프레임워크실습 기말 텀 프로젝트

캐릭터 정보 조회 시스템

프레임워크실습

2204110218 정아형

1. 프로젝트 개요

1.1. 개발 배경 및 목적

메이플스토리 게임의 다양한 캐릭터 정보를 쉽고 빠르게 조회할 수 있는 웹 기반 플랫폼을 사용만해보고 구조를 생각하거나 만들어본 경험이 없어 개인학습을 위해 개발하였습니다. Nexon Open API를 활용하여 실시간 캐릭터 정보를 제공하며, 사용자는 즐겨찾기와 검색 기록 관리 기능을 통해 정보를 관리할 수 있습니다.

2. 주요 기능

2.1. 사용자 인증 시스템

- **회원가입**: username, email, password을 통한 회원가입
- **로그인**: JWT 기반 토큰 인증 (Access Token 1시간, Refresh Token 7일)
- **보안**: BCrypt 비밀번호 암호화, Spring Security 적용

2.2. 캐릭터 정보 조회

- **실시간 조회**: Nexon Open API 연동으로 최신 정보 제공
- **상세 정보**:
    - 기본 정보 (레벨, 직업, 월드, 길드)
    - 장비 정보 및 세트 효과
    - 스탯 정보 (전투력, 주스탯 등)
    - 심볼, 하이퍼스탯, 어빌리티
    - 스킬, 유니온, 업적 정보
- **캐싱**: Redis를 활용한 API 응답 캐싱

2.3. 즐겨찾기 관리

- 관심 캐릭터 즐겨찾기 등록/삭제
- 즐겨찾기 목록 조회
- 빠른 재검색 기능

2.4. 검색 기록 관리

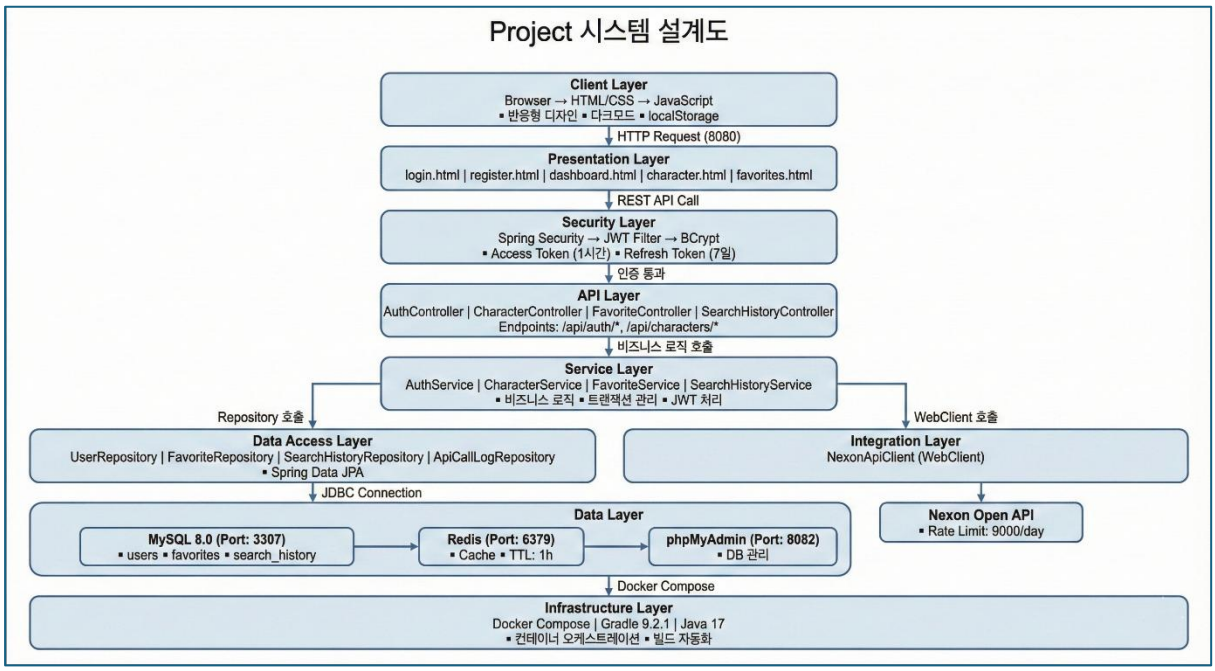
- 로그인 사용자 검색 기록 자동 저장
- 최근 검색 기록 조회 (최대 50개)
- 검색 기록 기반 빠른 재검색
- 개별/전체 삭제 기능

