

안녕하세요, 김종재입니다.

1995. 07. 03

gghs703@naver.com / 01053901348



- 최적화 및 성과
1. WebRTC 라이브 중 YOLO 기반 수신호 인식이 1FPS로 떨어지자, WebAssembly와 해상도 최적화로 17FPS까지 개선했습니다.
-개선: 입력 160×160 경량화 + WASM 적용 + 캡처 루프 수정 → 성능 0.90 유지, WebRTC 1-2 → 17 FPS.

2. Redis 캐싱을 적용해 낙찰가 차트 로딩 속도를 3초에서 0.2초로 줄이며 실시간성을 높였습니다.

3. 젠킨스를 통한 자동빌드/배포, Nginx 설정, 웹소켓 설정(실시간 추가), Https 설정






4. 1일 1 알고리즘 학습으로 백준 플래티넘 4티어이며 상위 2.21%에 도달했습니다.

- 프로젝트 요약
1. 농산물 경매 사이트 (Freshbid)- WebRTC기반 실시간 농산물 경매사이트/ 수신호,가격예측, 신선도 체크 AI
-역할: 백엔드(상품, 카테고리, 리뷰, 문의 설계 및 구현)/ 수신호 및 가격예측 AI 구현

2. 모의투자 사이트 (주식나침반)- 실시간성 주식 모의 투자 시스템/ 800만개의 뉴스에서 각 기업별 핵심 키워드 추출(SPARK)
-역할: 인프라 (Jenkins, Nginx, 웹소켓) / 백엔드 (모의투자 설계 및 구현)

희망 직무

백엔드 엔지니어, 클라우드 엔지니어

주요 기술	Strong	 Java	<div>하</div> <div>중</div> <div>상</div>
		 SpringBoot	<div>하</div> <div>중</div> <div>상</div>
		 MySQL	<div>하</div> <div>중</div> <div>상</div>
	Known	 Vue	<div>하</div> <div>중</div> <div>상</div>
		 React	<div>하</div> <div>중</div> <div>상</div>

경력

삼성 청년 소프트웨어 아카데미(SSAFY)

2025. 01. 01 - (진행 중)

셀트리온제약 (생산관리)

2022. 06. 27 - 2024. 09. 26 (2년 3개월)

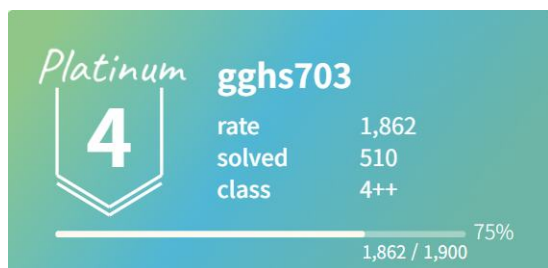
학력

건국대학교 서울캠퍼스

2015- 2021 (융합생명공학과)

Link

<https://github.com/KimJongjae1>





서비스

- 농산물 실시간 온라인 경매 사이트
- WebRTC 기반 라이브 경매
- 수신호 인식 /신선도 체크/ 가격 예상 AI

백엔드 기술 스택



인프라



역할

BackEnd

- 상품/카테고리 도메인 설계 및 API 구현
- 후기/리뷰 API 구현
- 낙찰가 차트 데이터 API 설계 및 가격 예측 도입
- WebRTC 라이브 중 입찰 수신호 자동 인식 도입
- 맡은 부분의 DB 설계/ 구축

