평판 기반 거래 플랫폼

Jake Tae-moon Jo · 개발 리드 · https://jake-jo-portfolio.vercel.app/ko/work/reputation-market/

사용자 앱, 운영툴, API 서버를 처음부터 만들어 첫 커밋 13일 뒤 운영툴까지 갖췄고, 20일 동안 기능을 여덟 묶음으로 나눠 묶음마다 만들고 확인한 뒤 커밋했습니다(출시 전). 그동안 기획 질문 242건 중 238건의 답을 명세에 반영했습니다.

출시 전	238 / 242	25 / 25	2,583
20일 개발, 운영 환경 배포 전	기획 질문 중 답을 받은 수	UX 점검 1순위 문제 해결	자동 테스트, 재실행에서 모두 통과

결과

새 기획서로 시작해 첫 커밋 13일 뒤 운영툴까지 만들었고, 20일 뒤인 10월 30일에 마지막 커밋을 남겼습니다. 푸시 알림 연결은 보류했고, 기획 질문 4건은 답을 기다립니다.

고객의 수정 요청은 모두 24시간 안에 반영했습니다.

사용자 앱(Flutter, iOS·Android), 운영툴(Next.js), API 서버와 워커(NestJS)를 만들었습니다. 출시 전이고, 운영 환경은 구성 설계와 롤백 런북 초안까지 했습니다.

10월 10일 시작한 지 13일 뒤 운영툴까지 커밋했고, 20일 뒤인 10월 30일이 마지막 커밋입니다. 푸시 알림 연결은 보류했고, 기획 질문 4건은 답을 기다리며, DB가 필요한 백엔드 테스트는 11월 10일 재실행에 넣지 않았습니다.

고객사 기획 담당과 8차에 걸쳐 질문 242건을 주고받아 238건의 답을 명세에 반영했습니다. 4건은 답을 기다리는 중입니다.

실제 SMS 발송과 주소 검색, 두 외부 연동을 실 어댑터로 동작시켰고, 스테이징·운영 설정도 같은 실 드라이버를 쓰도록 했습니다(10월 25일).

UX 점검에서 찾은 1순위 문제 25건을 모두 고쳤습니다.

자동 테스트 2,583개가 11월 10일 재실행에서 모두 통과했습니다.

기간

2026.10 (20일)

역할

사람은 본인 1명 · AI 에이전트 팀 설계·운영

팀

본인 1명 + AI 에이전트 13개 역할

플랫폼

iOS·Android 앱, 웹 운영툴, API 서버·워커

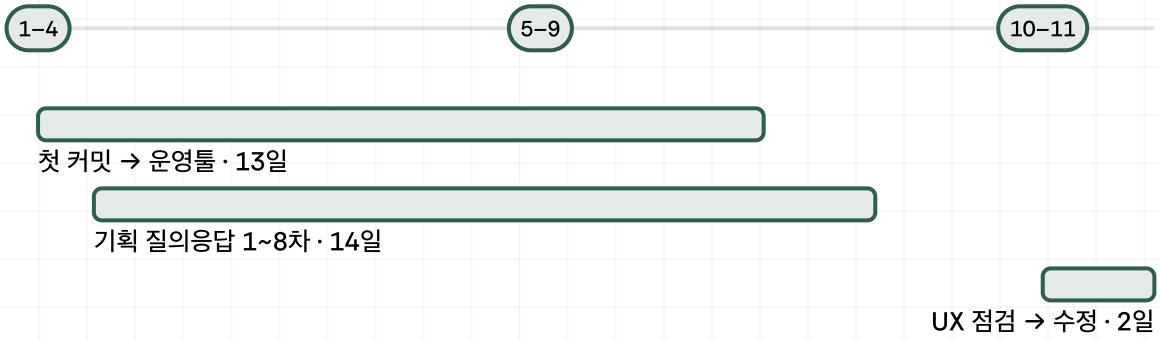
배경

출시를 준비하는 거래 플랫폼 회사. 기획서와 QA 체크리스트는 고객사 기획 담당이 씀

첫 커밋부터 20일

새 기획서로 시작한 지 13일 만에 운영툴까지 만들었고, 20일 동안 기획 질문 242건 중 238건의 답을 명세에 반영했다.

13일	20일	238 / 242
운영툴까지	개발 기간	답을 명세에 반영한 기획 질문



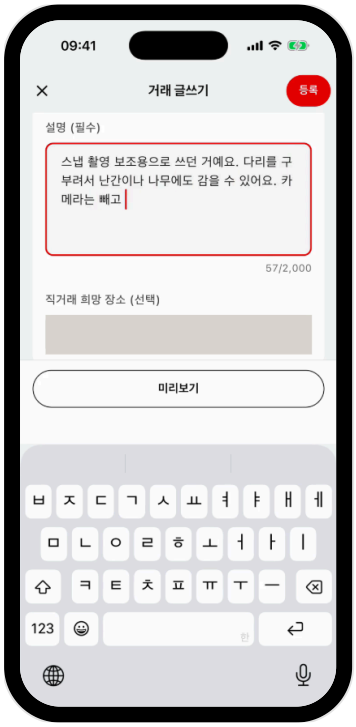
1. 2026년 10월 10일 — **착수**
   새 기획서 17종, 레포 4개, 앞선 프로젝트의 스택·에이전트 하네스
2. 2026년 10월 11일 — **운영 구조·데이터 모델**
   1차 기획 질문 109건, 구조 결정 기록 2건
3. 2026년 10월 14일 — **묶음 1~3: 인증·관계·피드**
   명세 v0.2, 푸시 큐 결정
4. 2026년 10월 16일 — **묶음 4: 거래·DM·신고**
   거래 확정을 한 트랜잭션으로
5. 2026년 10월 19일 — **묶음 5: 알림**
   기획서 전수 대조 반영(명세 v0.5~0.7)
6. 2026년 10월 21일 — **묶음 6: 티어·랭킹·MY**
7. 2026년 10월 23일 — **묶음 7: 운영툴**
   첫 커밋 13일 뒤, 명세 v0.8
8. 2026년 10월 24일 — **8차 질의 답변**
   질문 242건 중 238건 반영, 4건 대기
9. 2026년 10월 25일 — **실 외부 연동 2건**
   실 SMS·주소 검색. 스테이징은 만든 날 삭제(내부 배포 연기)

