BACKEND DEVELOPER

문제의 원인을 구조에서 찾아나가는

백엔드 개발자 신동걸

Email geolyun@gmail.com
GitHub github.com/geolyun

소개

SUMMARY

- Spring Boot 기반 REST API 설계·구현 경험
- Spring Security(JWT/OAuth2)로 인증/인가 구조 설계 및 구현
- JPA/QueryDSL로 도메인 모델링, 동적 검색, 성능 튜닝 경험
- Docker Compose로 로컬 인프라 구성 → 관측(Prometheus/Grafana)·부하테스트(k6)까지 경험

CONTACT

Email	geolyun@gmail.com
GitHub	https://github.com/geolyun

"실행 가능한 설계와
측정 가능한 개선"을 지향합니다.

기술 스택

주요 프로젝트에서 직접 사용한 기술 중심으로 정리했습니다.

Languages

Java
JavaScript/TypeScript
Python

Framework / Library

Spring Boot, Spring MVC, Spring Security
JPA(Hibernate), QueryDSL, MapStruct

Database / Storage

MySQL, H2
Redis(캐시/락)
AWS S3(이미지 업로드)

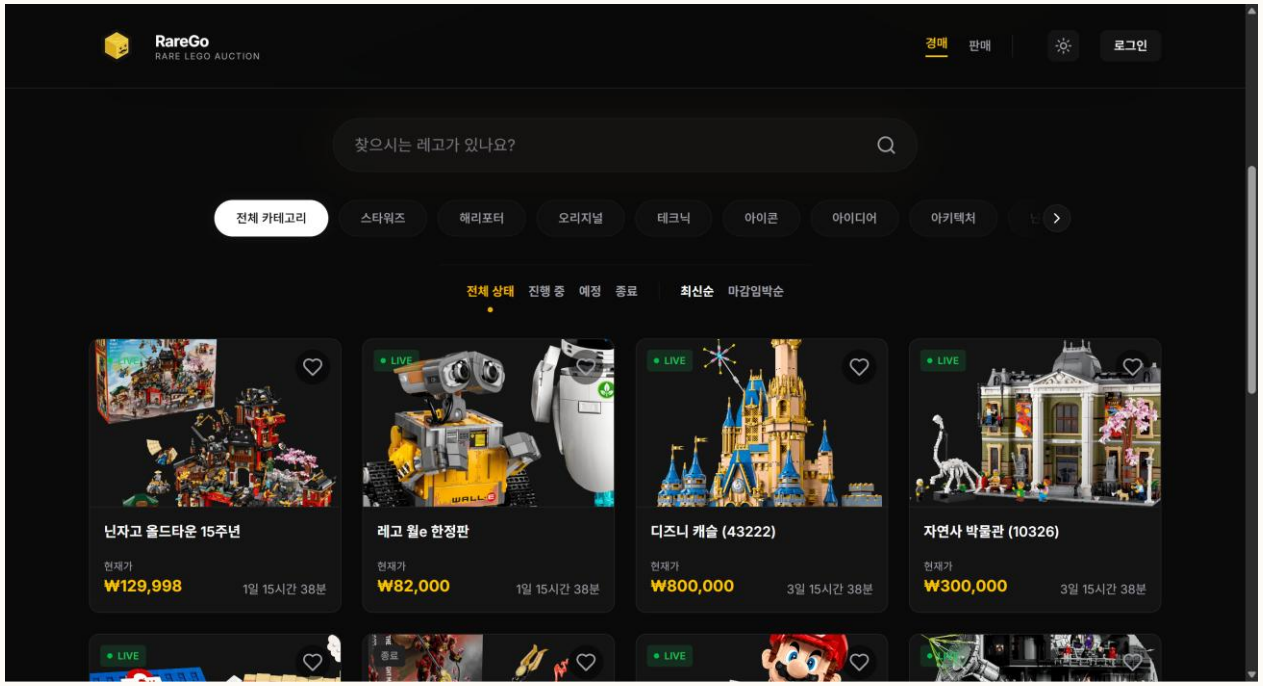
Infra / Etc

Kafka, Elasticsearch
Docker, Docker Compose, Kubernetes(학습/실습)
Micrometer, Prometheus, Grafana, k6
GitHub Actions, Swagger/OpenAPI

