베트남 급여 선지급(EWA) 핀테크 플랫폼

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

근로자 앱, 운영 CMS, 고용주·파트너 포털을 5개월 동안 처음부터 만들고, 운영 중이던 v1의 데이터를 v2 운영 서버로 옮겼습니다.

운영 데이터 이전	5개월	대사 일치	7,080
v1 → v2, 11월 7일	첫 커밋 → 이전 실행	옮긴 건수·금액을 변환 기대값과 대조	단언 테스트, 실행 기록 기준

결과

5개월 만에 v1의 운영 데이터를 v2로 옮겼고, 적재한 건수·금액이 변환 기대값과 맞는지 보는 대사가 일치했습니다.

고객의 수정 요청은 모두 24시간 안에 반영했습니다.

근로자 앱(iOS·Android), 운영사 CMS, 고용주·영업 파트너 포털을 3개 언어로 만들었습니다. API와 웹은 스테이징과 운영 서버에서 돌고, 앱은 운영 서버를 가리키는 빌드까지 만들었습니다.

11월 7일 v1의 운영 데이터를 v2로 옮겼고, 그 뒤 v1에 들어온 거래를 11월 8일에 더 옮긴 뒤에도 대사가 일치했습니다(대사 범위는 「v1 → v2 데이터 이전」 결정 카드). v1 앱에서 오는 요청에는 업데이트 안내를 돌려줘 v1 앱을 은퇴시켰습니다.

결제사 2곳과 연동했습니다. 8월 23일 샌드박스에서 첫 이체를 끝까지 성공시켰고, 10월 30일 운영 환경 연결과 정상 상태를 확인했습니다. 한 곳이 멈춰도 출금이 이어지게 장애 전환을 넣었고, 결과가 모호하면 그 결제사에 남겨 두 곳이 겹쳐 지급하지 않게 설계했습니다.

단언이 있는 자동 테스트 7,080개가 실행 기록과 함께 있습니다. 백엔드 단위·웹·앱 테스트는 11월 10일에 모두 통과했고, 실제 DB로 API를 끝까지 호출하는 백엔드 테스트는 11월 3일 전체 실행에서 1,508개 중 1개만 실패했습니다(다시 돌려 통과).

기간

2026.06 – 현재

역할

사람은 본인 1명 · AI 에이전트 팀 설계·운영

팀

본인 1명 + AI 에이전트 13개 역할

플랫폼

iOS·Android 앱, 웹 CMS·포털, API 서버 (3개 언어)

배경

베트남의 급여 선지급 서비스 운영사. 다른 개발자가 만든 v1이 이미 운영 중

첫 커밋에서 운영 이전까지

명세와 돈 모델은 첫 사흘에 정했고, 9월 첫 서버 릴리스에서 11월 운영 데이터 이전까지 두 달이 걸렸다.

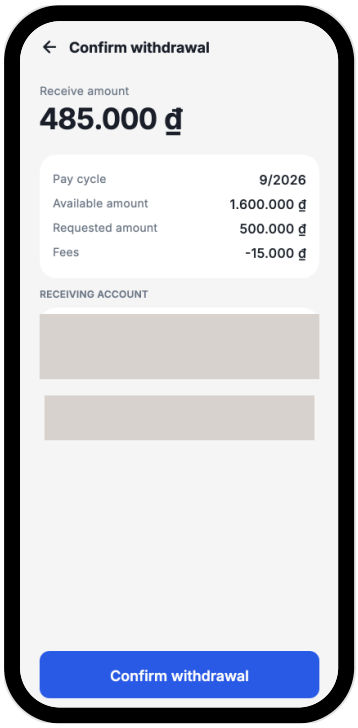


1	2026년 6월 13일	**명세 v0.2와 첫 ADR** V1·V2 기획을 합친 명세
2	2026년 6월 14일	**레포 4개, 에이전트 팀 가동** 13개 역할, 8단계 파이프라인
3	2026년 6월 15일	**돈 데이터 모델** 정수 금액, 추가 전용 원장, 출금 트랜잭션
4	2026년 7월 26일	**스테이징 환경** VPS + Caddy
5	2026년 8월 11일	**결제사 2곳 실 어댑터** 8월 23일 샌드박스 이체 끝까지 성공
6	2026년 9월 13일	**결제사 장애 전환, 첫 서버 릴리스 0.0.2** 같은 날 앱 강제 업데이트 규칙
7	2026년 9월 30일	**개인정보 암호화, v1 이전 도구** 실제 v1 데이터로 첫 리허설(롤백)
8	2026년 10월 14일	**운영 서버 구성**
9	2026년 11월 7일	**v1 → v2 운영 데이터 이전** 같은 날 오전 롤백 리허설 후 실행
10	2026년 11월 8일	**추가분 이전, v1 앱 은퇴** 대사 일치. v1 앱 요청에는 업데이트 안내
11	2026년 11월 10일	**현재(진행 중)** 서버 0.4.22(릴리스 43회), 결정 기록 199건