10	2026년 10월 28일	**UX 점검** 발견 110건, 1순위 25건
11	2026년 10월 30일	**마지막 커밋** 1순위 25건 해결

화면

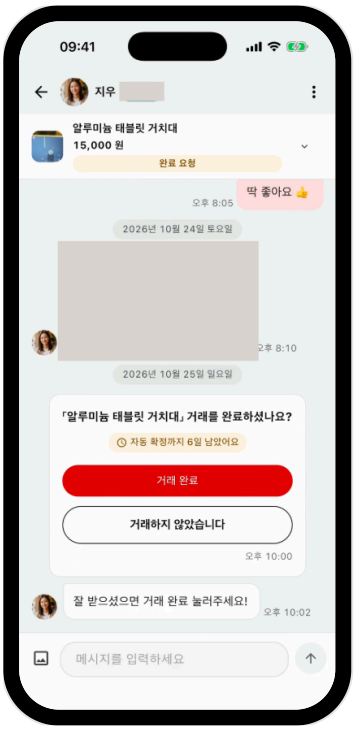
거래 글쓰기

사진, 제목, 가격, 설명을 적고 직거래 희망 장소는 선택으로 둔다. 가상 데모 시드이며 장소 입력칸은 가렸다.



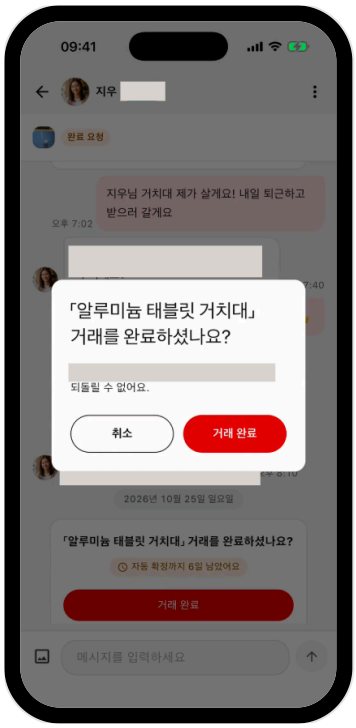
DM 안의 거래 바

상단에 거래 상태가 뜨고, 대화 중에 「거래 완료」와 「거래하지 않았습니다」를 고르는 카드가 오며 자동 확정까지 남은 날이 보인다. 관계 단계 표시는 가렸다.



「거래 완료」를 누르면 한 번 더 묻는다

확정 뒤에는 되돌릴 수 없다는 안내가 있다. 점수 안내 문장, 관계 단계 표시, 채팅 속 장소 이름은 가렸다.



문제와 배경

제품을 10월에 새 기획서와, 앞선 프로젝트와 같은 스택·에이전트 하네스로 처음부터 만드는 일이었습니다.

고객사는 관계망 기반 거래 플랫폼의 출시를 준비하는 회사입니다. 사용자는 서로 수락한 친구로 1단계 관계망을 만듭니다. 거래·연결·피드 게시글을 1단계 친구가 자기 친구들에게 「전달」하면, 글이 2·3단계 관계까지 퍼집니다. 거래 당사자는 DM으로 이어지고, 활동은 평판 점수로 쌓입니다.

앱은 당사자를 이어 줄 뿐입니다. 앱 안에서 결제·에스크로·배송을 하지 않고, 평판 점수는 사거나 넘기거나 돈으로 바꿀 수 없습니다. 고객사가 겨냥한 문제는 모르는 사람과 거래할 때 생기는 불신이었고, 답은 관계망으로 이어진 사람끼리만 거래하게 하는 것이었습니다.

10월 10일 처음부터 시작했습니다. 기준은 고객사 기획 담당이 새로 쓴 기획서 17종과 QA 체크리스트 1,450항목이었습니다. 제품은 1~3단계 관계를 시스템이 계산하고, 앱 안 결제가 없으며, 점수는 사거나 쓸 수 없는 평판 점수입니다. 스택은 앞선 프로젝트(급여 선지급 플랫폼)와 같은 NestJS·Next.js·Flutter입니다. 설계 문서에 남긴 이유는 「에이전트 하네스·운영 지식 공유」입니다.

거래 한 건이 지나가는 길

앱은 돈을 옮기지 않고 사람을 이어 줄 뿐이며, 거래가 확정되는 순간 거래 점수와 중개 몫이 한 트랜잭션으로 함께 기록된다.

① **글 작성** 〇사람
판매자가 거래 글을 쓴다

② **전달** 〇사람
1단계 친구가 자기 친구들에게 넘겨 2·3단계까지 퍼진다

③ **거래하기** 〇사람
누른 순간 전달 경로와 관계 단계를 고정, 글 행 잠금

④ **DM 연결** 〇사람
당사자만 DM으로 이어진다. 앱 안 결제·배송 없음

⑤ **완료 요청** 〇사람
판매자가 상대를 골라 요청

⑥ **확인** 〇사람
구매자 확인, 「거래하지 않았습니다」, 또는 7일 뒤 자동 확정(2일 전 예고)

⑦ **한 트랜잭션 확정** □시스템
완료 + 같은 글의 다른 거래 자동 취소 + 거래 점수 + 중개 점수(경로 참여자)

⑧ **집계 기간 점수 → 티어** □시스템
나보다 집계 기간 점수가 큰 활성 사용자 수로 티어를 정한다

제 역할

기획서는 고객사가 썼고, 저는 개발 전에 기획서의 빈틈을 질문으로 바꾸고 에이전트 팀과 구조 결정을 맡았습니다.

