데이터 구조와 서비스 아키텍처를 설계하는개발자 곽영헌입니다.

CONTACT

Phone 010-9889-5275

E-mail yhyh5275@naver.com

Github https://github.com/YoungHoney

데이터 정합성과 아키텍처를 설계하는 개발자 입니다.

공공 데이터 통합, 위치 기반 조회 성능 개선, 오픈소스 기여 경험을 통해 데이터 구조와 서비스 흐름을 설계하는 역량을 길렀습니다. 구현뿐 아니라 기술 선택 이유와 협업 방식을 문서화해 팀 개발 효율까지 높이는 백엔드 개발자를 지향합니다.

SKILLS

Java ■■■■□

- Java 버전에 따른 기능 차이를 이해하고 있음
- Star 5000++의 오픈소스 Armeria기여 경험

Spring ■■■□□

- Spring Boot로 REST API 서버 개발 가능
- Spring Security로 JWT인증 적용 가능
- Spring AI를 활용하여 RAG 구현 가능

MySQL ■■■□□

- InnoDB에 의한 인덱싱을 알고있음
- 실행계획을 통해 쿼리가 어떻게 실행될지 예측 가능

JPA ■■■□□

- N+1문제 해결 및 영속성 컨텍스트 이해

MyBatis ■■■□□

- 동적 쿼리(if/foreach)를 활용한 조건별 조회 구현
- resultMap 기반 객체 매핑

Vue.js ■■□□□

- Pinia 상태관리 적용
- Vue Router를 통한 페이지 흐름 및 인증 제어
- Axios기반 API연동 경험

CERTIFICATES / Languages

2025.12 SQLD

2025.12 정보처리기사

2024.05 TOPCIT(수준3)

2024.10 TOEIC (880)

2024.09 TOEIC Speaking (AL 160)

EDUCATION / EXPERIENCE

(2025.07 ~ 2026.) 삼성청년SW·AI아카데미 14기 재학중

- Java, Spring Boot, MyBatis, Vue.js

(2019 ~ 2025.02) 중앙대학교 소프트웨어학과 졸업(우등상)

PROJECT

(2026.01 ~ 2026.02) PAMMAP

지역축제 정보 제공 서비스

4p – 8p

(2025.12 ~ 2025.12) Nesto

부동산 청약정보 제공 서비스

9p – 16p

(2026.02 ~ 2026.03) CapSure

구독형 보험 플랫폼 구현

15p – 19p

(2025.08 ~ 2026.03) Armeria 오픈소스 기여

API 문서화 서비스에 상속, 다형성 정보를 반영하도록 PR제출

20p – 29p

지역축제 정보제공 서비스 **PAMMAP**

2026.01 ~ 2026. 02 (인원 : 6명)

▢ 프로젝트 개요

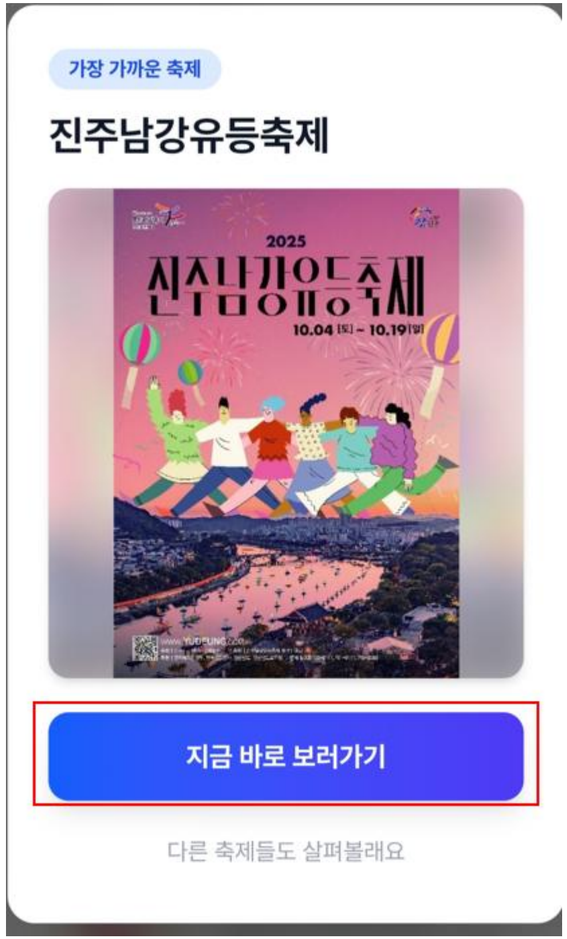
- 목표 : 주요 축제를 선정하여, 손 안의 팜플렛처럼 편하게 정보 보기
- 배경 : 순수 개발시간 2주 이내에, 부족한 축제관련 데이터를 보충하기 위해 수동으로 데이터 세팅 작업을 실시해야 했습니다.
- 주요 기능 : 지도앱으로 편하게 보는것과 같이 지역축제의 정보를 편하게 보는 것입니다.

▢ 담당 역할(역할 작성)

- BE리딩을 맡아 기술문서 작성. 컨벤션 지정. BE개발을 맡았습니다.
- BE 3
- FE 3

▢ 개인 기여도

- 축제 필터링 및 검색 기능 구현 (80%)
  → 다양한 조건 기반으로 축제를 탐색할 수 있는 검색 기능 개발
- 사용자 위치 기반 축제 조회 기능 구현 (90%)
  → 현재 위치 기준으로 가까운 축제를 제공하는 기능 개발



처음 서비스에 진입하면, 지리적으로 가까운 축제가 제공됩니다. 지금 바로 보러가기를 누르면 축제 지도 UI가 입혀진 지도 화면으로 이동합니다.

▢ 사용 기술 / 적용 맥락 및 이유

- **JPA / QueryDSL**

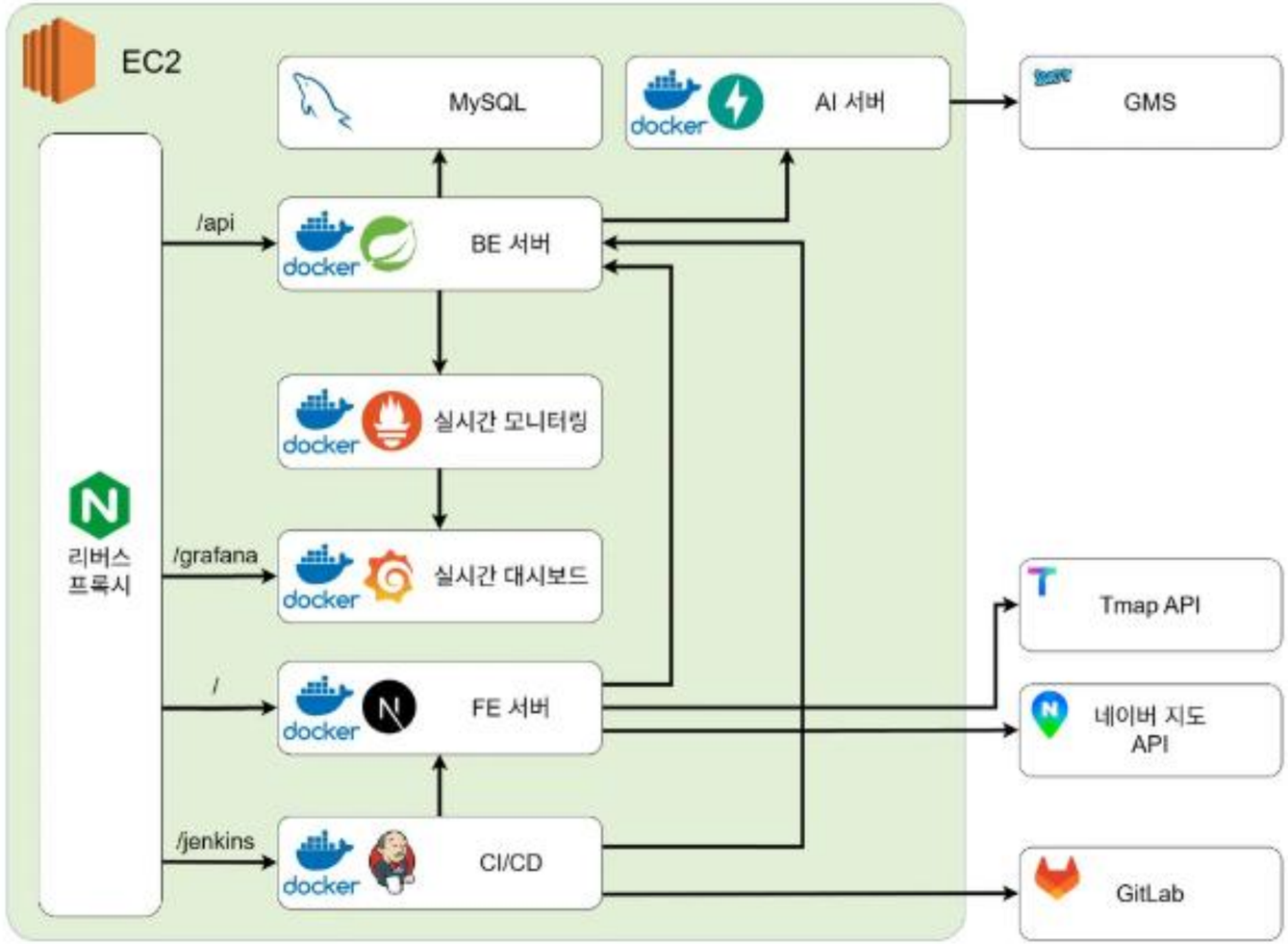
→ JDBC, MyBatis와 비교하여 객체 중심 설계 가능, 생산성을 위해 JPA도입. 축제 검색/필터링에서 동적 쿼리를 위한 QueryDSL

- **Spring Actuator + Micrometer + Prometheus + Grafana**

→ API 서버의 성능 및 상태를 실시간 모니터링 하기 위함 요청 수, 응답 시간, JVM상태 등 핵심 지표를 수집하여 병목구간 파악. Grafana 대시보드로 시각화

- **MySQL**

→ 지역 축제에 관한 정보가 구조가 명확하여 RDB 선택 팀원 대부분이 익숙했으며, MySQL의 인덱스의 동작방식을 이해하고있어서 성능 개선점을 찾기 편했습니다.



▢ 구현 사항

[실시간 인기축제 Top N 조회]

- 현재 진행 중인 축제를 우선하고, 방문객 수 기준으로 상위 N개를 조회하는 API 구현
- QueryDSL을 활용해 진행 여부 + 방문객 수 기반 복합 정렬 로직 설계

[축제 동적 검색 / 필터링]

- 지역, 월, 검색어 조건을 조합하여 축제를 유연하게 검색하는 API 구현
- QueryDSL 기반 동적 where절 구성으로 확장 가능한 검색 구조 설계.