Project 1. RareGo | 실시간 경매 서비스

MSA 기반으로 동시성·검색·관측을 함께 다룬 팀 프로젝트

OVERVIEW



기간 2026.01 ~ 2026.02

팀 6인 (BE 6)

한 줄 소개

실시간 입찰과 검색을 중심으로 확장 가능한 레고 경매 플랫폼을 구현

주요 역할

- 입찰 도메인 로직 및 트랜잭션 경계 설계
- 검색(Elasticsearch) 연동 및 인덱싱 흐름 설계
- 로컬 인프라(Docker Compose)·모니터링(Prom/Grafana) 구성

Tech

Spring Boot · MySQL · JPA · Redis · Kafka · Elasticsearch
Docker Compose · Prometheus/Grafana · k6 · GitHub Actions

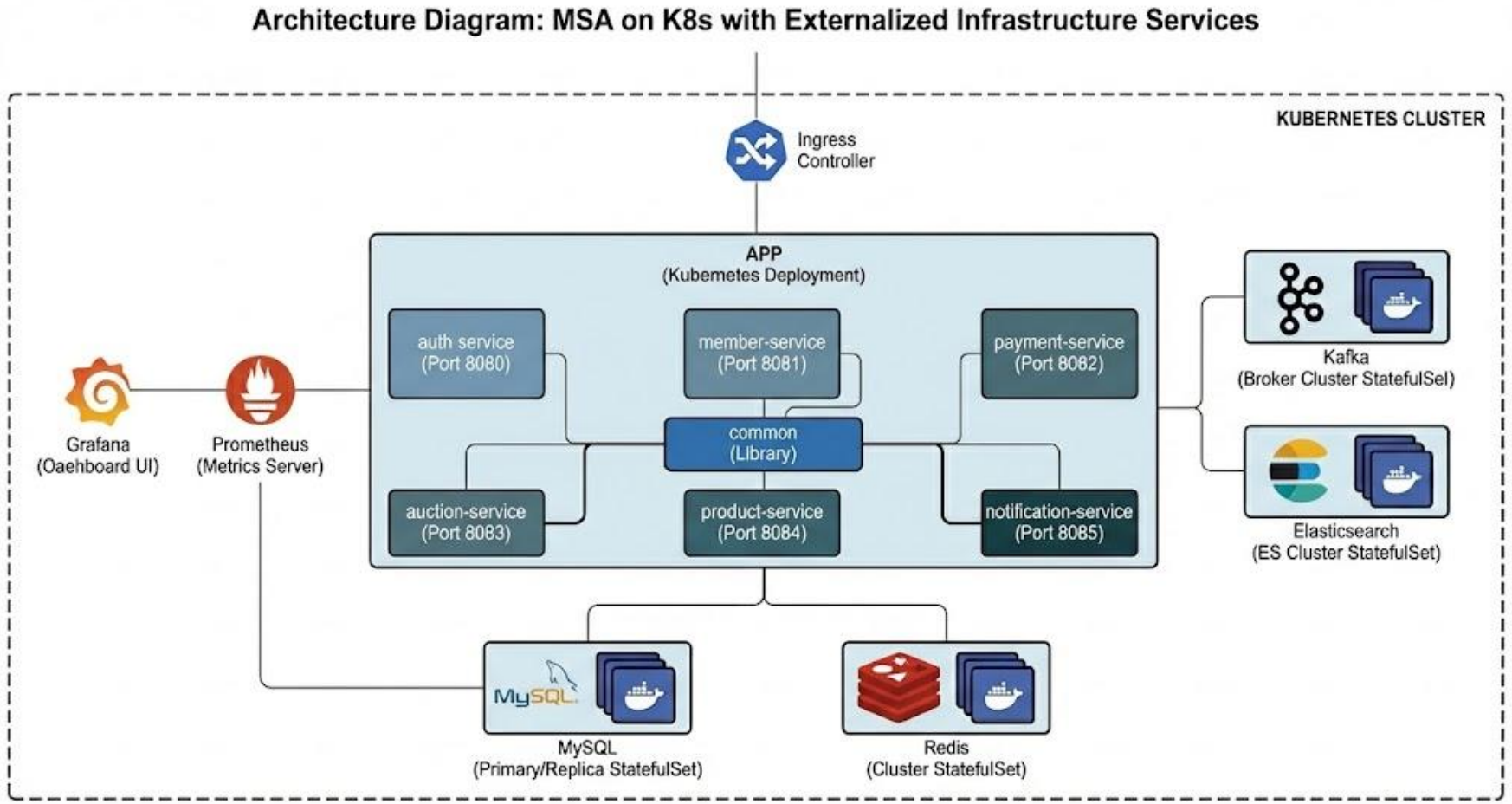
Repo

github.com/geolyun/beadv4_4_BugZero_BE

RareGo | 아키텍처

MSA + 이벤트 기반 흐름을 고려한 구성(로컬은 Docker Compose로 재현)

ARCH



RareGo | 구현 ① 입찰 처리(동시성/정합성)

동시에 여러 사용자가 입찰해도 최고가/낙찰 상태가 꼬이지 않도록 설계

IMPLEMENT

핵심 포인트

• 입찰 요청을 Redis 분산 락 획득 → 트랜잭션 처리 → 최고가 재검증 → 저장 흐름으로 설계

• DB Row Lock 대신 Redis 분산 락으로 선제적 경합 제어 → DB 커넥션 풀 보호

• 입찰 도메인 규칙(최소 입찰 단위, 마감 시간, 현재가 재검증)을 서비스 계층에 집중시켜 정책 변경에 유연하게 대응

데이터 정합성 관점

• 분산 락 + 트랜잭션 조합으로 Lost Update(갱신 손실) 0건 보장

• 락 획득 이후 재검증 로직을 통해 "최고가 역전/중복 낙찰" 방지

• 비즈니스 Reject를 명확히 구분하여 시스템 오류와 정책 거절을 분리

예시 코드(요약)

```
# 입찰 UseCase
@DistributedLock(key = "'auction:bid:' +
