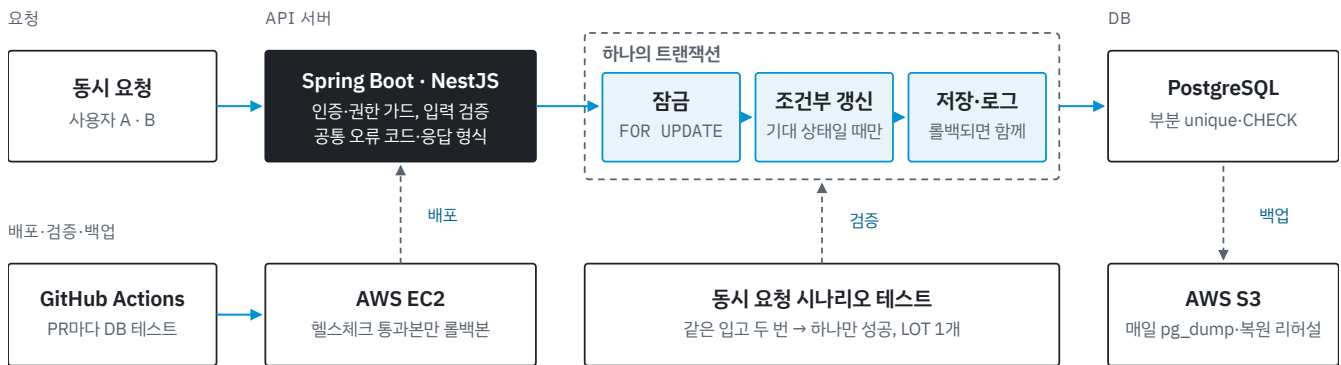


Backend Developer

같은 요청이 두 번 와도, 결과는 한 번만 남깁니다

Spring Boot·NestJS와 PostgreSQL로 API, 트랜잭션, 배포와 운영까지 맡아 온 백엔드 개발자입니다.

5인 팀 ERP에서 서버 공통 기반과 구매·Message → ERP 모듈을 만들었고, 1인 서비스에서 수집 배치·조회 API·운영을 맡았으며, 공공 포털 백엔드를 12개월간 유지보수했습니다. 중복과 경합은 잠금·조건부 UPDATE·unique 제약으로 막고, 동시 요청 테스트로 증명합니다.



주력 기술

Java · Spring Boot

TypeScript · NestJS · Prisma

PostgreSQL

MyBatis · JPA

AWS EC2 · S3

GitHub Actions

Docker · Nginx

Jest · Vitest

동시성

동시에 들어온 입고는 발주 행 잠금으로 초과 입고를 막고, 동시 거래의 번호 경합은 advisory lock으로 풀었습니다.

FantaSteel · 4쪽

트랜잭션 설계

초안 확정과 실행을 트랜잭션으로 나누고 LLM 호출은 트랜잭션 밖에 뒤, 실패해도 기록을 남기고 다시 실행할 수 있게 했습니다.

FantaSteel · 5쪽

성능

행 단위 upsert를 배치 INSERT ... ON CONFLICT로 묶고 DB 리전을 앱 서버와 같은 서울로 옮겨, 기초 데이터 적재를 200초에서 71초로 줄였습니다.

PitchLog · 9쪽

운영

EC2 배포와 롤백, 매일 백업과 복원 리허설, PR마다 실제 PostgreSQL에서 도는 CI를 직접 운영합니다.

PitchLog · 9쪽

실무 12개월

공공 포털(Spring Legacy·PostgreSQL)의 장애를 추적해 고치고, 약 3,000만 건 연례 갱신 배치를 멈춘 지점부터 다시 돌 수 있게 바꿨습니다.

실무 · 13쪽

기술과 경력

기술

백엔드	Spring Boot, Spring Security, MyBatis, JPA, Spring Batch 공공 포털 유지보수(Spring Legacy), LessonTree NestJS, Prisma(TypedSQL) FantaSteel ERP, PitchLog League
언어	Java, TypeScript, SQL, Python (Python은 ETL·GIS 데이터 처리: QGIS, ArcGIS, 좌표계 변환)
데이터베이스	PostgreSQL(Supabase, PostGIS), MySQL 트랜잭션 경계, 행 잠금(FOR UPDATE)-advisory lock, 조건부 UPDATE, 부분 unique 인덱스, 단계별 마이그레이션
API 설계	REST API, 쿠키 JWT 인증과 권한 가드, 공통 오류 코드와 응답 형식 입력 검증, ETag 캐시, 외부 API 클라이언트의 타임아웃·재시도·호출 한도
인프라와 배포	AWS EC2와 S3, Docker, Nginx, systemd, PM2, GitHub Actions, Vercel 블루/그린 무중단 배포, 헬스체크 기반 롤백, pg_dump 백업과 복원 리허설, PR마다 도는 CI
테스트	Jest, Vitest, Supertest, JUnit 실제 PostgreSQL에서 도는 통합 테스트, 동시 요청 시나리오 테스트, 업무 불변식 점검 스크립트
AI 활용	LLM 기능 구현, Claude Code 에이전트로 개발 조회 도구 어시스턴트(PitchLog), 메시지에서 업무 초안 추출(FantaSteel), 작성·검증 에이전트 분리

경력

2024.09 – 2025.08	도시재생 종합정보체계 유지관리 사업 주택도시보증공사 발주, 아이티커넥 소속 프리랜서(주임), 12개월
-------------------	--

프로젝트

2026.09 – 2026.10	FantaSteel ERP 포스코 K-뉴딜 아카데미 팀 프로젝트(5인). 서버 공통 기반, 구매·Message → ERP 모듈
2026.08 – 2026.10	PitchLog League 개인 프로젝트. 축구 데이터 서비스의 수집·API·운영을 혼자 맡음
2024.04 – 2024.05	과외나무 LessonTree 팀 프로젝트(3인) 팀장

학습

2026.08 – 현재	포스코 K-뉴딜 아카데미 LLM 기반 스마트팩토리 AI Assistant 개발 과정, 수강 중
2026 – 현재	AI-Byte-Club 스터디 『주니어 백엔드 개발자가 반드시 알아야 할 실무 지식』 토론
2024.12 – 2025.05	백준 알고리즘 스터디 같은 문제를 Java와 JavaScript로 풀며 주 단위 마감 유지
2023.12 – 2024.05	코리아IT아카데미 Java·Spring Boot 기반 웹 개발 과정(빅데이터·AI 헬스케어 플랫폼)
2019.03 – 2023.02	동의과학대학교 졸업