2.5. UI/UX 기능

- 반응형 웹 디자인
- 다크모드/라이트모드 테마 전환
- 직관적인 사용자 인터페이스

3. 시스템 설계도

3.1 전체 시스템 아키텍처



3.2 계층별 설명

Client Layer (클라이언트 계층)

- **구성**: 웹 브라우저
- **기술**: HTML, CSS, JavaScript

- **특징**: 반응형 디자인, 다크모드 지원

Presentation Layer (프레젠테이션 계층)

- **페이지 구성**:
    - login.html: 로그인 페이지
    - register.html: 회원가입 페이지
    - dashboard.html: 메인 대시보드
    - character.html: 캐릭터 상세 페이지
    - favorites.html: 즐겨찾기 관리 페이지

Security Layer (보안 계층)

- **Spring Security**: 전체 보안 프레임워크
- **JWT Filter**: 토큰 추출 및 유효성 검증
- **BCrypt**: 비밀번호 암호화

API Layer (REST API 계층)

- **AuthController**: 회원가입/로그인 (/api/auth/*)
- **CharacterController**: 캐릭터 조회 (/api/characters/*)
- **FavoriteController**: 즐겨찾기 관리 (/api/favorites/*)
- **SearchHistoryController**: 검색 기록 (/api/search-history/*)

Service Layer (비즈니스 로직 계층)

- **AuthService**: 사용자 인증 및 JWT 생성
- **CharacterService**: 캐릭터 정보 조회 및 데이터 가공
- **FavoriteService**: 즐겨찾기 비즈니스 로직
- **SearchHistoryService**: 검색 기록 관리

Data Access Layer (데이터 접근 계층)

- **UserRepository**: 사용자 정보 CRUD
- **FavoriteRepository**: 즐겨찾기 CRUD

- **SearchHistoryRepository**: 검색 기록 CRUD
- **ApiCallLogRepository**: API 호출 로그 관리
- **기술**: Spring Data JPA

Integration Layer (외부 API 연동)

- **NexonApiClient**: WebClient 기반 API 호출
- **Nexon Open API**: 메이플스토리 캐릭터 정보 제공

Data Layer (데이터 계층)

- **MySQL**:
    - users: 사용자 정보
    - favorites: 즐겨찾기
    - search_history: 검색 기록
    - api_call_log: API 호출 통계
- **Redis (Port 6379)**: 캐릭터 정보 캐싱

Infrastructure Layer (인프라 계층)

- **Docker Compose**: MySQL, Redis, phpMyAdmin 컨테이너 오케스트레이션
- **Gradle 9.2.1**: 빌드 및 의존성 관리
- **Java 17**: 런타임 환경