[사용자 위치 기반 주변 축제 조회]

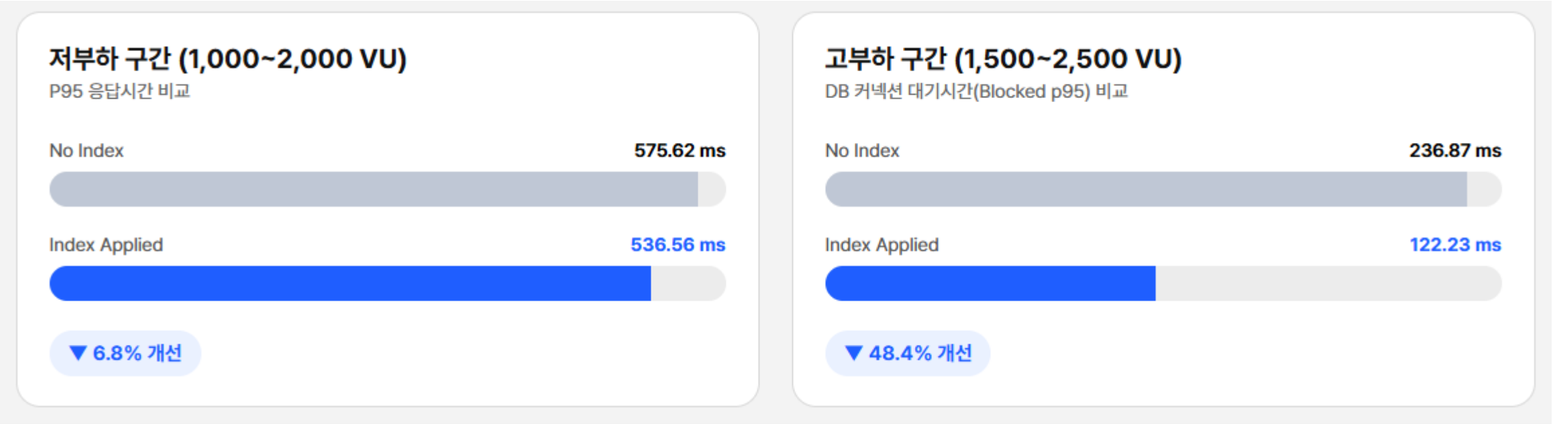
- 사용자 위/경도를 기준으로 가까운 축제를 거리순으로 조회하는 API 구현
- Haversine 공식으로 거리 계산 후 QueryDSL로 정렬, 진행 중 필터 옵션 지원

▢ 문제 해결 사례

1. 오류 분석/문제 분석

- 문제 현상 : 위치기반 주변축제 조회기능(/nearby) 에 대한 성능 테스트에서 동시 요청 증가 시 응답 지연과 DB 대기가 커졌습니다.
- 원인 : 테스트에 사용된 /nearby 기능이 인덱스를 타지 못했다고 생각했습니다.

2. 개선 과정



3. 해결 과정

1) 문제 진단 : 구현 방법을 정할 때 /nearby가 인덱스를 잘 타지 못할거라고 생각하고, 추후 고도화 문서에 남겨뒀습니다. 기능 구현이 마무리 되고 Explain으로 살펴보니 **테이블 풀 스캔(ALL)**이 발생했습니다.

2) 전략 수립 : 기존 기능은 **전체 축제를 가져와서 거리 계산 후 정렬하는 구조**였고, 실제 서비스 예시들을 살펴봤을 때, 네이버 지도의 경우 "카페"를 검색하면 반경 수 Km의 결과만 가져옵니다. 마찬가지로 **bounding box로 후보군을 줄이고**, 동일한 거리 계산을 수립했습니다.

3) 기술 적용 : (위도, 경도, 축제id) 순으로 인덱스를 설정했습니다.

4) 검증 : 임의의 3곳의 위치에서 EXPLAIN ANALYZE를 비교한 결과, 기존 ALL(테이블 풀 스캔)이 range스캔으로 변했고, ROWS_EXAMINED도 79~93% 가량 감소했습니다.

▢ PAMMAP 추가 활동 - 문서화

ADRs

ADR	Authors
[ADR-1] API 응답 및 에러 처리 표준화	@영헌 곽
[ADR-2] DTO 구현 전략	@영헌 곽
[ADR-3] 예외 처리 및 로깅 전략	@영헌 곽
[ADR-4] 모니터링 및 성능 측정 전략	@영헌 곽
[ADR-5] Swagger 문서화 전략	@영헌 곽

[ADR-2] DTO 구현 전략

🎯 Context

- Java 16부터 불변형 객체이자 DTO 지원을 위한 Record가 추가됨
- Record vs Class 어떤 구현방식을 사용할 지가 쟁점
- 사람마다 DTO 작성 스타일이 다름 (누구는 `@Data`, 누구는 `@Getter` + `@Builder`).
- 단순 조회(Read)가 많은 서비스 특성상, 데이터가 중간에 변조되지 않음을 보장하는 불변성(Immutability)이 중요함.

🤝 Decision

- **Controller와 Service 계층을 오가는 모든 Request/Response DTO는 Java `record`를 사용하여 구현한다.**
- JPA Entity는 기존대로 `class`를 사용하되, DTO는 무조건 `record`로 통일한다.
- 데이터 매핑(Entity ↔ Record)은 Record 내부에 `from` (Entity -> DTO) 또는 `toEntity` (DTO -> Entity) 메서드를 두어 처리한다.

📄 상세 규격

- 지양해야 할 방식

```
// 팀원마다 스타일이 갈리고, 코드가 길어짐
@Getter
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class FestivalResponseDto {
    private String name;
    private String location;
    // ... 필드 추가될 때마다 수정 범위가 큼
}
```

- 이렇게 합시다

```
// 선언과 동시에 생성자, Getter, equals, hashCode, toString 자동 생성
public record FestivalResponse(
    String name,
    String location,
    LocalDate startDate,
    LocalDate endDate
) {
    // Entity -> DTO 변환 로직 (Static Factory Method)
    public static FestivalResponse from(Festival entity) {
        return new FestivalResponse(
            entity.getName(),
            entity.getLocation(),
            entity.getStartDate(),
            entity.getEndDate()
        );
    }
}
```

▢ 활동 내용

- 이번 프로젝트에서는 문서화를 통해 팀원간 정보공유를 원활하게 하고자 했습니다.
->BE 리딩을 맡아서 주요 구현사항들에 대해 위와 같이 ADR형태로 만들어 공유했습니다.

▢ PAMMAP 추가 활동 -PoC

PoC Demo

생성 일시 1월 15

선택 BE

작성자 영 영헌 곽

+ 속성 추가

댓글

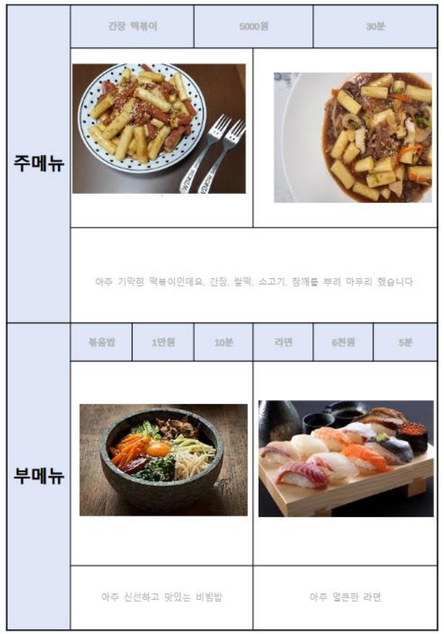
영 댓글 추가

▼ 부스 신청서 OCR

OCRPython.zip 1.7 MiB

▼ 네이버 검색-축제 후기 수집 및 정제

reviewTest_localAndGroq.zip 963.2 KiB




```
1. 메뉴 및 가격
- 간장 떡볶이 - 5000원
- 볶음밥 - 1만원
- 라면 - 6천 원

2. 사진과 설명 일치 여부
- 간장 떡볶이: 사진에는 파스타 느낌의 길쭉한 면이 보이지만 설명에는 쌀떡이 언급되어 불일치
- 볶음밥: 사진에는 돌구이가 보이지만 설명에는 비빔밥이 언급되어 불일치
- 라면: 사진에는 생선회가 보이지만 설명에는 얼큰한 라면이 언급되어 불일치

3. 최종 결론
- 메뉴 정보와 사진이 일치하지 않아 신청서를 다시 확인하고 수정할 필요가 있습니다.
Page 3 (1.31s)
1. 메뉴 및 가격
- 떡볶이 - 5천원
- 라면 - 1만원
```

```
### AI Festival Review Agent ###
분석 대상 축제: 서울세계불꽃축제
[search] 실제 리뷰 검색을 시작합니다: 서울세계불꽃축제
  -> 검색어: 서울세계불꽃축제 솔직 후기
검색된 리뷰 URL 수: 6
[scrape] https://blog.naver.com/brainy_smurf/224024271803
[scrape] https://blog.naver.com/illsang0224/224026085658
[scrape] https://blog.naver.com/wjddla1011/224024135610
[scrape] https://blog.naver.com/simile0124ever/223636419650
[scrape] https://blog.naver.com/gnayeon/224025809881
[scrape] https://blog.naver.com/kickjinsu/224033297093

--- [수집된 리뷰] ---
Title: 역대급이었다!" 2025 서울세계불꽃축제 솔직 후기 (명당, 귀가 꿀팁, 내년 전망)
URL: https://m.blog.naver.com/brainy_smurf/224024271803
----------
Title: 2025 서울 세계 불꽃 축제 직접 다녀온 솔직 후기
URL: https://m.blog.naver.com/illsang0224/224026085658
----------
Title: 2025 서울세계불꽃축제 보고 왔어요 | 관람장소 서울불꽃축제 솔직후기
URL: https://m.blog.naver.com/wjddla1011/224024135610
----------
Title: 2024 서울세계불꽃축제, 솔직후기!?? -여의도/영등포구/서울시
URL: https://m.blog.naver.com/simile0124ever/223636419650
----------
Title: 2025 서울세계불꽃축제 여의도한강공원 명당 : 솔직후기
URL: https://m.blog.naver.com/gnayeon/224025809881
----------
Title: [솔직후기] 서울세계불꽃축제 2025 +명당, 꿀팁
URL: https://m.blog.naver.com/kickjinsu/224033297093
```

```
==========================================
 [ 서울세계불꽃축제 분석 결과 ]
==========================================
>> 3줄 요약
- 2025 서울세계불꽃축제는 화려한 불꽃 쇼와 인파로 큰 인기를 끌었습니다.
  - 축제의 하이라이트는 대한민국 팀의 피날레 쇼였으며, 63빌딩을 배경으로 한강 전체를 캔버스 삼아 압도적인 스케일과 웅장한 음악으로 현장의 모든 관객에게 깊은 감동과 전율을 선사했습니다.
  - 축제는 9월 27일부터 29일까지 여의도 한강공원에서 개최되었으며, 많은 방문객들이 자리를 선점하기 위해 일찍 출발했습니다.

2. **
----------------------------------------
>> 핵심 장점 (BEST)
 - 화려한 불꽃 쇼: 축제의 하이라이트는 대한민국 팀의 피날레 쇼였으며, 63빌딩을 배경으로 한강 전체를 캔버스 삼아 압도적인 스케일과 웅장한 음악으로 현장의 모든 관객에게 깊은 감동과 전율을 선사했습니다.
 - 다양한 국가의 불꽃 쇼: 축제에는 중국, 폴란드, 캐나다, 이탈리아 등 다양한 국가의 불꽃 쇼가 existed했습니다.
 - 아름다운 행사 장소: 여의도 한강공원은 아름다운 자연 경관과 함께 불꽃 쇼를 감상할 수 있는 이상적인 장소입니다.
----------------------------------------
>> 핵심 단점 (WORST)
 - 인파: 축제는 큰 인기를 끌었으며, 많은 방문객들이 자리를 선점하기 위해 일찍 출발했습니다.
 - 주차장 폐쇄: 행사 당일에는 주차장이 폐쇄되어 있었고, 많은 방문객들이 주차장 진입을 시도했습니다.
 - 사진 촬영: 사진 촬영에 대한 정보가 부족하여, 많은 방문객들이 사진 촬영에 어려움을 겪었습니다.
----------------------------------------
>> 방문 꿀팁
 * 일찍 출발: 자리를 선점하기 위해 일찍 출발하는 것이 좋습니다.
 * 주차장 폐쇄를 미리 확인: 행사 당일에는 주차장이 폐쇄되어 있으므로, 미리 확인하는 것이 좋습니다.
 * 사진 촬영에 대한 정보를 미리 확인: 사진
==========================================
```

▢ 활동 내용

- Demo를 통해 PoC를 진행하여, 불필요한 시간을 줄이고, 프로젝트 리스크를 줄였습니다.

  -> 위의 예시는, 왼쪽과 같은 부스 신청서 PDF를 입력으로 줬을 때, **메뉴정보를 추출하는 Demo**입니다.

  -> 언뜻 보기엔 텍스트로 적혀진 정보는 잘 추출했지만, 사진과 연결시켰을때 **이해도가 떨어집니다**. (돌솥비빔밥을 돌구이 라고 인식함. 떡볶이를 파스타로 인식함)

  -> 또한 "서울세계불꽃축제" 와 같은 축제 이름을 입력했을 때 솔직한 **후기를 받아오는 Demo**도 성공적으로 동작하나, 이 경우 **법적 문제**때문에 도입하진 못했습니다.