팀 프로젝트

포스코 K-뉴딜 아카데미
2026.09 – 2026.10

5인

서버 공통 기반,

구매·Message → ERP

NestJS 12, Prisma 7,
PostgreSQL(Supabase),
Next.js 16, Jest, Vitest,
GitHub Actions



FantaSteel ERP

포스코 K-뉴딜 아카데미 과정에서 5인 팀으로 만든, 원료 입고부터 제선, 제강, 연주, 열연, 출하까지 일관제철 공정을 다루는 철강 제조 협업 ERP입니다. 설계 문서와 ERD를 기준으로 서버 공통 기반을 세팅했고, 구매·MRP·조직·업무 알림·Message → ERP 모듈의 API와 동시성 처리를 맡았습니다.

github.com/SteelFlowLLM/fantasteel-erp

GitHub 기록

2026.10.27 기준

110 PR 작성

250 PR 병합 담당(팀 전체 448)

8,472 줄 가짜 DB 삭제

16 개 PR로 화면 서버 전환

맡은 일

서버 공통 기반

ERD를 Prisma 스키마와 초기 마이그레이션(CHECK 제약, 부분 unique 인덱스)으로 옮기고 오류 처리, 응답 형식, 쿠키 인증, 권한 검사, 채번, 작업 로그 기록 공통 모듈을 만들었습니다. 팀원이 같은 기준으로 구현하도록 모듈별 구현 안내 문서 15개를 썼습니다.

직접 구현

구매(요청, 부서장 승인, 발주, 입고), MRP, 조직, 업무·알림, Message → ERP 모듈의 서버 API와 화면 연결.

화면 서버 전환

처음에는 화면이 브라우저 안 가짜 DB로 돌아갑니다. PR 16개에 걸쳐 영역별로 실제 서버 API에 붙이고 가짜 DB가 검증하던 동작은 서버 DB 테스트로 옮긴 뒤, 가짜 DB(35개 파일, 8,472줄)를 지웠습니다.

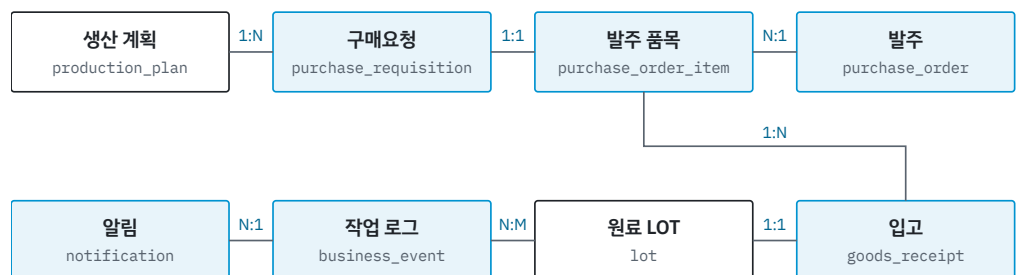
협업

Git 흐름과 PR 규칙을 정하고 팀 PR 병합과 공용 DB 반영을 맡았습니다. PR마다 테스트를 뺀 변경 줄 수와 머지 순서를 적어 작게 나눠 올렸습니다.

구매 데이터 모델

테이블 흐름

구매요청부터 발주, 입고, 원료 LOT, 작업 로그까지 이어지는 흐름을 테이블로 나눴습니다. 입고 한 건은 원료 LOT 하나를 만들고, 작업 로그는 LOT와 N:M으로 연결됩니다.

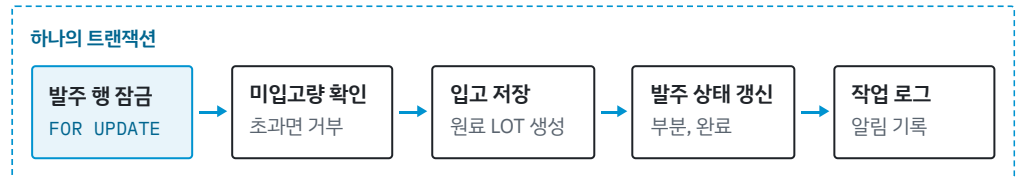


색칠한 테이블은 직접 구현한 구매·알림 모듈과 공통 작업 로그입니다.

같은 발주에 동시에 들어온 입고에서 초과 입고 막기

문제 발주 상태(부분 입고, 입고 완료)는 같은 발주의 모든 품목 입고 합계로 정해집니다. 같은 발주의 다른 품목이 동시에 입고되면 각 거래가 상대 품목을 "아직 안 들어옴"으로 보고 잘못된 상태를 저장할 수 있었고, 같은 품목이면 미입고량을 넘겨 입고될 수 있었습니다.

해결 잠금 대상을 발주 품목이 아니라 상위 발주 행으로 정하고(TypedSQL `SELECT ... FOR UPDATE`), 확인부터 기록까지 한 트랜잭션으로 묶었습니다.



결과 같은 품목에 60t를 동시에 두 번 입고하면 하나만 성공하고 다른 하나는 PUR-003으로 롤백되며 원료 LOT도 하나만 생기는 것을 테스트로 고정했습니다. 버튼을 두 번 누르는 경우까지 막도록 수주 모듈의 역등 키 저장소를 공통 모듈로 옮기자고 팀에 제안했습니다.

동시 거래의 작업 로그 번호가 겹쳐 거래 전체가 실패하던 문제

문제 채번은 "같은 앞부분의 최댓값 + 1"이라 거의 모든 거래가 남기는 작업 로그 번호(EV-)가 두 거래 사이에서 겹칠 수 있었습니다. 겹치면 unique 위반으로 본 거래 전체가 실패했고, 재시도가 있던 구매·수주와 달리 출하·품질·재고·생산은 그대로 실패했습니다.

해결

- 최댓값을 읽기 전에 번호 앞부분을 `pg_advisory_xact_lock(hashtext(prefix))`로 잠가, 뒤 거래가 앞 거래의 커밋을 기다린 뒤 다음 번호를 받게 했습니다. 잠금은 커밋·롤백 때 함께 풀립니다.
- 업무 번호(수주·생산계획·구매요청·발주·입고·출하요청·작업 로그)와 LOT 번호를 모두 같은 공통 채번으로 처리합니다.
- 잠금이 거래 끝까지 유지돼 생길 수 있는 교착 관계를 따져, 기존 unique 인덱스 대기와 같은 관계라 새 위험이 아님을 PR에 남겼습니다.