에이전트 팀은 앞선 프로젝트에서 옮겨 온 13개 역할입니다. 역할 이름은 그대로 두고, 정의 파일은 이 프로젝트에 맞게 하나하나 고쳤습니다. 파이프라인도 같은 8단계(접수 → 아키텍트 → 디자인 → 개발 → QA → 리뷰 → 통합 → 문서화)입니다. git 작성자는 저 한 사람이고, 커밋 226개 모두에 AI 공동 작성 표시가 있습니다. 세션 기록으로 센 에이전트 호출은 275회입니다. 오케스트레이터는 메인 세션이 맡았습니다.

코드보다 질문을 먼저 했습니다. 10월 11일, 개발 전에 기획서의 모호한 곳과 서로 어긋나는 곳을 질문 목록으로 만들기로 정했고, 1차로 109건을 보냈습니다. 개발하면서 일곱 차례 더 물어 질문은 모두 242건이 됐고(2차의 기획서 정정 11건은 별도), 그중 238건의 답을 명세에 반영했습니다. 답이 오면 해당 스토리의 수용 기준을 고치고 확정 표시를 단 뒤 명세 버전을 올렸습니다. 기획서끼리 어긋나면 규칙 문서 > 화면 문서 > 공통 UX > 개요 순으로 따랐습니다. 명세는 스토리 207개(Must 197 · Won't 10)이고, QA 체크리스트 1,450항목을 스토리에 연결했습니다. 기획서, QA 체크리스트, 질문의 답은 고객사 기획 담당의 몫입니다.

구조와 운영의 큰 결정은 제가 내렸습니다. 10월 10일 스택·에이전트·레포 구성을 앞선 프로젝트와 같게 하고, 배포 대상을 AWS로 정했습니다. 10월 11일에는 운영을 트래픽에 따라 늘어나는 서버와 공용 DB·Redis로, 개발·스테이징을 서버 1대 도커로 정했습니다. 10월 14일 푸시가 몰릴 때를 대비해 메시지 큐를 요구했고, 10월 23일 광고성 푸시의 야간 금지 시간을 설정값(기본 21:00~08:00)으로 두게 했습니다. 10월 25일에는 앱 버전 규칙과 강제 업데이트, 배포 노트 자동 생성을 정했습니다.

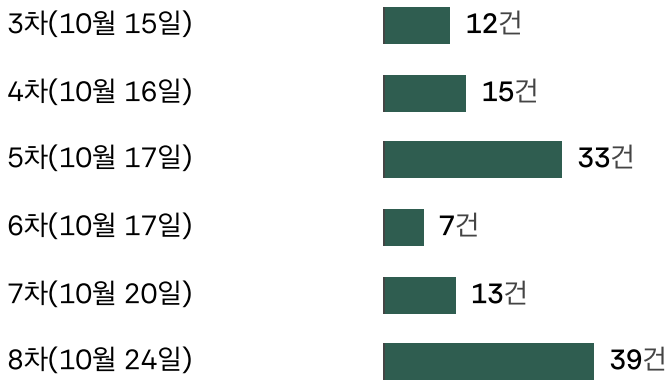
일하는 방식도 정했습니다. 10월 19일, 개발자가 아닌 사람도 진행 상황과 일정을 가늠할 수 있게 1쪽짜리 현황 보고서를 요구했습니다. 10월 26일에는 화면 수정을 「고침 → 확인 캡처 → 독립 QA」로 반복하고, 명세와 구현이 다르면 명세를 따르며, 3회를 넘기면 저에게 묻게 했습니다. 10월 25~27일에는 시연용 시드 데이터(가상 사용자 45명의 약 4개월 활동을 서비스 코드로 재생)와 Figma 디자인 시스템·화면을 만들었습니다. 원본 디자인은 고객사 쪽에서 만든 것입니다. 10월 29일에는 UX 점검에서 나온 기획 질문 19건 중 18건의 권장안을 채택하고 1건을 보류했습니다. 결정 카드의 번호는 설계 문서의 결정 번호(D10)와 ADR 번호입니다.

기획 질의 8차

개발 전에 모호·충돌 질문 109건을 먼저 냈고, 나머지 133건은 개발하며 일곱 차례에 나눠 물었다.

출처: 차수별 질문 목록 파일의 질문 ID 수. 2차의 기획서 정정 11건은 뺐다. 8차 39건 중 35건 답변, 4건 대기.





누가 무엇을 했나

방향과 운영 결정은 사람이 내렸고, 커밋 226개 모두에 AI 공동 작성 표시가 있으며, 검토는 보안·코드 리뷰어 에이전트가 41번 맡았다. CI와 PR은 없었다.

본인

스택·에이전트·레포 구성을 앞선 프로젝트와 같게(10월 10일)
운영 구조: 트래픽 따라 늘어나는 서버 + 공용 DB·Redis, 개발·스테이징은 서버 1대(10월 11일)
개발 전에 기획서의 모호·충돌 질문부터(10월 11일)
푸시가 몰릴 때를 대비한 메시지 큐(10월 14일)
스테이징을 최소 비용으로 만들고 같은 날 삭제(10월 25일)
UX 점검 기획 질문 19건 중 18건 권장안 채택(10월 29일)

에이전트 13개 역할

앞선 프로젝트와 같은 이름의 13개 역할, 정의는 파일마다 고쳐 씀
호출 275회(기본 제공 범용·탐색 67 포함): 앱 59 · 백엔드 51 · 프런트엔드 17 · 문서 15 · 디자이너 13 · 아키텍트 9 등
커밋 226개 전부 AI 공동 작성 표시