#auctionId")
public BidReponseDto createbid(Long
auctionId, String memberPublicId, int
bidAmount) {
          Auction auction =
support.findAuctionById(auctionId);

          // 유효성 검증
          Optional<Bid> lastBid =
validateBid(auction, bidder, bidAmount);

# 경매 Facade
Auction auction =
support.findAuctionById(auctionId);
```

RareGo | 구현 ② Elasticsearch 기반 검색

DB LIKE/조인 중심 검색의 한계를 보완하고 확장 가능한 검색 구조로 전환

SEARCH

검색 흐름

※ 검색은 ES에서 수행하고, 상세 데이터는 DB에서 정합성 있게 조회하는 패턴으로 구성

무엇을 해결했나

- 키워드/필터/정렬 조건이 늘어나도 확장 가능한 검색 구조 확보
- DB 부하를 검색 트래픽으로부터 분리하여 병목 지점을 줄임

설계 포인트

- 인덱스 문서 설계: 검색/정렬에 필요한 필드만 ES에 반영
- 일관성: 상품 변경 이벤트 → 인덱스 동기화(최소 지연) 흐름 구성
- 실패 대비: 재시도/모니터링으로 인덱싱 누락을 감지 가능하게 구성

RareGo | 트러블슈팅

문제 정의 → 시도 → 결과가 한 눈에 보이도록 정리

TROUBLE

1 문제 정의

- 동시 입찰(수십~수백 RPS) 상황에서 최고가가 "역전"되는 케이스가 재현
- 입찰 검증/저장이 여러 트랜잭션으로 쪼개져 경쟁 조건이 발생
- 서비스 간 이벤트 처리 중 중복 처리(재시도) 가능성도 존재

2 시도(해결)

- 입찰 처리 로직을 단일 트랜잭션으로 묶고, 경매 엔티티에 락을 적용
- 재시도 시에도 동일 결과가 되도록 멱등 키/상태 체크 로직을 추가
- k6 시나리오를 작성해 동시 입찰을 반복 재현하고, 지표를 Prometheus/Grafana로 관측