▢ 프로젝트 회고

- ADR과 PoC를 통해 협업, 의사소통의 효율을 높일 수 있었습니다.

부동산 청약정보 제공 서비스 **Nesto**

2025.12 ~ 2025. 12 (인원 : 2명)

▢ 프로젝트 개요

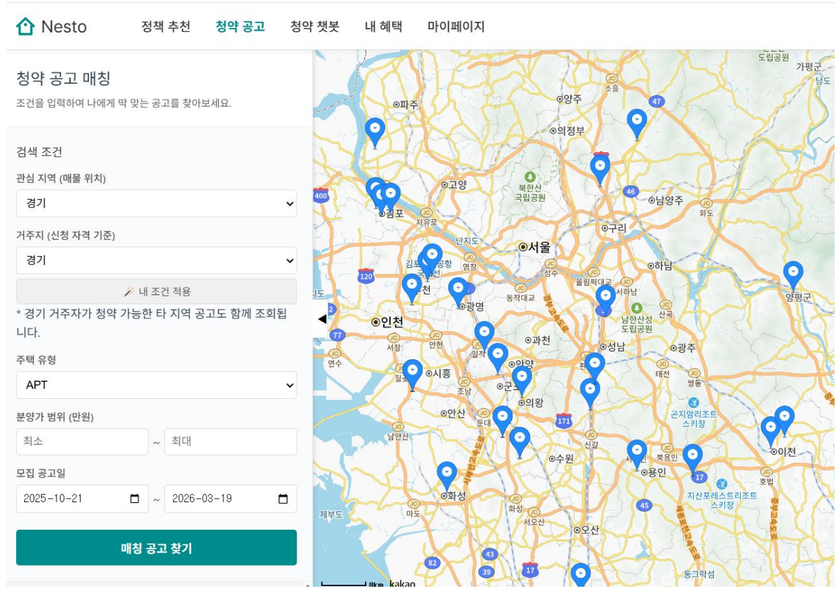
- 목표 : 청년층을 위한 부동산 청약공고 정보 제공
- 배경 : 프로젝트 중반부에 팀원이 이탈하여, 혼자 진행했습니다.
- 주요 기능 : 과거~현재의 청약 매물들을 지도에서 확인 가능, 청약공고 QnA를 학습한 RAG챗봇과의 대화 가능

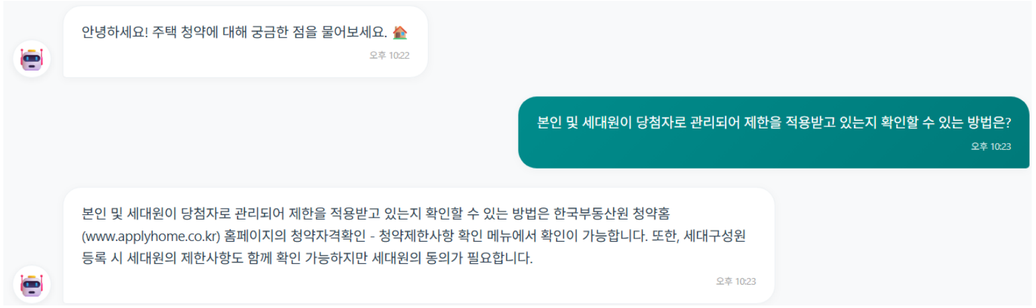
▢ 담당 역할(역할 작성)

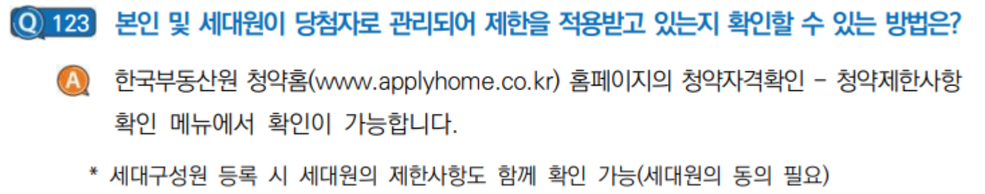
- Java Spring, Vue.js로 풀스택 개발을 진행 했습니다.
- 기획 이후 개발단계에서 팀원이 이탈하여 혼자 개발했습니다.

▢ 개인 기여도

- 청약공고 API 통합 스키마 구성 및 지도에 표시하기(100%)
  → 다양한 조건으로 필터링 후 지도에 표시 하는 기능 개발
- 청약공고 QnA 정부자료를 학습한 RAG챗봇 개발(100%)
  → 확실한 출처의 자료를 읽어와서 정확한 대답을 하는 챗봇 개발







▢ 사용 기술 / 적용 맥락 및 이유

- Java Spring AI / ChromaDB
  → 비교적 최근 출시한 Spring AI의 시험 적용에 가까웠습니다. BE, AI를 같이 개발할 수 있다면 관리가 편해질 것이라고 생각했고, 실제로도 그러했습니다.
  → 데이터 규모가 크지 않았고, FAISS, Qdrant를 쓰기보다는 쓰기 편하고, 빠르게 적용하는 것이 중요하다고 생각하여 ChromaDB를 선택했습니다.

- MyBatis
  → 청약 매물을 검색할 때 지역, 날짜, 주택 유형, 분양가 등 여러 요소로 동적 쿼리를 작성해야 했는데, JPA보다 편리했습니다.

- Vue.js
  → 백엔드 개발을 주로 해서, 진입장벽이 낮은 프레임워크를 찾다가 사용하게 되었습니다. 구조가 명확해서 컴포넌트 기반의 개발을 직관적으로 이해할 수 있었습니다.

▢ 구현 사항

[복잡한 필터링 로직으로 청약정보 지도에 띄우기]
- 지역, 주택 유형, 접수 일정, 소득 기준 등 다양한 조건을 조합해 청약 공고를 조회하고, 결과를 지도 UI와 연동했습니다.
- MyBatis Dynamic SQL의 <if>, <choose>, <foreach>를 활용해 동적 쿼리를 구성했고, 관심 등록 여부 LEFT JOIN과 가격 범위 MIN/MAX 조회를 단일 쿼리에서 함께 처리했습니다.

[청약공고 PDF의 RAG챗봇 구현]
- 청약 공고 PDF를 벡터화해 질문에 근거 기반 답변을 제공하는 RAG 챗봇을 구현했습니다.
- SpringAI와 ChromaDB로 임베딩, 검색 파이프라인을 구성했고, QnADocumentSplitter를 구현하여 검색 정확도를 높였습니다.

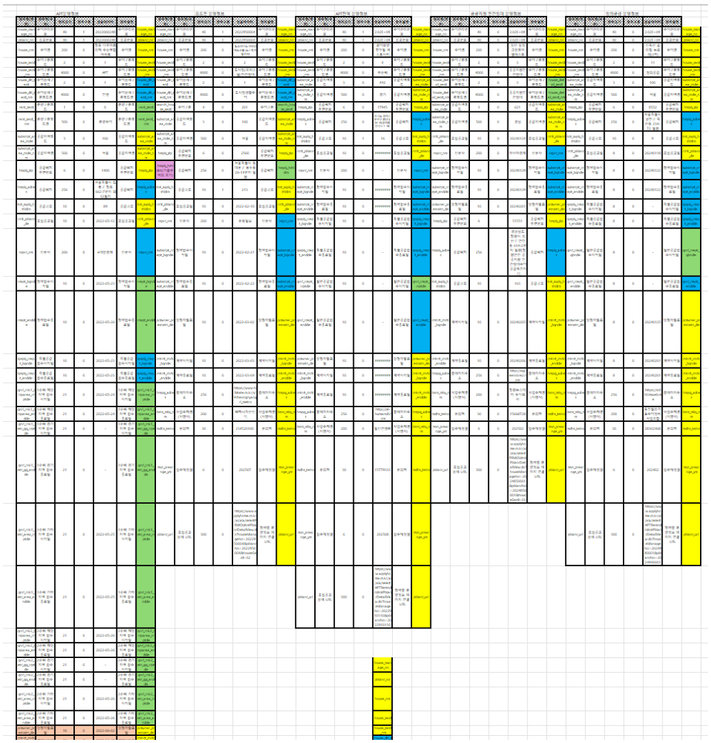
▢ 문제 해결 사례

1. 오류 분석/문제 분석

- 문제 현상 : 공공 API 10종이 주택 유형별로 분리되고, 컬럼명도 불일치했습니다.
- 해결 : 공통/부분공통/단독 컬럼 기준으로 통합 스키마 2개로 설계 했습니다.
- 이유 : 10종을 그대로 테이블로 변환한다면, 불필요한 Join연산이 발생할 수 있다고 판단했습니다. 주택 유형별 스키마를 보수적으로 합집합 개념으로 접근하여 통합했습니다.

▪ 다. 상세기능 목록

번호	API 명(국문)	상세기능명(영문)	상세기능명(국문)
1	청약홈 분양정보 조회 서비스	getAPTLttotPblancDetail	APT 분양정보 상세조회
2	청약홈 분양정보 조회 서비스	getUrbtyOfctlLttotPblancDetail	오피스텔/도시형/민간임대/생활숙박시설 분양정보 상세조회
3	청약홈 분양정보 조회 서비스	getRemndrLttotPblancDetail	APT 잔여세대 분양정보 상세조회
4	청약홈 분양정보 조회 서비스	getAPTLttotPblancMdl	APT 분양정보 주택형별 상세조회
5	청약홈 분양정보 조회 서비스	getUrbtyOfctlLttotPblancMdl	오피스텔/도시형/민간임대/생활숙박시설 분양정보 주택형별 상세조회
6	청약홈 분양정보 조회 서비스	getRemndrLttotPblancMdl	APT 잔여세대 주택형별 상세조회
7	청약홈 분양정보 조회 서비스	getPblPvtRentLttotPblancDetail	공공지원 민간임대 분양정보 상세조회
8	청약홈 분양정보 조회 서비스	getPblPvtRentLttotPblancMdl	공공지원 민간임대 분양정보 주택형별 상세조회
9	청약홈 분양정보 조회 서비스	getOPTLttotPblancDetail	임의공급 분양정보 상세조회
10	청약홈 분양정보 조회 서비스	getOPTLttotPblancMdl	임의공급 분양정보 주택형별 상세조회



노란색 칼럼
> 5개 공통으로 나타난 칼럼

파랑색 칼럼
> 2개 이상 나타난 칼럼

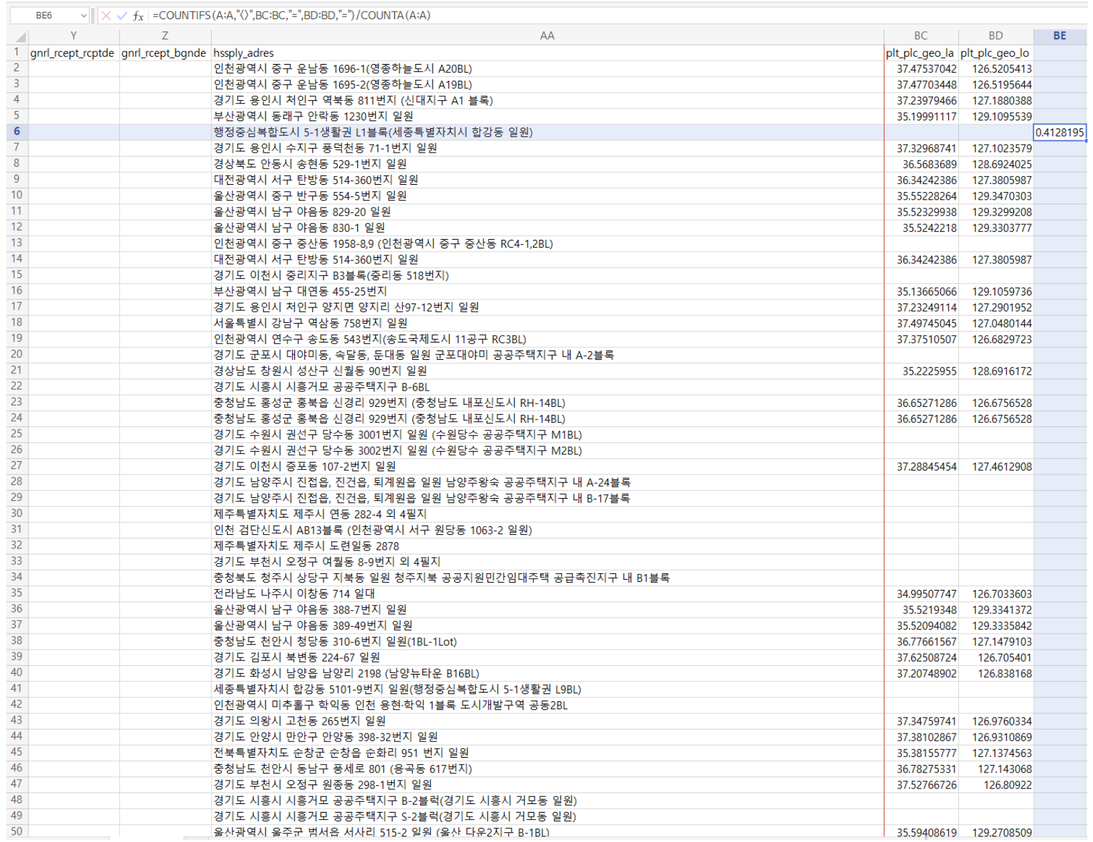
초록색 칼럼
> 한 번만 나타난 칼럼

그리하여. 오른쪽과 같은 통합 테이블이 나타났고, 나머지 5개에 대해서도 유사한 방식으로 진행하여 총 2개의 청약 표준 테이블을 도출했습니다.