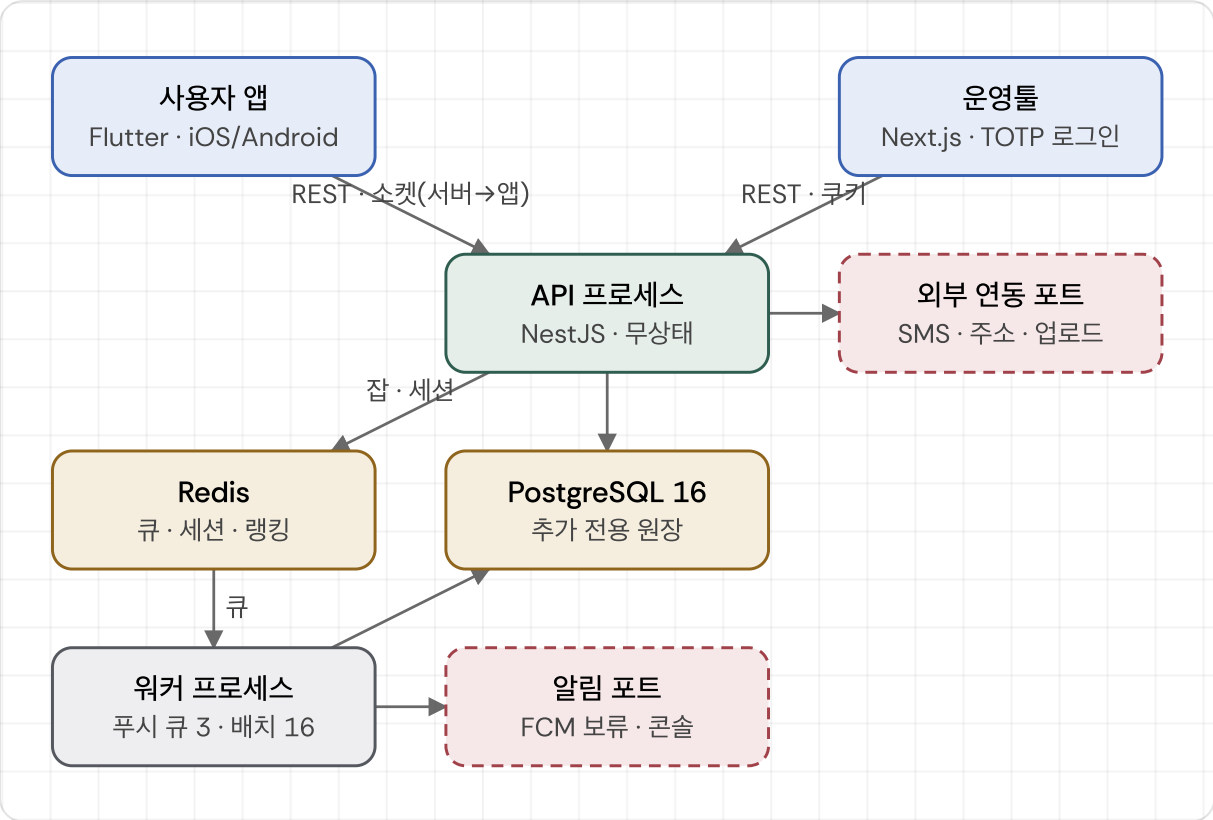
리뷰 게이트

보안 리뷰어 16회 · 코드 리뷰어 25회 · 명세 범위 감사 1회(10월 14~29일)
ADR에 리뷰 지적 인용 25회, 알려진 한계 22개 기록
force push, reset --hard, rm -rf, .env 읽기 차단
CI·PR 없음

시스템 구조

시스템 구성

API는 상태를 Redis에 두어 몇 대로든 늘릴 수 있고, 푸시와 정기 배치는 큐를 거쳐 워커 프로세스에서만 돈다.



사용자 앱

Flutter·Riverpod. 보내기는 REST, 받기는 socket.io(서버 → 앱 알림 전용). 앱 모델은 손으로 쓰고 OpenAPI 스냅샷과 대조한다. 연락처는 단방향 해시로만 맞춘다.

운영툴

Next.js. ID·비밀번호·TOTP로 로그인하면 Redis 세션 쿠키를 받는다. 타입은 백엔드 OpenAPI에서 생성한다.

API 프로세스

NestJS 모듈 42개. 세션·레이트 리밋·멱등 키를 Redis에 두어 몇 대를 띄워도 상태가 없다. 기능 모듈은 외부 SDK를 직접 부르지 않고 포트를 거친다. 트래픽에 따라 늘어나는 운영 배치는 설계만 있고 구축하지 않았다.

외부 연동 포트

포트 7개 중 실 어댑터 3개(SMS, 주소 검색, LLM 배포 노트), 나머지는 목·콘솔·로컬 디스크. 개발 환경의 외부 호출은 0건이고 운영 설정은 개발용 어댑터를 거부한다.

Redis

BullMQ 브로커, 세션·레이트 리밋, socket.io 어댑터, 티어·랭킹 스냅샷. 큐가 있어 메모리 축출을 끈다. 그래서 큰 캐시는 DB에 둔다.

PostgreSQL 16

모델 41개. 원장은 UPDATE·DELETE를 트리거가 거부하고, 이벤트마다 한 번만 들어가며, 끝난 집계 기간에는 더할 수 없다. 2·3단계 관계는 기준 사용자당 배열 한 행으로 캐시한다.

워커 프로세스

같은 코드의 별도 프로세스. 푸시 큐 3종(즉시·5분 묶음·500토큰 일괄)과 반복 배치 16종을 Redis에 저장된 일정으로 돌려, 인스턴스가 몇 대든 한 일정을 나눠 쓴다. API 안에는 cron이 없다.

알림 포트

스토어 개발자 계정이 늦어 FCM 연결을 미뤘다. 그동안 알림 포트는 콘솔 어댑터로 돈다.