3 결과

- 동시 입찰 테스트에서 최고가 역전/중복 낙찰 케이스가 재현되지 않도록 개선
- 관측 지표(p95, error rate)를 기준으로 병목을 확인하고 개선 포인트를 빠르게 찾을 수 있게 됨
- "정합성"을 코드 규칙으로 고정해 이후 기능 추가 시에도 안전하게 확장 가능

RareGo | 프로젝트 리뷰

성과·배운 점·다음 계획을 구분해서 정리

REVIEW

잘한 점 / 기여

- 동시성/정합성 이슈를 "재현 가능한 테스트"로 만들고 해결 과정을 문서화
- 검색 구조를 ES 중심으로 분리해 확장성(필터/정렬 추가)을 확보
- 로컬 인프라를 Docker Compose로 표준화해 팀 온보딩/재현성을 개선

배운 점

- 트랜잭션 격리/락 전략에 따라 데이터 정합성이 달라짐을 체감
- 관측 지표가 있어야 개선이 "추측"이 아니라 "검증"이 됨

다음 단계(성장 계획)

- Outbox/Saga 등 분산 트랜잭션 패턴을 실제 적용해 "이벤트 정합성"까지 확장
- 캐시/검색/DB 인덱스까지 포함한 성능 튜닝 루프를 더 체계화(측정→개선→회귀)
- Kubernetes로 운영 배포 파이프라인을 구성하고 롤링 업데이트/오토스케일 실습