결과 앞 거래가 커밋하기 전에 뒤 거래가 기록해도 두 거래 모두 저장되고 번호가 이어지는 테스트를 만들었고, 잠금을 빼면 이 테스트가 unique 위반으로 실패하는 것을 확인했습니다. 병합 전 로컬 리뷰에서 `sleep`으로 순서를 맞춘 테스트가 가끔 실패한다는 지적을 받아 신호로 순서를 정하게 고쳤습니다.

그 밖의 동시성 규칙

상태 전이 승인, 반려, 재요청은 "현재 상태가 기대한 상태일 때만" 바꾸는 조건부 UPDATE로 처리해 동시에 들어와도 한쪽만 반영됩니다. 여러 구매요청으로 발주를 만들 때는 id 순서로 잠가 교착을 피했습니다.

알림의 원자성 작업 로그와 알림을 본 거래와 같은 트랜잭션에서 기록해 거래가 롤백되면 알림도 남지 않습니다. 중복 알림은 부분 unique 인덱스로 막았습니다.

핵심 기능

FantaSteel ERP

2026.10

NestJS, Prisma,
PostgreSQL, LLM

Message → ERP

채팅 메시지에서 구매요청 초안을 만들고, 요청자가 확정하면 실제 구매요청을 만드는 기능입니다. 서버 API와 화면을 만들었고, 이후 LLM이 메시지에서 초안 칸을 채우도록 넓혔습니다.

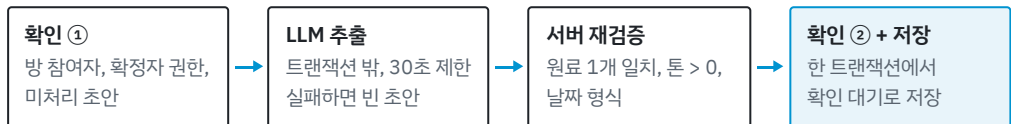
`POST messages/:id/action-drafts`

`POST action-drafts/:id/confirm`

트랜잭션 경계를 어디에 둘지

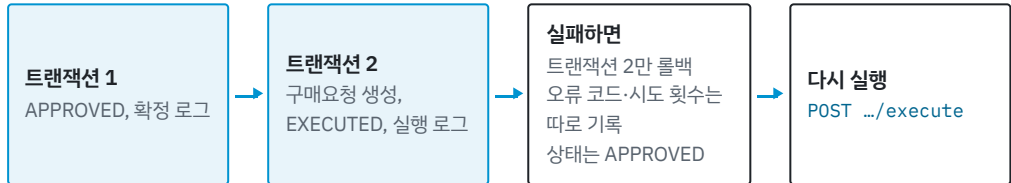
초안 만들기

외부 LLM 호출은 느리고 실패할 수 있어 DB 트랜잭션 밖에서 부르고, 호출 전과 트랜잭션 안에서 같은 조건을 두 번 확인합니다.



확정과 실행

구매요청 생성은 승인권자가 없거나 채번이 겹치면 실패할 수 있습니다. 확정을 먼저 커밋하고 실행은 다른 트랜잭션으로 나뉘, 실행이 실패해도 확정은 남고 다시 실행할 수 있게 했습니다.



중복 실행 방지

상태 확인, `updatedAt` 조건 갱신, 결과 구매요청의 `action_draft_id` unique 제약 세 겹으로 막았습니다. 같은 메시지·유형의 미처리 초안은 부분 unique 인덱스로 하나만 둡니다. 다시 확정·실행해도 구매요청은 1건만 생기는 것을 테스트로 확인했습니다.

업무 유형 등록부

유형마다 확정자, 필요한 권한, 칸 해석, 실행 방법을 정의 하나로 등록하는 구조로 바꿔, 수주·출하요청 같은 새 유형은 정의 하나만 더하면 붙게 했습니다. AI에는 DB 쓰기 권한을 주지 않고 실행은 사용자 권한과 업무 검증을 거칩니다.

AI 경유 기록

AI가 채운 칸 이름을 초안에 남기고(사람이 고쳐도 유지), 그런 초안의 확정·실행 작업 로그는 AI 경유로 표시해 나중에 추적할 수 있게 했습니다.

트랜잭션

경고 5건 뒤에 있던 쿼리 겹침 789회

문제

테스트에서 pg 드라이버 경고 `client is already executing a query`가 나왔습니다. 다음 메이저 버전에서는 오류로 바뀌는 경고였습니다.

원인

Prisma 트랜잭션 안에서 `include`로 여러 관계를 한 번에 읽으면 한 연결에 쿼리가 동시에 들어갔습니다. 경고는 테스트 파일마다 한 번만 찍혀 5건으로 보였지만, 쿼리 호출을 가로채 세어 보니 실제로 겹친 횟수는 789회였습니다.

해결·결과

트랜잭션 안에서는 관계를 하나씩 읽고 응답 모양을 만드는 조회는 커밋 뒤로 옮겼습니다. 겹침 조건을 직접 세는 테스트(수정 전 51회로 실패, 수정 후 0회)로 고정했고, 같은 원인이 있는 다른 모듈의 위치를 표로 정리해 팀에 공유했습니다.

문제	개발 서버 실행 명령이 <code>DATABASE_URL</code> 이 가리키는 DB에 마이그레이션과 시드를 자동으로 실행해, <code>.env</code> 를 공용 DB로 바꾼 사람이 병합 전 브랜치에서 서버를 켜면 그 변경이 팀 공용 DB에 들어갈 수 있었습니다.
해결	대상 DB가 로컬인지 판정하는 스크립트로 마이그레이션과 시드를 감싸고, 공용 DB 반영은 develop 브랜치, 깨끗한 작업 트리, 원격과의 일치를 확인하는 명령(<code>db:deploy</code>) 하나로만 하게 했습니다.
결과	원격 주소로 실행하면 막히는 것을 확인했고, 그 과정에서 로컬(PostgreSQL 18)과 공용 DB(17)의 버전 차이를 찾아 리뷰 체크리스트에 넣었습니다.