`src/trade/trade.service.ts` · 모든 거래 트랜잭션은 사용자 → 게시글 → 거래 순으로 잠그고, 확정·형제 거래 취소·점수 지급을 한 트랜잭션에 묶는다. 구매자와 자동 확정 배치가 겹치면 두 번째는 이미 확정된 거래를 보고 멈춘다. (주석은 원문을 줄여 옮김, 일부 줄 생략, 이름은 바꿈)

```
  await runTx(this.prisma, async (tx) => {
    // Lock order of every trade transaction (ADR 0005 §4.2):
    // users (id order) → posts → trades → dm_rooms → the rest.
    const payees = [head.buyerId, head.sellerId, ...plan.relayers];
    await lockParties(tx, payees);
    await lockPost(tx, head.postId);
    const t = await lockTrade(tx, tradeId);
    // Buyer and auto-confirm batch race: the second finds COMPLETED.
    if (t.status === 'COMPLETED') return;
    // …
    await writeTrade(tx, t.id, 'COMPLETE_REQUESTED', {
      status: 'COMPLETED', confirmedAt: now,
      confirmMethod: auto ? 'AUTO_CONFIRM' : 'BUYER_CONFIRM',
    });
    // One completion request per post: the others are in progress only.
    const siblings = await tx.trade.updateManyAndReturn({
      where: {
        postId: t.postId, id: { not: t.id }, status: 'IN_PROGRESS',
      },
      data: {
        status: 'CANCELLED', cancelReason: 'OTHER_TRADE_COMPLETED',
      },
    });
    await this.points.pay(tx, t, plan, now); // trade + broker points
    await rederivePost(tx, t.postId);
    // … DM system cards for the trade and every cancelled sibling
  }); // any failure rolls all of it back
```

`prisma/migrations/…_init_schema/migration.sql` · 점수 원장의 UPDATE·DELETE는 트리거가 거부하고, 새 줄은 그 시각을 포함하는 끝나지 않은 집계 기간에만 들어간다. 이 밖에 (사용자, 출처 키) 유일 제약으로 이벤트마다 한 번만 들어가고, 집계 기간끼리는 배제 제약으로 겹치지 못하며, 점수 캐시는 CHECK로 0 밑으로 내려가지 않는다. (첫 마이그레이션의 트리거 두 개, 따옴표와 이름은 줄임)

```
-- The points ledger is append-only: no UPDATE or DELETE, ever.
CREATE FUNCTION forbid_update_delete() RETURNS trigger
LANGUAGE plpgsql AS $$
BEGIN
  RAISE EXCEPTION '% is append-only (% rejected)', TG_TABLE_NAME, TG_OP
    USING ERRCODE = 'restrict_violation';
END $$;
CREATE TRIGGER ledger_append_only
  BEFORE UPDATE OR DELETE ON ledger_entries
  FOR EACH ROW EXECUTE FUNCTION forbid_update_delete();
-- …
-- A row goes only to the open season whose range holds created_at.
CREATE FUNCTION ledger_check_season() RETURNS trigger
LANGUAGE plpgsql AS $$
BEGIN
  IF NOT EXISTS (
    SELECT 1 FROM seasons s
    WHERE s.id = NEW.season_id
      AND NEW.created_at >= s.starts_at AND NEW.created_at < s.ends_at
      AND s.finalized_at IS NULL
  ) THEN
    RAISE EXCEPTION 'season % does not cover % or is finalized',
      NEW.season_id, NEW.created_at USING ERRCODE = 'check_violation';
  END IF;
  RETURN NEW;
END $$;
CREATE TRIGGER ledger_season_chk BEFORE INSERT ON ledger_entries
  FOR EACH ROW EXECUTE FUNCTION ledger_check_season();
```

핵심 결정

다섯 결정 중 DM과 푸시 큐 두 건은 제 질문과 요청에서 시작했고, 자동 확정 배치는 고객사 기획 담당의 답으로 고쳤으며, 관계 캐시와 티어 계산은 에이전트의 추천 기본값입니다.

DM은 직접 구현하고 원본은 Postgres에 D10 본인 입력

문제

거래 당사자는 DM에서 만나고, 거래 상태가 바뀔 때마다 DM 안에 시스템 메시지가 남아야 했습니다. 10월 11일 저는 채팅을 직접 구현해 운영하는 것보다 외부 API가 더 안정적이고 운영 비용에서도 유리하지 않은지 물었습니다. 직접 만든다면 스택에 메시지 큐가 더해져야 한다는 점도 짚었습니다.

검토한 선택지

외부 채팅 SaaS | 관리형 실시간 서비스(API Gateway WebSocket, AppSync, 관리형 pub/sub) | ✓ 직접 구현: 보내기는 REST, 소켓은 서버 → 앱 알림만

결정

직접 구현을 골랐습니다. 필요한 것은 1:1 텍스트와 이미지뿐이었고, 거래 시스템 메시지는 거래 트랜잭션과 함께 써야 했습니다. 외부 SaaS는 예상 사용자 규모에 따라 월 비용이 붙었습니다. 방마다 순번(seq)을 두고 방 행을 잠가 커밋 순서를 정했기 때문에, 읽음·안 읽음·놓친 메시지 따라잡기를 정수 하나로 처리합니다. 보내기는 REST로 하고, socket.io는 서버가 앱에 알리는 데만 씁니다. 소켓으로 보내기와 long-polling도 검토해 기각했습니다. 소켓으로 보내면 검증·멱등·오류 처리를 다시 만들어야 했고, long-polling에는 sticky session 문제가 있었습니다. DM 푸시는 알림 목록에 보이지 않는 알림 행으로 만들어, 다른 알림과 같은 아웃박스를 탑니다.

결과

거래 확정 트랜잭션이 DM 시스템 카드까지 함께 씁니다. 확정이 실패하면 카드도 남지 않습니다. 메시지 원본은 PostgreSQL에 있고, 앱이 꺼져 있으면 푸시로 알리게 했습니다(FCM 연결은 보류).

대가·남은 위험

같은 방 안의 보내기는 방 행 잠금 때문에 차례로 처리됩니다. 두 사람의 대화라 문제가 없다고 ADR에 적었습니다.

푸시는 알림 행을 아웃박스로, 이미 쓰는 Redis 위 BullMQ 큐 3종으로 ADR-0003 본인 입력

문제

집계 기간이 끝날 때나 야간에 보류한 광고성 푸시를 아침에 풀 때처럼, 알림은 한꺼번에 몰릴 수 있습니다. 10월 14일 저는 설계 단계에서 푸시가 몰릴 상황에 대비해 메시지 큐를 쓰자고 요구했습니다.

