백엔드 개발 지원

PORTFOLIO

김광훈

Contact

010-8668-9294

hosookkh@gmail.com

https://github.com/kwanghunk

CONTENT

1. 자기소개 - 1분 자기 소개

1분 자기소개

신입 풀스택 개발자 김광훈입니다.

8년간의 군 복무를 통해 책임감과 팀워크를 익혔으며,

IT 기술이 만들어내는 변화와 가능성에 끌려 개발을 시작했습니다.
책임감과 팀워크 능력을 바탕으로 팀에 기여하는 개발자로 성장하고 싶습니다.

풀스택 과정과 프로젝트 경험을 통해 백엔드와 프론트엔드를 모두 경험했으며,

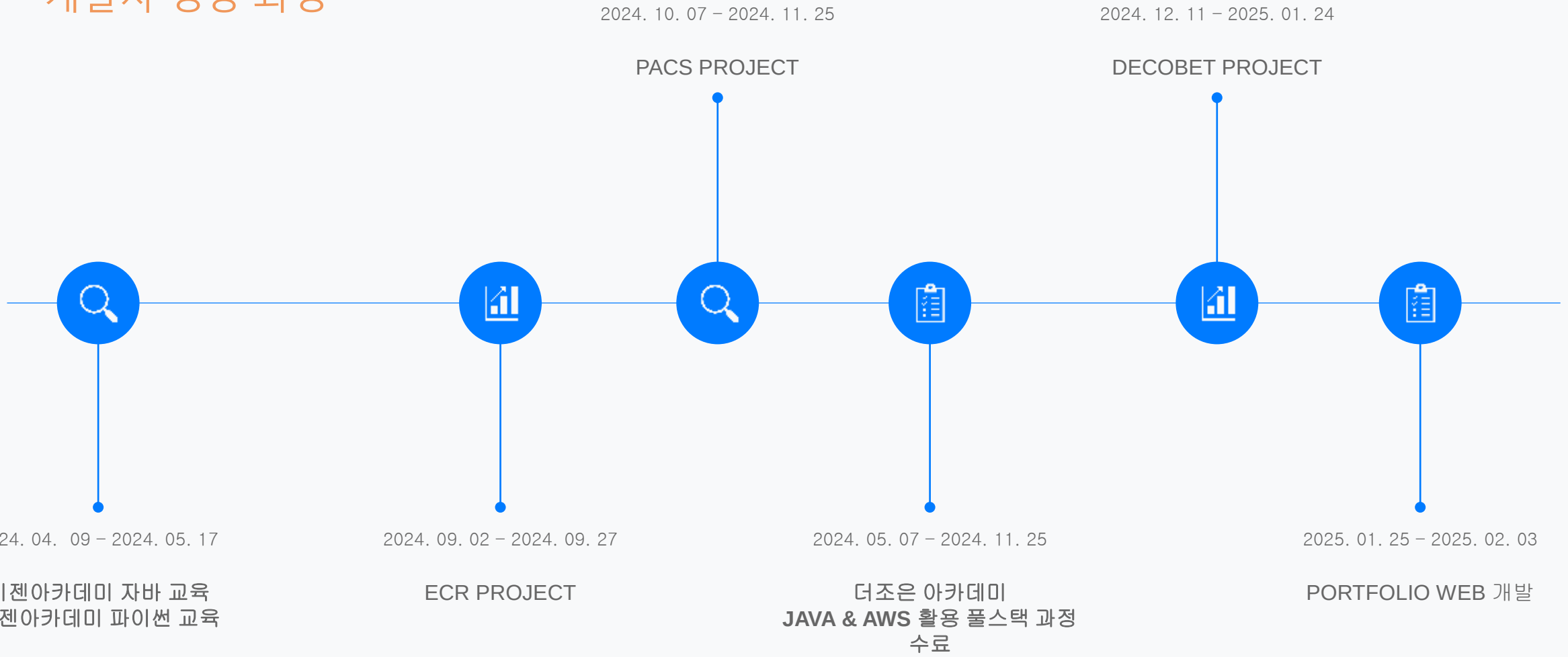
협업과 코드 품질을 중요하게 생각합니다.
특히, 아키텍처 설계와 관심사 분리를 적용하여 유지보수성과 확장성을 고려한 개발을 지향합니다.