화면

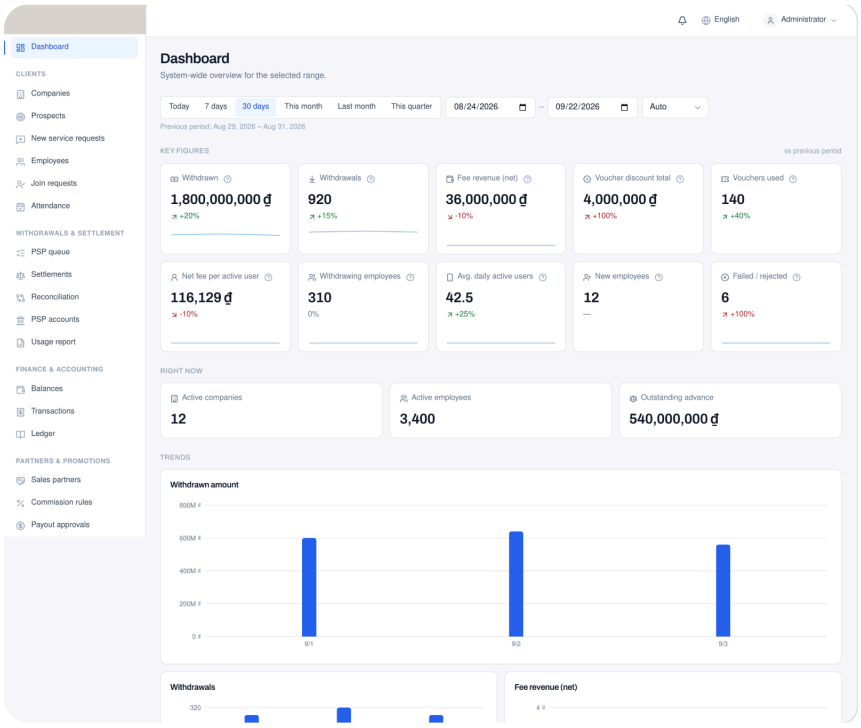
출금 확인 화면

지급 주기·가능 금액·신청 금액·
수수료와 받을 금액을 한 장에 보여
준다. 받는 계좌 정보는 가렸다.



운영사 CMS 대시보드: 출금액·출금 건수·수수료 수익 같은 핵심 지표와 기간별 추이 차트

모의 데이터이며 로고는 가렸다.



수동 출금 승인 대기열

요청 시각·직원·회사·금액·실수령액·대기 시간이 한 줄에 있고, 줄마다 승인·반려 버튼이 있다. 모의 API 데이터이며 로고는 가렸다.



문제와 배경

돈이 실제로 나가는 서비스를, 이미 운영 중인 v1 옆에서 처음부터 다시 만드는 일이었습니다.

베트남의 근로자는 월급날 전에 돈이 필요하면 고리 대출이나 회사의 수기 급여 가불에 기대야 했습니다. 수기 가불은 회사 급여 담당자의 시간도 잡아먹었습니다.

EWA(Earned Wage Access)에서는 근로자가 이미 일한 만큼의 급여를 앱으로 먼저 꺼내 쓰고, 회사가 다음 급여에서 공제해 운영사와 정산합니다. 근로자는 출금마다 수수료를 냅니다. 고용주에게는 비용이 들지 않는 복리후생입니다.

운영사의 관리자·재무·인사 담당자, 고용주, 근로자, 영업 파트너까지 6개 행위자가 각자 다른 화면과 권한으로 같은 돈의 흐름을 다룹니다.

돈이 실제로 나가는 서비스라 중복 지급이나 잘못된 성공 처리를 허용할 수 없었습니다. 규정상 데이터는 베트남 안의 서버에 두어야 했습니다. 다른 개발자가 만든 v1이 이미 운영 중이었고, v2는 처음부터 새로 만드는 시스템이었습니다. 이전의 원칙은 v1의 어떤 정보도 지우지 않는 것이었습니다.

돈이 도는 한 달

운영사가 먼저 지급하고 고용주가 다음 급여에서 공제해 갚으며, 근로자는 출금마다 수수료만 낸다.

① **근무** ○사람
이미 일한 만큼의 급여가 꺼낼 수 있는 돈이 된다

② **출금 요청** ○사람
앱에서 금액·바우처 선택, 추가 인증

③ **지급** □외부
운영사의 결제사 계좌 → 근로자 은행 계좌(출금액 − 수수료)

④ **수수료 인보이스** □시스템
수수료에 VAT 전자인보이스 발행, 바우처로 감면

⑤ **급여일 공제** ○사람
고용주가 「급여 − EWA 공제」 정산 파일 제출

⑥ 정산 승인 ○사람
운영사 1차·고용주 2차 승인 뒤 처리(자동 처리 꺼 둠)

⑦ 정산 확인 □시스템
상태 모니터링, 미정산 알림

되돌아가는 길
↻ ⑦ 정산 확인 → ① 근무: 다음 급여 주기

제 역할

팀과 파이프라인은 제가 설계해 운영했고, 운영 반영은 제 지시로 실행하는 것을 규칙으로 두었습니다. 돈이 오가는 경로의 코드는 제가 직접 읽고 승인했고, 에이전트가 쓴 코드도 운영에 내보낸 책임은 제게 있습니다.