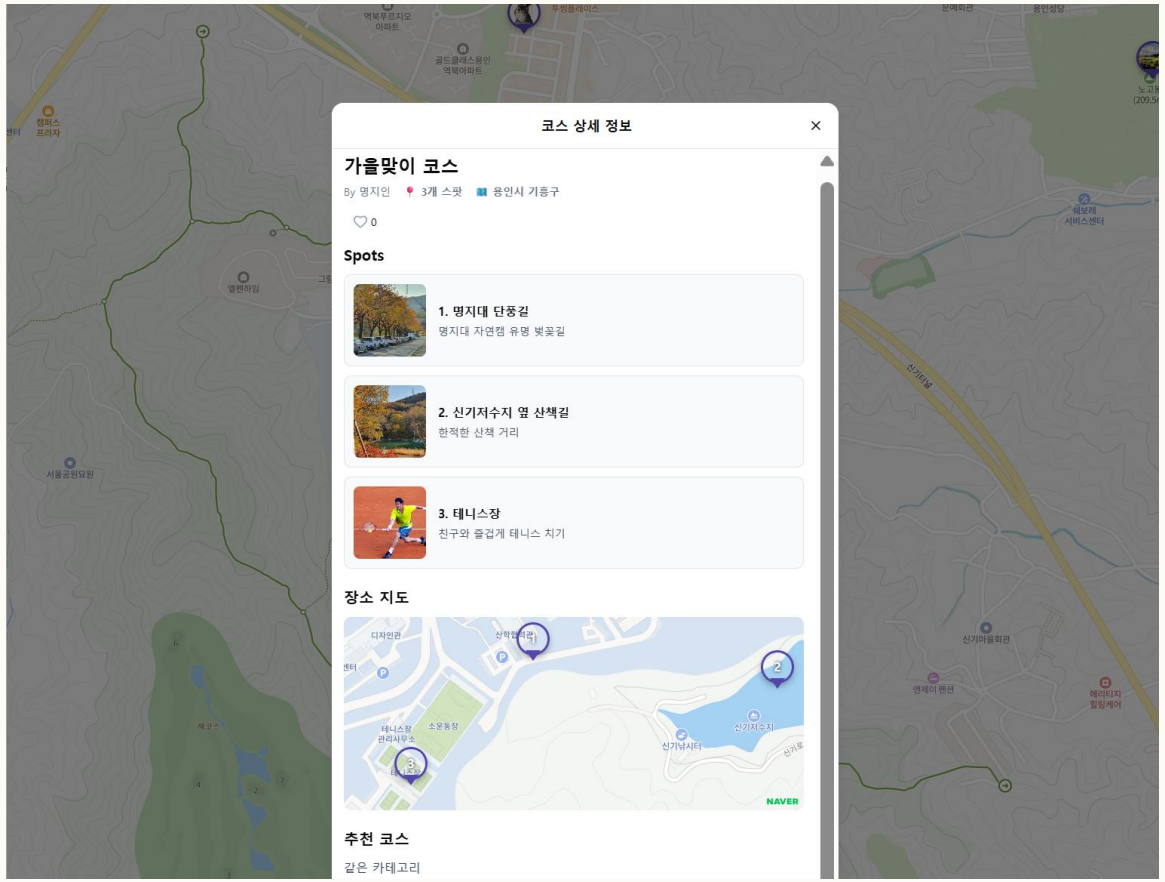
Repo

github.com/geolyun/beadv4_4_BugZero_BE

Project 2. Ringco | 사용자 코스 추천 서비스

지도 기반 코스 생성/공유 · 조건 검색 · 소셜 로그인

OVERVIEW



기간	2025.03 ~ 2025.06
팀	3인 (FE 1, BE 1, PM 1) · 백엔드 단독 담당
주요	코스/스팟 CRUD · 조건 검색 · 지도 API · 이미지 업로드

구현 기능

- JWT 기반 API 인증/인가
- Google/Kakao OAuth2 로그인
- QueryDSL 동적 검색(카테고리/예산/지역/키워드)
- S3 이미지 업로드, 이메일 인증

Tech

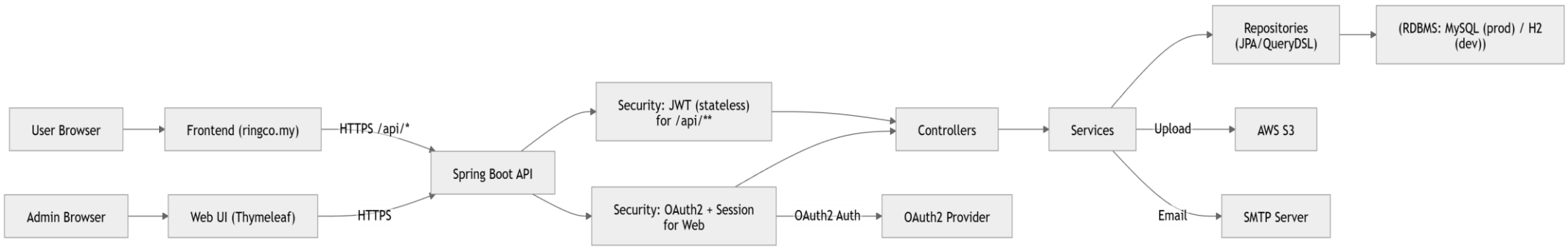
Spring Boot · Spring Security · JPA · QueryDSL · MySQL
OAuth2 · JWT · MapStruct · AWS S3 · Spring Mail