스키마 변경과 데이터 검증

NOT NULL 칸 추가	기존 행이 있는 업무 테이블에 필수 칸(등록자)을 더할 때, 바로 NOT NULL로 추가하는 자동 생성 마이그레이션 대신 빈 칸 허용으로 추가 → 기존 행을 담당자로 채움 → NOT NULL 순서로 나눴습니다. ERD와 용어 사전을 먼저 고친 뒤 반영했습니다.
업무 불변식 점검	입력 검증이 빠졌던 동안 들어간 데이터를 찾으려고, 재고 수량과 LOT 수, 예약 합계와 주문 수량, 히트 톤과 용량, 설비 작업 시간 겹침 같은 11가지 규칙을 SELECT만으로 점검하는 스크립트를 만들었습니다. 위반이 있으면 종료 코드 1을 냅니다. 공용 DB에서 투입 순서 위반 20건과 설비 시간 겹침 61쌍을 찾았습니다.
운영 이력 시드	발표·시연용으로 3주차 거래·협업 이력을 넣는 시드를 만들었습니다. 전역 시계를 바꿔 실제 서비스 코드를 날짜 순서대로 불러, 생성 시각과 날짜가 들어간 번호까지 그 시점으로 남깁니다. 넣은 뒤 불변식 11개와 참조 관계 99개 점검을 위반 0으로 통과했습니다.

계산과 권한

MRP 계산	생산 계획의 히트 톤, 제강 수율, 원단위로 원료 소요량을 구하고 원료 LOT 잔량과 입고 예정을 빼 순소요량을 계산합니다. 같은 공급을 두 계획에서 중복으로 빼지 않도록 공급 사용 순서를 정했고 Decimal로 계산한 뒤 마지막에 한 번만 반올림합니다.
권한	승인은 권한 코드가 아니라 요청자 부서의 부서장인지로 판단하고 본인 요청은 승인할 수 없습니다. 권한은 요청마다 DB에서 다시 읽어 변경이 바로 반영됩니다.

CI와 배포

CI	PR마다 GitHub Actions가 PostgreSQL 17 서비스에 마이그레이션을 적용하고 타입 검사, 서버 단위 테스트, 화면 테스트, 서버 DB 테스트 7묶음을 돌립니다. 손으로 돌리던 검사를 빠뜨려 develop이 깨지던 일을 막으려고 만들었습니다.
자동 배포	main 병합 때 러너가 Tailscale로 사설망에 들어가 EC2에 SSH하는 배포 워크플로를 만들었습니다(SSH 포트는 밖에 열지 않음). 배포 스크립트는 반영되지 않은 마이그레이션이 있으면 멈추고, 빌드 뒤 <code>pm2 reload</code> 와 헬스 확인까지 합니다. 사설망 설정 뒤 첫 실행을 앞두고 있습니다.



로컬 개발 환경에서 시드 데이터로 실행해 캡처했습니다. 제가 맡은 부분은 이 화면들이 쓰는 구매, MRP, 조직, 알림 API입니다.

대시보드

MRP

구매요청

발주

입고

재고

생산계획

기준정보

조직도

LOT 추적

작업 로그

업무 알림

메신저

구매 - 10-13 (화) 15:08

입고

Q 수주번호·LOT 번호·출하요청 번호 검색

AI AI 어시스턴트

4

정다운

구매부 · 구매

입고예정 1건

발주

발주번호·원료·공급업체 검색

입고예정 1 전체 9

PO-2610-0009 PR-2610-0013

부분 입고

입고 예정 10-18

실리코망가니즈 · 하람합금철

입고예정 3.000 t

최근 입고

GR-2610-0009

5.000 t

실리코망가니즈 · 하람합금철 · 2026-10-13

RM-SMN01-261013-001

GR-2610-0008

20.000 t

실리코망가니즈 · 하람합금철 · 2026-10-10

RM-SMN01-261010-001

GR-2610-0007

20.000 t

실리코망가니즈 · 하람합금철 · 2026-10-06

RM-SMN01-261006-001

GR-2610-0006

150.000 t

석회석 · 소담광물 · 2026-10-10

RM-LIM01-261010-001

GR-2610-0005

150.000 t

석회석 · 소담광물 · 2026-10-06

RM-LIM01-261006-001

입고 등록이 곧 입고 확정이에요. 원료 LOT이 바로 만들어지고 고칠 수 없어요. 입고 검사는 하지 않아요.

입고

PO-2610-0009 PR-2610-0013

부분 입고

실리코망가니즈 SMN01 · 하람합금철 · 입고 예정 2026-10-18 · 구매요청 PR-2610-0013

발주 8.000 t

입고 누계 5.000 t

발주 대비 63%

입고예정 3.000 t

발주 - 입고 누계 · 나눠서 입고할 수 있어요

입고 등록 등록하면 바로 확정돼요

입고 톤 * 3 t

입고일 * 2026-10-13

미입고량 이하 · 소수 3자리까지

오늘까지 · 원료 LOT 번호의 날짜가 돼요

아드

원료 1아드

원료의 기본 아드로 자동 지정돼요

확정한 원료 LOT이 만들어지고 재고-MRP 잔량에 바로 반영돼요. 확정된 입고는 고칠 수 없어요.

입고 확정

원료 LOT 확정할 때 생성

RM-SMN01-YYMMDD-NNN

번호는 입고를 확정할 때 매겨요 (YYMMDD = 입고일) · 입고 1건마다 LOT 1개 · 잔량 = 입고 톤

입고 내역

이 발주 품목 전체

입고번호	발주번호	원료	입고일	입고 톤	아드	원료 LOT	확정
GR-2610-0009	PO-2610-0009	실리코망가니즈	2026-10-13	5.000 t	원료 1아드	RM-SMN01-261013-001	정다운 · 10-13 15:08

입고 발주 8t 중 5t이 들어온 부분 입고 상태입니다. 입고를 확정하면 발주 행을 잠그고 미입고량을 다시 확인한 뒤 원료 LOT을 만듭니다.