FrontEnd

- 차트 페이지 구현
- 후기/리뷰 페이지 구현

기타

- 발표 수행

작업 기간

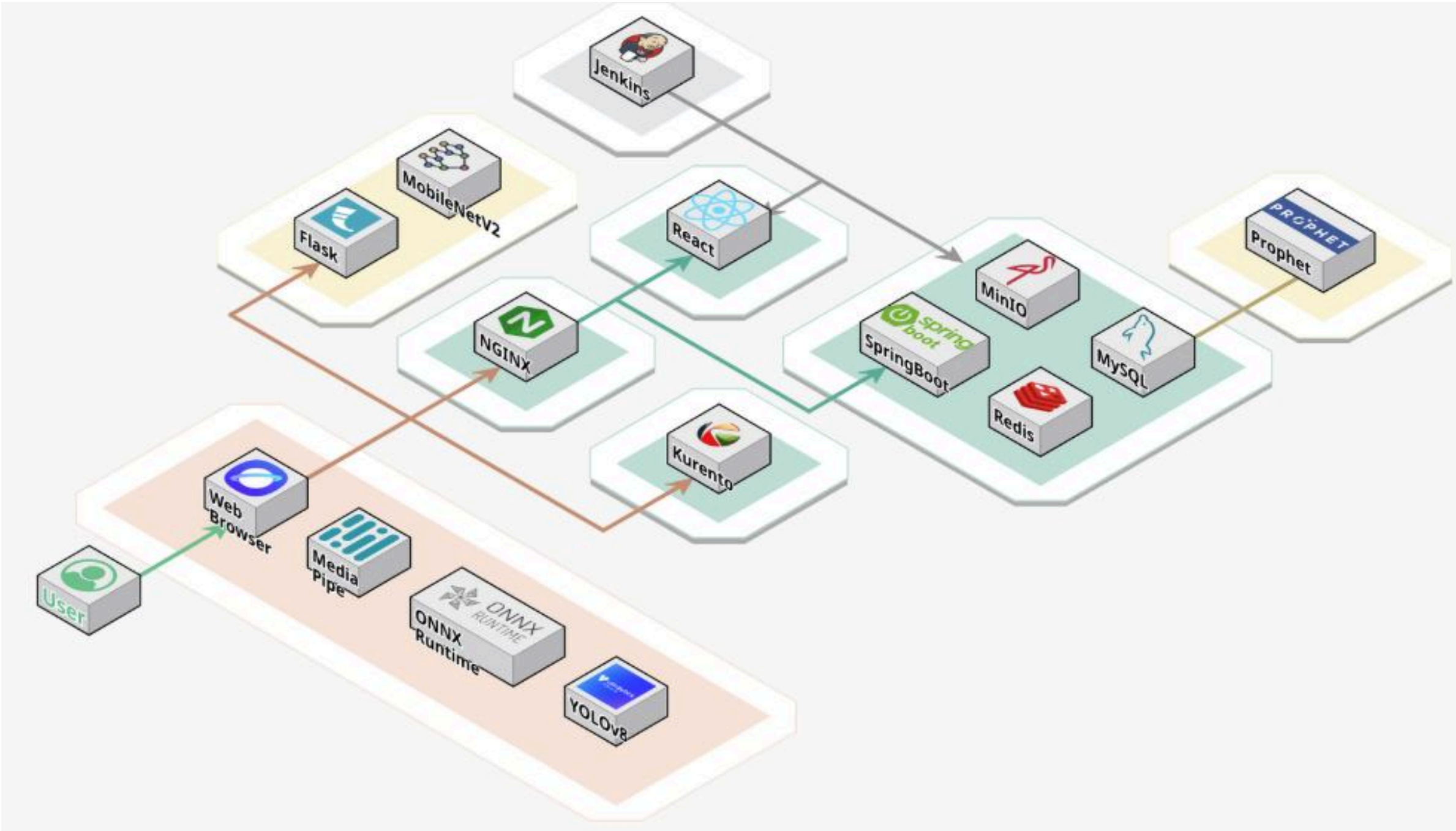
2025.07.07 ~ 2025.08.18 (6주)

구성원

- FE (이상민 / 윤승주)
- BE (김종재/조예진/김이루/ 박준서)

프로젝트 회고

- 프로젝트를 통해 **Git, jira**를 이용하여 **에자일 방식의 팀 프로젝트 수행**
- Docker/ jenkins 사용으로 자동 배포
- 수신호 AI를 실시간 경매에 적용하면서 성능 최적화 경험
- Redis 캐싱 도입으로 낙찰가 차트 응답 속도 3.0초 → 0.2초로 실시간 데이터 캐싱 적용, 차트 로딩 시간 90%이상 단축
- Swagger 사용으로 프론트 개발자와 쉽게 소통하고 API 테스트 수행
- JPA를 사용함으로써 간단한 CRUD는 Repository 메서드 호출로 간단 해짐. 만약에 복잡해지면 QueryDSL 사용.
- JWT 인증과 Spring Security를 사용함으로써 보안 강화