Repo github.com/geolyun/Capstone-2

Ringco | 아키텍처

인증/외부 연동(지도·OAuth2·S3)을 포함한 백엔드 구성

ARCH

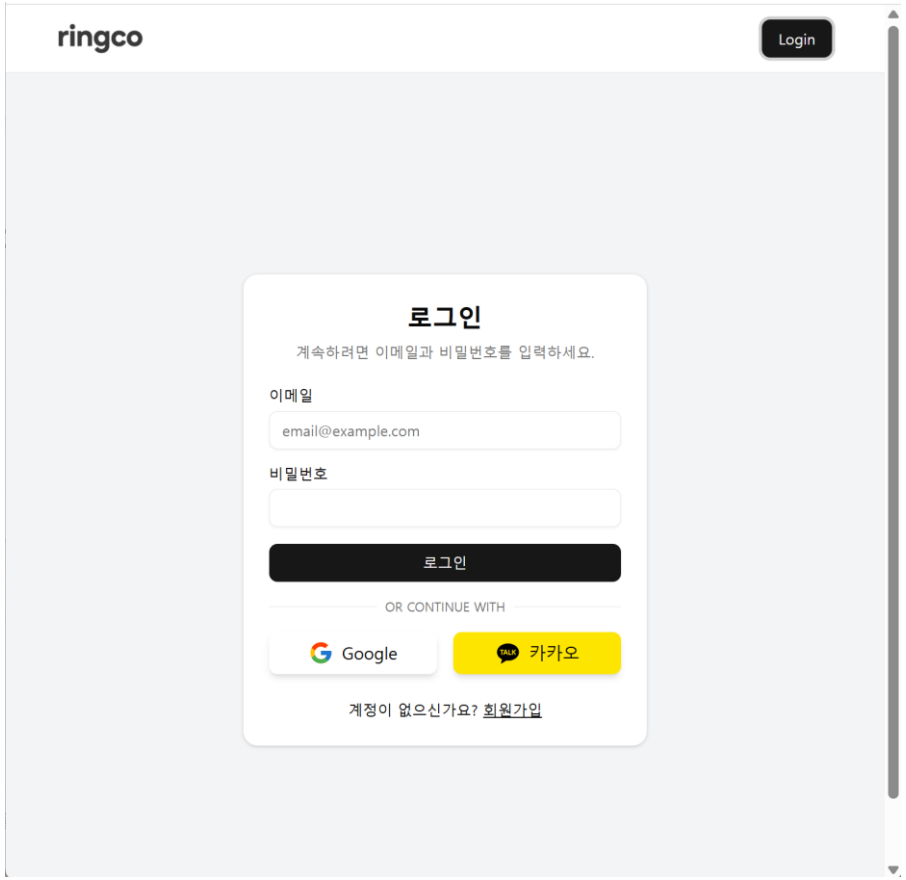


환경 구성: dev(H2 In-Memory) / prod(MySQL) 프로파일 분리

Ringco | 구현 ① 인증/인가 (JWT + OAuth2)

로그인/회원 가입 플로우를 Spring Security로 일관되게 구성

IMPLEMENT



구현 내용

- Spring Security FilterChain 구성으로 API 접근 제어
- OAuth2 로그인 성공 핸들러에서 사용자 생성/매핑 후 JWT 발급
- 권한 기반 엔드포인트 분리 및 예외 처리 표준화

보안/운영 관점

- .env 기반 시크릿 관리(JWT Secret, OAuth Client)
- CORS/Redirect URI 설정으로 OAuth2 연동 안정화

Ringco | 트러블슈팅 & 개선

기능 구현뿐 아니라 "유지보수성"을 개선한 사례

IMPROVE

문제: 조건 검색 쿼리 관리 난이도

- 카테고리/예산/지역/키워드 등 선택 조합이 많아 JPQL 문자열이 복잡해짐
- 요구사항 추가 시 쿼리 수정 범위가 커져 버그 발생 가능

해결: QueryDSL 도입

- 조건을 BooleanExpression으로 분리해 조합/재사용 가능하게 구성
- 정렬/페이징까지 QueryDSL로 통합하여 Repository 계층을 단순화