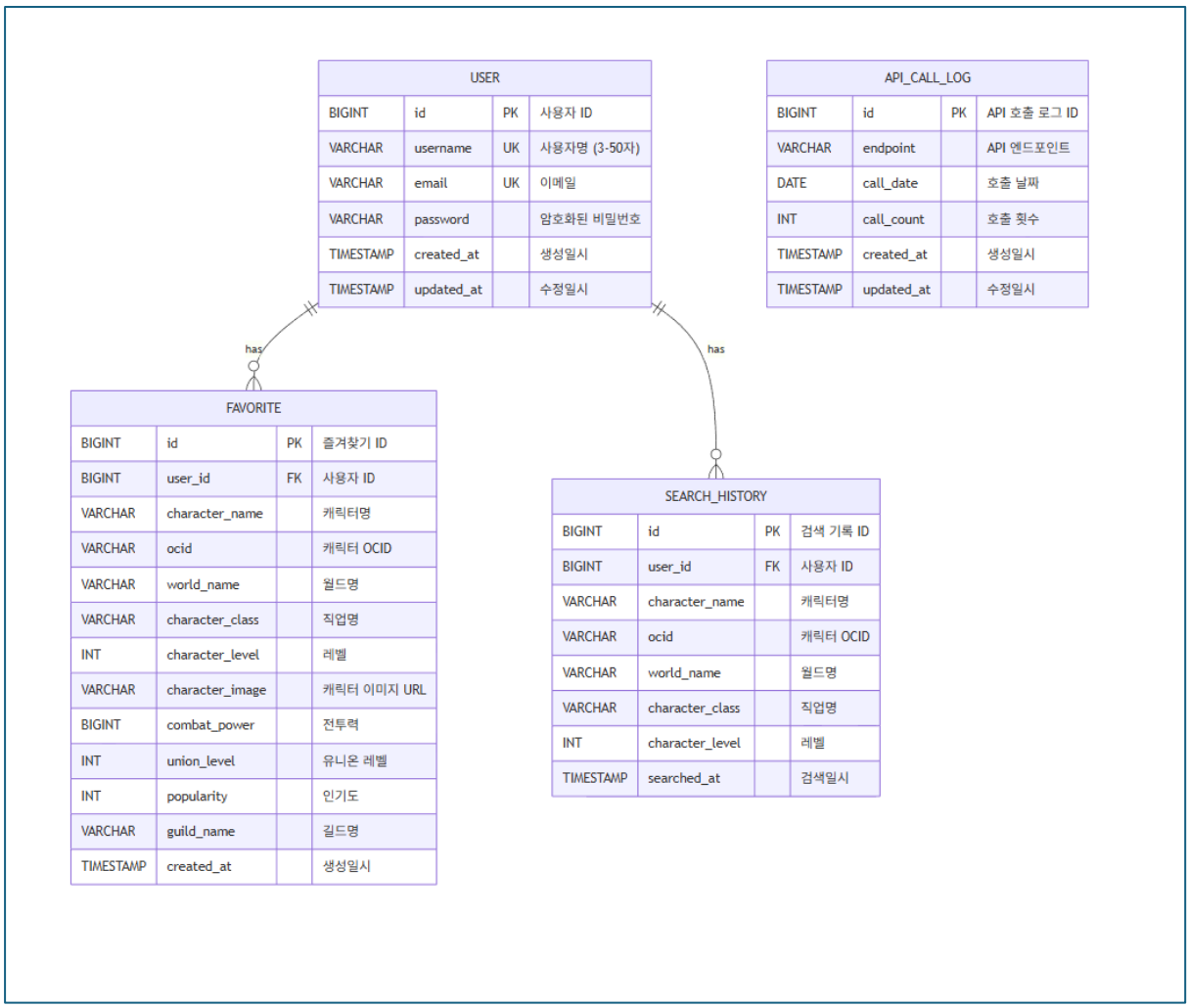
3.3 데이터 흐름

1. 사용자가 웹 브라우저에서 요청
2. Presentation Layer에서 REST API 호출
3. Security Layer에서 JWT 토큰 검증
4. API Layer에서 요청 라우팅
5. Service Layer에서 비즈니스 로직 처리
6. Data Access Layer를 통해 데이터베이스 접근
7. 필요 시 Integration Layer를 통해 외부 API 호출

8. 응답 데이터를 클라이언트에 반환

4. DB 설계

4.1 E-R 다이어그램



4.2 테이블 설명

USER (사용자)

- **역할**: 시스템 사용자 정보 관리
- **주요 컬럼**:
  - id: 사용자 고유 ID
  - username: 사용자명 (3-50자)
  - email: 이메일 주소
  - password: BCrypt 암호화된 비밀번호

    - created_at: 가입일시
    - updated_at: 수정일시
- **제약조건**: username과 email은 중복 불가

FAVORITE (즐겨찾기)

- **역할**: 사용자가 즐겨찾기한 캐릭터 정보 저장
- **주요 컬럼**:
    - id: 즐겨찾기 고유 ID
    - user_id: 사용자 ID
    - character_name: 캐릭터명
    - ocid: Nexon Open Character ID
    - world_name: 월드명 등등

SEARCH_HISTORY (검색 기록)