검토한 선택지

Kafka·RabbitMQ·SQS 같은 별도 브로커 | SQS + Lambda 또는 SNS 모바일 푸시 | API에서 FCM을 바로 부르고 재시도 | ✓ BullMQ + 이미 쓰는 Redis, 알림 행 아웃박스 + 큐 3종

결정

BullMQ와 이미 쓰는 Redis를 골랐습니다. SQS 계열은 AWS 서비스와 운영 지점을 늘리고, 5분 묶음·지연·속도 제한을 직접 만들어야 했습니다. Kafka는 스트리밍 파이프라인이 생기기 전에는 과하다고 봤습니다. API에서 바로 부르면 몰릴 때 응답이 늦어지고 쿼터가 바닥나는 것을 막지 못합니다. 그래서 알림 행 자체를 아웃박스로 삼았습니다. 기능 코드는 NotificationService.create(tx, …)만 부르고, 워커가 즉시·5분 묶음·일괄(500토큰 단위) 큐 3종에 나눠 큐마다 속도 제한을 걸어 보냅니다. 야간 보류를 푸는 08:00 발송과 집계 기간 종료 일괄 발송은 10분에 나눠 보냅니다.

결과

API 프로세스 안에는 cron이 하나도 없습니다. 반복 배치 16종도 워커가 BullMQ job scheduler로 돌리고, 일정이 Redis에 저장돼 워커를 여러 대 띄워도 한 일정을 나눠 씁니다. 실제 FCM 연결은 스토어 개발자 계정이 늦어 미뤘고, 그동안 알림 포트는 콘솔 어댑터로 돕니다.

대가·남은 위험

Redis가 큐를 맡으므로 메모리 축출을 꺼야 해서(noeviction), 큰 캐시는 Redis에 둘 수 없습니다(다음 카드). 사용자가 100만 명을 넘거나, 서비스를 나누거나, 다른 채널이 같은 파이프라인을 쓰게 되면 다시 보기로 적었습니다.

2·3단계 관계 캐시는 기준 사용자당 배열 한 행 ADR-0002 에이전트 기본값 · 확정 기록 없음

문제

피드는 2·3단계 관계까지 보여 주므로, 관계 하나가 바뀌면 수백에서 수천 명의 캐시가 영향을 받습니다. 피드 정렬은 게시글 테이블과의 SQL 조인 한 번으로 끝나야 했습니다.

검토한 선택지

Redis 캐시 | (기준, 상대) 쌍마다 한 행 | ✓ 기준 사용자당 배열 한 행

결정

기준 사용자당 배열 한 행을 골랐습니다. Redis에 두면 피드 정렬을 게시글과 한 번에 조인할 수 없고, 큐 때문에 메모리 축출을 끈 공용 Redis에는 GB 단위 캐시를 둘 수 없었습니다. 쌍 행은 사용자 1만 명을 가정해도 3천만~1억 행으로 추정됐고, 관계 1건이 바뀔 때 수백만~수천만 행을 지워야 했습니다. 배열이면 영향받는 기준 행 수천 개의 computed_at을 NULL로 바꾸는 것으로 끝납니다. 무효화는 관계 변경과 같은 트랜잭션에서 하고, 교착을 막으려고 id 순서로 upsert합니다. 다시 계산은 읽을 때 합니다. 이미 다시 계산 중이면 300ms까지 기다린 뒤 이전 배열을 쓰고, 작업을 큐에 넣습니다.

결과

관계가 바뀌는 트랜잭션은 영향받는 행에 표시만 남깁니다. 무거운 계산은 그 사용자가 다음에 피드를 읽을 때 일어납니다.

대가·남은 위험

읽는 쪽이 다시 계산을 최대 300ms까지 기다릴 수 있습니다. 가장 촘촘한 관계망에서 응답 p95가 목표를 넘으면 완전한 stale-while-revalidate로 바꾸겠다고 천장을 적어 두었습니다.

자동 확정은 5분 배치, 확정은 한 트랜잭션 ADR-0005 기본값 → 이해관계자 답으로 수정

문제

거래를 확정하면 거래 완료, 같은 글의 다른 거래 자동 취소, 거래·중개 점수, 기여자 기록, DM 시스템 카드가 함께 바뀝니다. 구매자의 확인과 7일 자동 확정 배치가 같은 거래에 동시에 닿을 수도 있었습니다. 일부만 반영되면 점수가 거래와 어긋납니다.

검토한 선택지

자동 확정을 거래마다 BullMQ job으로 예약 ✓ 5분마다 도는 배치가 기한이 지난 거래를 찾아 확정

결정

자동 확정은 5분 배치로 했습니다. 거래별 job은 큐와 DB 두 곳의 상태를 맞춰야 하지만, 배치는 여러 번 돌아도 결과가 같습니다. 모든 거래 트랜잭션은 사용자 → 게시글 → 거래 → DM 방 순서로 잠급니다. 확정은 이 모두를 한 트랜잭션에서 하고, 하나라도 실패하면 전부 되돌립니다. 중개 경로는 구매자의 1단계 전달에서 판매자 쪽으로 재귀 CTE로 거슬러 오르고, 전달 시각이 계속 줄어드는 조건으로 순환을 끊습니다. 일일 제한을 카운터 표나 Redis 카운터로 세는 안은 원장과 어긋날 수 있어서 기각했습니다. 사용자별 advisory lock도 사용자 행 잠금과 같은 직렬화라서 쓰지 않았습니다.

결과

10월 19일 기획 질의(Q6-01)의 답에 따라, 중개 경로와 관계 단계를 확정보다 앞선, 「거래하기」를 누른 시점에 고정했습니다. 거래 중에 관계를 바꿔 점수 총량을 조작하거나 확정 직전에 끼어드는 것을 막기 위해서입니다. 10월 23일에는 운영자가 어뷰징으로 판정하면 거래 점수를 회수하는 경로를 제재와 함께 넣었습니다.

대가·남은 위험

같은 밀리초에 생긴 전달끼리는 시각만으로 순환을 끊을 수 없습니다. 그래서 방문 집합과 깊이 50을 안전망으로 둡니다.