대시보드

MRP

구매요청

발주

입고

재고

생산계획

기준정보

조직도

LOT 추적

작업 로그

업무 알림

메신저

구매 - 10-13 (화) 15:09

구매요청

Q 수주번호·LOT 번호·출하요청 번호 검색

AI AI 어시스턴트

4

정다운

구매부 · 구매

구매요청 13 / 13

+ 구매요청 등록

요청번호·원료·요청자 검색

전체 13 승인 대기 2 승인 1 반려 1

발주 완료 9

출처 MRP 계획 0 Message → ERP 0

직접 13

내 요청만 보기

PR-2610-0013

발주 완료

희망 10-18

실리코망가니즈 · 8.000 t

정다운 직접 승인 최종력 발주 완료

PR-2610-0012

반려

희망 10-21

실리코망가니즈 · 50.000 t

정다운 직접 반려 최종력 고쳐 다시 요청 필요

PR-2610-0011

승인

희망 10-22

철광석 · 800.000 t

정다운 직접 승인 최종력 10-13 · 발주 대기

PR-2610-0010

승인 대기

희망 10-25

석탄 · 300.000 t

정다운 직접 구매부 부서장 승인 대기

PR-2610-0009

승인 대기

희망 10-23

석회석 · 120.000 t

정다운 직접 구매부 부서장 승인 대기

PR-2610-0008

발주 완료

희망 10-10

실리코망가니즈 · 20.000 t

정다운 직접 승인 최종력 발주 완료

구매요청

PR-2610-0013

발주 완료 직접

상세 열기

실리코망가니즈 · 8.000 t · 정다운 · 2026-10-13 15:05 등록

등록

부서장 승인

발주

발주 완료 · PO-2610-0009

요청 내용 직접

원료 실리코망가니즈 SMN01

수량(톤) 8.000 t

근거 생산계획 -

발주 PO-2610-0009

희망 입고일 2026-10-18

요청자 정다운 사원 · 구매부

승인권자 최준혁

요청 근거 시연용 시드 (구매 화면)

승인 일시 2026-10-13 15:05

출처 직접

요청자가 직접 등록한 요청이에요.

연결 발주 입고 5.000 t / 발주 8.000 t

발주번호	원료	공급업체	발주	입고	입고예정	입고 예정일	상태
PO-2610-0009	실리코망가니즈	하람합금철	8.000 t	5.000 t	3.000 t	2026-10-18	부분 입고

구매요청 등록, 부서장 승인, 발주로 이어지는 상태 흐름입니다. 승인과 반려는 현재 상태가 기대한 상태일 때만 바꾸는 조건부 UPDATE로 처리합니다.



PitchLog League

유럽 5대 리그와 UEFA 대회, 국내 컵까지 19개 대회의 최근 5시즌 경기·선수 데이터를 외부 API에서 모아 제공하는 축구 데이터 서비스입니다. 수집 배치, 조회 REST API, 라이브 스코어 갱신, AI 어시스턴트, 배포와 백업까지 혼자 설계하고 운영합니다.

pitchlog-league.vercel.appgithub.com/donasman/pitchlog-league

규모

13,724 경기

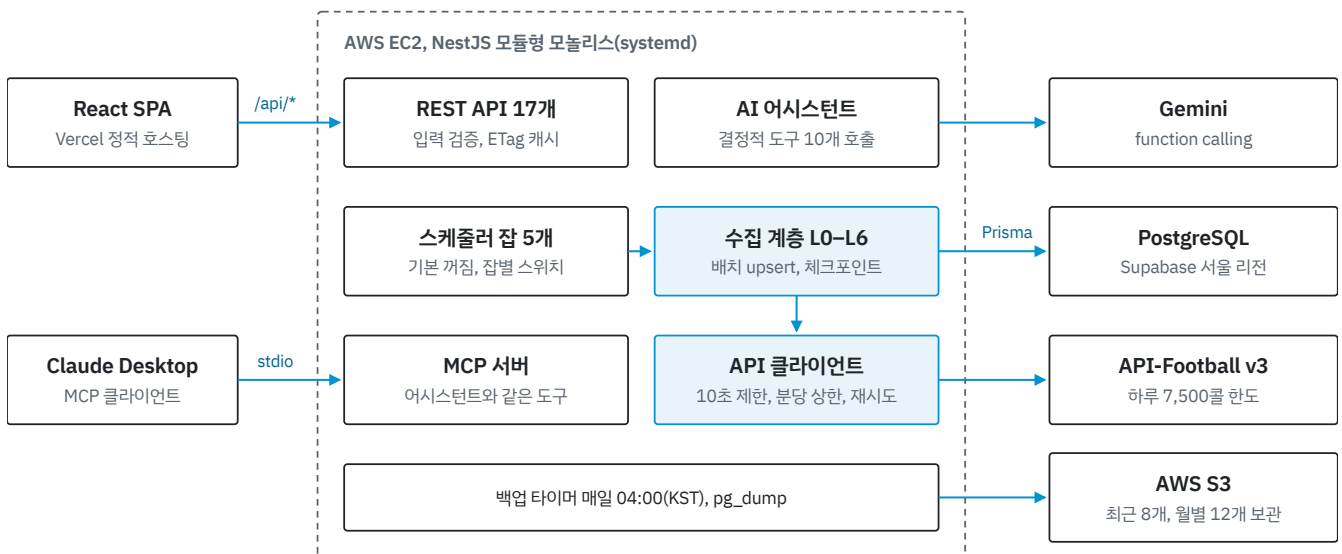
2,110 팀

13,608 선수

148 PR

약 500 백엔드 테스트

데이터 2026.09.17 실측



핵심 설계 결정

외부 API 보호

외부 호출을 한 클래스(`ApiFootballClient`)로 모으고 10초 제한 시간, 분당 자체 상한, 네트워크 오류·429·5xx에만 1, 2, 4초 간격 재시도를 걸었습니다. 응답이 200이어도 오류 필드가 있으면 실패로 처리하고 하루 한도에 닿으면 즉시 멈춰 '조용한 실패'를 막았습니다.

큐 없는 백필

약 4만 3천 콜 규모의 경기 상세 백필을 메시지 큐 대신 `backfill_jobs` 테이블의 커서와 단일 루프로 처리했습니다. 하루 자율 상한(환경변수, 기본 5,700콜)과 중단 사유를 기록해 멈춘 지점부터 이어 가고, 라이브 수집은 백필이 끝난 뒤 커서 같은 한도를 두고 다투지 않게 했습니다.

DB 설계

"대회당 현재 시즌은 하나" 같은 규칙은 부분 unique 인덱스로 DB에서 강제했습니다. 대량 적재 속도를 위해 외래 키를 두지 않는 대신 참조 컬럼 인덱스 69개와 고아 행 검사(CI, 매일)로 무결성을 확인하고, 테스트는 운영과 같은 PostgreSQL에서 돌립니다.

행 단위 upsert가 원격 DB 왕복에 막힌 문제

문제
행마다 upsert하자 원격 DB 왕복 때문에 5초 트랜잭션 한도에 걸렸습니다. 20개 팀을 저장하는 데만 약 4초가 걸려 수십만 행 규모의 백필은 불가능했습니다.

원인
행마다 약 60ms의 네트워크 왕복이 생겼고 ORM의 upsert는 경우에 따라 조회와 저장 두 번의 쿼리로 나갔습니다.