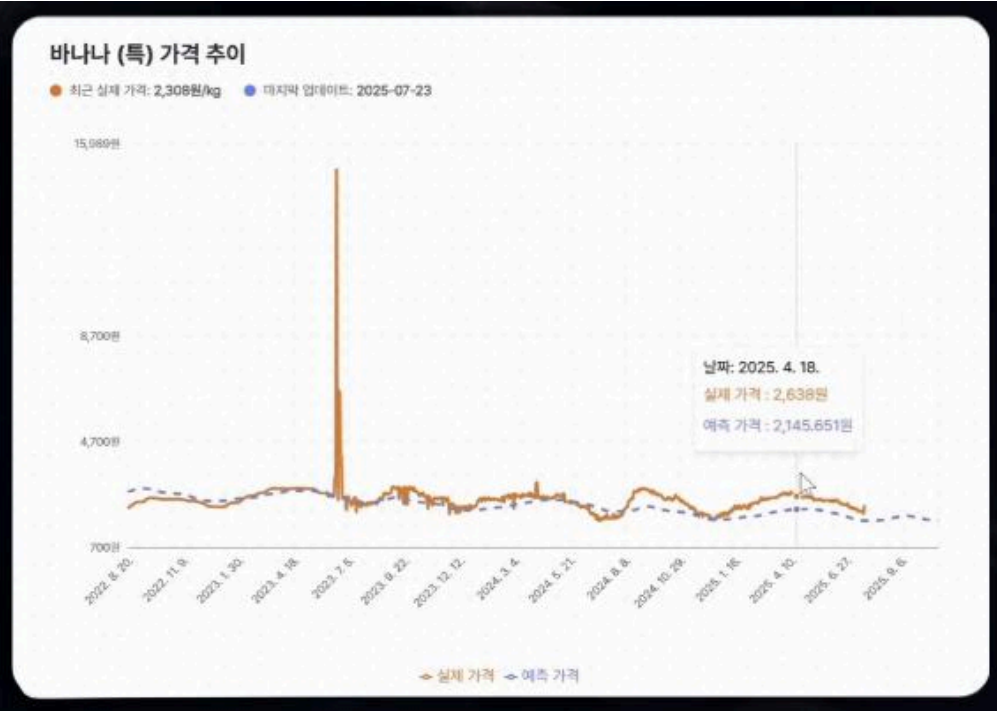
트러블 슈팅

- 1. 수신호 AI 성능 최적화 (3FPS → 17FPS)
 - 검증: Ultralytics YOLOv8 val로 테스트셋 4,854장 평가 → F1=0.91, 추론 지연 수 ms → 모델 이상 없음.
 - 원인: 병목은 브라우저 파이프라인(캡처-인코딩-전송-디코드-전/후처리-추론-렌더).
 - 개선: 입력 160×160 경량화 + WASM 적용 + 캡처 루프 수정 → F1=0.90 유지, WebRTC 1-2 → 11-13 FPS(기록).
- 2. 차트 API 속도 개선 (3초 → 0.2초)
 - Redis 캐싱 도입: 카테고리/등급 및 차트 데이터 캐싱
 - 배치 처리: 일일 배치로 예측 데이터 미리 생성

```
TS typescript
// front/src/hooks/useGestureModel.ts
if (isInferringRef.current) {
  return []; // 이미 추론 중이면 스킵
}
isInferringRef.current = true;
```

추론 중복 방지 코드와 수신호 인식 예시

```
J java
// back/src/main/java/FreshBid/back/service/impl/CategoryCacheServiceImpl.java
@Override
public void cacheSuperCategories(List<PriceDataRequestDto> superCategories) {
  String json = objectMapper.writeValueAsString(superCategories);
  redisTemplate.opsForValue().set(SUPER_CATEGORIES_KEY, json, CACHE_TTL);
}
```



배치 처리 코드와 실제 낙찰가 차트 예시

- 3. JWT 인증/인가 통합
 - 문제: 프론트/백엔드/웹소켓 간 토큰 전달 방식 불일치로 인증 실패 및 연결 오류 발생
 - 백엔드 토큰 처리 표준화 (만료 시 RefreshToken 자동 갱신)
 - 프론트 Axios 인터셉터로 모든 요청에 토큰 자동 포함
 - WebSocket 연결 시 동일한 검증 로직 적용