티어는 「나보다 점수가 큰 사람 수」로 ADR-0006 에이전트 기본값 · 확정 기록 없음

문제

티어는 집계 기간 점수의 분포로 정합니다. 집계 기간 초반처럼 활성 사용자가 적을 때도 티어가 말이 되어야 했고, 점수가 바뀌면 경계도 따라 움직입니다.

검토한 선택지

순위 백분위: (순위)/n ≤ 경계 사용자 행에 티어를 저장 ✓ 나보다 집계 기간 점수가 큰 활성 사용자 수로 경계 비교

결정

세 번째를 골랐습니다. 순위 백분위는 활성 사용자가 1명이면 그 사람이 가장 낮은 티어가 되고, 100명 미만이면 가장 높은 티어가 비는 콜드 스타트 문제가 있었습니다. 티어를 사용자 행에 저장하면 경계가 1분마다 움직일 때마다 전원을 UPDATE해야 했습니다. 동점은 위 티어로 올리고, 1위는 항상 가장 높은 티어에 둡니다. 워커가 60초마다 분포와 랭킹을 Redis 스냅샷으로 통째로 바꿔 넣습니다.

결과

활성 사용자가 몇 명이든 1위는 가장 높은 티어에 있고, 같은 점수는 같은 티어에 듭니다. 사용자 행을 고치지 않고 스냅샷 하나만 바꿉니다.

대가·남은 위험

경계 근처에서는 티어가 최대 1분 늦게 바뀔 수 있습니다.

알림 한 건이 나가는 길

알림은 업무 데이터와 같은 트랜잭션에 행으로 남고, API와 따로 도는 워커 프로세스가 큐와 속도 제한을 거쳐 보낸다.

① **업무 이벤트** 시스템
기능 코드는 알림 서비스의 create(tx, …)만 부른다

② **알림 행 = 아웃박스** 시스템
업무 데이터와 같은 트랜잭션. DM 푸시도 목록에 안 보이는 행으로 같은 길

③ **큐 고르기** 시스템
즉시 · 5분 묶음 · 일괄(500토큰 단위)

④ **속도 제한·야간 보류** 시스템
큐별 속도 제한. 광고성 푸시는 설정한 야간(기본 21:00~08:00)에 보류, 08:00 해제는 10분에 나눠 보낸다

⑤ **워커 → 알림 포트** 외부
스토어 계정 대기로 FCM 보류, 그동안 콘솔 어댑터

품질과 검증

UX 점검에서 찾은 1순위 문제 25건은 실패하는 테스트를 먼저 쓰고 모두 고쳤고, 자동 테스트 2,583개는 11월 10일 다시 돌려 모두 통과했습니다.

자동 테스트 2,583개를 11월 10일 다시 돌려 모두 통과했습니다. 백엔드 단위 991개(Jest 스위트 100개), 운영툴 단위 359개(Vitest 파일 45개), 앱 위젯·단위 1,233개(Flutter 파일 130개)입니다. 앱은 레포에 추적되는 테스트 파일만 셌습니다. 같은 세 수치가 10월 29일 UX 점검 보고서에도 통과로 기록돼 있습니다.

e2e는 11월 10일 재실행에 넣지 않았습니다. 백엔드 e2e 스펙 54개 파일(정적으로 센 it 806개)은 Postgres와 Redis가 필요합니다. 앱 통합 테스트 1개 파일은 화면 갤러리 캡처용입니다.

리뷰는 에이전트가 맡았습니다. 10월 14~29일에 보안 리뷰어를 16회, 코드 리뷰어를 25회 돌렸고, 10월 19일에 명세 범위 감사를 1회 돌렸습니다. API 계약 검사기는 한 번도 부르지 않았습니다. ADR에는 리뷰 지적을 25번 인용했고, 알려진 한계 22개를 적었습니다. CI와 PR은 없었고, 병합 커밋도 0개입니다.

되돌리기 어려운 명령은 설정으로 막았습니다. 권한 설정은 앞선 프로젝트와 글자 그대로 같아서, force push, reset --hard, rm -rf, .env 읽기를 막습니다. 에이전트가 DB를

초기화하는 명령(migrate reset)은 Prisma의 AI 안전 가드가 막는다고 README에 운영 메모로 남겼습니다.

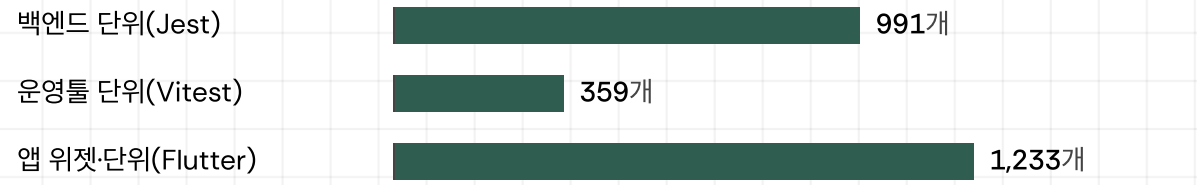
외부 연동은 모두 포트 7개를 거칩니다. 실 어댑터는 SMS, 주소 검색, LLM 배포 노트 3개이고, 나머지는 목·콘솔·로컬 디스크입니다. 개발 환경의 외부 호출은 0건이고, 운영 설정은 개발용 어댑터를 거부합니다.

10월 28일 UX 점검에서는 기존 위젯 테스트를 7개 화면 조건(작은 폭, 큰 글자, 키보드)으로 다시 돌려 넘침을 모았습니다. 110건을 찾아 1순위 25건, 기획 확인 19건 등으로 나눴습니다. 항목마다 실패하는 테스트를 먼저 썼습니다. 1순위 25건을 모두 고쳤고, 큰 글자에서 넘치던 33곳은 키보드가 없는 6개 조건에서 0곳이 됐습니다. 수정에 대한 코드 리뷰 지적 23건은 고침 14건, 버그 아님 6건, 범위 밖 3건으로 정리했습니다.