해결
- 테이블당 한 문장의 `INSERT ... ON CONFLICT DO UPDATE`로 묶는 `batchUpsert`를 만들고, 충돌 키 중복 제거, 명시적 타입 변환, 파라미터 65,535개 한도에 맞춘 분할, `COALESCE`로 기존 값 보호까지 처리했습니다.
- DB 리전을 앱 서버와 같은 서울로 옮겼습니다.

결과
배치 upsert로 대화·시즌 하나당 쿼리가 3문장으로 줄었고, 리전 이전과 합쳐 기초 데이터 적재 시간이 200초에서 71초로 줄었습니다. 경기 13,724건과 상세 백필을 무인으로 끝냈습니다.

라이브 스코어 역행과 동시 저장 충돌

문제
15초 주기 라이브 폴러와 매일 도는 일정 수집 잡이 같은 경기 상태를 덮어써 진행 중인 스코어가 이전 상태로 되돌아갈 수 있었습니다. 즐겨찾기를 동시에 저장하면 unique 충돌로 500 오류가 났습니다.

해결
- 상태 순위와 경과 시간을 `WHERE` 절에서 비교하는 조건부 `UPDATE` 한 문장으로 바꿔, 읽고 쓰는 사이의 경합 없이 역행하는 쓰기를 막았습니다. 쓸 때마다 `data_version`을 올립니다.
- 잡마다 `matches` 테이블에서 쓸 수 있는 컬럼을 나눠 서로 덮어쓰지 않게 했습니다.
- 즐겨찾기 저장은 트랜잭션 안에서 사용자 단위 `pg_advisory_xact_lock`으로 직렬화했습니다.

결과
막힌 쓰기는 `/health`에 차단 건수로 집계되고 동시 저장 시나리오는 테스트로 검증합니다.

운영

백업과 복원
매일 04:00(KST)에 `pg_dump`로 백업해 S3에 올리고 최근 8개와 월별 12개를 보관합니다. 일회용 컨테이너에 복원해 원본과 행 수를 대조하는 리허설에서 31개 테이블 93,337행이 두 번 연속 일치했습니다.

배포
필수 환경변수가 빠진 배포로 서비스가 재시작을 반복했고 이어서 재배포한 깨진 빌드가 롤백본을 덮어쓴 일이 있었습니다. 이후 배포 스크립트가 헬스체크를 통과한 버전만 롤백본(`dist.prev`)으로 남기도록 고쳤습니다.

CI
빈 PostgreSQL 16에 마이그레이션을 적용하고 부분 인덱스 확인, 타입 검사와 린트, 단위 테스트 약 300개와 E2E 테스트 약 200개, 고아 행 검사를 통과해야 병합됩니다. 작업 브랜치에 보호 규칙을 걸고 PR 148개 중 144개를 병합했고, 10월 16일 dev에서 main으로 첫 릴리즈를 냈습니다.

운영 중 수정
id가 없는 선수 데이터가 다른 선수와 합쳐지는 것을 막고, 상세 백필이 같은 경기에서 끝없이 재시도하지 않게 경기 단위 재시도 상한을 두는 등 운영 중 찾은 수집 문제를 고쳤습니다.

핵심 기능

PitchLog League

2026.09.27 기준

Gemini 3.5 Flash, Flash-Lite,
function calling, NestJS, ajv



AI 어시스턴트

LLM은 도구를 고르기만 하고, 숫자는 화면과 같은 조회 API에서 가져옵니다. 답변마다 근거 도구와 기준 시각을 붙입니다.

POST /api/assistant

규모

10 도구

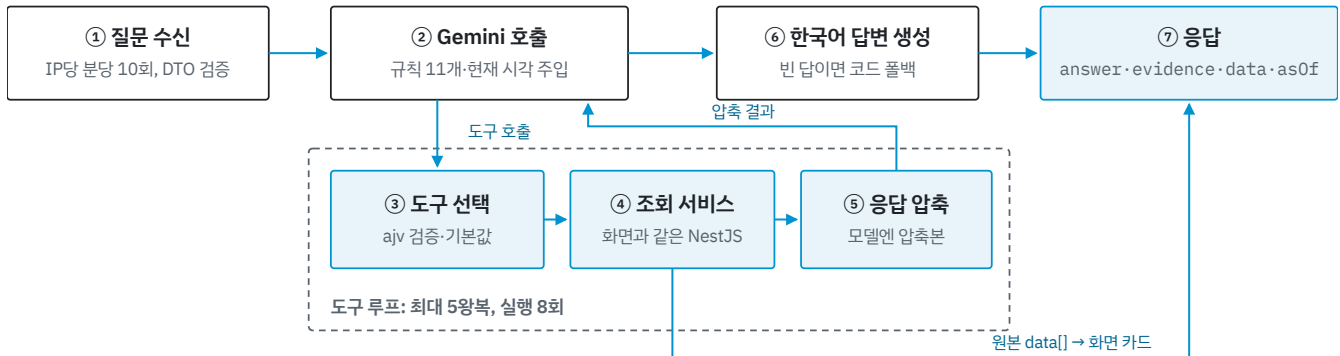
11 규칙

24 golden 케이스

65% 응답 압축

5·8 왕복·실행 상한

75초 타임아웃



모델에는 압축본(로그·홈/원정 분할 제거, null 유지)을, 화면에는 원본 data[]를 그대로 보내 순위표·경기 카드로 그림니다.

도구 10개

대회·팀·경기·순위·랭킹·선수 조회 도구입니다. 설명마다 "0은 실제 0회, null은 측정 안 됨"을 붙였고, 대회 목록은 프롬프트에 하드코딩하지 않고 도구를 정보으로 씁니다.

환각을 막는 규칙

11개 중 핵심

- 도구 결과에 없는 숫자·순위·기록은 쓰지 않고, "압도적"·"부진" 같은 판정 대신 조회된 수치만 씁니다.
- 답할 수 없으면 네 갈래로 나눠 답합니다. 도구 없음 "우리 데이터에 없습니다", 미확정 "아직 확정되지 않았습니다", 필드 null "제공되지 않습니다", 잘못된 전제 "그런 것은 없습니다".
- 합계·차이는 계산 과정을 함께 적고, 나이·D-day처럼 직접 계산해야 하는 값은 만들지 않습니다.

검증

golden 회귀 24케이스. 근거에 없는 숫자(환각)·빈 답변·없는 도구명은 hard, 도구 호출 횟수 차이 같은 LLM 비결정성은 soft로 판정합니다. 면제 목록을 두지 않은 건 면제가 늘면 golden이 무력화되기 때문입니다.