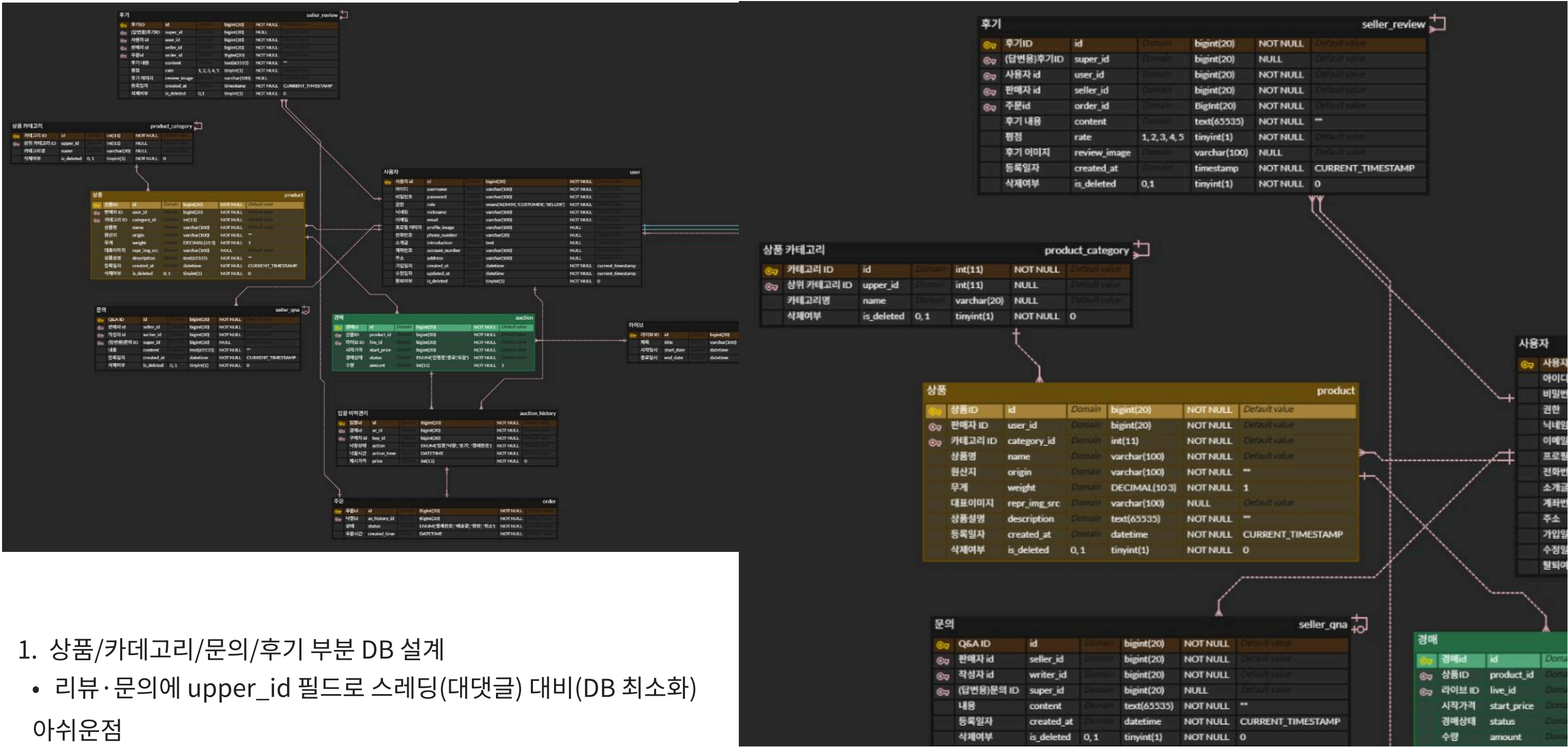
```
J java
// back/src/main/java/FreshBid/back/filter/JwtAuthenticationFilter.java
if (authorizationHeader != null && authorizationHeader.startsWith("Bearer ")) {
  String token = authorizationHeader.substring(7);
  jwtTokenProvider.validateToken(token);
}
```

백엔드 필터에서 토큰 표준화 및 자동 갱신

```
TS typescript
// front/src/api/axiosInstance.ts
axiosInstance.interceptors.request.use((config) => {
  const token = useUserStore.getState().accessToken;
  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  return config;
});
```