입사 후에도 실무에서 필요한 기술과 협업 역량을 끊임없이 발전시키며,

팀과 함께 성장하는 개발자가 되고 싶습니다.

1. 자기소개 – 개발자 성장 과정

개발자 성장 과정



기술스택

주요 스택

[Backend]
- Language: Java 17
- Framework: Spring Framework, Spring Boot, Spring Security, JWT
- Hibernate: MyBatis, JPA(+ NativeQuery, JPQL)
- DBMS: Oracle DB, MySQL
- Template Engine: JSP, Thymeleaf
- Build Tool: Maven, Gradle
- Server: Apache, Tomcat
- DVCS: Git
- IDE: Eclipse, STS

[Frontend]
- HTML, CSS, JavaScript, jQuery, React
- Bootstrap

보유 역량

[Skills]
- Spring Framework 기반의 RESTful API 설계 및 구현 경험
- Native Query와 JPQL을 활용한 복잡한 데이터 처리 및 최적화
- TDD를 기반으로 JUnit, Mockito를 활용한 테스트 코드 작성 능력
- CI/CD 파이프라인 구축 경험 (GitHub Actions)
- AWS EC2를 활용한 배포 및 운영 경험
- 팀 협업 프로젝트를 통한 효율적인 Git 관리 및 코드 리뷰 경험
- 아키텍처 설계와 관심사 분리를 적용한 유지보수성과 확장성을 고려한 설계 경험
- HTML, CSS, JavaScript, jQuery, Bootstrap을 활용한 프론트엔드 개발 경험

프로젝트 목록

Escape Room(ECR)

2024. 09. 02 – 2024. 09. 27

인원: 백엔드&프론트엔드 5명

사용기술: Java, Spring Boot, JPA, React, Oracle, AWS

설명: 방탈출 카페 스마트 스토어

[담당역할]

- 동시성 제어를 활용한 안정적인 예약 시스템 구축
- 사용자 및 관리자 예약 관리 기능 구현
- 제휴업체 회원 관리 시스템 개발

HealScope

2024. 10. 07 – 2024. 11. 25

인원: 백엔드&프론트엔드 5명

사용기술: Java, Spring Security, JPA, Thymeleaf, HTML5, Oracle

설명: PACS

(의료영상정보전달시스템)

[담당역할]

- 의료 영상 조회, 조작, 전송 시스템 개발
- 대규모 데이터 처리 최적화
- 사용자 친화적 UI/UX 구현

DECOBET

2024. 12. 11 – 2025. 01. 24

인원: 백엔드&프론트엔드 3명

사용기술: Java, Spring Security, JWT, JPA, React, Redis, MySQL

설명: 주니어 개발자 코드 번역기

[담당역할]

- 아키텍처 설계 및 데이터베이스 설계
- Spring Security 설정 및 JWT 인증 구현
- 코드 번역 및 검색 API 개발
- 번역 히스토리 및 1:1 문의 관리 기능 개발

ESCAPE ROOM
[ECR] Web Dev

What is this Project?

다양한 방탈출 카페 데이터를 통합 관리하며,
검색, 예약, 리뷰 기능을 제공하는 종합 플랫폼입니다.

What are the main features?

- 방탈출 카페 검색 및 필터링
- 예약 가능한 시간대 실시간 확인 및 예약
- 사용자 리뷰 관리 및 평가

3-1. 프로젝트 – 프로젝트 소개

프로젝트 소개 (ECR)

[Escape Room]

- 방탈출 스마트 스토어 웹 서비스 개발 (백엔드 & 프론트엔드 개발)
- 팀 프로젝트 총 5명
- 2024. 09.02 – 2024. 09. 27
- GitHub - https://github.com/kwanghunk/ECR-WEB-Project