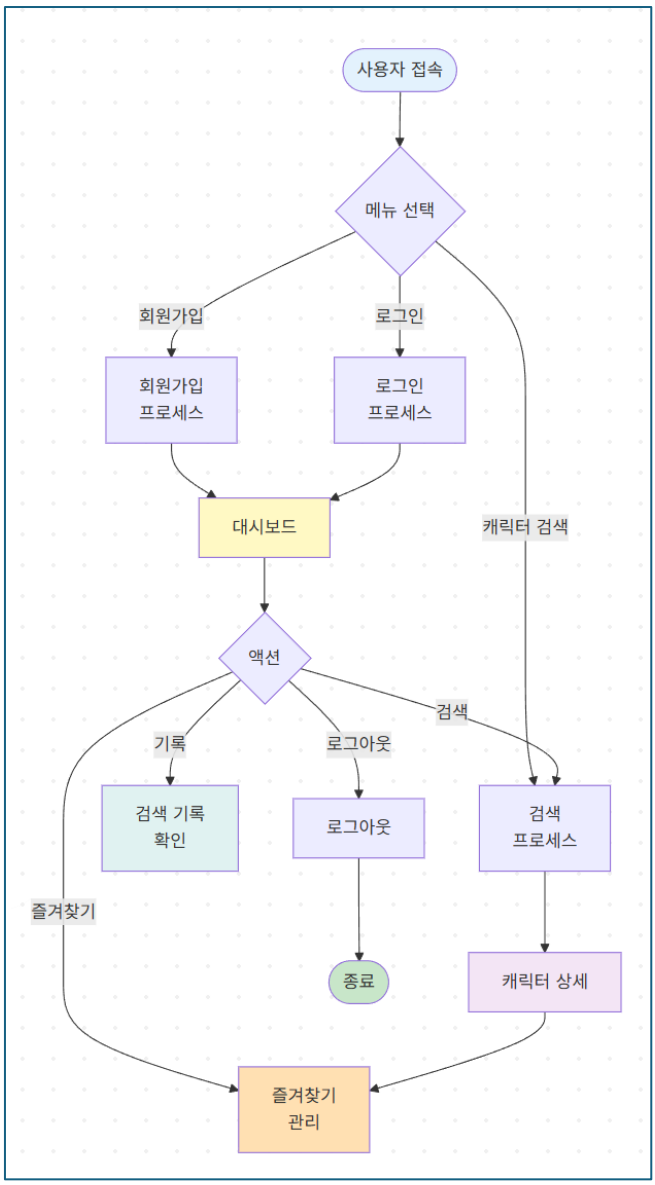
- **역할**: 로그인 사용자의 캐릭터 검색 기록 저장
- **주요 컬럼**:
    - id: 검색 기록 고유 ID
    - user_id: 사용자 ID
    - character_name: 검색한 캐릭터명
    - searched_at: 검색일시

API_CALL_LOG (API 호출 로그)

- **역할**: Nexon API 호출 통계 관리
- **주요 컬럼**:
    - id: 로그 고유 ID
    - endpoint: API 엔드포인트
    - call_date: 호출 날짜
    - call_count: 호출 횟수

5. 시스템 흐름도

5.1 전체 시스템 흐름도

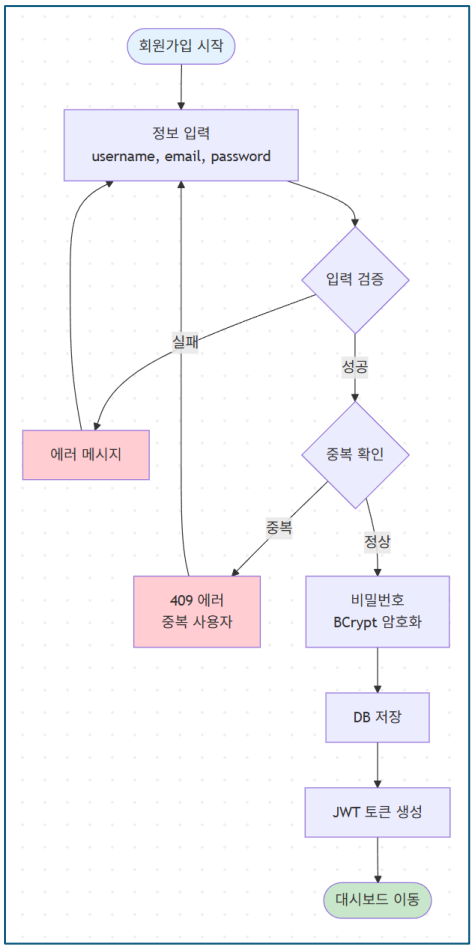


전체 흐름 설명

1. 사용자가 시스템에 접속
2. 메뉴 선택 (회원가입/로그인/캐릭터 검색)
3. 인증 후 대시보드 이동
4. 주요 기능 수행:

캐릭터 검색 및 상세 조회, 즐겨찾기 관리, 검색 기록 확인, 로그아웃

5.2 회원가입 흐름도

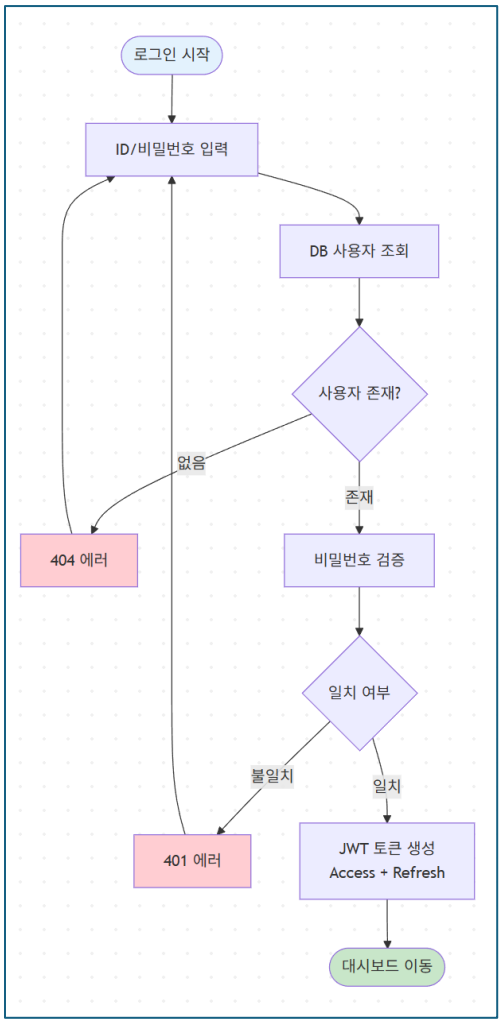


회원가입 프로세스

1. **정보 입력**: username, email, password 입력
2. **입력 검증**: 클라이언트 및 서버 유효성 검사
    - username: 3-50자
    - email: 유효한 이메일 형식
    - password: 6자 이상
3. **중복 확인**: DB에서 username, email 중복 확인
    - 중복 시: 409 에러 반환
4. **DB 저장**: users 테이블에 저장
5. **JWT 생성**: Access Token + Refresh Token

6. **대시보드 이동**: 자동 로그인 처리

5.3 로그인 흐름도



로그인 프로세스

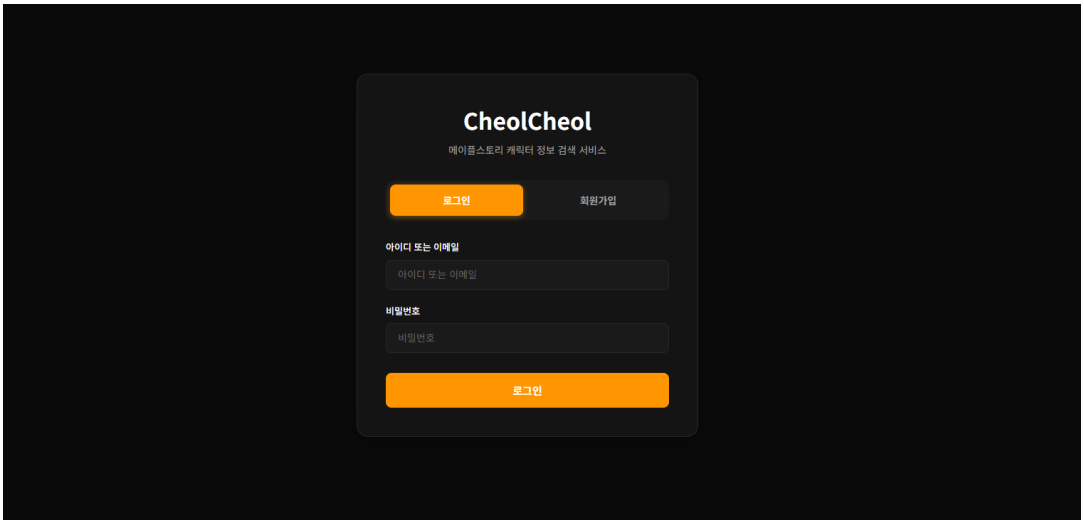
1. **ID/비밀번호 입력**: 사용자 정보 입력
2. **DB 사용자 조회**: username 또는 email로 조회
3. **사용자 존재 확인**:
    - 없음: 404 에러 반환
4. **비밀번호 검증**: BCrypt matches로 비교
5. **일치 여부 확인**:
    - 불일치: 401 에러 반환
6. **JWT 토큰 생성**: Access + Refresh Token 생성