▢ 문제 해결 사례

2. 오류 분석/문제 분석

- 문제 현상 : 위도/경도에 나타난 Null값이 41%에 달해 지도상에 제대로 표기할 수 없는 문제
- 원인 : 카카오맵 API를 통해 주소를 넣고 위도/경도 값을 받아오는데, "행정중심복합도시 5-1 생활권 L1블록(세종특별자치시 합강동 일원)" 과 같은 경우 NULL이 되게 됩니다.
- 해결 : 행정중심복합도시 5-1 생활권 L1블록(세종특별자치시 합강동 일원) 과 같은 주소가 있는 경우, NULL이 나오면 괄호 안쪽을 파싱하여 넣어보고, 그래도 NULL이 나온다면 주소 문자열을 N개의 블록으로 나누고, 모든 N블록, N-1개의 블록.... 순으로 넣어보며 NULL에 대한 Fallback 로직을 구현했습니다. 그 결과 NULL 값 비율을 41% -> 0.6%로 감소시켰습니다.



BE6 =COUNTIFS(A:A,"()",BC:BC,"=",BD:BD,"=")/COUNTA(A:A)

	Y	Z	AA	BC	BD	BE
1	gnrl_rcept_rcptde	gnrl_rcept_bgnde	hssply_adres	plt_plc_geo_la	plt_plc_geo_lo	
2			인천광역시 중구 운남동 1696-1(영종하늘도시 A20BL)	37.47537042	126.5205413	
3			인천광역시 중구 운남동 1695-2(영종하늘도시 A19BL)	37.47703448	126.5195644	
4			경기도 용인시 처인구 역북동 811번지 (신대지구 A1 블록)	37.23979466	127.1880388	
5			부산광역시 동래구 안락동 1230번지 일원	35.19991117	129.1095539	
6			행정중심복합도시 5-1생활권 L1블록(세종특별자치시 합강동 일원)			0.4128195
7			경기도 용인시 수지구 풍덕천동 71-1번지 일원	37.32968741	127.1023579	
8			경상북도 안동시 송현동 529-1번지 일원	36.5683689	128.6924025	
9			대전광역시 서구 탄방동 514-360번지 일원	36.34242386	127.3805987	
10			울산광역시 중구 반구동 554-5번지 일원	35.55228264	129.3470303	
11			울산광역시 남구 야음동 829-20 일원	35.52329938	129.3299208	
12			울산광역시 남구 야음동 830-1 일원	35.5242218	129.3303777	
13			인천광역시 중구 중산동 1958-8,9 (인천광역시 중구 중산동 RC4-1,2BL)			
14			대전광역시 서구 탄방동 514-360번지 일원	36.34242386	127.3805987	
15			경기도 이천시 중리지구 B3블록(중리동 518번지)			
16			부산광역시 남구 대연동 455-25번지	35.13665066	129.1059736	
17			경기도 용인시 처인구 양지면 양지리 산97-12번지 일원	37.23249114	127.2901952	
18			서울특별시 강남구 역삼동 758번지 일원	37.49745045	127.0480144	
19			인천광역시 연수구 송도동 543번지(송도국제도시 11공구 RC3BL)	37.37510507	126.6829723	
20			경기도 군포시 대야미동, 속달동, 둔대동 일원 군포대야미 공공주택지구 내 A-2블록			
21			경상남도 창원시 성산구 신월동 90번지 일원	35.2225955	128.6916172	
22			경기도 시흥시 시흥거모 공공주택지구 B-6BL			
23			충청남도 홍성군 홍북읍 신경리 929번지 (충청남도 내포신도시 RH-14BL)	36.65271286	126.6756528	
24			충청남도 홍성군 홍북읍 신경리 929번지 (충청남도 내포신도시 RH-14BL)	36.65271286	126.6756528	
25			경기도 수원시 권선구 당수동 3001번지 일원 (수원당수 공공주택지구 M1BL)			
26			경기도 수원시 권선구 당수동 3002번지 일원 (수원당수 공공주택지구 M2BL)			
27			경기도 이천시 중포동 107-2번지 일원	37.28845454	127.4612908	
28			경기도 남양주시 진접읍, 진건읍, 퇴계원읍 일원 남양주왕숙 공공주택지구 내 A-24블록			
29			경기도 남양주시 진접읍, 진건읍, 퇴계원읍 일원 남양주왕숙 공공주택지구 내 B-17블록			
30			제주특별자치도 제주시 연동 282-4 외 4필지			
31			인천 검단신도시 AB13블록 (인천광역시 서구 원당동 1063-2 일원)			
32			제주특별자치도 제주시 도련일동 2878			
33			경기도 부천시 오정구 여월동 8-9번지 외 4필지			
34			충청북도 청주시 상당구 지북동 일원 청주지북 공공지원민간임대주택 공급촉진지구 내 B1블록			
35			전라남도 나주시 이창동 714 일대	34.99507747	126.7033603	
36			울산광역시 남구 야음동 388-7번지 일원	35.5219348	129.3341372	
37			울산광역시 남구 야음동 389-49번지 일원	35.52094082	129.3335842	
38			충청남도 천안시 청당동 310-6번지 일원(1BL-1Lot)	36.77661567	127.1479103	
39			경기도 김포시 북변동 224-67 일원	37.62508724	126.705401	
40			경기도 화성시 남양읍 남양리 2198 (남양뉴타운 B16BL)	37.20748902	126.838168	
41			세종특별자치시 합강동 5101-9번지 일원(행정중심복합도시 5-1생활권 L9BL)			
42			인천광역시 미추홀구 학익동 인천 융현·학익 1블록 도시개발구역 공동2BL			
43			경기도 의왕시 고천동 265번지 일원	37.34759741	126.9760334	
44			경기도 안양시 만안구 안양동 398-32번지 일원	37.38102867	126.9310869	
45			전북특별자치도 순창군 순창읍 순화리 951 번지 일원	35.38155777	127.1374563	
46			충청남도 천안시 동남구 풍세로 801 (용곡동 617번지)	36.78275331	127.143068	
47			경기도 부천시 오정구 원종동 298-1번지 일원	37.52766726	126.80922	
48			경기도 시흥시 시흥거모 공공주택지구 B-2블럭(경기도 시흥시 거모동 일원)			
49			경기도 시흥시 시흥거모 공공주택지구 S-2블럭(경기도 시흥시 거모동 일원)			
50			울산광역시 울주군 범서읍 서사리 515-2 일원 (울산 다운2지구 B-1BL)	35.59408619	129.2708509	

AA	BC	BD	BE	BF
hssply_adres	plt_plc_geo_la	plt_plc_geo_lo		
인천광역시 중구 운남동 1696-1(영종하늘도시 A20BL)	37.47537042	126.5205413		
인천광역시 중구 운남동 1695-2(영종하늘도시 A19BL)	37.47703448	126.5195644		
경기도 용인시 처인구 역북동 811번지 (신대지구 A1 블록)	37.23979466	127.1880388		
부산광역시 동래구 안락동 1230번지 일원	35.19991117	129.1095539		
행정중심복합도시 5-1생활권 L1블록(세종특별자치시 합강동 일원)	36.52512762	127.3303042		
경기도 용인시 수지구 풍덕천동 71-1번지 일원	37.32968741	127.1023579		
경상북도 안동시 송현동 529-1번지 일원	36.5683689	128.6924025		0.00619914762
대전광역시 서구 탄방동 514-360번지 일원	36.34242386	127.3805987		
울산광역시 중구 반구동 554-5번지 일원	35.55228264	129.3470303		
울산광역시 남구 야음동 829-20 일원	35.52329938	129.3299208		
울산광역시 남구 야음동 830-1 일원	35.5242218	129.3303777		
인천광역시 중구 중산동 1958-8,9 (인천광역시 중구 중산동 RC4-1,2BL)	37.49183628	126.5710648		
대전광역시 서구 탄방동 514-360번지 일원	36.34242386	127.3805987		
경기도 이천시 중리지구 B3블록(중리동 518번지)	37.27226759	127.4350893		
부산광역시 남구 대연동 455-25번지	35.13665066	129.1059736		
경기도 용인시 처인구 양지면 양지리 산97-12번지 일원	37.23249114	127.2901952		
서울특별시 강남구 역삼동 758번지 일원	37.49745045	127.0480144		
인천광역시 연수구 송도동 543번지(송도국제도시 11공구 RC3BL)	37.37510507	126.6829723		
경기도 군포시 대야미동, 속달동, 둔대동 일원 군포대야미 공공주택지구 내 A-2블록	37.32912671	126.9184082		
경상남도 창원시 성산구 신월동 90번지 일원	35.2225955	128.6916172		
경기도 시흥시 시흥거모 공공주택지구 B-6BL	37.35670182	126.7741749		
충청남도 홍성군 홍북읍 신경리 929번지 (충청남도 내포신도시 RH-14BL)	36.65271286	126.6756528		
충청남도 홍성군 홍북읍 신경리 929번지 (충청남도 내포신도시 RH-14BL)	36.65271286	126.6756528		
경기도 수원시 권선구 당수동 3001번지 일원 (수원당수 공공주택지구 M1BL)	37.2887193	126.940217		
경기도 수원시 권선구 당수동 3002번지 일원 (수원당수 공공주택지구 M2BL)	37.2887193	126.940217		
경기도 이천시 중포동 107-2번지 일원	37.28845454	127.4612908		
경기도 남양주시 진접읍, 진건읍, 퇴계원읍 일원 남양주왕숙 공공주택지구 내 A-24블록	37.72630049	127.1898865		
경기도 남양주시 진접읍, 진건읍, 퇴계원읍 일원 남양주왕숙 공공주택지구 내 B-17블록	37.72630049	127.1898865		
제주특별자치도 제주시 연동 282-4 외 4필지	33.48941262	126.4923648		
인천 검단신도시 AB13블록 (인천광역시 서구 원당동 1063-2 일원)	37.59394103	126.7215029		
제주특별자치도 제주시 도련일동 2878	33.50560809	126.5892888		
경기도 부천시 오정구 여월동 8-9번지 외 4필지	37.5183234	126.8015516		
충청북도 청주시 상당구 지북동 일원 청주지북 공공지원민간임대주택 공급촉진지구 내 B1블록	36.59496555	127.5087195		
전라남도 나주시 이창동 714 일대	34.99507747	126.7033603		
울산광역시 남구 야음동 388-7번지 일원	35.5219348	129.3341372		
울산광역시 남구 야음동 389-49번지 일원	35.52094082	129.3335842		
충청남도 천안시 청당동 310-6번지 일원(1BL-1Lot)	36.77661567	127.1479103		
경기도 김포시 북변동 224-67 일원	37.62508724	126.705401		
경기도 화성시 남양읍 남양리 2198 (남양뉴타운 B16BL)	37.20748902	126.838168		
세종특별자치시 합강동 5101-9번지 일원(행정중심복합도시 5-1생활권 L9BL)	36.52512762	127.3303042		
인천광역시 미추홀구 학익동 인천 융현·학익 1블록 도시개발구역 공동2BL	37.44101799	126.6543808		
경기도 의왕시 고천동 265번지 일원	37.34759741	126.9760334		
경기도 안양시 만안구 안양동 398-32번지 일원	37.38102867	126.9310869		
전북특별자치도 순창군 순창읍 순화리 951 번지 일원	35.38155777	127.1374563		
충청남도 천안시 동남구 풍세로 801 (용곡동 617번지)	36.78275331	127.143068		
경기도 부천시 오정구 원종동 298-1번지 일원	37.52766726	126.80922		
경기도 시흥시 시흥거모 공공주택지구 B-2블럭(경기도 시흥시 거모동 일원)	37.35670182	126.7741749		
경기도 시흥시 시흥거모 공공주택지구 S-2블럭(경기도 시흥시 거모동 일원)	37.35670182	126.7741749		
울산광역시 울주군 범서읍 서사리 515-2 일원 (울산 다운2지구 B-1BL)	35.59408619	129.2708509		