13개 역할의 AI 에이전트 팀과 8단계 파이프라인(접수 → 아키텍트 → 디자인 → 개발 → QA → 리뷰 → 통합 → 문서화)을 설계해 6월 14일에 올렸고, 그 뒤로 이 팀을 운영했습니다. git 작성자는 저 한 사람이고, 커밋 2,130개 중 1,853개(87%)에 AI 공동 작성 표시가 있습니다.

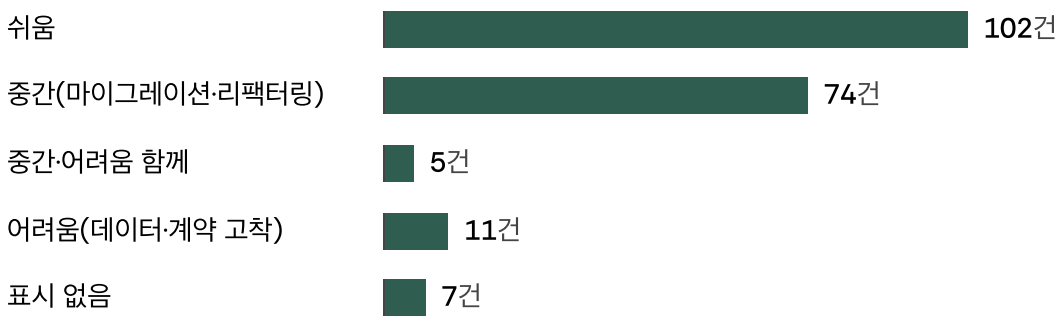
결정은 막히지 않게 추천 기본값으로 진행하고, 결정 기록에 남긴 뒤 나중에 검토하는 방식으로 정했습니다. 199행 중 68행(34%)에 사람의 확정·입력이 적혀 있고, 그중 제 이름이 적힌 행은 37행입니다. 되돌리기 어려운 11건 중 3건(D-036, D-159, D-161)은 기록할 때 확정했고, 8건은 기본값으로 진행했습니다. 7월 3일 이해관계자 결정 시트의 답은 7월 4~5일 결정 D-128~D-131로 한꺼번에 반영했습니다.

push 승인·운영 배포·스토어 제출은 제 지시로 실행하는 것을 규칙으로 두었고, 실행은 에이전트 세션이 맡았습니다. 11월 7일 운영 데이터 이전은 같은 날 오전 롤백 리허설 뒤에 실행했고, 이전 도구는 대사가 하나라도 틀리면 커밋하지 않습니다. 이전의 원칙 「v1의 어떤 정보도 삭제되면 안 된다」는 제 입력으로 결정 기록(D-173)에 남아 있습니다.

결정 기록 199건의 되돌리기 난이도

결정 199건 중 192건에 되돌리기 난이도를 적었고, 데이터·계약이 고착되는 어려운 결정은 11건이다.

출처: 결정 기록 D-001~D-199의 되돌리기 칸을 행마다 셌다. 한 행에 중간과 어려움이 함께 적힌 5건은 따로 묶었다.



누가 무엇을 했나

본인이 13개 역할의 에이전트 팀을 설계해 운영했고, 결정 확정·입력과 운영 반영 지시를 맡았다.

저 — 1 빌드 팀의 사람(저)

13개 역할 에이전트 팀과 8단계 파이프라인 설계(6월 14일)
결정 확정·입력: 199행 중 사람 표기 68행, 그중 본인 이름 37행(확정 13 · 입력 24)
이전 원칙: v1의 어떤 정보도 지우지 않는다
운영 반영 지시: push 승인·운영 배포·스토어 제출

에이전트 — 9 역할 — 87% AI 공동 작성 커밋(전체 2,130 중 1,853)