7. **대시보드 이동**: 로그인 성공

6. 실행 화면

실행 및 결과 화면

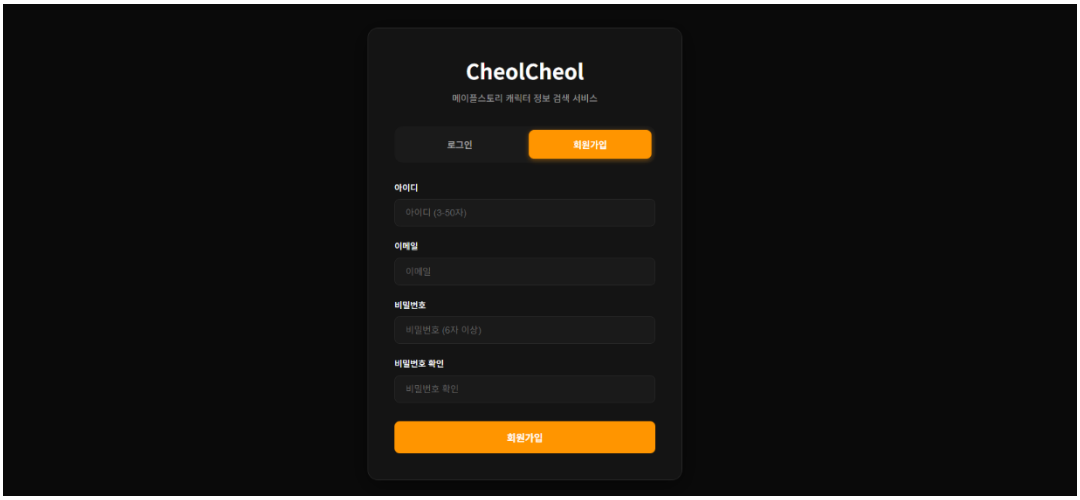
6.1 로그인 화면



주요 기능:

- 아이디/비밀번호 입력
- 유효성 검증
- 로그인 실패 시 에러 메시지 표시
- 회원가입 페이지 링크

6.2 회원가입 화면

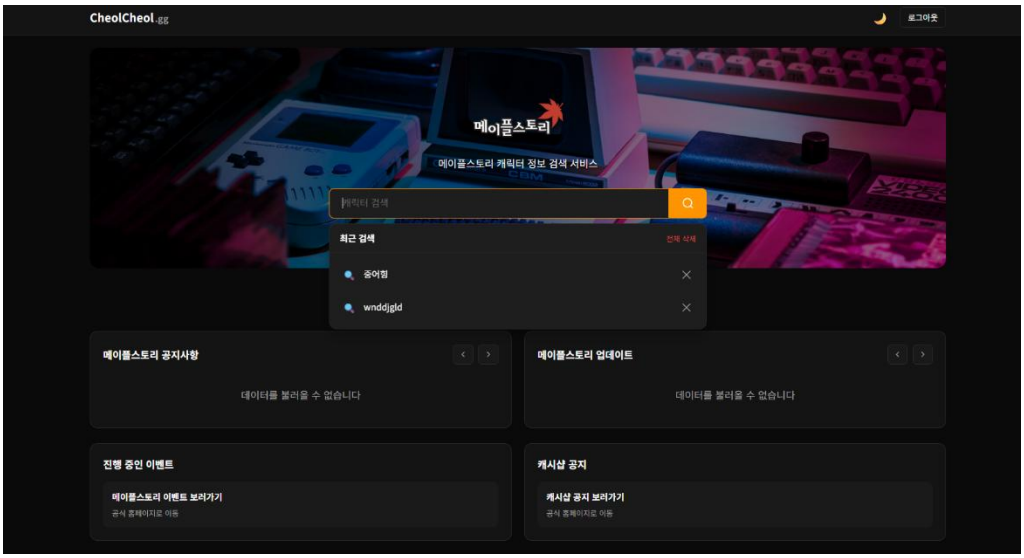


주요 기능:

- username, email, password 입력
- 실시간 유효성 검증

- 중복 확인 메시지
- 회원가입 성공 시 자동 로그인

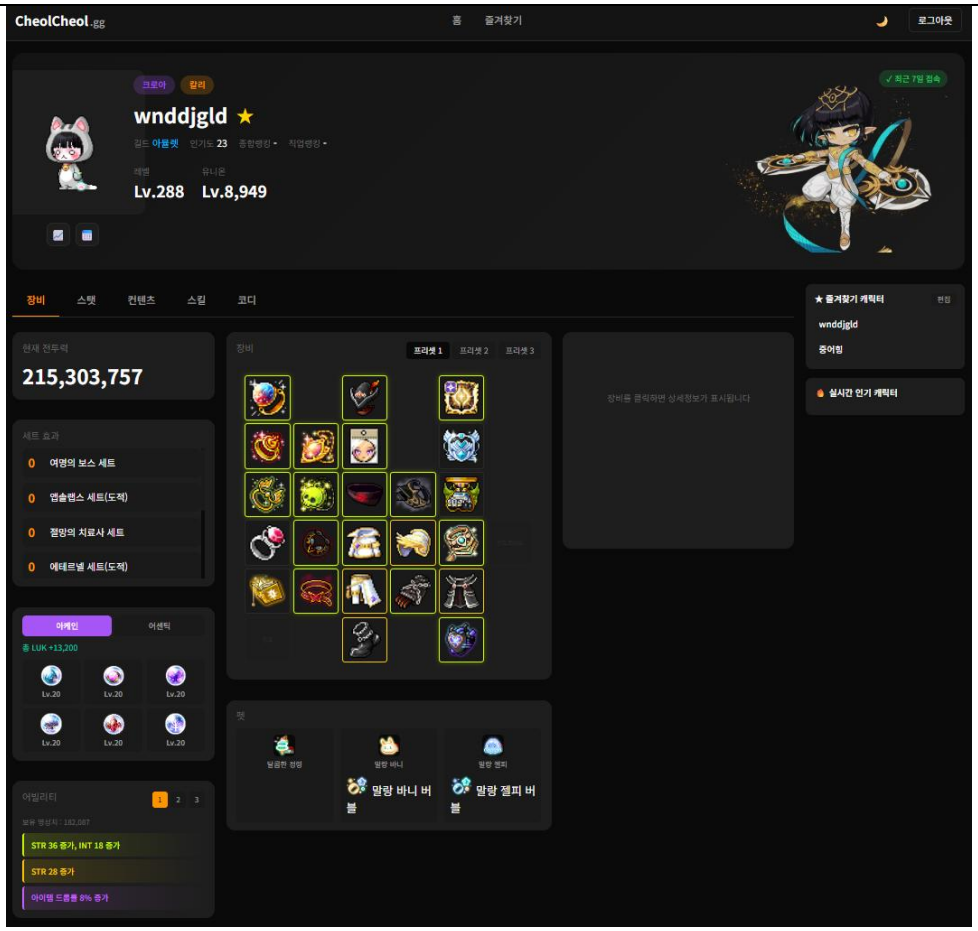
6.3 대시보드 (검색 화면)



