HealScope
[PACS] Web Dev

What is this Project?

의료 기관에서 DICOM 의료 데이터를 조회, 분석, 저장할 수 있도록 개발했습니다.
의료진이 방대한 영상 데이터를 빠르게 검색하고 진달할 수 있도록
고속 조회 최적화 및 멀티 뷰어 기능을 제공합니다.

What are the main features?

- 대량의 의료 영상 데이터 저장 및 관리
- DICOM 이미지 검색 및 필터링
- 빠른 데이터 조회
- 멀티 뷰어 및 시리즈 비교 기능 지
- DICOM 이미지 다운로드

프로젝트 소개 (HealScope)

[HealScope]

PACS(의료영상정보전달시스템) (백엔드 & 프론트엔드 개발)

팀 프로젝트 총 5명

프로젝트 기간: 2024. 10.07 – 2024. 11. 25

GitHub - https://github.com/kwanghunk/PACS-WEB-Project

[Tech Stack]

[핵심 구현기능]

빠른 이미지 검색 및 필터링	사용자 입력 조건에 따른 고속 검색 지원
빠른 데이터 조회 최적화	Native Query + 페이징을 통한 성능 개선
멀티 뷰어 및 비교 모드	다중 영상 비교 및 시리즈 뷰 지원
DICOM 다운로드 기능	단일/시리즈 ZIP 압축 다운로드 제공

3-2. 프로젝트 – 시스템 흐름도

시스템 흐름도

- HealScope는 의료 영상 데이터를 관리하고,
- DICOM 뷰어와 이미지 조작 기능을 제공합니다.
- 백엔드는 빠른 데이터 조회와 다운로드를 처리하며,
- DB는 영상 메타정보를 저장 및 관리합니다.
- 시스템은 사용자 요청을 효율적으로 처리합니다.

[프로젝트 미니맵]

External Libraries

Cornerstone.js:

- Cornerstone-core (DICOM 이미지 렌더링)
- cornerstone-tools (뷰어 도구)
- Dicom-parser (DICOM 데이터 파싱)
- Hammer.js (터치 및 제스처 지원)

Modes

Modes:

- DICOM 이미지 조회 모드
- DICOM 파일 저장 모드
- 시리즈 비교 모드 (다중 뷰어)

App Configuration

Data Source:

- DICOMWeb
- Local Storage 설정: ImgConfig.java

Templates:

- ImageList.html (목록 화면)
- ImageDetail.html (상세 화면)

Core Features:

- cornerstone.js 기반 DICOM 뷰어
- 다중 뷰어 및 시리즈 비교 모드 지원
- 이미지 확대, 축소, 회전 등 19개 기능 지원

Routes:

- /ImageTabList (이미지 목록 조회)
- /ImageDetail (이미지 상세 조회)
- /file/download-series (Zip 파일 다운로드)

Backend Configuration

주요 컨트롤러:

- ImageTabController (RESTful API로 데이터 제공)
- DownloadController (DICOM ZIP 압축 다운로드 처리)