▢ 문제 해결 사례

3. 오류 분석/문제 분석

- 문제 현상 : 청약공고 PDF기반 RAG챗봇에서 질문 의도와 맞지 않는 답변이 발생
- 원인 : 기본 길이 기반 splitter를 사용 시 질문과 답변이 서로 다른 청크로 분리되어 문맥이 끊김
- 한계 : 페이지 단위 또는 고정 길이 분할만으로는 QnA구조와 섹션 문맥을 보존하기 어려웠음

개선 과정

- QnADocumentSplitter를 직접 구현하여 PDF를 페이지별 Document로 읽고 줄 단위로 순회
- 질문 시작 패턴과 섹션 헤더를 정규식으로 탐지하여 질문~답변 묶음을 하나의 청크로 유지
- 청크 생성 시 기존 페이지 메타데이터에 section, category=Q&A를 추가하여 검색 시 문맥 정보를 함께 활용하도록 설계
- 긴 답변은 4000자 기준으로 보조 분할하여 과도한 청크 생성을 방지

```
// 질문 시작 패턴:
// "Q1. 질문?" 또는 "1 질문?"처럼 번호와 물음표로 끝나는 질문 라인을 감지한다.
// 질문의 시작점을 기준으로 Q&A 단위 청크를 구성하기 위해 사용한다.
private static final Pattern QUESTION_PATTERN = Pattern.compile(
        regex: "^\\s*(?:Q\\s*)?(\\d+)[.\\s]+(.*\\?)\\s*$",
        Pattern.MULTILINE
);

// 섹션 헤더 패턴:
// "I. 제목", "1. 제목"처럼 문서의 대분류/소분류 제목을 감지한다.
// 청크 분리 기준으로 바로 사용하지 않고, 현재 문맥(section) 추적용으로 사용한다.
private static final Pattern SECTION_HEADER_PATTERN = Pattern.compile(
        regex: "^\\s*([IVX]+|\\d+)\\.\\s+(.*)$",
        Pattern.MULTILINE
);
```

```
@Override
public List<Document> apply(List<Document> documents) {
    List<Document> splitDocuments = new ArrayList<>();

    String currentSection = "General";   // 현재 섹션 문맥
    StringBuilder qnaBuffer = new StringBuilder(); // 질문 + 답변을 누적할 버퍼
    Map<String, Object> currentMetadata = null;

    // 페이지 전체를 먼저 합칠 수도 있지만,
    // 그 경우 질문이 시작된 페이지 번호 같은 메타데이터가 흐려질 수 있다.
    // 따라서 페이지를 순회하면서 섹션 상태를 유지하고,
    // 질문이 시작되는 시점을 기준으로 Q&A 버퍼를 관리한다.
    for (Document doc : documents) {
        String content = doc.getContent();
        String[] lines = content.split(regex: "\\r?\\n");

        for (String line : lines) {
            // 1. 섹션 헤더 감지: 현재 문맥만 갱신한다.
            Matcher sectionMatcher = SECTION_HEADER_PATTERN.matcher(line);
            if (sectionMatcher.find()) {
                currentSection = sectionMatcher.group(group: 0).trim();
                logger.debug("섹션 헤더 감지: {}", currentSection);

                // 섹션 헤더는 즉시 청크를 끊는 기준이 아니라,
                // 이후 생성될 Q&A 청크에 붙일 문맥 정보로만 활용한다.
            }

            // 2. 질문 시작 라인 감지
            Matcher qMatcher = QUESTION_PATTERN.matcher(line);
            if (qMatcher.find()) {
                // 새 질문이 시작되면, 이전까지 누적한 Q&A를 하나의 청크로 저장한다.
                if (qnaBuffer.length() > 0) {
                    splitDocuments.add(createDocument(qnaBuffer.toString(), currentSection, currentMetadata));
                    qnaBuffer.setLength(newLength: 0);
                }

                // 새 질문으로 버퍼를 시작한다.
                qnaBuffer.append(line).append(str: "\n");
```

분할 예시

원문
I. 청약통장
1. 청약통장이 꼭 있어야 하나요?
청약 유형에 따라 다릅니다.
일반공급에서는 필요한 경우가 많습니다.

- 기본 splitter
[1. 청약통장이 꼭 있어야 하나요?]
[청약 유형에 따라 다릅니다. 일반공급에서는...]

- QnADocumentSplitter
[section=I. 청약통장, category=Q&A]
1. 청약통장이 꼭 있어야 하나요?
청약 유형에 따라 다릅니다.
일반공급에서는 필요한 경우가 많습니다.

FAQ의 질의응답에 해당하는 내용을 찾아 정확히 대답하는 챗봇

▢ 프로젝트 성과 및 결과

- SpringAI를 사용해서 프로젝트를 처음으로 진행했는데, 최근에 나와서 활발하게 발전하는 프로젝트라서 AI가 작업할 경우 이전 스냅샷 버전을 사용한다던지. 의존성 문제가 많았습니다.
- RAG를 활용하여 내부 문서를 활용하고자 하는 기업들이 많은데, 이번 시도를 통해 RAG가 단순히 적당히 토큰으로 문서를 쪼개고 VectorStore에 집어넣는게 아니라 굉장히 다양한 변주가 가능하다는 것을 알게 되었습니다.

▢ 프로젝트 회고

- 중간에 팀원이 이탈하여 혼자 진행하게 되었고, FE연동, BE작업, AI 작업 모두 맡다보니 개별 완성도가 떨어지는 점이 있습니다.
- 만약 여기서 더 발전시킨다면, RAG챗봇의 정확도를 더 다양한 방법론을 사용하여 향상시킬 것 같습니다.(TOK-K의 매직넘버 찾기, 또 다른 Split방법 찾기 등)

구독형 보험 서비스 CapSure

2026.03 ~ 2026. 03 (인원 : 6 명)

▢ 프로젝트 개요

- 목표 : 최근 보험업계의 화두에 오른 “구독형 보험” 구현하기
- 배경 : 기능의 동작 여부만큼 로깅, 인증도 중요하다고 판단하여 진행
- 주요 기능 : 마이데이터 API 동작 방식과 유사한 서버 구성
  보험을 구독하고 해제하는 과정에서의 인증처리

▢ 담당 역할(역할 작성)

- 생명보험, 손해보험 데이터를 가져와서 통합 스키마 설계
- 마이데이터 가이드라인에 따라 최대한 유사하게 진행하기 위한 Mock서버 구성과 그 과정에서의 인증설계

▢ 개인 기여도

•Main–Mock 서버 분리 기반 마이데이터 조회 흐름 구현 : 80%
→ 전송요구 저장, access token 발급, Bearer token 기반 계약/보장 조회 API 구성

•보험 계약/보장 데이터 통합 스키마 설계 : 80%
→ 생명보험/손해보험 응답 구조를 공통 스키마로 정리하고 메인 조회 API와 연동

•가입·변경 과정의 감사 로그 아키텍처 설계 : 60%
→ Transactional Outbox, DLQ, idempotency 기반 비동기 감사 로그 적재 구조 설계

▢ 사용 기술 / 적용 맥락 및 이유

- PostgreSQL

→ 보험 데이터, 사용자 개인/계약/보장정보 등 데이터간 관계성이 명확했고
보험 약관 요약 원문 등을 유동성있게 JSON으로 관리하기위해 사용했습니다.

- Spring Security + JWT

→ 서비스를 이용하는 일반 사용자 로그인, MyData조회용 access Token 등 인증로직
구현에 사용했습니다.

- Transactional Outbox

→ 보험 가입·변경 과정의 감사 로그 적재 실패가 핵심 비즈니스 요청 실패로
전이되지 않도록, 비즈니스 트랜잭션과 감사 로그 적재를 분리하기 위해 적용

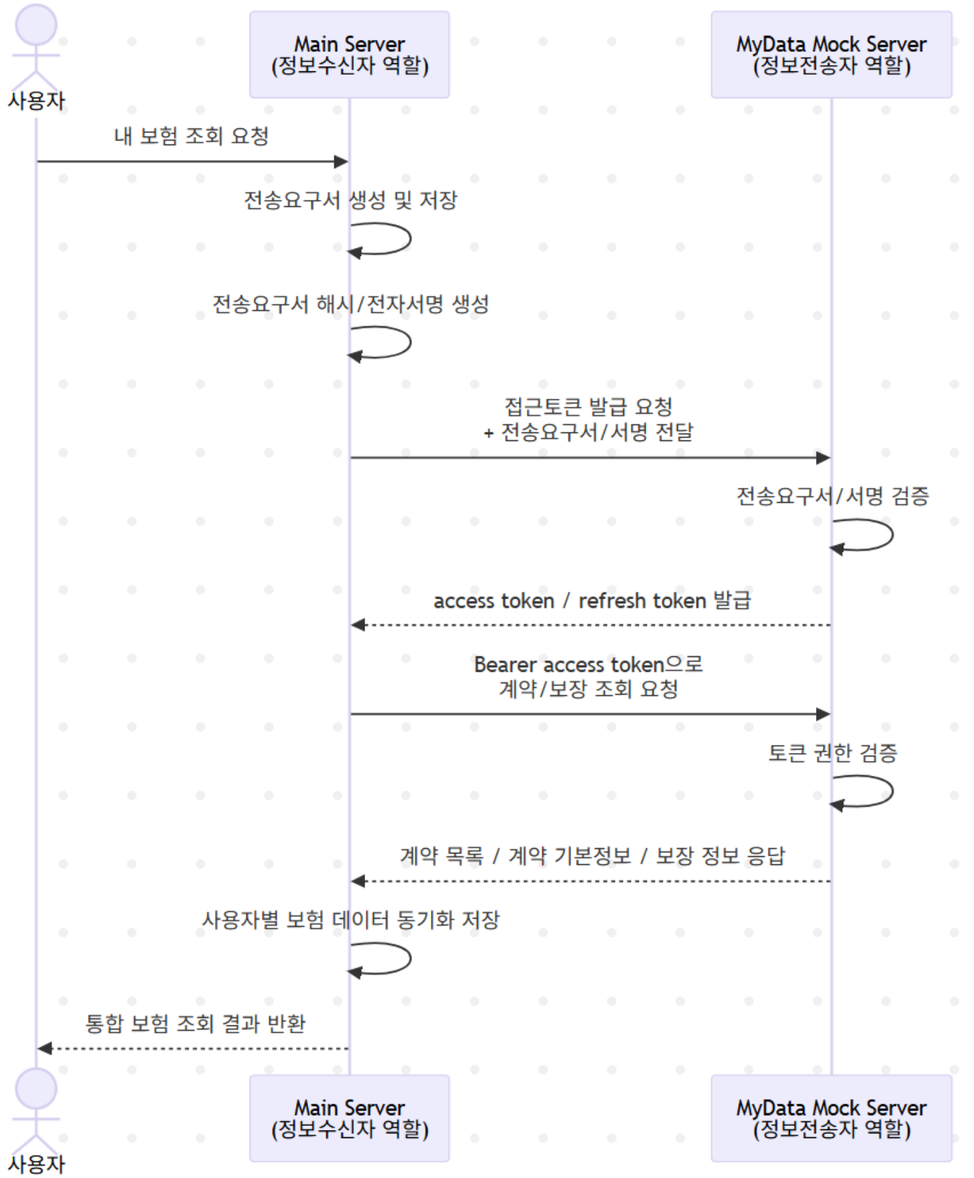
▢ 구현 사항 - 1

[Main–Mock 서버 분리 기반 MyData 조회 흐름 구현]