개선: DTO 매핑/보일러플레이트 감소

- MapStruct 도입으로 Entity↔DTO 변환 코드를 선언적으로 관리
- API 응답 스펙 변경 시 매핑 규칙을 한곳에서 관리 가능

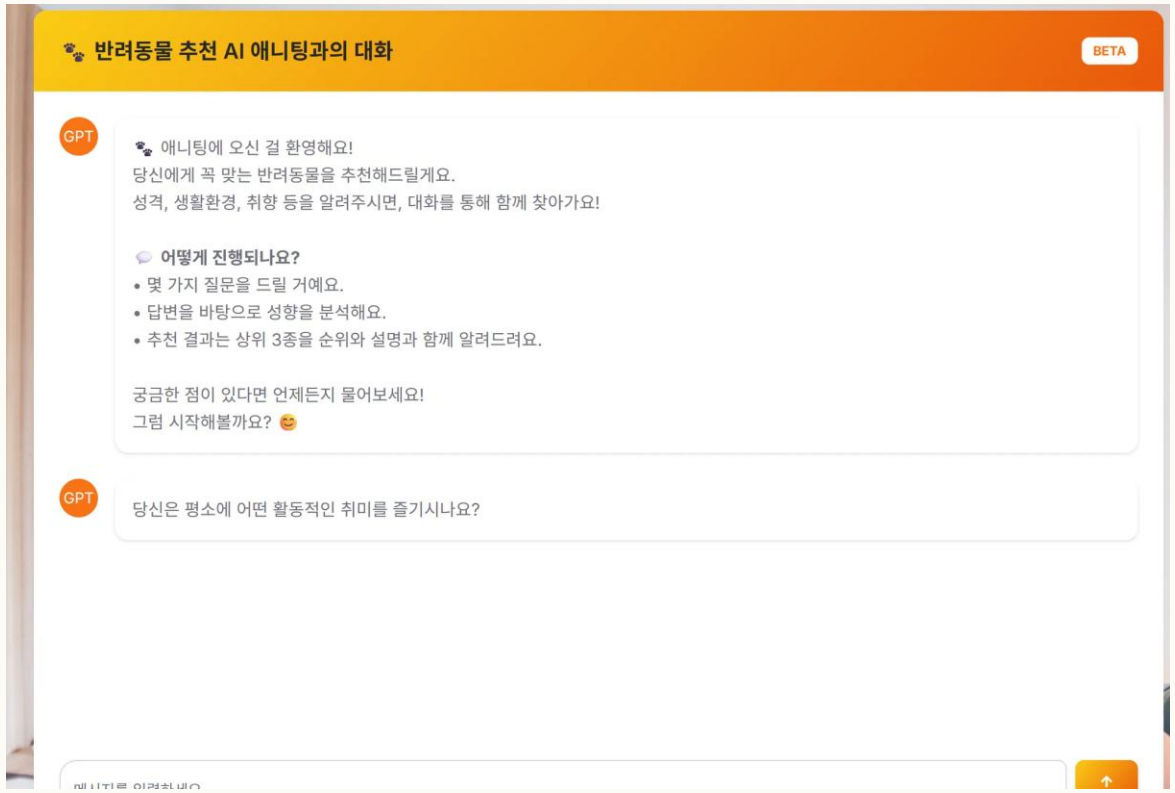
결과

- 기능 추가/수정 시 수정 범위를 줄이고, 테스트/리뷰가 쉬운 구조로 개선

Project 3. Aniting | 반려동물 추천 서비스

대화 흐름 설계 + 외부 API 연동 경험을 보여주는 프로젝트

OVERVIEW



기간	2025.03 ~ 2025.06
팀	2인 (BE 2)
역할	대화/추천 흐름 API 설계 및 결과 저장

핵심 구현

- 사용자 답변 기반 점수화 → 반려동물 추천 결과 산출
- Spring Security로 로그인/권한 처리
- JPA로 대화/추천 결과 저장 및 조회

Tech

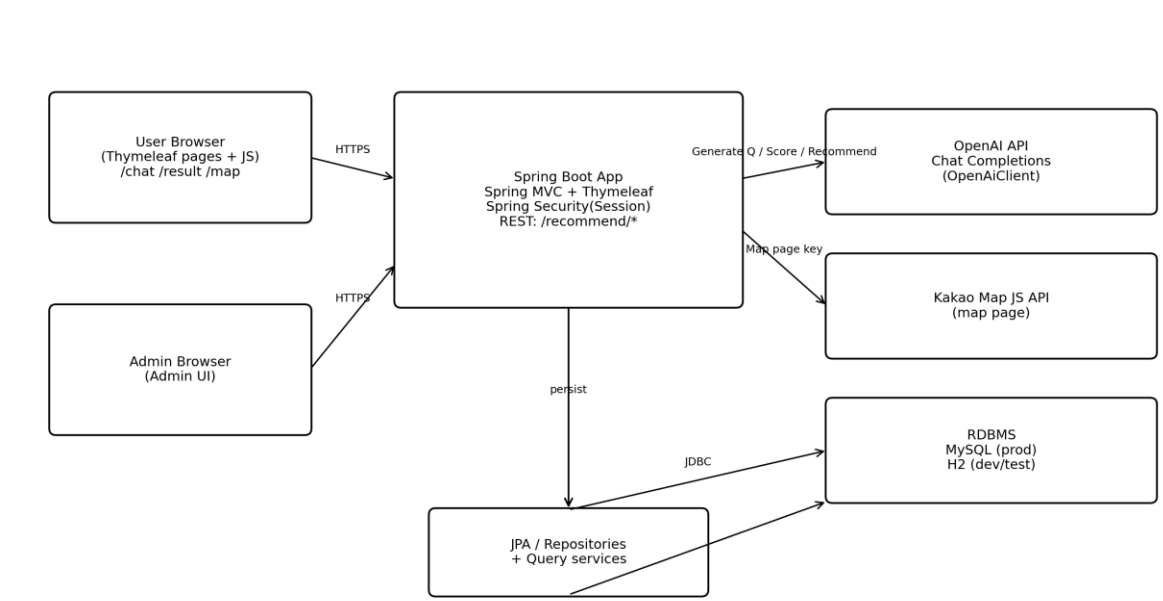
Spring Boot · Spring Security · JPA · MySQL · Thymeleaf

