

임철순

Frontend · Full-stack Developer

✉ cjtfn01@gmail.com | 🏠 서울 강동구

📄 GitHub: github.com/POcodingWER | 📝 Blog: pocodingwer.github.io | Project: backend-visualized.vercel.app

■ 요약

React + TypeScript 기반으로 B2C 모바일 앱(학기 중 DAU 2~3 만)의 결제·포인트·충전 서비스를 개발·운영해온 프론트엔드 개발자입니다.

10 개 서비스의 Nuxt.js → React CSR 마이그레이션을 2 인 팀으로 리드하며 **AWS 비용 92% 절감(월 \$1,500 → \$120, 연 약 \$16,500)**과 **로딩 속도 60% 개선**을 달성했고, Sentry 기반 모니터링으로 장애 대응 시간을 65% 단축했습니다.

Lambda@Edge·S3·CloudFront 인프라와 모노레포 CI/CD 를 직접 다뤘으며, 프로젝트 컨텍스트 문서·행동 제약 룰·PR 리뷰 자동화 스킴으로 구성된 AI 에이전트 하네스를 설계·운영하고 있습니다.

■ 핵심 성과

- **AWS 비용 92% 절감**: 월 \$1,500 → \$120, 연 약 \$16,500 절감 (SSR→CSR 전환 65%, CloudFront 최적화 76% 추가 절감)
- **CloudFront 요청 90% 감소 · 캐시 hit 90% 달성**: 1,200 만 → 100 만 요청/월, 로딩 속도 60% 개선(3 초 → 1.2 초)
- **10 개 서비스 점진적 마이그레이션 리드**: Nuxt SSR → React CSR, 10 개월, 2 인 팀
- **장애 대응 시간 65% 단축**: Sentry 실시간 모니터링(2 시간 → 40 분), 블루투스 연결 실패율 80% 감소, 일 40~50 건 문의 → 3~4 건
- **AI 리뷰로 Critical 버그 사전 검출**: 150 개 파일·8,000 줄 규모 대형 PR 에서 사람이 놓치기 쉬운 버그를 AI 리뷰로 검출·수정

■ 경력 사항

경력 시작 2022.06 ~ 현재

(주)핑거랩스 | Frontend Developer | 2025.10 ~ 현재

※ (주)핑거버스(2022.08~2025.02), 기프트레터(주)(2025.02~2025.10) — 계열사 조직 개편으로 인한 소속 변경. 실제로는 메타클럽 서비스를 계속 개발했습니다.

캠퍼스 중심 B2C 모바일 앱 "메타클럽"(학기 중 DAU 2~3 만) 프론트엔드 개발 및 아키텍처 전환 주도. React, TypeScript, React Query, Context API 기반.

1. 결제·포인트·충전 서비스 및 도메인별 서비스 개발

- 문화상품권, 가상계좌, 카드, 편의점 충전 등 다중 결제 수단 통합, 포인트 시스템 개발·운영
- Sentry 에 결제/환불 플로우 사용자 컨텍스트를 태깅해 이슈를 사용자 단위로 추적
- 세탁: 블루미터(블루투스 IoT) 연동 세탁기 제어, 앱·웹 브릿지 통신 / 티켓: 클립(Clip) 연동 공연 예매 / 게임: 실시간 랭킹

2. Nuxt.js → React 마이그레이션 리드 (2025.01 ~ 2025.11, 10 개월)

- 10 개 핵심 서비스(main, charging, conversion, laundry, point, ticket 등) 점진적 전환, 2 인 팀 리드 (주간 싱크업, 페어 프로그래밍)
- Lambda@Edge + S3 기반 점진적 배포 전략 설계
- 기술 선택: Redux 대신 React Query(서버 상태) + Context API(UI 상태), Vite 번들러(Webpack 대비 빠른 HMR), Lambda@Edge(낮은 운영 비용)

성과: AWS 비용 1 단계 65% 절감(\$1,500 → \$500), 유지보수 비용 감소

3. CloudFront 캐싱 최적화 및 비용 절감 (2025.01 ~ 2025.11)

- 캐싱 정책 세분화(HTML/CSS/JS 별도 전략), S3 정적 파일 Brotli + Gzip 압축, Lambda@Edge 동적 라우팅

성과: CloudFront 비용 76% 절감(\$500 → \$120), 요청 90% 감소(1,200 만 → 100 만/월), 캐시 hit 90% 달성, 총 AWS 비용 92% 절감

4. Sentry 모니터링 구축 & 블루투스 IoT 장애 해결 (2025.04 ~ 2025.06)

- 세탁기 블루투스 연결 실패(일 40~50 건 이상 발생) 2 개월 디버깅, Sentry 도입 제안 및 구축
- 앱 개발자와 로그 수집 구조 설계(기기 정보·OS 버전·연결 상태) → 2 주·150 여 건 분석 → 특정 기기/OS 버전에서만 발생 확인 → 타임아웃 조정 + 재시도 로직

성과: 장애 대응 시간 65% 단축(2 시간 → 40 분), 연결 실패율 80% 감소, 일 40~50 건 문의 → 3~4 건(약 90% 감소)

5. AI 에이전트 하네스 설계·운영

- 프로젝트 컨텍스트 문서: 아키텍처, 앱별 도메인 책임, 공유 패키지 영향범위, 작업 완료 체크리스트
- 행동 제약 룰: 요청 범위 외 리팩토링 금지, 신규 추상화 시 사유 설명, 공유 패키지 변경 시 영향 앱 명시 → 과잉 작업과 리뷰 비용 방지
- MCP(Sentry/Figma) 작업별 수동 토크, 상시 연결 서버 2~4 개로 제한해 토크 예산 관리
- GitHub PR 리뷰 자동화 스킴: Critical/Medium/Low 분류, 요약·인라인 코멘트 자동 등록, 리뷰 로그 저장·통계화