- 메인 서버를 정보수신자 역할, Mock 서버를 보험사(정보제공자) 역할로 분리
- 전송요구서 생성·저장 → access token 발급 → Bearer token 기반 계약/보장 조회 흐름 구현
- 실제 사업자 연동이 불가능한 상황에서 마이데이터 가이드라인 절차를 참고해 조회 과정을 최대한 유사하게 재현

[보험 계약/보장 데이터 통합 스키마 설계]

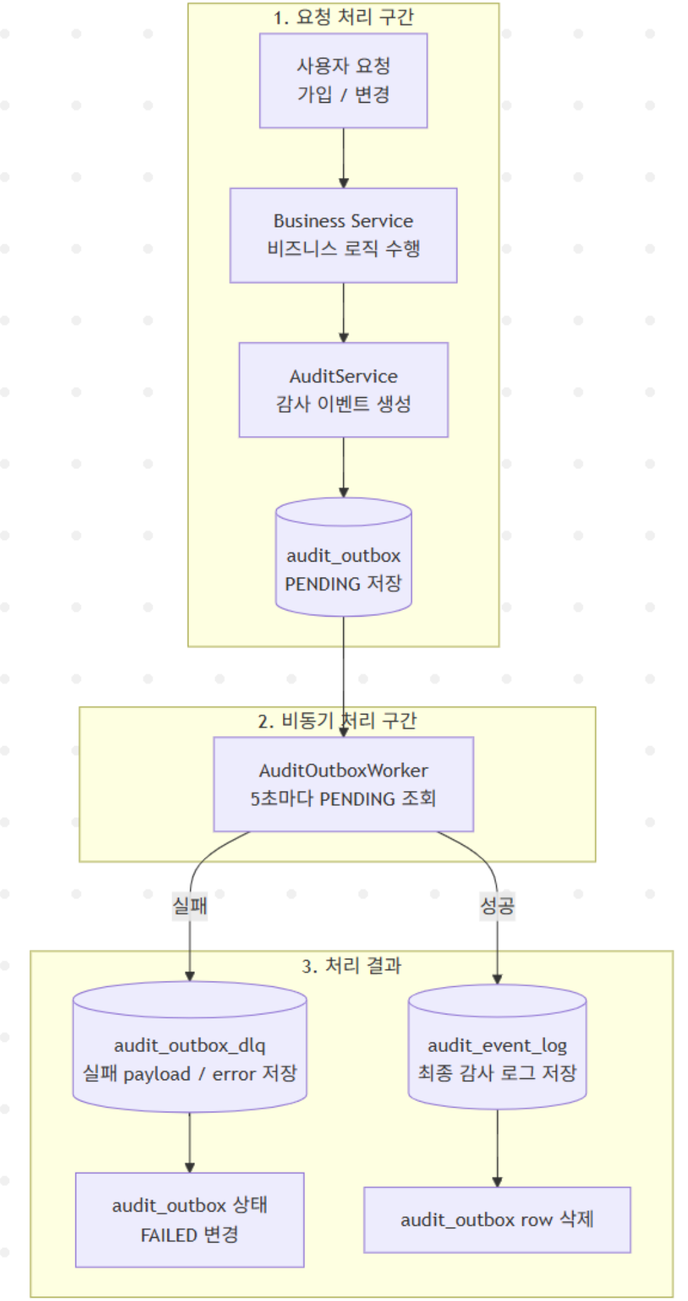
- 보험사 Mock 응답 데이터를 메인 서버 기준의 공통 스키마로 저장하도록 설계
- 사용자별 계약 정보와 상품성 데이터의 저장 경계를 구분하여 조회 API와 연동
- 계약 목록, 기본 정보, 보장 정보가 하나의 사용자 보험 조회 흐름으로 이어지도록 구성



▢ 구현 사항 - 2

[가입·변경 과정의 감사 로그 아키텍처 설계]

- 감사 로그를 요청 경로에서 바로 적재하지 않고 outbox에 먼저 저장한 뒤 별도 워커가 최종 로그로 이관
- 실패 이벤트는 DLQ에 분리 저장하고, idempotency 기준으로 중복 적재를 방지
- 정합성, 추적 가능성, 장애 격리를 고려한 구조로 설계



▢ 문제 해결 사례

1. 오류 분석/문제 분석

- 문제 현상 : 보험 가입·변경 요청에서 감사 로그를 **동기 적재할 경우**, 로그 저장 실패가 핵심 비즈니스 요청 실패로 **전이될 수 있었습니다.**
- 원인 : 비즈니스 처리와 감사 로그 적재가 같은 요청 경로에 결합되어 있어 실패를 독립적으로 격리하거나 재처리하기 어려웠습니다.
- 추가 확인 사항 : 보험/금융 서비스에서는 비즈니스 성공 여부와 별개로 감사 이력이 남아야 하므로, 로그 실패를 별도 관리하는 구조가 필요했습니다.

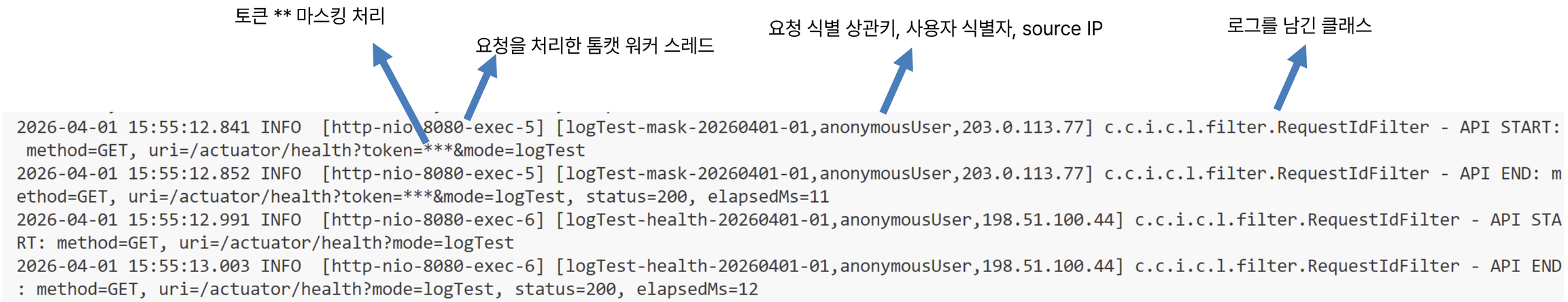
2. 개선 과정

- 감사 이벤트를 먼저 audit_outbox에 PENDING 상태로 저장
- AuditOutboxWorker가 5초마다 PENDING 이벤트를 조회해 최종 감사 로그로 이관
- 실패 이벤트는 audit_outbox_dlq에 payload와 오류 정보를 저장하고 상태를 FAILED로 변경
- 중복 적재는 idempotency 기준으로 방지.

항목	Before	After
적재 방식	요청 처리 경로에서 감사 로그를 바로 최종 테이블에 저장	outbox에 먼저 저장한 뒤 worker가 비동기 이관
장애 영향	로그 적재 실패가 비즈니스 요청 실패로 전이될 수 있음	비즈니스 처리와 로그 적재를 분리해 장애 전파를 완화
실패 처리	실패 원인 추적 및 후속 재처리가 어려움	DLQ 저장으로 실패 payload와 오류 사유를 별도 관리
중복 대응	재시도 시 중복 적재 가능성 존재	idempotency 기준으로 중복 적재를 방지
추적성	요청과 감사 로그를 연결해 추적하기 어려움	requestId 기반으로 요청 단위 추적 가능

3. 해결 과정

1) 문제 진단 : 감사 로그의 중요성과 비즈니스 요청의 안정성을 동시에 만족해야 하는 구조적 문제를 확인
2) 전략 수립 : 비즈니스 처리와 로그 적재를 분리하고, 실패 이벤트를 격리 저장하는 방향으로 구조를 재설계
3) 기술 적용 : DB Outbox, Worker polling, DLQ, idempotency, MDC 기반 requestId 추적 적용
4) 검증 : 로그 저장 실패가 핵심 비즈니스 요청에 직접 전이되지 않도록 분리하고, 실패 원인 분석과 후속 재처리 가능성을 확보



□ 프로젝트 성과 및 결과

- 마이데이터 가이드라인을 참고해 Main–Mock 서버 분리 기반 보험 조회 흐름을 구현하고, 전송요구·토큰 발급·보험 조회 절차를 프로젝트 구조 안에서 재현했습니다.
- DB Outbox + Worker + DLQ 기반 감사 로그 구조를 설계해 비즈니스 안정성과 추적 가능성을 함께 고려한 처리 흐름을 경험했습니다.
- 금융 서비스에서는 단순 기능 구현보다 인증 절차, 데이터 정합성, 감사 이력, 장애 격리를 함께 설계하는 것이 중요하다는 점을 배웠습니다.

□ 프로젝트 회고

- 실제 정보제공기관과의 연동이 불가능해 일부 절차는 가이드라인 기반 Mock 구조로 대체했다는 한계가 있었습니다.
- 다만 제약 조건 속에서도 절차를 단순 흉내 내는 데 그치지 않고, 왜 이 인증 흐름과 저장 구조가 필요한지 이해하고 설계해 본 경험이 의미 있었습니다.
- 로그를 남기는 것 이후, 모니터링 전략으로 연결되지 못해 아쉬웠습니다.
- 앞으로는 Grafana, Prometheus 등 운영 모니터링, 재처리 전략, 다중 인스턴스 환경까지 포함한 구조로 더 발전시켜 보고 싶습니다.

Java OSS Armeria 기여

2025.08 ~ 2026. 03 (인원 : 1명)

▢ 기여 개요

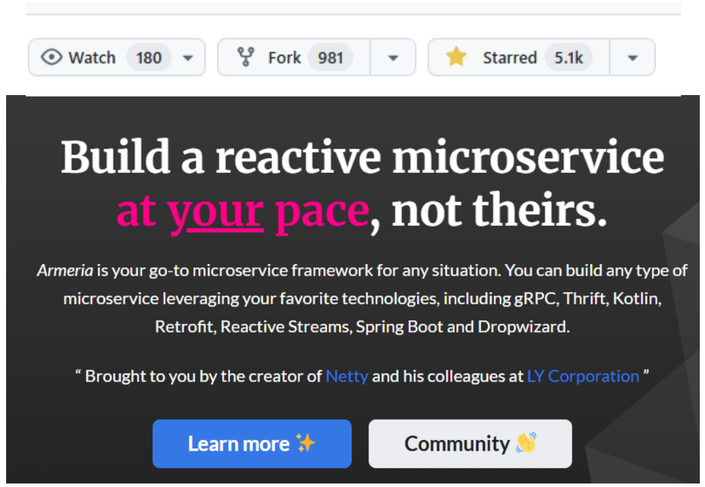
- 대상 : LINE/Armeria 오픈소스 (Start 5000++)
- 주제 : 자체 API문서화 기능인 DocService의 inheritance / polymorphism 문서화 지원
- 기여: Jackson annotation 기반 polymorphism 지원 구현
- 현재 상태 : Merge되어 Contributor가 되었음.