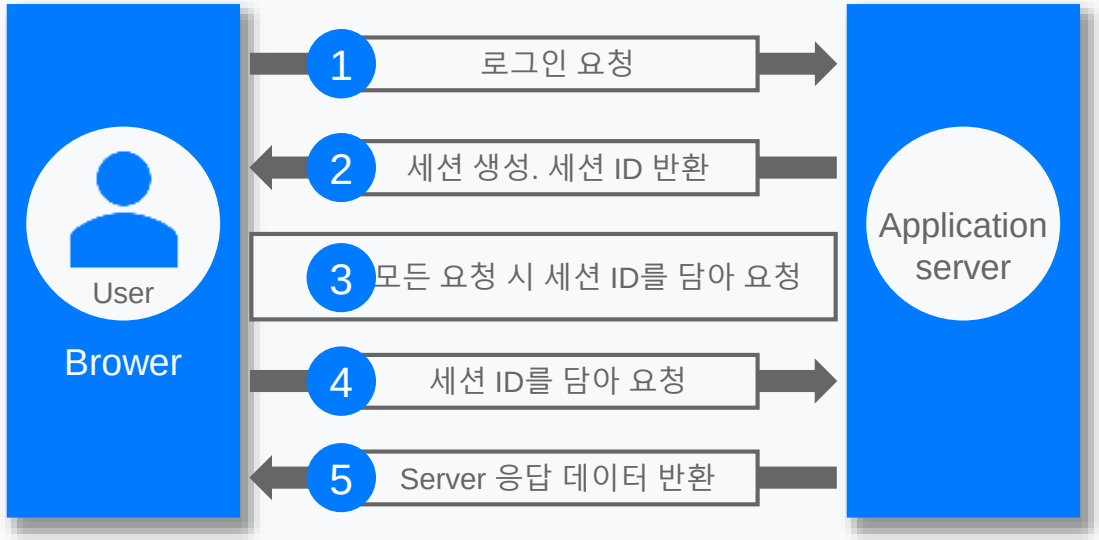
[Tech Stack]

공통	사용자	관계자	관리자
회원가입, 로그인 상품 목록 및 상품 상세 조회	상품 예약 관리 예약 내역 관리 리뷰 관리	상품 등록/삭제	사용자/관계자 회원관리 예약 상품 관리 상품, 리뷰, 공지사항 관리

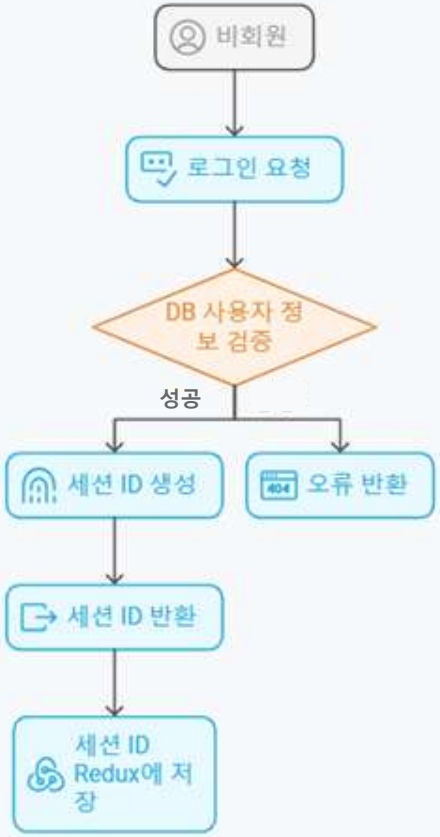
3-1. 프로젝트 – 시스템 흐름도

시스템 흐름도

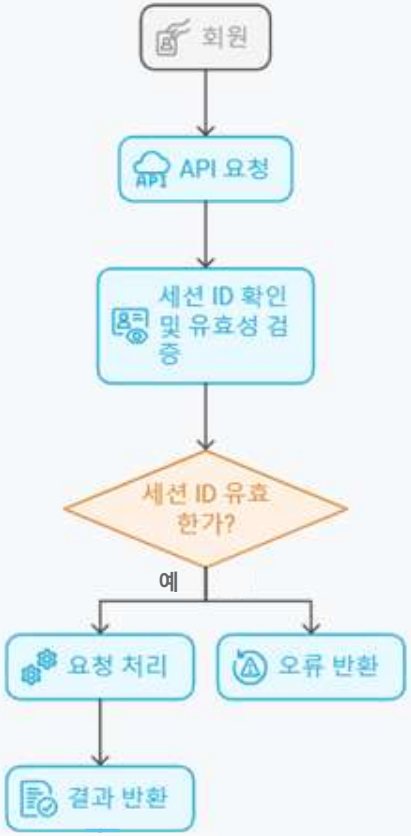
- 사용자 로그인 및 API 요청 처리 과정을 시각화 한 것입니다.
- 로그인 시 세션 ID를 생성하고, Redux에 저장하여 인증 상태를 관리합니
- API 요청은 세션 ID 유효성을 검증한 뒤 처리 결과를 반환합니다.
- 효율적인 세션 관리와 데이터 일관성을 위해 설계되었습니다.
- 확장성과 보안을 고려한 구조입니다.