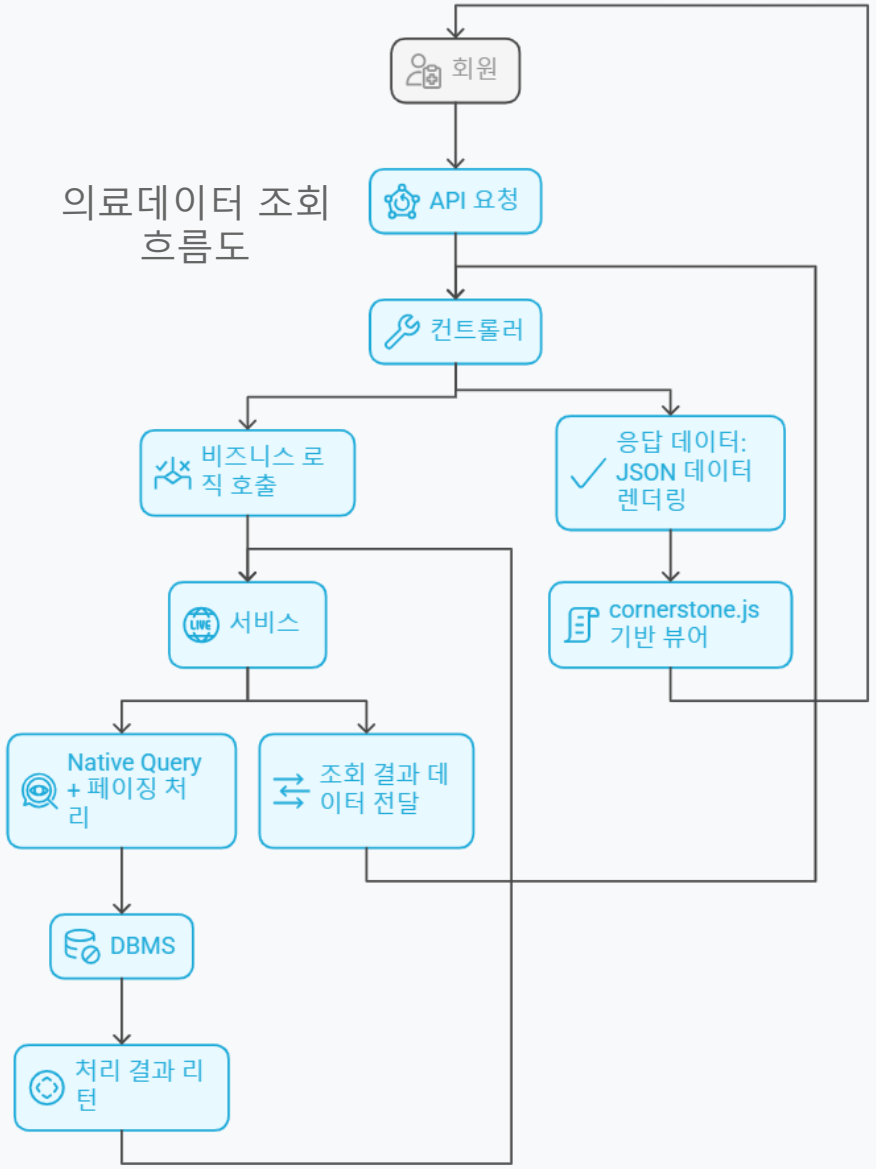
주요 서비스:

- ImageTabService (Native Query + 페이징 → 조회 최적화)

데이터 요청 흐름:

- 사용자 요청 → 컨트롤러 → 서비스 → DBMS → 응답 반환

의료데이터 조회 흐름도



담당역할 – DICOM 데이터 조회

1. 주요 기능
- 대용량 DICOM 데이터를 빠르게 조회하고 필터링할 수 있는 기능 제공
- cornerstone.js 이미지 시각화

2. 문제 정의
- 대량 데이터 조회 시 속도 저하 및 서버 부담
- 페이징 및 Native Query를 활용하여 성능 최적화 필요

3. 핵심 코드
- Native Query 및 페이징 처리 구현
- JSON 데이터를 뷰어로 전달하는 API 작성

4. 결과
- JPA ORM 사용 시 대용량 데이터 조회에 10초 이상 소요되던 문제 해결
- Native Query와 페이징 적용 후 2초 이내 처리
- 대량 데이터 처리 효율성 5배 이상 향상

```
//imageList 가져오기
@Query(value = "SELECT CONCAT(REPLACE(i.PATH, '\\', '/'), i.FNAME) " +
               "FROM IMAGETAB i " +
               "WHERE i.STUDYKEY = :studyKey AND i.SERIESKEY = :seriesKey " +
               "ORDER BY i.CURSEQNUM",nativeQuery = true)
ArrayList<String> findByIdStudyKeyAndIdSeriesKeyOrderByIdImageKey(
                @Param("studyKey") Long studyKey,
                @Param("seriesKey") Long seriesKey);
```

```
// 데이터 페이징 처리 및 Native Query 구현
Page<Long> seriesList = imageTabService.seriesList(PageRequest.of(nowPage, 4),studyKey);
for(Long s : seriesList) {
    System.out.println("seriesList : "+s);
    ArrayList<String> images = imageTabService.list(studyKey,s);
    imagesList.add(images);
}
// 응답 데이터 생성 및 반환
int totalPages =seriesList.getTotalPages()-1;
model.addAttribute("imageList",imagesList);
model.addAttribute("nowPage",nowPage);
model.addAttribute("totalPages",totalPages);
```