▢ 담당 역할

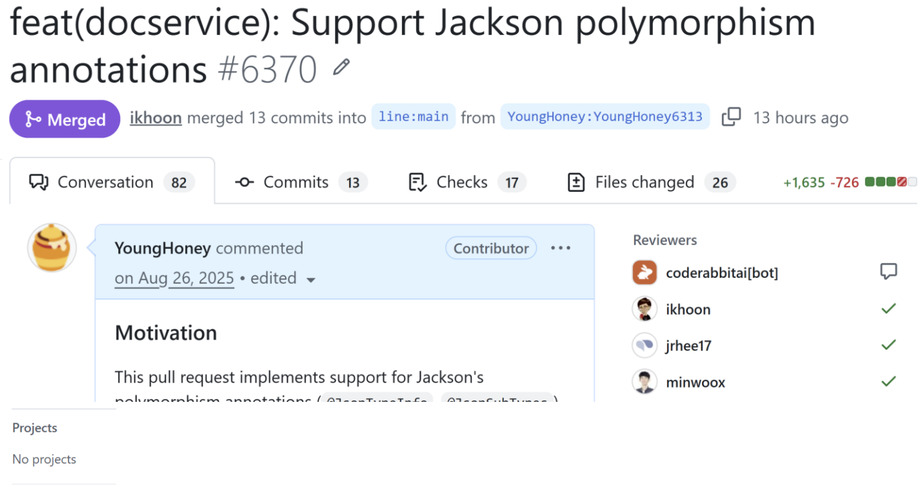
- 이슈 분석 및 구현 방향 제안
- DocService 내부 구조 추적 및 설계 판단
- Provider 구현 및 SPI 등록
- 공통 타입 변환 로직 리팩토링
- PR/Discussion 작성, maintainer 피드백 반영
- gradle parallel build 실패 재현 및 로그 정리 후 보고

▢ 개인 기여도

- Jackson기반 다형성, 상속 기능 및 테스트 추가, Schema 구조 개편
- 리뷰 반영 및 빌드 문제 조사
- 문서 렌더링은 Reviewer의 도움을 받음



Github Issue



제출한 PR

▢ 사용 기술 / 적용 맥락 및 이유

- Jackson Annotation

→ 다형성 문서화의 근거가 필요했고, Maintainer에게 문의한 결과
Jackson Annotation 방식을 선호한다고 응답받음

Support for inheritance and polymorphism in DocService #6313

Open Feature

icepeppermint opened on Jul 17, 2025 Contributor

I'm using `DocService` to document our API. Currently, the response classes of the annotated HTTP service use inheritance and polymorphism. There doesn't seem to be any support for this case at the moment. It would be great if Armeria automatically detect annotations like `@JsonSubTypes`. Or, it's good to create a new annotation to specify it manually. Below is an example of a Discriminator in Swagger.

- https://swagger.io/docs/specification/v3_0/data-models/inheritance-and-polymorphism/

YoungHoney on Aug 10, 2025

Hi! I'm interested in working on this feature and would like to confirm the preferred direction.

For `DocService`, I see two possible approaches:

1. Generate polymorphism strictly from **Jackson annotations** (`@JsonTypeInfo`, `@JsonSubTypes`, `@JsonTypeName`)
2. Introduce **Armeria-specific annotations** as an explicit override or extension point.

I'd like to hear advice or the direction the Armeria team would prefer.
If nobody is currently working on this, I'd be happy to take it on.

ikhoon on Aug 11, 2025 Contributor

I prefer using Jackson annotations rather than introducing a new annotation for it.

👍 1

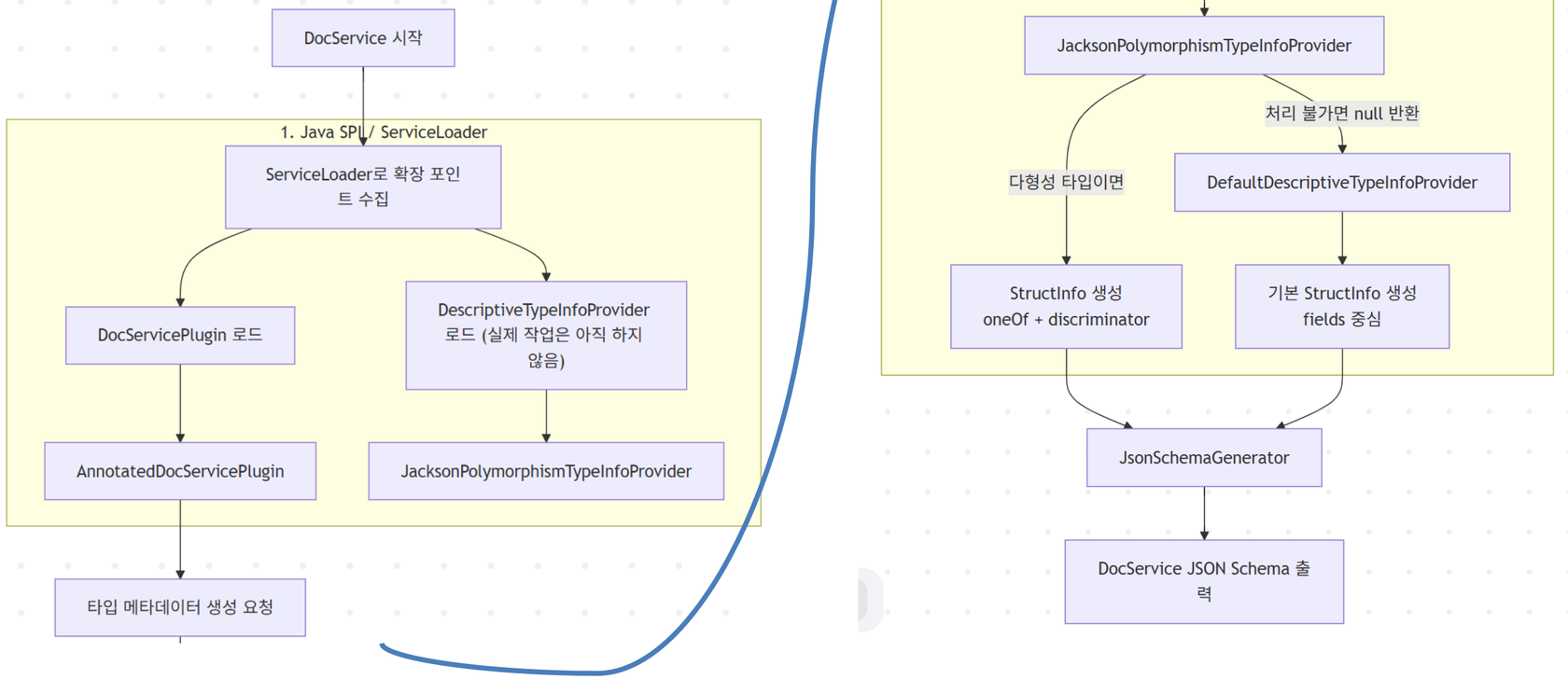
YoungHoney mentioned this on Aug 26, 2025

feat(docservice): Support Jackson polymorphism annotations #6370

▢ 사용 기술 / 적용 맥락 및 이유

- Java SPI / ServiceLoader

→ 문서화를 담당하는 DocService가 Provider들을 런타임에 조합하는 구조임을 확인. 기존 Default구현체를 변경하지 않고 Provider를 구현하는 것이 설계 사상에 부합한다고 생각함.



- Provider Chain / Chain of Responsibility

→ DocService가 Provider를 런타임에 조합할 때, 다형성 타입일 때만 Provider로 동작시키고 이외엔 Default Provider로 넘기기 위함

▢ 구체적 작업 내용

- 새로운 Provider – JacksonPolymorphismTypeInfoProvider추가

```java
public final class JacksonPolymorphismTypeInfoProvider implements DescriptiveTypeInfoProvider {
    @Override
    @Nullable
    public DescriptiveTypeInfo newDescriptiveTypeInfo(Object typeDescriptor) {
        requireNonNull(typeDescriptor, message: "typeDescriptor");
        if (!(typeDescriptor instanceof Class)) {
            return null;
        }

        final Class<?> clazz = (Class<?>) typeDescriptor;
        final JsonTypeInfo jsonTypeInfo = clazz.getAnnotation(annotationClass: JsonTypeInfo.class);
        final JsonSubTypes jsonSubTypes = clazz.getAnnotation(annotationClass: JsonSubTypes.class);

        if (jsonTypeInfo == null || jsonSubTypes == null) {
            return null;
        }

        String propertyName = jsonTypeInfo.property();
        if (propertyName.isEmpty()) {
            final JsonTypeInfo.Id use = jsonTypeInfo.use();
            if (use == JsonTypeInfo.Id.CLASS) {
                propertyName = "@class";
            } else if (use == JsonTypeInfo.Id.MINIMAL_CLASS) {
                propertyName = "@c";
            } else if (use == JsonTypeInfo.Id.NAME || use == JsonTypeInfo.Id.SIMPLE_NAME) {
                propertyName = "@type";
            } else {
                return null;
            }
        }

        if (jsonSubTypes.value().length == 0) {
            return null;
        }
```

"클래스 타입 " 만 처리
Class<?>가 아니면 처리 대상이 아니므로 Null 반환

Jackson다형성 관련 Annotation조회

둘 중 하나라도 없으면 Jackson 타입이 아니므로 null 반환
후, 다음 Provider로 넘기기

Ex) @JsonTypeInfo(property="species") 추출

Property가 비어있으면 Fallback 이름을 정함

문서화 할 하위 타입이 없으면 다형성 정보 생성 불가

```java
@JsonTypeInfo(use = JsonTypeInfo.Id.NAME, property = "species")
@JsonSubTypes({
        @JsonSubTypes.Type(value = Dog.class, name = "dog"),
        @JsonSubTypes.Type(value = Cat.class, name = "cat")
})
interface Animal {
    String name();
}
```

예시 구조

▢ 구체적 작업 내용

- 새로운 Provider – JacksonPolymorphismTypeInfoProvider추가

```
final Map<String, String> mapping = new LinkedHashMap<>();
Arrays.stream(jsonSubTypes.value()).forEach(subType -> {
    final Class<?> subClass = subType.value();
    final String key = isNullOrEmpty(subType.name()) ? subClass.getSimpleName() : subType.name();
    final String schemaName = TypeSignature.ofStruct(subClass).name();
    mapping.put(key, "#/$defs/models/" + schemaName);
});

final DiscriminatorInfo discriminator = DiscriminatorInfo.of(propertyName, mapping);

final List<TypeSignature> oneOf = Arrays.stream(jsonSubTypes.value())
                                        .map(subType -> TypeSignature.ofStruct(subType.value()))
                                        .collect(toImmutableList());

final JavaType javaType = mapper.constructType(clazz);
final BeanDescription description = mapper.getSerializationConfig().introspect(javaType);
final List<BeanPropertyDefinition> properties = description.findProperties();

final List<FieldInfo> fields = properties.stream()
                                         .map(prop -> FieldInfo.of(prop.getName(),
                                                                   toTypeSignature(
                                                                           prop.getPrimaryType())))
                                         .collect(toImmutableList());

final Description classDescription = clazz.getAnnotation(annotationClass: Description.class);

final DescriptionInfo descriptionInfo = classDescription == null ? DescriptionInfo.empty()
                                                                 : DescriptionInfo.from(
        classDescription);

return new StructInfo(clazz.getName(), alias: null, fields, descriptionInfo, oneOf, discriminator);
```

@JsonSubTypes에 선언된 각 하위타입을 순회하며

Dog 객체 -> “#/defs/models/com.example.dog 와 같이 discriminator의 값을 실제 schema 경로로 변환

앞에서 계산한 discriminator property 이름과 subtype Mapping을 묶어 DiscriminatorInfo라는 내부 객체로 생성

one of의 목록 생성, 이 타입은 실제로 어떤 하위 타입 중 하나인가를 나타냄. 부모가 Animal, 자식이 Dog, Cat이라면, oneOf=[Dog,Cat]

부모타입 자체의 필드 정보도 추출

최종적으로 부모타입을 StructInfo라는 내부 객체로 생성
1. 부모 타입 이름
2. 부모타입의 공통 필드 목록
3. 설명
4. oneOf(하위 타입 목록)
5. Discriminaor (어떤 필드로 타입을 구분하는지)