지휘: 오케스트레이터
설계: 아키텍트·디자이너
구현: 백엔드·웹·앱 개발자
QA·데브옵스·문서

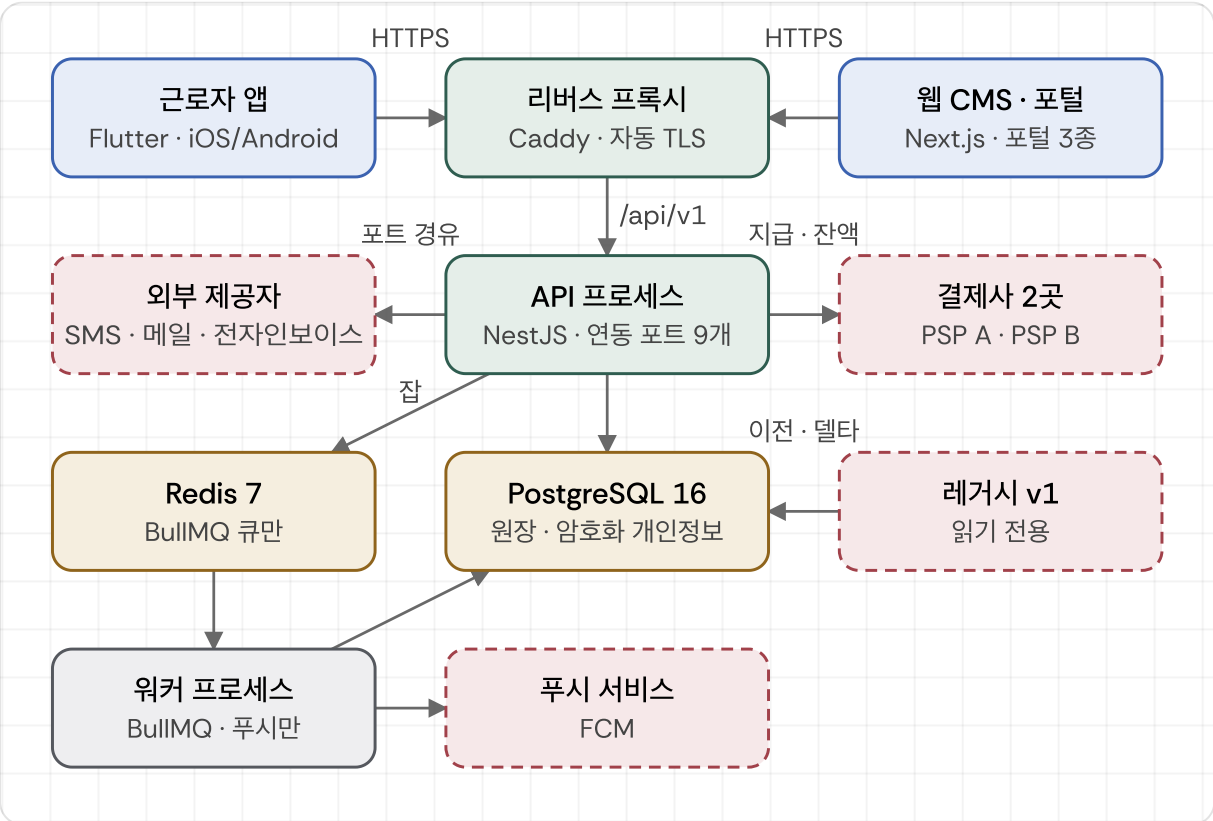
리뷰 게이트 — 4 리뷰어(9 + 4 = 13개 역할, 파이프라인 안 3종)

보안 리뷰·보안 수정을 언급한 수정 커밋 30개(6월 14일~10월 29일)
파이프라인 리뷰어 3종: 보안(돈·인증·개인정보 변경 때), 코드(항상), API 계약(계약이 바뀔 때)
결제사 장애 전환: 보안 리뷰 3회에서 나온 지적 4건을 고침

시스템 구조

시스템 구성

앱과 웹이 API 하나를 함께 쓰고, 돈을 옮기는 일은 API 프로세스 한 곳에서만 일어난다(워커는 푸시만 보낸다).



근로자 앱

근로자용 Flutter 앱(iOS·Android). 가입, 한도 조회, 출금(2단계 인증·생체·PIN), 바우처, 지도 출근, 기기 안에서 하는 신분증 글자 인식. 화면 45개. 앱 모델은 손으로 작성한다.

리버스 프록시

Caddy가 TLS를 자동으로 붙인다. 스테이징은 한 도메인에서 /api/v1만 API로 보내고, 운영은 API와 CMS 도메인을 나눈다. CMS 페이지도 이 프록시가 내보낸다.

웹 CMS · 포털

운영사 CMS, 고용주 포털, 영업 파트너 포털을 Next.js 하나에 담았다. 페이지 123개. API 타입은 OpenAPI 스냅샷(오퍼레이션 604개)에서 생성한다.

외부 제공자

SMS 인증번호, 이메일, 수수료 VAT 전자인보이스, 사업자 조회. 포트마다 목 어댑터와 실 어댑터가 있어 개발 환경의 외부 호출은 0건이다. 결제는 목으로 바뀌는 경로 자체를 없앴다.

API 프로세스

NestJS 모듈 42개, 컨트롤러 96개, 라우트 605개. 예약 작업 18종(정산 대사, 바우처 발급, 전자인보이스, 스토어 버전 확인, 계약 만료 등)도 이 프로세스에서만 돈다. 외부 호출은 모두 연동 포트 9개를 거친다.

결제사 2곳

결제사 2곳. 요청에 RSA 서명을 붙이고, 한 곳은 결과를 콜백 웹훅으로 돌려준다. 2분마다 잔액 조회로 상태를 보고 장애 전환한다(D-142).

Redis 7

BullMQ 큐 브로커만 맡는다.

PostgreSQL 16

PostgreSQL 16 + Prisma. 모델 112개, 마이그레이션 150개. 돈 테이블은 추가만 하는 원장이고, 직원 개인정보 6개 필드는 저장할 때 암호화한다.

레거시 v1

이전 운영 시스템. 이전 도구는 읽기만 한다. 11월 7일 1회 이전, 11월 8일 델타로 추가분을 반영했다.

워커 프로세스

같은 코드로 띄우는 별도 프로세스로, 큐에서 꺼낸 푸시만 보낸다. 돈을 옮기는 예약 작업이 두 프로세스에서 동시에 돌지 않도록 워커에서는 예약 작업을 끈다.