안전장치

- 로그의 API 키·Bearer 마스킹, 빈 답일 때 코드 폴백("요청하신 정보를 찾지 못했습니다").
- POST 응답에 no-store: 옛지 60초 캐시가 다른 질문에 첫 답을 재사용하던 결함을 고쳤습니다.
- MCP 서버(stdio)로 같은 도구 10개를 Claude 등 MCP 클라이언트에도 노출합니다.

비용·한도 실측

응답 압축 45~78%, 그리고 429의 실제 원인

측정

순위표 6대회(120행) 도구 응답이 76KB에서 27.6KB로, 세션당 비용이 \$0.024에서 \$0.006으로 줄었습니다. 질문 1개에 Gemini 요청은 도구 루프 때문에 3~6회 나가서, X-Gemini-Requests 헤더로 따로 셉니다.

429 원인

페이로드 크기를 의심했지만 로그를 확인해 보니 무료 티어의 하루 요청 한도(RPD)였습니다. 압축은 비용에는 효과가 있었지만 429에는 효과가 없었고, RPD를 다 쓰면 Flash와 Flash-Lite를 교체합니다.



PitchLog League 화면

운영 중인 서비스(pitchlog-league.vercel.app)에서 캡처했습니다. 기획, 백엔드, 프론트엔드, 운영을 모두 직접 했습니다.

The screenshot shows the match page for Coventry vs Newcastle United. The score is 4-1 at the 90th minute. The page includes a line-up section, a match timeline, and a player profile for Nico González. The line-up shows Coventry in a 4-2-3-1 formation and Newcastle in a 4-2-3-1 formation. The match timeline shows the score changing from 0-0 to 4-1. The player profile for Nico González shows he played 90 minutes, scored 0 goals, and had 68 passes.

경기 상세 라인업, 경기 통계, 타임라인은 경기 상세 백필로 채운 데이터입니다. 진행 중인 경기를 15초 주기로 갱신하는 폴러를 만들었고 이전 상태로 되돌아가는 쓰기는 조건부 UPDATE로 막습니다.

The screenshot shows the league table for the Premier League. The table lists the top 11 teams, their positions, and their statistics. The teams are ranked by points, with Manchester City at the top. The table includes columns for matches played, wins, draws, losses, goals scored, goals conceded, goal difference, and points. The last column shows the last 5 match results for each team.

#	팀	경	승	무	패	득	실	득실	승점	최근 5경기
1	맨체스터 시티	6	6	0	0	16	5	+11	18	W W W W W
2	아스널	6	5	0	1	10	5	+5	15	W W W L W
3	브라이턴	6	4	1	1	18	5	+13	13	L D W W W
4	브렌트퍼드	6	2	4	0	12	6	+6	10	D D D W D
5	에버턴	6	2	4	0	8	5	+3	10	D D D W D
6	첼시	6	3	1	2	15	13	+2	10	W L D L W
7	리즈 유나이티드	6	2	3	1	8	5	+3	9	D D W D L
8	헐 시티	6	2	3	1	8	6	+2	9	W D D L D
9	리버풀	6	2	3	1	7	7	0	9	D W D W L
10	임스위치 타운	6	3	0	3	9	12	-3	9	L L L W L
11	뉴캐슬 유나이티드	6	2	2	2	10	13	-3	8	W L D L W

순위 대회와 시즌별 순위표입니다. "대회당 현재 시즌은 하나" 같은 규칙은 부분 unique 인덱스로 DB가 지킵니다.

개발 방식

PitchLog League

2026.09.28 기준

Claude Code CLI(에이전트

10, 혹 2, 스킵 4),

Cowork(Claude 데스크톱)



AI와 함께 개발하기

구현은 Claude Code 에이전트가 하고, 무엇을 만들지 정하고 승인하는 일은 제가 합니다. 코드를 쓴 AI와 검증하는 AI를 나누고, 구현 전에는 반드시 사람이 확인하는 단계를 두었습니다.

[docs/AGENT_WORKFLOW.md](#)

[docs/AGENT_RUNS.md](#)

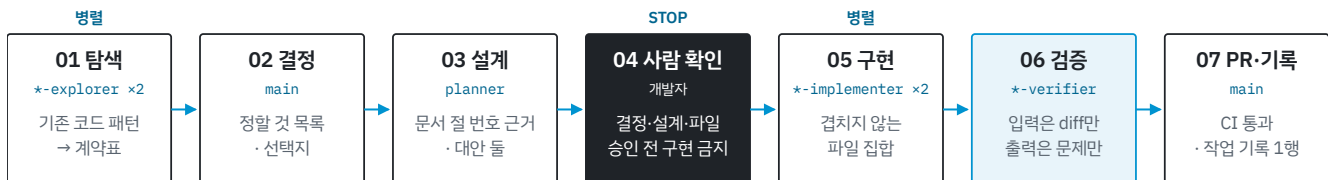
규모

431 커밋

64 작업 기록

10 에이전트

3 겹 안전장치



작업 한 번(판)의 흐름입니다. 04에서는 반드시 멈추고, 판마다 규모·에이전트 토큰·사람 시간을 작업 기록에 한 줄씩 남깁니다.

한각을 막는 규칙 4

- 읽지 않은 것은 없는 것 함수·컬럼·환경변수는 grep으로 확인하고 보고에 파일:줄을 인용합니다.
- 쓴 에이전트가 검증하지 않음 검증은 diff만 받은 별도 에이전트가 합니다.
- 판단 근거는 문서 절 번호 규칙을 기억이 아니라 파일에서 읽게 합니다.
- 테스트가 진실 "통과할 것"은 보고에 못 쓰고, 통과한 출력만 씁니다.

지시문 6칸

목표(완료 정의 한 문장), 읽을 것(문서 절 번호·파일 경로), 건드릴 파일(목록 밖 수정 금지), 금지(Mock 수정, dev 직접 커밋 등), 완료 조건(실행할 명령과 기대 출력), 보고 형식(바꾼 파일·근거·출력·못 한 것). 빈 칸이 있으면 보내지 않습니다.

서로 견제

CLI가 구현하면 Cowork(Claude 데스크톱)가 검증하고, 머지는 제가 승인합니다. Cowork는 저장소를 클론해 verify를 재현하고, Playwright로 화면을 렌더하고, 배포 API를 실제로 호출해 확인합니다.