성과: 150 개 파일·8,000 줄 대형 PR 에서 Critical 버그 사전 검출·수정, 반려된 리뷰 의견의 사유를 기록해 리뷰 기준을 지속 개선

6. 모노레포 · 디자인 시스템 · 개발 프로세스 자동화

- PNPM workspace 기반 10 개 서비스 통합 관리, 공통 UI 라이브러리(packages/ui, styles, utils), 페이지별 독립 빌드/배포, Turborepo 빌드 캐시
- Figma 디자인 토크(style-dictionary) + Storybook 도입, 주간 기술 공유 세션 제안·진행
- Node.js 스크립트로 언어팩(Excel→JSON), 약관(Notion→HTML) 업데이트 자동화: 각각 70%, 90% 시간 단축

성과: CI 빌드 80% 단축(변경된 앱만 빌드), 배포 35% 단축, 번들 사이즈 25% 감소

(주)이모션글로벌 | Blockchain Developer | 2022.06 ~ 2022.08 (3 개월)

- Solidity 스마트 컨트랙트 + React 프론트엔드 (Delica NFT 민팅 플랫폼), Bellygom NFT 발행 시스템, NFT 정산시스템 개발

■ 개인 프로젝트

backend-visualized — 백엔드 개념 시각화 사이트 (2026.10, 1 인 개발)

Site: backend-visualized.vercel.app | GitHub: github.com/POcodingWER/backend-visualized

실제 서버·DB 에서 동시성·인증·결제 문제를 일부러 재현하고, 해결 방법을 2~4 가지씩 실측 비교해 결과를 화면으로 보여주는 6 장 구성 사이트입니다. 화면·API·인프라·배포를 혼자 구축했고, 장마다 선택한 방법과 포기한 방법을 근거 수치와 함께 문서화했습니다.

Stack: Next.js 15(App Router, Vercel) · NestJS · Prisma · MySQL 8.4 · Redis 7.4 · Docker Compose · Caddy · AWS EC2 · GitHub Actions

백엔드

- 동시성 제어:** 선착순 쿠폰 500 명 동시 요청에서 원자적 조건부 UPDATE 로 초과 발급 0 장, 비관적 락 대비 약 4 배 빠름. 비관적 락 대기 시간의 96%가 커넥션 풀 대기임을 측정
- 인증:** JWT 액세스 토크(메모리) + 리프레시 토크 회전·재사용 감지(httpOnly 쿠키, DB 엔 SHA-256 해시만 저장). 클라이언트 single-flight 로 동시 재발급 요청 5 회 → 1 회

- **결제 멍등성:** Idempotency-Key + UNIQUE 제약으로 중복 결제 4 회 → 1 회. 처리 중 장애로 남은 결제는 조건부 UPDATE 정리 잡으로 종료해 다중 인스턴스에서도 한 번만 처리
- **포인트 원장:** SELECT ... FOR UPDATE 트랜잭션 + 대사(잔액 vs 내역 합계)로 동시 200 건에서 유실 10,100P → 0P
- **이벤트 수집·캐시:** 자체 SDK 배치 전송 + createMany 로 10,000 건 18.7 초 → 0.29 초(병목이 DB 커밋 횟수임을 분석), Redis cache-aside + 분산 락으로 캐시 스탬피드 시 DB 조회 1,000 회 → 1 회

인프라 · 운영

- **배포:** GitHub Actions → GHCR(ARM 이미지) → OIDC → SSM 구조로 배포용 포트·저장된 AWS 키 없이 운영, blue-green 교체로 배포 3 회 동안 실패 요청 0 건, 서버 빌드를 Actions 로 옮겨 서버 배포 job 135 초 → 37~51 초
- **관측성:** pino 구조화 로그 + x-request-id, 5xx Sentry 전송, 15 분 주기 헬스체크, MySQL 일일 백업(S3) 및 복원 테스트
- **수평 확장:** k6 로 병목이 Node 이벤트 루프 하나임을 확인 후 API 2 대 + Caddy 로드 밸런싱, 공유 상태(잠금·cron·SSE 알림)를 Redis 분산 락·pub/sub 으로 이전. 800 VU p95 7~10 초 → 1.9 초, 시간 초과 0.89% → 0%

■ 기술 스택

Frontend: React, TypeScript, JavaScript, HTML5, SCSS / Vue.js·Nuxt.js (마이그레이션 경험) / Next.js 15 App Router (개인 프로젝트)

상태 관리: React Query(서버 상태), Context API(UI 상태), Jotai(세탁 서비스 전역 상태)

Backend (개인 프로젝트): Node.js, NestJS, Prisma, MySQL, Redis

Infra & CI/CD: AWS Lambda@Edge·S3·CloudFront(실무), AWS EC2·Docker Compose·Caddy(개인 프로젝트), GitHub Actions, Vite, PNPM workspace, Turborepo

Monitoring · Design: Sentry, pino / Figma, Storybook, style-dictionary, Notion

AI 도구: Claude Code, MCP 서버 구현(SSE·stdio, Docker) — 카카오톡 MCP 공모전용 개인 프로젝트

■ 학력

경북전문대학교 | 철도전자과 졸업 | 2011.03 ~ 2015.02 | 학점 4.01 / 4.5