프론트 Axios 인터셉터로 모든 요청에 토큰 자동 포함

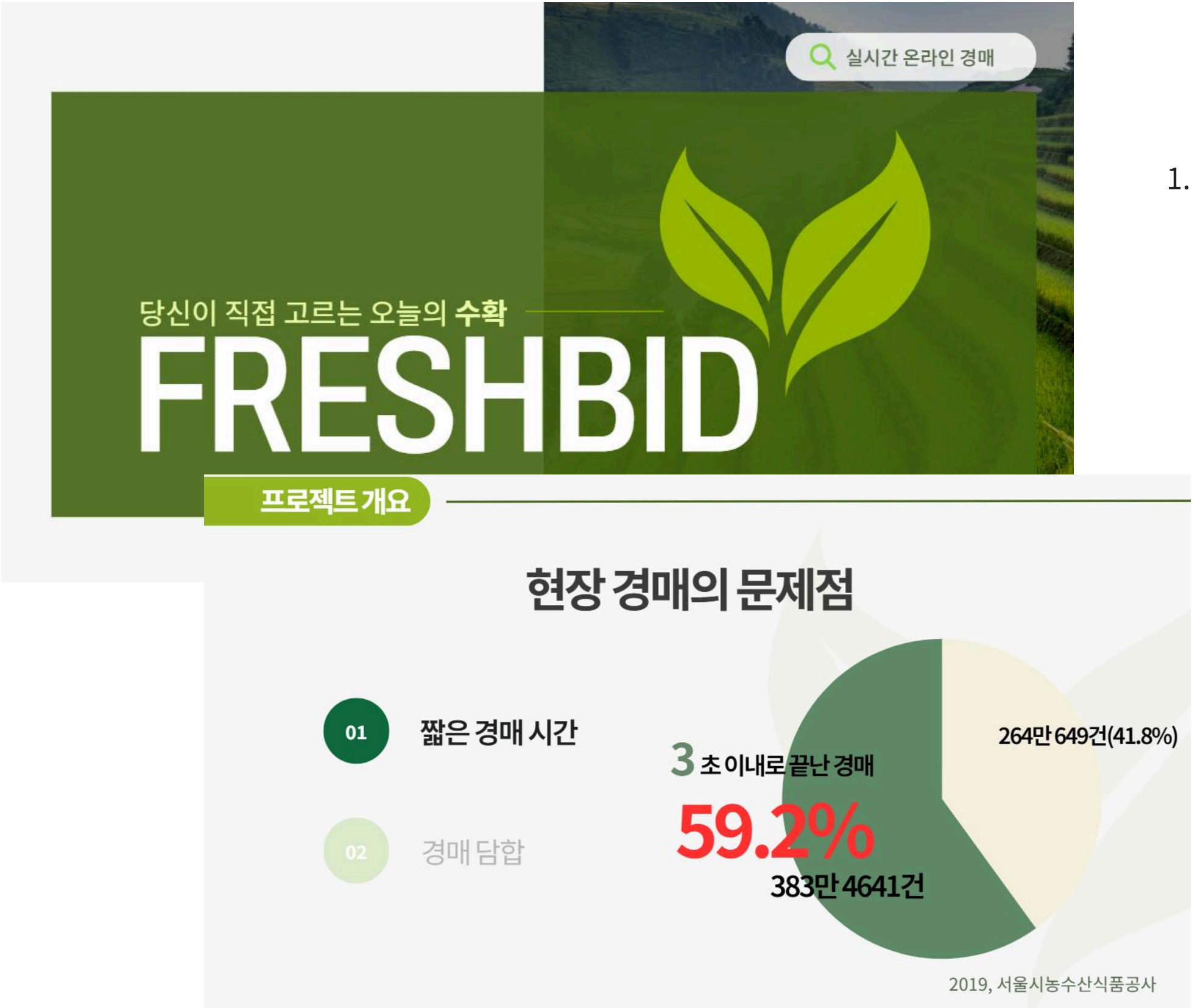
문서화



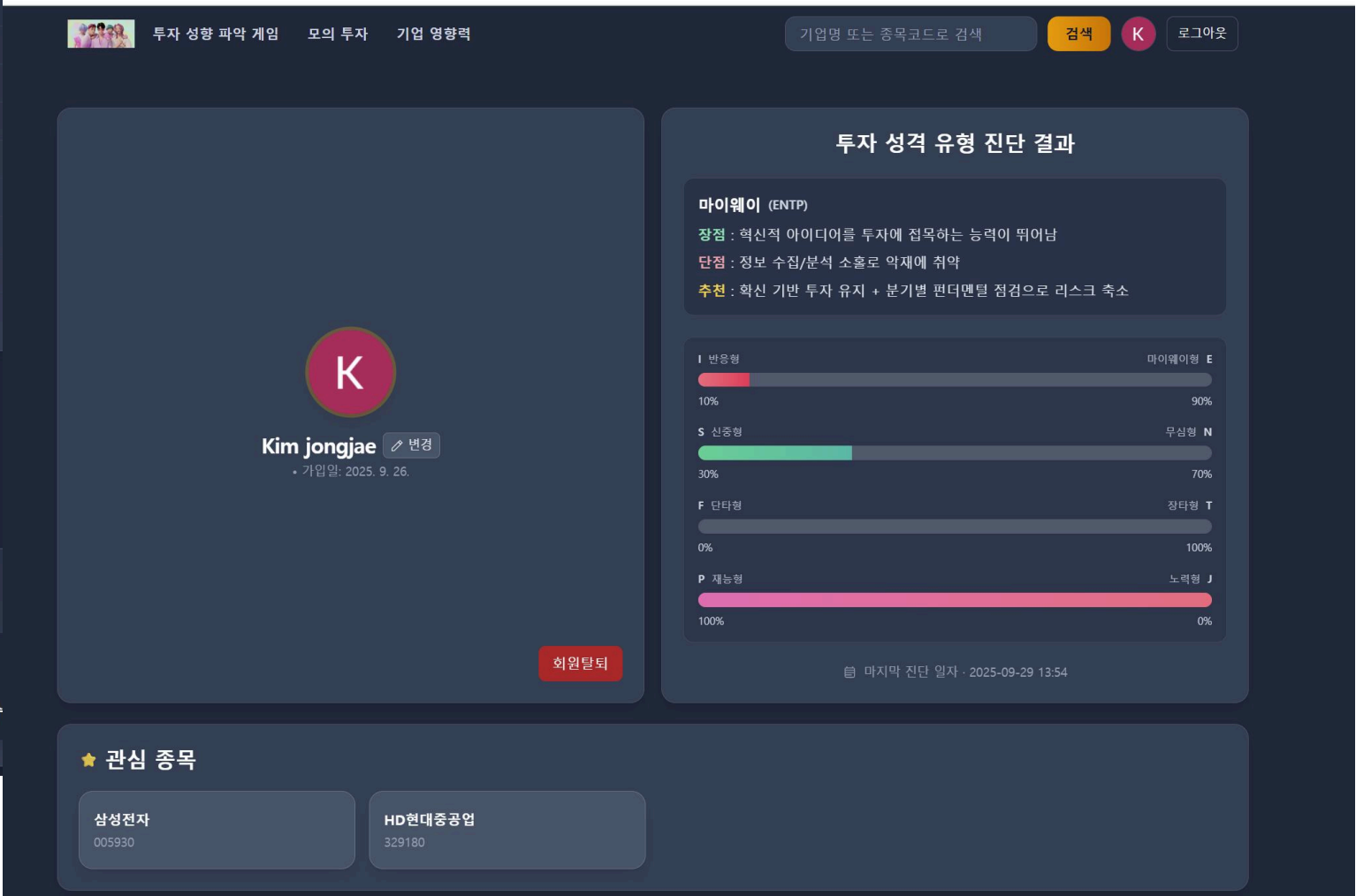
- 1. 상품/카테고리/문의/후기 부분 DB 설계
 - 리뷰·문의에 upper_id 필드로 스테딩(대댓글) 대비(DB 최소화) 아쉬운점
 - 타임스탬프 일관성 부족: updated_at, is_deleted 등 변화에 대한 타임스탬프가 적어서 관리자입장에서 확인하기 매우 힘들

발표

- 성과: SSAFY 공통프로젝트 우수상 (10팀중 2등)
- 800명이 있는 라이브 앞에서 발표 수행



- 1. 발표자료 및 발표 대본 작성
 - 발표에 앞서 저의 생각이 아닌 청자의 입장에서 작성
 - 통계와 수치를 위주로 작성
 - 개요-기능-시연-기술과 트러블슈팅-기대효과 순으로 맨처음 수치를 토대로 프로젝트의 정당성을 부여.
 - 특히 사회적인 이슈를 이야기하면 청중들은 쉽게 납득하며 이 점을 이용하여 개요부분을 전체 발표의 30퍼센트를 할당함.
 - 마지막으로 적극적으로 주변분들한테 발표를 보여주면서 말투, 흐름, 제스처를 적극 고치기.