가 모두 들어감

▢ 사용 기술 / 적용 맥락 및 이유

- JUnit / 통합 테스트

→ 앞선 과정에서 추출한 Schema.json 생성 확인만으로는 부족하다고 판단하여
테스트 코드를 통해 예시 객체들로 테스트 진행

```
// --- DTOs and Service for the test ---

@JsonTypeInfo(use = JsonTypeInfo.Id.NAME, property = "species")
@JsonSubTypes({
        @JsonSubTypes.Type(value = Dog.class, name = "dog"),
        @JsonSubTypes.Type(value = Cat.class, name = "cat")
})
interface Animal {
    String name();
}

abstract static class Mammal implements Animal {
    @JsonProperty
    private final String name;

    protected Mammal(String name) {
        this.name = requireNonNull(name, message: "name");
    }

    @Override
    public String name() {
        return name;
    }

    public abstract String sound();
}
```

```
static final class Dog extends Mammal {
    @JsonProperty
    private final int age;
    @JsonProperty
    private final String[] favoriteFoods;
    @JsonProperty
    private final Toy favoriteToy;
    @JsonProperty
    private final VetRecord vetRecord;

    @JsonCreator
    Dog(@JsonProperty("name") String name, @JsonProperty("age") int age,
        @JsonProperty("favoriteFoods") String[] favoriteFoods, @JsonProperty("favoriteToy") Toy
        @JsonProperty("vetRecord") VetRecord vetRecord) {
        super(name);
        this.age = age;
        this.favoriteFoods = requireNonNull(favoriteFoods, message: "favoriteFoods");
        this.favoriteToy = requireNonNull(toy, message: "favoriteToy");
        this.vetRecord = requireNonNull(vetRecord, message: "vetRecord");
    }
```

```
static final class Cat extends Mammal {
    @JsonProperty
    private final boolean likesTuna;
    @JsonProperty
    private final Toy scratchPost;
    @JsonProperty
    private final VetRecord vetRecord;

    @JsonCreator
    Cat(@JsonProperty("name") String name, @JsonProperty("likesTuna") boolean likesTuna,
        @JsonProperty("scratchPost") Toy scratchPost, @JsonProperty("vetRecord") VetRecord vetRecord)
        super(name);
        this.likesTuna = likesTuna;
        this.scratchPost = requireNonNull(scratchPost, message: "scratchPost");
        this.vetRecord = requireNonNull(vetRecord, message: "vetRecord");
    }

    @Override
    public String sound() {
        return "meow";
    }
```

▢ 사용 기술 / 적용 맥락 및 이유

- JUnit / 통합 테스트

  → 앞선 과정에서 추출한 Schema.json 생성 확인만으로는 부족하다고 판단하여
  테스트 코드를 통해 예시 객체들로 테스트 진행


```
{
  "name": "com.linecorp.armeria.internal.server.annotation.PolymorphismDocServiceTest$Animal",
  "fields": [
    {
      "name": "name",
      "typeSignature": "string",
      "requirement": "REQUIRED"
    }
  ],
  "descriptionInfo": {
    "docString": "",
    "markup": "PLAIN_TEXT"
  }
}
```



```
{
  "name": "com.linecorp.armeria.internal.server.annotation.PolymorphismDocServiceTest$Animal",
  "fields": [
    {
      "name": "name",
      "typeSignature": "string",
      "requirement": "REQUIRED"
    }
  ],
  "descriptionInfo": {
    "docString": "",
    "markup": "PLAIN_TEXT"
  },
  "oneOf": [
    "com.linecorp.armeria.internal.server.annotation.PolymorphismDocServiceTest$Dog",
    "com.linecorp.armeria.internal.server.annotation.PolymorphismDocServiceTest$Cat"
  ],
  "discriminator": {
    "propertyName": "species",
    "mapping": {
      "dog": "#/$defs/models/com.linecorp.armeria.internal.server.annotation.PolymorphismDocServiceTest$Dog",
      "cat": "#/$defs/models/com.linecorp.armeria.internal.server.annotation.PolymorphismDocServiceTest$Cat"
    }
  }
}
```


Animal 인터페이스에 Jackson Annotation에 작성된 대로 그 자식 객체인 dog, cat의
정보가 반영됨

▢ 기타 활동

1. 오류 분석/문제 분석

- 문제 현상 : 일반 빌드는 시간이 40분 이상 소요되어
  Armeria 프로젝트의 공식 문서에서는 병렬 빌드를 추천하는데,
  병렬 빌드 시 서로 다른 이유로 매번 실패함.
  재시도 시 성공하다가 JVM 관련 TimeOut발생
- 원인 : shadedTest들은 unitTests들을 shaded classes로 다시 돌리는 태스크인데, 비동기, 통합테스트가 시간 안에 완료되지 않아 실패하는것
  병렬 빌드 시 max workers를 줄이면 발생하지 않는 것으로 보아
  과도한 Context Switching으로 인해 실패하는 것으로 생각됨.

  테스트를 할 때 JVM이 60초 이내의 TimeOut을 두고있음.
  Armeria 서버가 graceful Shutdown을 위해 테스트 마다 최대 5초의 간격을 두는데, 이것이 모여 60초를 초과하여 TimeOut 에러 발생.

2. 개선 과정

- 에러 발생 과정 및 로그를 정리하여 Discussion에 보고

Request Guidance on Build Issues in my feature branch #6369

YoungHoney on Aug 26, 2025 · 2 comments · 2 replies

The Problem: Build Instability

However, this feature has uncovered significant and complex build stability issues when running a full parallel build (`./gradlew clean build --parallel`). While a build on the `main` branch succeeds, my feature branch consistently fails, with the failure points being non-deterministic.

Summary of Investigation

To demonstrate the instability, I ran the full `clean build --parallel` process three times on my feature branch. Here are the results:

Run #	Full Build Result	Failing Task(s) & Error Type	Individual Re-run Result
1	Failed	`:kubernetes:shadedTest` `:grpc:shadedTest` (`AssertionError`)	Succeeded
2	Failed	`:xds:shadedTest` (`TimeoutException`)	Succeeded
3	Failed	`:spring:boot2-webflux-autoconfigure:shadedTest` `:spring:boot3-webflux-autoconfigure:shadedTest` (JVM did not terminate cleanly)	Failed

As the data shows, the build fails at different points with different errors. Most failures appear to be flaky tests that pass when run in isolation (Run 1 & 2). However, some failures are more persistent and suggest a severe resource leak, failing even on an individual re-run (Run 3).

My Environment:

- **OS:** Linux 6.8.0-78-generic amd64
- **JDK:** OpenJDK 21.0.8 (Ubuntu 21.0.8+9-Ubuntu-0ubuntu124.04.1)
- **Gradle:** 8.10.2

▢ 기타 활동

2. 개선 과정

- 에러 발생 과정 및 로그를 정리하여 Discussion에 보고
- 보고 이후 Maintainer가 fix
- fix 반영 이후 상황 재 보고

Request Guidance on Build Issues in my feature branch #6369

YoungHoney on Aug 26, 2025 · 2 comments · 2 replies

minwoox on Aug 29, 2025 **Maintainer**

영어로 된 원본 주석 - 한국어로 번역

> :spring:boot2-webflux-autoconfigure:shadedTest
> :spring:boot3-webflux-autoconfigure:shadedTest
> (JVM did not terminate cleanly)

This will be fixed by this PR:
#6379

YoungHoney on Aug 29, 2025 **Author**

영어로 된 원본 주석 - 한국어로 번역
Thank you !

YoungHoney on Aug 30, 2025 **Author**

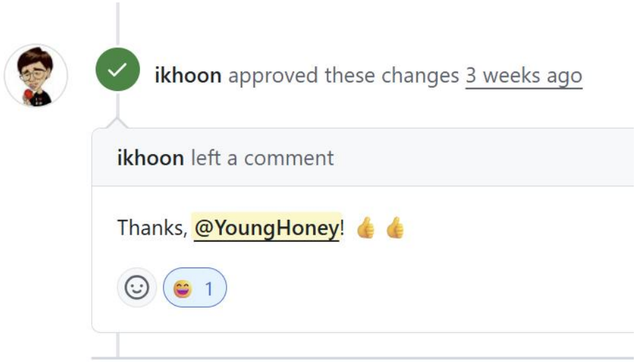
영어로 된 원본 주석 - 한국어로 번역
I've done some local testing by merging your PR #6379 into my feature branch to verify the fix, and I have some interesting findings to share.

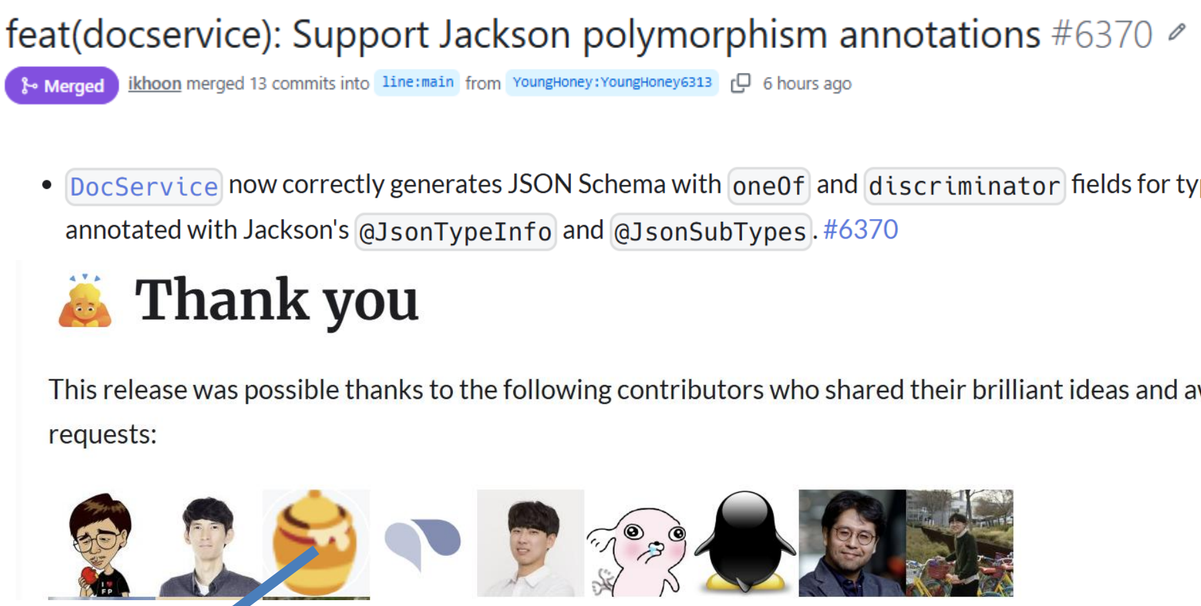
For context, my local environment is running on an **Intel Core Ultra 7 155H CPU (16 cores, 22 logical threads)**.

Here is a summary of my build attempts after merging your fix:

Build Command	Result	Observations
`./gradlew clean build --parallel`	🔴 FAILED	worked perfectly for the `JVM did not terminate cleanly` error, as it no longer occurs. However, the build still fails with the other errors like `:kubernetes`, `:grpc`, and `:xds`.
`./gradlew clean build --parallel --max-workers=6`	🟢 SUCCESSFUL	By explicitly limiting the number of parallel workers to 6, the build now completes successfully and consistently.

Sharing this in case it helps with the investigation. Thank you again!





▢ 프로젝트 성과 및 결과

- 3명의 Maintainer에게 Approve를 받았으며, Merge되어 Contributor로서 등록 되었습니다.

▢ 프로젝트 회고

- 오픈소스 기여에 관심이 컸지만, 감히 내가 해도 될까 하는 두려움이 있었습니다. 그러나 Maintainer들이 적극적으로 도와주셨고, 어느덧 작업을 마무리하게 되었습니다.
- 코드를 분석하면서 Mermaid로 클래스 다이어그램으로 나타내곤 했는데. DocService를 구성할 때 Builder패턴을 사용하고, Provider들을 사용하는 부분에서 Strategy 패턴과 CoR 패턴이 보였는데, 설계패턴 과목을 듣지 못했다면 알아채지 못했을 것 같습니다.
- 작업과정에서 null값 처리. 고려하지 못한 Case들에 대한 처리 등 실제 구현만큼 예외처리가 중요하다는 생각이 들었고, 실제로 Product를 생산하는 실무자가 되려면 이런 점까지 다 챙기는 능력을 가져야 겠다 생각했습니다.