로그인 흐름도



API 요청 처리 흐름도



3-1. 프로젝트 – 담당역할

담당역할 - 예약시스템

1. 주요 기능
   - “동시성 제어를 활용한 예약 시스템 구축”
2. 문제 정의
   - 다중 사용자가 동시에 예약할 경우의 데이터 충돌을 방지하고 데이터 무결성을 보장하기 위해 동시성 제어를 적용했습니다.
3. 핵심 코드
   - @Transactional을 사용하여 예약 도중 오류가 발생하면 롤백되도록 설정
   - 예약 가능 여부를 먼저 확인하고, 같은 시간대에 중복 예약이 있으면 예약을 차단
4. 결과
   - 동시 요청 발생 상황에서도 데이터 무결성 유지와 안정적인 예약 시스템이 작동.

```java
@Transactional
public boolean reserve(Reservation reservation) {
    Optional<Reservation> existingReservation =
        reservationRepository.findByTema_TemaNoAndUseDateAndUseTime(
            reservation.getTema().getTemaNo(),
            reservation.getUseDate(),
            reservation.getUseTime()
    );

    if (existingReservation.isPresent()) return false;

    reservation.setPaymentStatus("결제 대기");
    reservationRepository.save(reservation);
    return true;
}
```

3-1. 프로젝트 – 담당역할

담당역할 – 예약관리시스템

1. 주요 기능
   - 사용자 예약 내역 조회 및 취소 요청
   - 관리자 예약 승인 및 취소 처리
2. 문제 정의
   - 다중 요청으로 인한 데이터 무결성 문제 발생
   - 인증되지 않은 사용자의 예약 접근 제한 필요
3. 핵심 코드
   - 트랜잭션 적용으로 데이터 무결성 보장
   - 인증 로직 추가로 보안 강화
4. 결과
   - 트랜잭션과 인증 로직으로 안정적인 예약 관리 구현

```java
@Transactional
public Reservation approveReservation(Long reservationCode) {
    Optional<Reservation> reservation =
            reservationRepository.findById(reservationCode);
    if(reservation.isPresent()) {
        Reservation res = reservation.get();
        res.setPaymentStatus("예약확정");
        return reservationRepository.save(res);
    }
    return null;
}
```

```java
@GetMapping("/findUserReserveAll")
public ResponseEntity<?> findUserReserveAll(
        @RequestParam(name = "page") int page,
        @RequestParam(name = "size") int size,
        HttpServletRequest request) {
    try {
        Member sessionMember = AuthenticationUtils.authClient(request);
        String  userId = sessionMember.getMemberId();

        List<Reservation> result =
                reservationService.findUserReservations(userId, page, size);
        return result != null && !result.isEmpty()
                ? ResponseEntity.ok(result)
                : ResponseEntity.status(204).body("예약 내역이 없습니다.");
    } catch (UnauthorizedException e) {
        return ResponseEntity.status(403).body(e.getMessage());
    } catch (Exception e) {
        return ResponseEntity.status(500).body("서버 내부 오류 발생: " + e.getMessage());
    }
}
```