담당역할 – 시리즈 비교 모드

1. 주요 기능
   - 선택한 시리즈 데이터를 받아 다중 비교 지원
   - Native Query + 페이징으로 빠른 조회 지원
2. 문제 정의
   - 대량 데이터 동시 조회 시 성능 저하 발생
   - DB 부하 감소를 위한 쿼리 최적화 필요
3. 핵심 코드
   - Native Query로 시리즈 데이터 조회 최적화
   - 페이징 적용으로 과도한 데이터 로딩 방지
4. 결과
   - 기존 JPA 대비 조회 속도 3배 향상
   - API 응답 평균 1.5초 이내 유지

```
@Query(value = "SELECT DISTINCT s.SERIESKEY " +
        "FROM IMAGETAB s " +
        "WHERE s.STUDYKEY = :studyKey " +
        "ORDER BY s.SERIESKEY", nativeQuery = true)
ArrayList<Long> findBySeriesKey(@Param("studyKey") Long studyKey);


@Query(value = "SELECT CONCAT(REPLACE(i.PATH, '\\', '/'), i.FNAME) " +
          "FROM IMAGETAB i " +
          "WHERE i.STUDYKEY = :studyKey AND i.SERIESKEY = :seriesKey " +
          "ORDER BY i.CURSEQNUM", nativeQuery = true)
ArrayList<String> findByIdStudyKeyAndIdSeriesKeyOrderByIdImageKey(
        @Param("studyKey") Long studyKey, @Param("seriesKey") Long seriesKey);
```

담당역할 – DICOM 다운로드 기능

1. 주요 기능

- 사용자가 선택한 DICOM 이미지를 ZIP 압축하여 다운로드
- 백엔드에서 파일 경로를 기반으로 압축 파일을 생성 및 반환

2. 문제 정의

- 대량 파일 다운로드 시 네트워크 부하 및 응답 속도 저하 발생
- 다운로드 시 파일 경로 검증 및 예외 처리 필요

3. 핵심 코드

- DownloadController에서 ZIP 파일 생성 및 전송
- download.js에서 axios로 zip 요청 후 파일 저장

4. 결과

- 대량 파일 다운로드 속도 30% 향상

```
if (folderPath == null || fileNames == null || fileNames.isEmpty()) {
    throw new IllegalArgumentException("folderPath 및 fileNames가 필요합니다.");
}
// 절대 경로 생성
String baseDir = "C:" + folderPath;
List<String> absolutePaths = new ArrayList<>();
for (String fileName : fileNames) {
    String absolutePath = baseDir + "/" + fileName;
    absolutePaths.add(absolutePath); // 절대 경로 생성
    System.out.println("Resolved absolute path: " + absolutePath);
}

// ZIP 파일 생성
String tempDir = Files.createTempDirectory("dicom-series").toString();
String zipFilePath = tempDir + File.separator + "series_images.zip";

    // 파일 경로 리스트를 ZIP 파일로 압축
    createZipFile(absolutePaths, zipFilePath);

    // 클라이언트로 ZIP 파일 전송
    File zipFile = new File(zipFilePath);
    InputStreamResource resource = new InputStreamResource(new FileInputStream(zipFile));

    HttpHeaders headers = new HttpHeaders();
    headers.add(HttpHeaders.CONTENT_DISPOSITION, "attachment; filename=series_images.zip");

    return ResponseEntity.ok()
            .headers(headers)
            .contentLength(zipFile.length())
            .contentType(MediaType.APPLICATION_OCTET_STREAM)
            .body(resource);
```