Repo github.com/geolyun/2025-capstone-aniting

Aniting | 아키텍처 & 트러블슈팅

외부 API 지연/실패를 고려한 사용자 경험(UX) 개선

ARCH



TROUBLE

문제

- 외부 API 호출 시간이 길어질 때, 사용자가 "멈춘 것"으로 인식

시도

- Timeout/Retry 정책을 추가하고 실패 시 사용자에게 재시도 가이드 제공
- 대화 상태를 DB에 저장해 새로고침/이탈 후에도 복구 가능하게 구현

결과

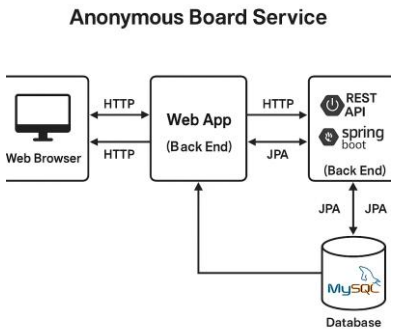
- 응답 지연/실패 상황에서도 서비스 이용 흐름이 끊기지 않도록 UX가 개선됨

Other Projects

개인적으로 진행한 프로젝트

SIDE

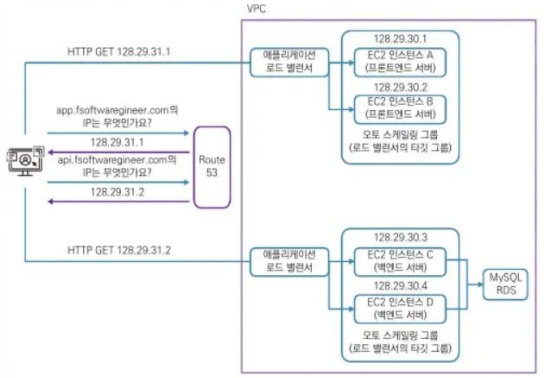
학교 분실물 센터(익명 게시판)



- 게시글 CRUD + 제목 검색, 비밀번호 기반 수정/삭제
- JPA + H2로 빠르게 구현, 단순하지만 완결된 서비스 흐름

github.com/geolyun/springboot-develop

Todo App (React + Spring Boot)



- JWT 기반 로그인/회원가입, Todo CRUD
- AWS(EC2/RDS) 배포를 포함해 전체 흐름을 경험
- React를 통한 Todo UI/UX 구현 경험

github.com/geolyun/boot-study github.com/geolyun/todo-react-app

학력 및 활동

EDU

학력

- 명지대학교 (2023.01 ~ 2026.03 예정) · 정보통신공학과 · 학점 3.86/4.5
- 강원대학교 (2020.03 ~ 2022.12) · 컴퓨터공학과

활동

- MCC 코딩 동아리 (2024.03 ~ 2024.12) – 세미나/프로젝트 참여
- 개인/팀 스터디 – Spring, JPA, CS, 백엔드 면접 질문 정리
- 백엔드 단기심화 데브코스 4기 (2025.12 ~ 2026.02) – 주문, 결제 플랫폼 구현 프로젝트 경험

GitHub github.com/geolyun

THANK YOU