주요 기능:

- 캐릭터명 검색
- 검색 기록 드롭다운
- 최근 검색 캐릭터 표시
- 다크모드/라이트모드 전환

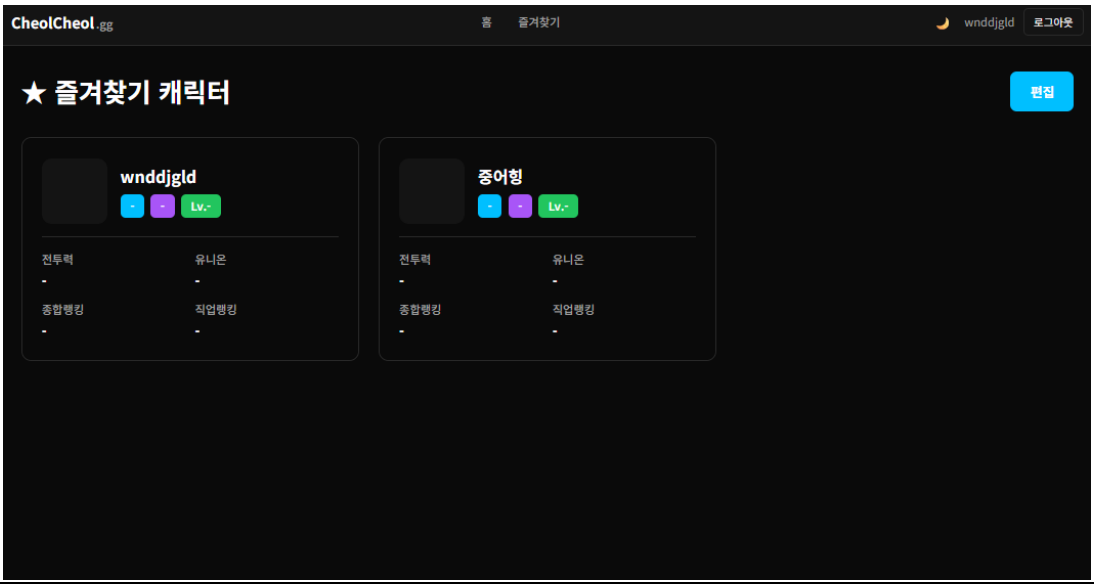
6.4 캐릭터 상세 화면



주요 기능:

- 탭 기반 정보 표시 (기본정보/장비/스탯/스킬)
- 캐릭터 이미지 표시
- 상세 스탯 정보
- 즐겨찾기 추가 버튼

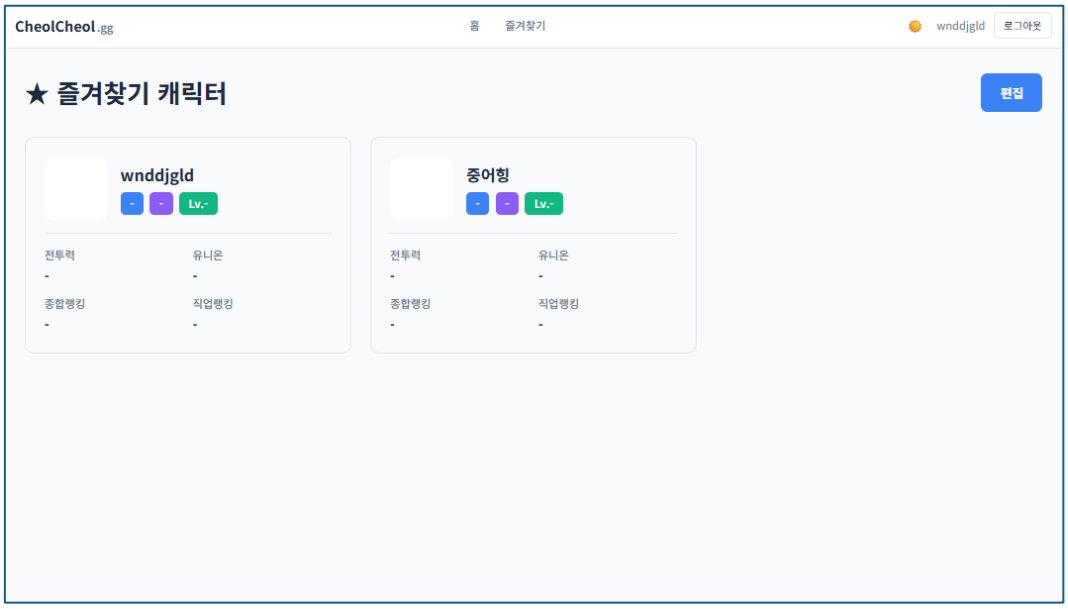
6.5 즐겨찾기 화면



주요 기능:

- 즐겨찾기 목록 카드 형태 표시
- 캐릭터 클릭 시 상세 페이지 이동
- 삭제 버튼
- 빈 상태 메시지

6.6 다크모드/라이트모드 비교



주요 기능:

- 우측 상단 테마 전환 버튼
- localStorage에 테마 설정 저장

7. 결론

7.1 프로젝트 성과

성과

1. **Spring Boot 프레임워크 숙달**
    - RESTful API 설계 및 구현
    - Spring Security를 활용한 인증/인가 시스템 구축
    - Spring Data JPA를 통한 데이터베이스 연동
2. **보안 시스템 구현**
    - JWT 기반 토큰 인증 시스템

- BCrypt 비밀번호 암호화
- CORS 설정 및 보안 필터 체인 구성

3. **외부 API 연동**
    - Nexon Open API 통합
    - WebClient를 활용한 HTTP 통신
4. **성능 최적화**
    - Redis 캐싱 전략 구현
    - JPA 최적화 (LAZY 로딩, 인덱스 활용)
    - API 응답 시간 단축
5. **Docker 기반 개발 환경**
    - Docker Compose를 활용한 컨테이너 관리
    - MySQL, Redis, phpMyAdmin 통합 구성

개발 환경 및 기술 스택
백엔드 (Backend) • 프레임워크: Spring Boot 4.0.0 • 언어: Java 17 • 보안: Spring Security 6.4.1, JWT • ORM: Spring Data JPA • API 클라이언트: Spring WebFlux • 빌드 도구: Gradle 9.2.1 **프론트엔드 (Frontend)** • 언어: Vanilla JavaScript • 마크업: HTML5 • 스타일링: CSS3 • 기능: Fetch API, localStorage, 반응형 디자인 **데이터베이스 (Database)** • 주 데이터베이스: MySQL 8.0 • 캐시: Redis **인프라 (Infrastructure)**

- 컨테이너: Docker Compose
- 버전 관리: Git/GitHub

외부 API (External API)

- 넥슨 오픈 API: 메이플스토리 캐릭터 정보

프로젝트 후기

이번 프로젝트를 통해 Spring Boot 프레임워크의 편리함을 체감할 수 있었습니다. Spring Data JPA를 활용한 데이터베이스 연동과 Spring Security를 통한 인증/인가 시스템 구현은 실무에서 중요한 기술임을 깨달았습니다. 또 Nexon Open API와 같은 외부 API를 연동하면서 실제 서비스 운영에 필요한 다양한 요소들을 고려하는 경험을 할 수 있었습니다. 개발의 환경을 구축하며 Docker Compose를 활용한 개발 환경 구축은 팀 프로젝트나 실무에서 환경 설정의 일관성을 유지하는 데 매우 유용하다는 것을 배웠습니다, 이번 프로젝트에서 배운 내용을 바탕으로 앞으로 더 복잡하고 대규모의 시스템을 개발할 수 있는 기반을 마련했다고 생각합니다