푸시 서비스

FCM. 워커만 호출한다.

`src/integrations/payment-router.ts` · 확정 거절·사용 불가 응답일 때만 다음 결제사로 넘어가고, 예외는 대기로 그 결제사에 남긴다. (주석은 원문을 줄여 옮김)

```
  async payout(
    plan: PaymentPartner[],
    req: PayoutRequest,
  ): Promise<RoutedPayout> {
    type Hop = { partner: PaymentPartner; result: PayoutResult };
    const attempts: PayoutAttempt[] = [];
    let last: Hop | null = null;
    let declined: Hop | null = null;
    for (const partner of plan) {
      const result = await this.hop(partner, req);
      attempts.push({ partner, status: result.status });
      last = { partner, result };
      // Only a returned 'failed' or 'unavailable' goes to the next PSP.
      // A throw inside hop() is ambiguous (it may have paid), so it
      // comes back as 'pending' and the walk stops on that PSP.
      if (result.status === 'success' || result.status === 'pending') {
        break;
      }
      if (result.status === 'failed') {
        declined = last;
      }
    }
    // …
  }
```

핵심 결정

돈 경로의 세 결정은 에이전트 기본값으로 시작해 둘을 이해관계자 시트와 보안 리뷰 지적으로 고쳤고, 암호화와 v1 이전은 제 입력에서 시작했습니다.

돈은 정수로, 원장은 추가만 ADR-0004 에이전트 기본값 · 확정 기록 없음

문제

베트남 동(VND)은 소수 단위가 없고, 한 회사의 월 정산 합계는 32비트 정수 한도(약 21억)를 넘을 수 있습니다.

검토한 선택지

완전한 복식부기: 라이선스 핀테크의 장기 모델로는 맞지만, 지금 스토리에는 무겁고 납기를 늦춤

✓ 단식 + 추가 전용 원장: 나중에 복식부기로 묶을 필드를 미리 둠

결정

원장은 단식이지만 수정·삭제 없이 추가만 하고, 취소는 반대 기입(REVERSAL)으로 합니다. 계정·방향·참조·배치 필드를 미리 두어, 나중에 복식부기로 바꿔도 과거 데이터를 다시 짤 필요가 없게 했습니다. 금액은

BigInt 동 단위, 요율은 정수 bp(60% = 6000)로 저장합니다. 동에는 소수 단위가 없어 돈 경로에서 Decimal과 float를 금지했고, 월 정산이 32비트 한도를 넘을 수 있어 Int가 아니라 BigInt를 썼습니다.

결과

한도와 수수료 계산이 모두 정수 연산이 되었습니다. 원장에 추가만 하므로, 나중에 경쟁 조건 테스트가 「원장 합계 = 0」을 불변식으로 검사할 수 있었습니다.

대가·남은 위험

엄격한 복식부기가 필요해지면 상위 거래 테이블과 균형 제약을 더하는 마이그레이션이 필요합니다(ADR은 「중간」 난이도로 적었습니다). ADR의 상태는 지금도 「잠정」입니다.

출금은 한 트랜잭션, 지급은 커밋 후 ADR-0006 기본값 → 이해관계자 답으로 수정

문제

중복 클릭이나 경쟁 조건으로 두 번 지급될 수 있고, 네트워크 호출 중에 DB 잠금을 쥐고 있으면 안 됩니다.

검토한 선택지

사가/아웃박스: 출금 예약이 DB 하나 안에서 끝나므로 1단계에는 필요 없다고 봄

✓ 단일 Postgres 트랜잭션 + 커밋 후 지급

결정

출금 예약은 Serializable 트랜잭션 하나로 합니다. 트랜잭션 안에서 이번 주기 출금액을 다시 합산해 한도를 계산하고, 근로자 상태를 다시 확인한 뒤 출금·바우처·원장을 기록합니다. 두 요청이 겹치면 PostgreSQL이 한쪽을 직렬화 실패로 되돌리고, 최대 5번 다시 시도해도 실패하면 409를 돌려줍니다. ADR 문서에는 FOR UPDATE 잠금으로 적었지만, 구현은 처음부터 이 방식이었습니다. 결제사 호출은 커밋 뒤 트랜잭션 밖에서 하고, 결과는 짧은 두 번째 트랜잭션으로 반영합니다. Idempotency-Key는 전체에서 영구히 유일합니다. 같은 키로 다시 보내면 원래 결과를, 직원이나 금액이 다르면 409를 돌려줍니다.

결과

같은 직원이 서로 다른 키로 동시에 두 번 확정하면 정확히 1건만 성공하는지 e2e로 검사합니다. 9월 4일에는 수동 반려 두 건이 동시에 들어오면 반대 기입이 두 번 쌓이는 경쟁 조건을 찾아 고쳤습니다. 상태가 기대값일 때만 갱신하고 갱신된 행이 1개일 때만 진행하는 조건부 선점으로 바꿨고, 원장 합계가 0인지 검사하는 경쟁 e2e 테스트를 더했습니다. 이 수정은 보안 리뷰 승인을 받았고, 보안 리뷰가 지적한 나머지 지급 경로에도 9월 13일 같은 패턴을 적용했습니다.

