────────────────────────────────────────────────┘

│

▼

┌─────────────┐

│Price\_History│

├─────────────┤

│ id (PK) │

│ symbol │

│ price │

│ timestamp │

└─────────────┘

**주요 테이블 스키마**

-- 사용자

CREATE TABLE users (

id BIGINT AUTO\_INCREMENT PRIMARY KEY,

email VARCHAR(255) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL,

name VARCHAR(100) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- API 키

CREATE TABLE api\_keys (

id BIGINT AUTO\_INCREMENT PRIMARY KEY,

user\_id BIGINT NOT NULL,

exchange\_type ENUM('UPBIT', 'KIWOOM') NOT NULL,

access\_key TEXT NOT NULL,

secret\_key TEXT NOT NULL,

is\_active BOOLEAN DEFAULT TRUE,

FOREIGN KEY (user\_id) REFERENCES users(id)

);

-- 보유 자산

CREATE TABLE holdings (

id BIGINT AUTO\_INCREMENT PRIMARY KEY,

user\_id BIGINT NOT NULL,

symbol VARCHAR(20) NOT NULL,

name VARCHAR(100) NOT NULL,

quantity DECIMAL(18,8) NOT NULL,

average\_price DECIMAL(18,2) NOT NULL,

asset\_type ENUM('STOCK', 'CRYPTO') NOT NULL,

FOREIGN KEY (user\_id) REFERENCES users(id)

);

**🔌 API 명세서 (API Specification)**

**인증 API**

| **Method** | **Endpoint** | **Description** | **Request** | **Response** |
| --- | --- | --- | --- | --- |
| GET | /oauth2/authorize/google | 구글 로그인 | - | redirect to Google |
| GET | /oauth2/authorize/kakao | 카카오 로그인 | - | redirect to Kakao |
| GET | /oauth2/authorize/naver | 네이버 로그인 | - | redirect to Naver |
| GET | /login/oauth2/code/{provider} | OAuth 콜백 | code | {token, user} |
| POST | /api/auth/refresh | 토큰 갱신 | {refreshToken} | {token} |

**포트폴리오 API**

| **Method** | **Endpoint** | **Description** | **Response** |
| --- | --- | --- | --- |
| GET | /api/portfolio/summary | 포트폴리오 요약 | {totalValue, profit, profitRate} |
| GET | /api/portfolio/holdings | 보유 자산 목록 | [{symbol, quantity, value, profit}] |
| POST | /api/portfolio/sync | 외부 데이터 동기화 | {status, message} |

**차트 API**

| **Method** | **Endpoint** | **Description** | **Response** |
| --- | --- | --- | --- |
| GET | /api/charts/distribution | 자산 분배 차트 | {labels[], datasets[]} |
| GET | /api/charts/trend | 수익률 추이 | {dates[], values[]} |

**🚀 개발 로드맵**

**Phase 1: 기반 구축 (3주)**

* [x] 프로젝트 셋업
* [x] 데이터베이스 설계
* [x] 사용자 인증 시스템
* [x] 기본 CRUD API
* [x] 프론트엔드 기본 구조

**Phase 2: 핵심 기능 (4주)**

* [ ] 외부 API 연동 (업비트, 키움)
* [ ] 포트폴리오 계산 로직
* [ ] 차트 데이터 API
* [ ] 실시간 가격 업데이트
* [ ] 기본 대시보드 UI

**Phase 3: 고도화 (3주)**

* [ ] WebSocket 실시간 통신
* [ ] Redis 캐싱 구현
* [ ] 가격 알림 기능
* [ ] 성능 최적화
* [ ] 테스트 코드 작성

**Phase 4: 배포 및 운영 (2주)**

* [ ] Docker 컨테이너화
* [ ] CI/CD 파이프라인
* [ ] 운영 환경 배포
* [ ] 모니터링 구축

**🔒 보안 고려사항**

**데이터 보안**

* API 키 AES-256 암호화 저장
* 비밀번호 BCrypt 해싱
* JWT 토큰 기반 인증
* HTTPS 통신 강제

**API 보안**

* Rate Limiting 적용
* CORS 정책 설정
* SQL Injection 방어
* XSS 공격 방어

**📝 개발 규칙**

**코딩 컨벤션**

* Java: Google Java Style Guide
* JavaScript: Airbnb JavaScript Style Guide
* 변수명: camelCase
* 클래스명: PascalCase
* 상수명: UPPER\_SNAKE\_CASE

**Git 브랜치 전략**

main (프로덕션)

├── develop (개발)

│ ├── feature/user-auth

│ ├── feature/portfolio-api

│ └── feature/chart-integration

└── hotfix/critical-bug-fix

**커밋 메시지 규칙**

feat: 새로운 기능 추가

fix: 버그 수정

docs: 문서 수정

style: 코드 포맷팅

refactor: 코드 리팩토링

test: 테스트 코드

chore: 빌드 업무 수정

**📋 테스트 계획**

**단위 테스트**

* JUnit 5 사용
* Service 계층 테스트
* Repository 계층 테스트
* 코드 커버리지 80% 이상 목표

**통합 테스트**

* TestContainers로 DB 테스트
* MockWebServer로 외부 API 테스트
* Spring Boot Test 활용

**성능 테스트**

* JMeter로 부하 테스트
* 동시 사용자 1000명 목표
* 응답시간 1초 이내 목표

**📚 참고 문서**

**외부 API 문서**

* [업비트 Open API](https://docs.upbit.com/)
* [키움증권 Open API+](https://www.kiwoom.com/nkw.templateFrameSet.do?m=m1408000000)
* [한국투자증권 KIS API](https://apiportal.koreainvestment.com/)

**기술 문서**

* [Spring Boot Reference](https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/)
* [Chart.js Documentation](https://www.chartjs.org/docs/latest/)
* [MySQL 8.0 Reference](https://dev.mysql.com/doc/refman/8.0/en/)

**📞 연락처 및 정보**

* **개발자**: [개발자명]
* **이메일**: [이메일]
* **프로젝트 저장소**: [GitHub Repository URL]
* **이슈 트래킹**: [GitHub Issues URL]
* **문서 버전**: v1.0.0
* **최종 수정일**: 2025-06-15