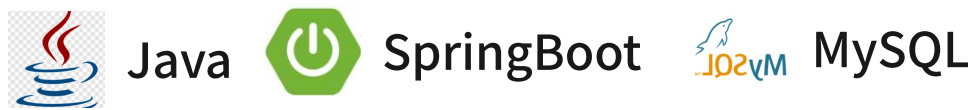
서비스

실시간성 주식 모의 투자 시스템

투자성향 분석 게임

800만건의 뉴스에서 각 기업별 핵심 키워드 추출

백엔드 기술 스택



인프라



역할

Infra

자동빌드/배포 파이프라인 구성 (Webhook → Jenkins → Docker Compose)

Nginx 리버스 프록시 설정 (HTTPS 리다이렉트, /api/*, /oauth2/* 라우팅)

Certbot을 이용한 SSL 인증서 적용 및 자동갱신 확인

EC2 서버 관리 (네트워크, Docker 서비스, 볼륨/로그 관리)

서버 내 웹소켓 설정(한국투자증권에서 실시간 주식 정부 수신)

BackEnd

모의 투자 설계

- TradeController-매수/매도, 자산조회
- StockController-실시간 데이터 및 히스토리 조회
- AccountService/StockService-자산 및 손익계산/주 가처리 및 데이터 수집

관련 DB 설계

뉴스 자동 크롤링 설정

작업 기간

2025. 08. 25 - 2025. 09. 29 (5주)

구성원

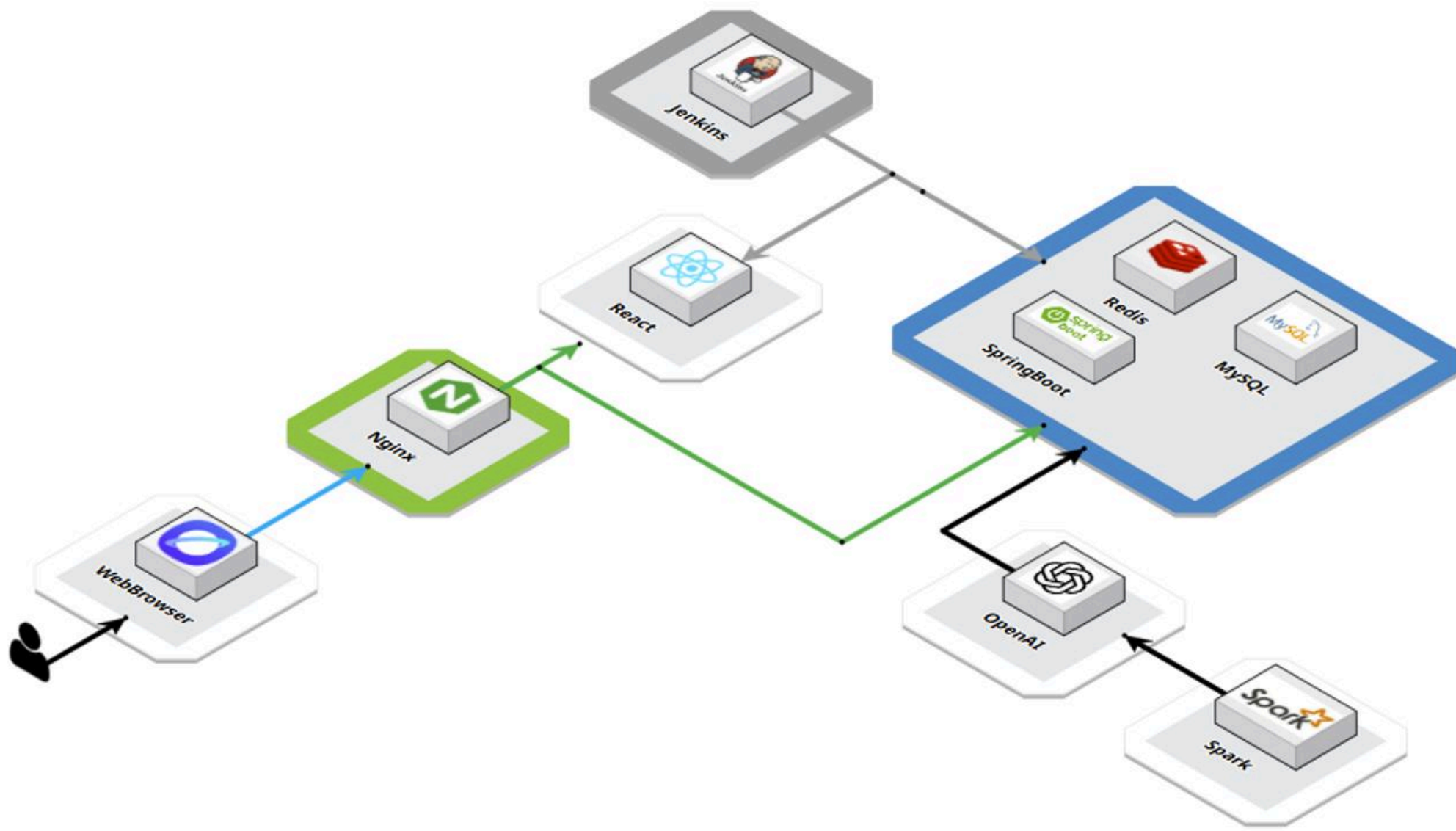
FE (김대정,장동현)