대가·남은 위험

행 잠금 대신 Serializable 충돌 감지를 쓰므로, 겹친 요청은 한쪽이 다시 계산되고 충돌이 5번을 넘으면 409를 돌려줍니다. Idempotency-Key는 지우지 않고 전체에서 영구히 유일하게 둡니다. ADR은 결제사가 여럿이 되면 이 방식을 다시 보겠다고 적었는데, 9월에 결제사 두 곳이 들어온 뒤에도 ADR은 고치지 않았습니다.

결제사 이중화와 헬스 기반 장애 전환 D-142 에이전트 기본값 · 확정 기록 없음

문제

결제사 한 곳이 멈추면 출금 전체가 막혔습니다.

결정

지급 결과를 성공·대기·실패·불가 네 가지로 나눴습니다. 허용 목록에 있는 확정 거절 코드만 '실패'로 보고, 나머지는 '대기'로 두어 다른 결제사로 넘기지 않게 했습니다. 시도 순서는 회사 지정 결제사 → 나머지이고, 설정됨·활성·정상·잔액 충분인 곳만 넣습니다. 실패나 불가면 다음 결제사로 넘어가고, 대기나 예외면 그 결제사에 남기고 멈춥니다. 헬스는 2분마다 잔액 조회로 확인하고, 전송 오류가 2회 연속이면 DOWN으로 봅니다. 모두 불가하면 2단계 인증(TOTP) 코드를 쓰기 전에 503을 돌려주고 원장에 아무것도 남기지 않습니다. 결제는 목으로 바뀌는 경로를 아예 없앴습니다.

결과

9월 13일 병합했습니다. 보안 리뷰를 세 번 반영해 수동 지급 경쟁, 모르는 코드를 실패로 봐서 생기는 이중 지급, 해소 후 다시 걸리는 플래그, 대기 영구 고착을 고쳤습니다.

대가·남은 위험

모호한 결과는 '대기'로 두어, 상태 조회로 풀리거나 운영자가 종료할 때까지 그 결제사에 남습니다. 종료해도 한도는 복구되지 않습니다. 쓸 수 있는 결제사가 없으면 목으로 넘기지 않고 2단계 인증 전에 503을 돌려줍니다.

직원 개인정보 저장 시 암호화 D-172 본인 입력

문제

v1이 일부 개인정보를 암호화해 왔고, DB 덤프만으로 개인정보가 드러나면 안 됐습니다.

결정

AES-256-GCM을 Prisma 쿼리 확장에 넣어, 쓸 때 암호화하고 읽을 때 풀게 했습니다. 서비스 코드는 평문만 봅니다. 대상은 전화번호, 계좌번호, 예금주, 신분증 번호, 생년월일, 기본급 6개입니다. 모든 필드에 랜덤 IV를 쓰면 같은 값으로 조회할 수 없어서, 조회가 필요한 전화번호와 신분증 번호(10월 27일부터)만 필드 이름과 값의 HMAC에서 만든 결정적 IV로 고유성과 동등 조회를 지키고, 나머지는 랜덤 IV를 씁니다. 검색용으로 전화번호 끝 4자리만 평문으로 둡니다.

결과

v1 이전 도구는 이전한 직원의 6개 필드가 모두 암호문인지(평문 0건) 대사에서 확인한 뒤에만 커밋합니다.

대가·남은 위험

키를 잃으면 복구할 수 없고, 부분 검색은 전화번호 끝 4자리만 됩니다. 결정적 IV를 쓰는 두 필드는 암호문만 보고도 같은 번호인지 알 수 있습니다.

v1 → v2 데이터 이전 D−173 본인 입력

문제

운영 중인 v1의 데이터를 v2로 옮겨야 했습니다. 원칙은 「v1의 어떤 정보도 삭제되면 안 된다」였습니다.

결정

이전 도구에는 v1 컬럼 목록과 규칙만 고정하고, 행은 실행하는 순간 전부 읽습니다. 결정적 ID를 쓰고, 규칙표에 없는 enum이 나오면 멈춥니다. 한 트랜잭션으로 적재한 뒤 대사하고, 하나라도 틀리면 커밋하지 않습니다. 대사 항목은 회사별 건수·금액·수수료, 직원별 지급 합계, 정산 완료 건수, 실패 건의 원장 상계, 평문 개인정보 0건입니다. 기대값은 v1에서 읽은 행을 변환하면서 함께 쌓은 값이라, 변환 결과와 실제 적재를 맞추는 검사입니다. 기본 실행은 적재·대사 뒤 롤백하는 리허설이고, v1에 보내는 문자열에 파괴적 단어가 있으면 테스트가 실패합니다.

결과

9월 30일과 11월 7일 오전 리허설에서 v1 쪽 이상(근태 기록의 날짜 필드 뒤바뀜, 깨진 계좌번호)을 찾았고, 도구 쪽은 출금 코드 유지, 이미 v2에 있는 번호 충돌, 미정산 달 목록화를 고쳤습니다. 11월 7일 1회 이전 뒤 v1에 들어온 거래는 11월 8일 델타 이전으로 더했고, 대사가 일치했습니다.