3-1. 프로젝트 – 담당역할

담당역할 – 회원관리시스템

1. 주요 기능

• 관리자: 일반회원 및 제휴업체 회원 관리

• 회원 조회(페이징 처리) 및 회원 삭제 기능 제공

2. 문제 정의

• 대량의 데이터를 처리할 때 조회 성능 저하 문제

• 회원 삭제 시 무결성 원칙 위배 오류 발생 문제

3. 핵심 코드

• 페이징 처리로 데이터 조회 성능 최적화

• 연관된 데이터 삭제를 위해 각 레포지토리를 순차적 호출

4. 결과

• 페이징 처리로 대량 데이터 처리 성능 개선

• @Transactional로 데이터를 안전하게 삭제

```
@Transactional
public void deleteMemberAndReviewsAndTemas(String memberId) {
    if(memberRepository.existsById(memberId)) {
        reviewRepository.deleteByUserId(memberId);
        temaRepository.deleteByMemberId(memberId);
        memberRepository.deleteById(memberId);
    } else {
        throw new IllegalArgumentException("해당 회원을 찾을 수 없습니다");
    }
}
```

```
@GetMapping("/findClientAll")
public ResponseEntity<?> getAllClients(
        @RequestParam(name = "loginType") int loginType,
        @RequestParam(name = "page") int page,
        @RequestParam(name = "size") int size,
        HttpServletRequest request) {
    try {
        AuthenticationUtils.authClient(request);
        List<Member> list = memberService.getAllClients(loginType, page, size);
        return ResponseEntity.ok(list);
    } catch (UnauthorizedException e) {
        return ResponseEntity.status(403).body(e.getMessage());
    }
}
```

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)

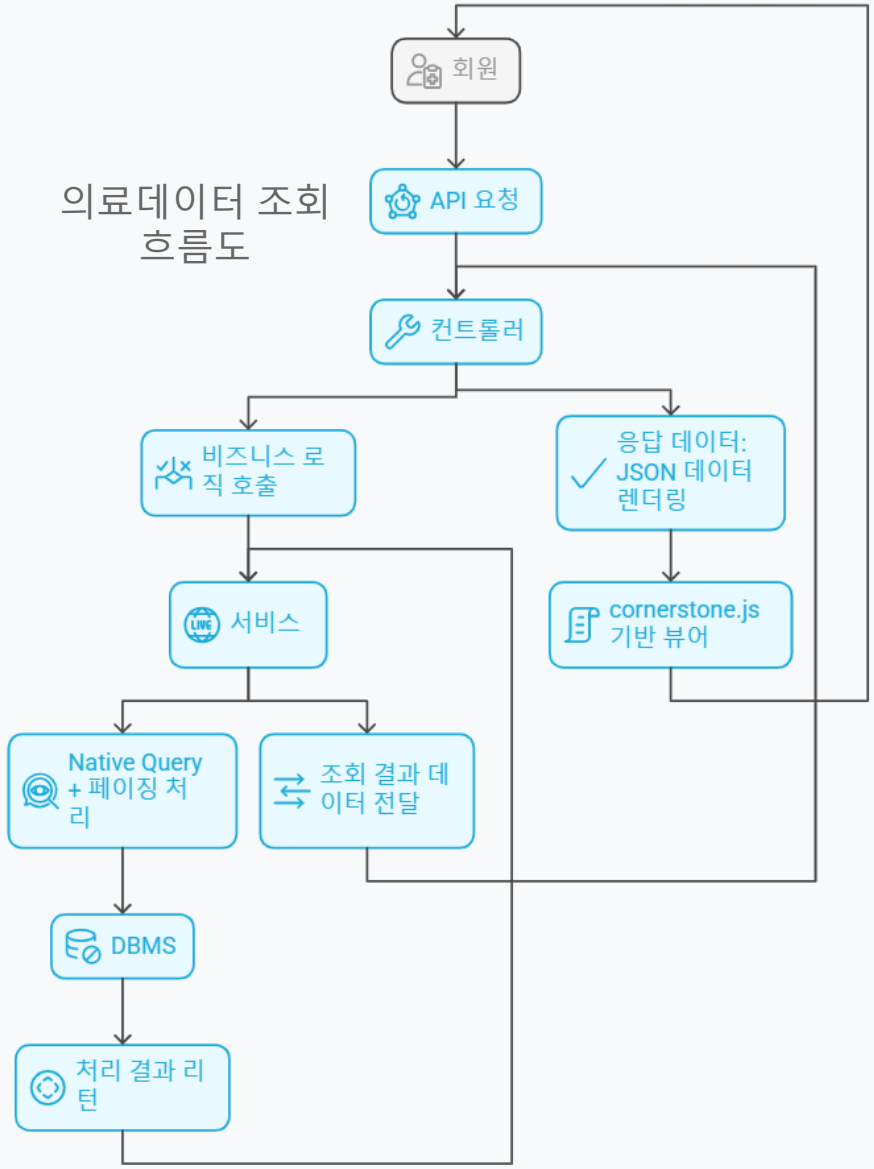
주요 서비스:

- 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);
```

Thank you for visting my portfolio

김 광 훈