BE (김성현 / 이상용 / 정연수)

Infra&BE (김종재)

프로젝트 회고

- 인프라에 대한 지식
- 리눅스 명령어, 젠킨스, Nginx에 대한 이해
- 모의투자시에 설계를 잘 못해서 시간을 많이 낭비함. 백엔드 설계의 중요성을 배움.
- WebSocket를 서버에서 설정하며 실시간 데이터를 외부에서 가져옴
- Spark와 하둡에 대해 배움. 직접 하지는 않았지만 데이터를 다루는 방법 중 하나를 알게됨.



트러블 슈팅-인프라

- 1. 하드코딩
 - Front/Back 모두 8080/8765 하드코딩 (로컬호스트 개발 기준)
 - 팀장으로써 이런 부분을 초기에 확인 못 한점. 또한, 코드를 쓸 때, 항상 서버 가동을 염두에 뒤야함을 배움.
- 2. Https에 대한 이해도 부족
 - oAuth(구글/카카오)를 하려면 Https를 사용해야하지만 http로 초기 서버를 설정해버림.
 - Cerbot과 Let's encrypt를 배움. HTTPS는 내가 서버가 맞는지 확인하는 서버 키와 각 API를 암호화하는 세션키로 있음. 이후, 설정 완료함. (HTTP/2)

전체적인 개요

GitLab(master push/MR merge)
| Webhook (token=deploy-hook, job=build-and-deploy-all)
▼
Jenkins (8081)
└─ SSH로 서버 접속 → 빌드/배포 스크립트 실행
└─ 백엔드: gradle bootJar → /srv/app/backend/app.jar
└─ 프론트: npm build → /srv/app/frontend/dist/
└─ (추가) 수집기: collector 코드 동작(실시간 주가 → 백엔드 전달)
└─ docker compose up -d backend nginx collector
|
▼
Nginx:80 ──▶ / (정적 배포물)
└─▶ /api/ → backend:8080 프록시
└─▶ /ws → backend:8080/ws (WebSocket)
|
▼
Backend(8080)
└─ REST: /api/*
└─ WS: /ws (STOMP/SockJS)
└─ 브로드캐스트: /topic/price 로 실시간 전송

Stage View

				Build & Deploy	Declarative: Post Actions
Average stage times: (full run time: ~37s)				35s	187ms
#48	9월 29 일 10:46	No Changes	🔍	32s	145ms
#47	9월 29 일 09:45	No Changes	🔍	31s	166ms
#46	9월 29 일 09:44	No Changes	🔍	31s	211ms

- 위의 사진처럼 모니터링의 중요성을 배움. 현재는 최소한으로 했지만 더 자세하고 고도화가 필요함을 느꼈음.
- Dockerfile: 각 컨테이너 빌드
- docker-compose.yml: 컨테이너 올리는 방법들 어떤 이미지로/ 포트/ 볼륨/ 환경변수 선택
- .env: 설정에 필요한 각 설정값들