대가·남은 위험

대사는 v1 원본을 다시 조회해 맞추지 않습니다. 시각, 근태 일수, 알림, 파일은 대사 항목이 아닙니다.

출금 한 건이 지나가는 길

돈은 기록이 커밋된 뒤에만 나가고, 확정 실패일 때만 다음 결제사로 넘어가게 설계했다. 결과가 모호하면 그 결제사에 남겨 두 곳이 겹쳐 지급하지 않게 한다.

① **출금 요청** ○사람
금액·바우처·2단계 인증. 같은 Idempotency-Key로 다시 보내면 원래 결과

② **자격 확인** □시스템
근태·한도·사용 이력, 회사·계좌 검증

③ **결제사 순서 정하기** □시스템
회사 지정 결제사 → 나머지. 설정됨·활성·정상·잔액 충분만. 하나도 없으면 인증 코드를 쓰기 전에 503, 원장 0행

④ **한 트랜잭션** □시스템
Serializable 트랜잭션 안에서 한도 재계산 → 근로자 상태 재확인 → 출금·바우처·원장 기록. 겹치면 한쪽을 되돌려 다시 계산(최대 5회)

⑤ **커밋 후 지급** □외부
트랜잭션 밖에서 서명한 요청. 결제사가 바뀌어도 같은 참조번호

⑥ **결과 판정** □시스템
성공·대기·실패·불가 4종. 대기와 예외는 그 결제사에 남기고 멈춤

⑦ **결과 반영** □시스템
짧은 두 번째 트랜잭션, 입금 알림, 대사

되돌아가는 길

↻ ⑥ 결과 판정 → ⑤ 커밋 후 지급: 확정 실패·불가 → 다음 결제사

품질과 검증

자동 테스트 7,080개 중 3분의 2가 돈 경로가 있는 백엔드에 있고, e2e는 돈과 권한의 경계를 겨냥합니다.

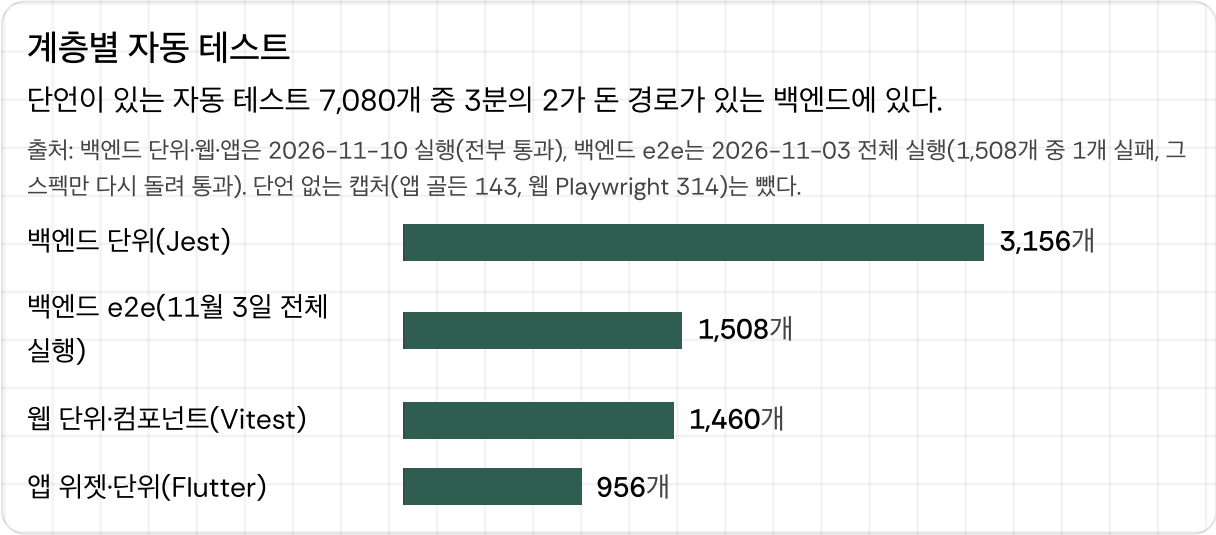
단언이 있는 자동 테스트는 7,080개입니다. 백엔드 단위 3,156개, 웹 1,460개, 앱 956개는 11월 10일에 실행해 모두 통과했습니다. 실제 DB가 필요한 백엔드 e2e는 11월 3일 전체 실행 기록을 씁니다. 1,508개 중 1개가 실패했고, 그 스펙만 다시 돌리자 통과했습니다.

e2e는 돈과 권한의 경계를 겨냥합니다. 테스트 이름으로 센 근삿값으로 테넌트 격리·타사 접근 32개, 403 응답 94개, 멱등·재시도·동시성 45개가 있습니다.

외부 호출은 모두 연동 포트 9개를 거칩니다. 포트마다 목 어댑터가 있어 개발 환경의 외부 호출은 0건입니다. 결제 포트만은 목으로 바뀌는 경로를 막았습니다.

파이프라인의 리뷰어 3종(보안·코드·API 계약)은 기록이 남은 9월 28일 이후 24회 실행했습니다. 보안 리뷰·보안 수정을 언급한 수정 커밋은 30개(6월 14일~10월 29일)입니다. 결제사 장애 전환(D-142)은 보안 리뷰를 세 번 거쳤습니다.