다시 돌린 자동 테스트

세 코드베이스의 단위·위젯 테스트 2,583개를 다시 돌려 모두 통과했고, DB가 필요한 백엔드 e2e는 11월 10일 재실행에 넣지 않았다.

출처: 2026-11-10 실행, 전부 통과. 앱은 레포에 추적되는 테스트 파일만. 백엔드 e2e 54개 파일은 Postgres·Redis가 필요해 재실행에 넣지 않았다.



기능 하나, 처음부터 끝까지

스토리 하나가 기획서에서 테스트까지 어떻게 이어지는지 보여 드립니다. 거래 확정과 중개 점수입니다.

1. **기획 — 기획서 → 명세 스토리 (10월 10일)**

   기획서의 거래 프로세스·점수 규칙과 상태 전이를 명세의 거래 스토리(USR-TRD-01~07, SYS-TRD-01~06)와 중개 점수 스토리 2개로 옮겼습니다.

2. **질의 — 여섯 항목 확정 (10월 13~24일)**

   중개 점수는 전달 경로에 참여한 사람들에게 나누기로 했습니다(Q-005). 남용을 막으려고 같은 쌍의 대기 기간, 같은 중개자–판매자의 집계 기간 횟수, 집계 기간 상한을 뒀습니다(Q5-01). 일일 제한을 따로 세고(Q5-08), 경로의 시작점을 정했습니다(Q5-09). 경로와 관계 단계를 「거래하기」 시점에 고정하고(Q6-01), 탈퇴한 사람의 몫을 어떻게 할지도 확정했습니다(Q7-06).

3. **설계 — ADR 0005 (10월 16일)**

   거래 상태 기계, 한 트랜잭션 확정, 순환에 안전한 중개 경로, 잠금 순서, 5분 배치 자동 확정을 정했습니다.

4 **백엔드 (10월 16일)**

상태 기계, 「거래하기」의 게시글 행 잠금, 완료 요청 제한, 한 트랜잭션 확정, 같은 글의 다른 거래 자동 취소, 보는 사람마다 다른 취소 사유, 배치 3종(자동 확정, 2일 전 예고, 3일 리마인드)을 한 커밋에 넣었습니다. 거래 e2e 스펙 4개도 함께 들어갔습니다.

5 **앱 (10월 16일)**

「거래하기」 버튼, 판매자 하단 바와 상대 선택 시트, DM 거래 바의 완료 요청·확인·거절·취소를 만들었습니다.

6 **규칙 변경 (10월 19일)**

질의 답에 따라 경로·관계 단계 고정, 쌍 대기 기간, 같은 판매자 집계 기간 최대, 중개 집계 기간 상한, 일일 제한 분리를 반영했습니다. 앱에는 거절 확인 대화상자와, 남은 횟수를 보여 주는 판매자 전용 거절 카드를 더했습니다.

7 **운영자 회수 (10월 23일)**

운영자가 어뷰징으로 판정하면 거래 점수를 회수하고, 이를 제재와 묶었습니다.

8 **UX 점검 (10월 28~30일)**

구매자가 거래를 취소할 유일한 길이 숨은 스와이프에 있었습니다. 펼친 거래 바에 「거래 취소」 버튼을 드러내는 권장안을 제가 채택했고(UXQ-07), 10월 30일 커밋했습니다.

9 **테스트 (11월 10일)**

거래를 포함한 백엔드 단위 테스트 991개가 모두 통과했습니다. 거래 e2e 스펙 4개는 있지만 11월 10일 재실행에는 넣지 않았습니다.

회고

세션 임시 폴더가 두 번 지워진 뒤 오래 걸리는 작업의 파일 위치를 바꿨고, CI는 아직 할 일로 남아 있습니다.

세션 임시 폴더가 두 번 지워졌습니다(10월 초, 10월 27일). 두 번째에는 Figma 화면 작업에 쓰던 참조 이미지와 목록, 규칙을 잃었습니다. 그 뒤로 여러 시간 걸리는 작업의 파일은 레포 안의 git 제외 폴더에 둡니다.

커밋 날짜는 작업 날짜가 아닙니다. 「요청할 때만 커밋한다」는 규칙 때문에 단계가 끝나면 기능별로 나눠 한꺼번에 커밋했고, 10월 14일 하루에 55건이 몰렸습니다. 이 페이지의 날짜는 커밋 기준이라 실제 작업보다 늦을 수 있습니다.

스테이징은 10월 25일 가장 싼 구성으로 만들었다가, 내부 배포를 미루면서 같은 날 지웠습니다. 예산 알림과 쿼터 요청은 남겨 두었습니다. 만드는 과정에서 VM 시작 스크립트가 POSIX sh로 돌아, bash 전용 옵션이 첫 부팅을 깨뜨리는 것을 찾았습니다.

개발 환경의 함정도 둘 있었습니다. Docker로 띄운 운영툴은 빌드 인자를 넘기지 않아 API 주소가 빈 값으로 굳어 로그인이 되지 않았고, compose에 빌드 인자를 넣어 고쳤습니다(10월 26일). 목 SMS는 인증번호를 로그에 남기지 않고 Redis에는 해시만 있어, 사람이 인증번호를 볼 방법이 없었습니다. 개발 환경과 목 SMS에서만 쓰는 고정 인증번호를 두었습니다.

CI는 아직 없습니다. 네 레포 모두 워크플로 파일이 없고, 할 일 목록의 CI/CD 단계는 시작하지 않았습니다. 지금은 로컬 전용인 UX 점검 하네스를 CI로 옮기는 것도 후속 작업으로 남았습니다.

일하는 방식

이 프로젝트도 역할별 AI 에이전트 팀으로 만들었습니다. 팀 구성과 게이트는 프로젝트마다 다르고, 이 프로젝트의 것은 위 「제 역할」에 적었습니다. 아래는 EWA 플랫폼을 만들며 정리한 방식입니다.

2026년 EWA 플랫폼을 만들며 세우고 운영한 방식 보기 →

https://jake-jo-portfolio.vercel.app/ko/process/

연락처

whxoans@gmail.com
+82 10-9233-7440