화면 상태 지도도 만들었습니다. 앱 골든 캡처와 웹 Playwright 캡처로 화면 131개, 상태 597개, 전이 1,245개를 와이어프레임 지도로 그렸습니다. 이 캡처에는 단언이 없어서 테스트 수에서 뺐습니다.
규칙은 실패하는 테스트를 먼저 쓰는 것이었습니다. 비병합 fix 커밋 398개 중 344개(86%)가 테스트 파일을 함께 바꿨습니다. 기존 테스트를 고친 경우도 들어간 수치라, 회귀 테스트를 더했다는 뜻은 아닙니다.



기능 하나, 처음부터 끝까지

스토리 하나가 문서에서 배포까지 어떻게 이어지는지 보여 드립니다. 앱 강제 업데이트 기능입니다.

1 스펙과 결정 — EMP-SET-07 / ADM-SET-03 → D-143

X.Y.Z 버전 규칙을 정했습니다. X나 Y가 오르면 스토어 버튼만 있는 차단 화면을, Z만 오르면 닫을 수 있는 안내를 5초 동안 띄웁니다. 확인에 실패하면 앱은 정상 실행(fail-open)됩니다. 공개 GET은 「모든 엔드포인트에 역할 가드」 규칙의 의도적인 예외로 정했습니다. 동기는 v1 앱에 있던 강제 업데이트가 v2에 없다는 점이었습니다.

2 백엔드 — 버전 레지스트리와 공개 확인 API

버전 레지스트리, 공개 GET /app-version, 감사 기록이 남는 관리자 PUT을 한 커밋에 넣었습니다. 단위 테스트 3개와 e2e 7개가 함께 들어갔습니다. 같은 커밋에서 역할 가드 예외의 이유를 프로젝트 규칙 파일에 적었습니다.

3 웹 CMS — 최신 버전 등록 화면

설정 화면, 첫 등록 때 강제 업데이트를 확인하는 대화상자, OpenAPI 스냅샷 재생성 순으로 커밋이 이어졌습니다. 테스트 7개와 수정 커밋 1개가 들어갔습니다.

4 앱 — 콜드 스타트 업데이트 게이트

위젯 테스트 6개(강제, 선택, 5초 자동 닫힘, API 오류 시 fail-open, 잘못된 본문, 안드로이드 행 없음)와 버전 비교 단위 테스트 6개를 넣었습니다. iOS 분기는 테스트 변형으로 범위를 좁혔습니다. 강제·선택 화면 캡처는 9일 뒤 와이어프레임용으로 따로 추가한 것이고, 회귀 테스트는 아닙니다.

5 첫 적용 — 스테이징에서 규칙대로 버전 판단

9월 21일, 초대 코드 라우트를 없애고 계좌 쓰기를 막으면서 설치된 앱이 반드시 업데이트해야 했습니다. 규칙대로 서버를 0.1.5에서 0.2.0으로, 앱도 Y 단위로 올렸습니다. 스테이징 라운드였고 스토어에 없는 빌드라서 CMS 등록은 일부러 하지 않았습니다. 운영 순서(스토어 공개 → 서버 배포 → CMS 등록)와, 순서를 거꾸로 하면 왜 더 나쁜지도 기록했습니다.

6 보강 — 규칙을 코드로 강제

「스토어에 공개된 뒤에만 등록」은 관례에 머물러 있었습니다. 10월 26일, 매일 06:00(ICT) 스토어 버전을 읽어 저장하고, 그보다 높은 버전을 등록하면 400(VERSION_ABOVE_STORE)을 돌려주게 했습니다. 강제 기준선 밑으로 떨어지면 관리자에게 알립니다.

회고

가끔 실패하는 e2e는 실패 모양으로 환경 문제인지 버그인지 가르는 기준을 세워 원인을 좁혔습니다.

가끔 실패하는 e2e에서 진단 기준을 얻었습니다. 약 90번에 3번, 매번 다른 테스트 하나가 실패했습니다. 「여러 개가 한꺼번에 실패하면 환경, 하나가 옮겨 다니면 아직 못 찾은 버그」라는 기준으로 원인을 앱이 아니라 테스트 전송(요청마다 임시 포트를 열고 닫는 방식)으로 좁혔고, 서버를 한 번만 바인드하게 바꿨습니다.

일하는 방식

2026년 EWA 플랫폼을 만들며 세우고 운영한 방식과, 프로젝트 6건에서 그 방식이 바뀐 과정을 적었습니다. 고객의 수정 요청은 모두 24시간 안에 반영했습니다. 13개 역할의 AI 에이전트 팀에 8단계 파이프라인으로 설계·구현·QA·리뷰를 맡겼습니다. 요구사항과 결정 시트의 답은 고객 쪽 이해관계자가 댔고, 저는 기본값으로 진행한 결정을 확정하거나 입력하고 운영 반영을 지시했습니다.

2026년 EWA 플랫폼을 만들며 세우고 운영한 방식 보기 →

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

연락처

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