작업 크기별 비용

09-15 실측

큰 판 01~07 전 단계



~700k

작은 판 결정 사전 기입



~141k

에이전트 토큰 기준. 사람 시간도 약 3시간에서 약 30분으로 줄어, 결정을 미리 적어 두면 작은 판으로 돌립니다.

안전장치

한 층의 구멍을 다른 층이 막는 3겹

3겹

- Claude Code 혹 도구 호출 직전에 차단: `--no-verify`, `reset --hard`, 강제 push, 보호 경로 쓰기
- git pre-commit 사람 커밋도 차단: API 키 형식(AIza...), `.env`, 깨진 UTF-8
- 브랜치 보호 + CI dev-main 직접 커밋 금지, PR 필수, 백엔드·프론트 검증 통과가 머지 조건

실제로 걸린 구멍

pre-commit이 변수 이름만 검사해 Gemini 키를 놓쳐 키 형식 검사를 추가했습니다. 읽기 전용 에이전트도 Bash의 `python -c` heredoc으로 파일을 쓸 수 있다는 것을 확인하고 혹에서 막았습니다.

실무

2024.09 - 2025.08
12개월

아이티커넥 소속

프리랜서(주임)

응용SW개발

발주

주택도시보증공사

Java(Spring Legacy),

PostgreSQL, Python,

QGIS, ArcGIS



도시재생 종합정보체계 유지관리

주택도시보증공사가 발주한 공공 포털 유지관리 사업에서 수행사 소속으로 Spring Legacy·PostgreSQL 백엔드의 장애 대응과 유지보수, 매년 돌리는 데이터 갱신 배치를 맡았습니다. 전국 도시재생 사업지와 토지·건축물 데이터를 다룹니다.

city.go.kr

규모

약 3,000만 건 건축물대장

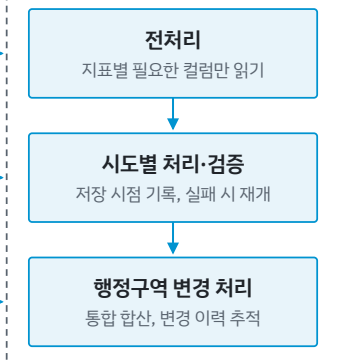
5 개 진단 지표

3 단계 집계(시군구·읍면동·격자)

원천 데이터



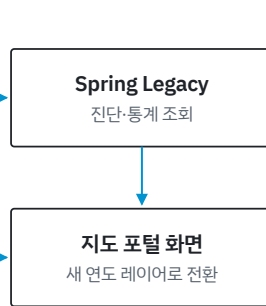
연계 갱신 배치(Python)



저장·배포



서비스



색칠한 부분은 매년 갱신에서 직접 만든 부분입니다. 지도 레이어는 ArcGIS로 배포해 화면 코드에서 바로 호출합니다.

장애 원인 추적

지도 레이어를 골라도 안 보이던 오류

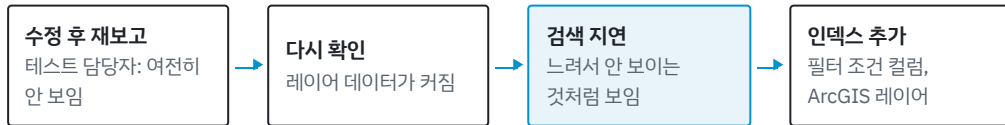
원인 추적

하나를 고친 뒤에도 같은 증상이 남아 원인을 두 번에 걸쳐 찾았습니다.

첫 번째 원인



두 번째 원인



결과

레이어 검색 속도가 빨라져 선택하면 바로 지도에 나타납니다.

문제

전년도 코드가 그대로 돌지 않았고, 지표마다 코드가 따로 실행돼 실패하면 어디서 왜 멈췄는지 알기 어려웠습니다. 멈추면 3,000만 건을 처음부터 다시 돌려야 했습니다.

재설계

이전

- 지표마다 코드를 따로 실행
- 실패 위치와 원인이 남지 않음
- 멈추면 처음부터 다시 실행

재설계 후

- 필요한 컬럼만 읽고 실행 코드를 한 흐름으로
- 시도별로 적재 전에 검증하고 저장 시점 기록
- 실패 원인을 로그로 남기고 마지막 저장 시점부터 재개
- 다시 실행해도 같은 결과

재실행 예시



결과

중간에 멈춰도 3,000만 건을 처음부터 다시 돌리지 않게 됐고 다음 해 갱신에도 그대로 쓸 수 있게 정리해 두었습니다.

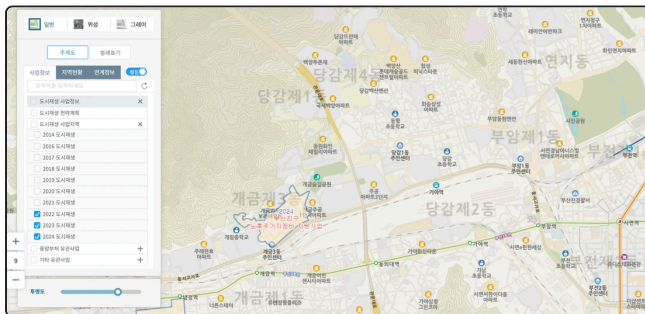
그 밖에 맡은 일

진단 로직 수정

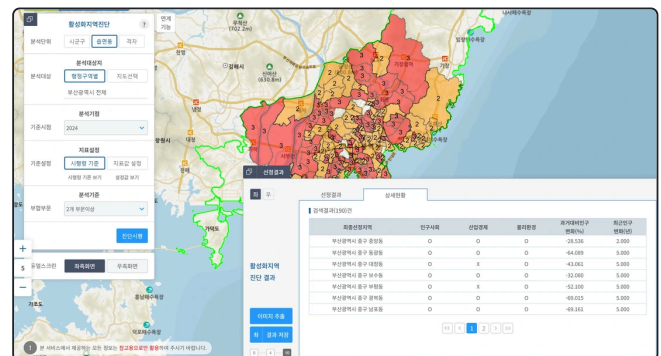
활성화지역진단의 부합 판정에서 경계값, 값이 없는 지역, 지표값 설정 처리 오류를 고쳤습니다. 갱신 뒤에는 시군구·읍면동·격자 값과 지도·표·차트의 건수를 대조해 어긋난 곳을 바로잡았습니다.

행정구역 변경 기준

여러 해를 비교하는 지표가 행정구역 통합·분리나 이름·코드 변경 때 끊기지 않도록 변경 유형을 통합(이전 구역 값 합산), 분동·신설(해당 연도부터 새로 기입), 명칭·코드 변경(이력 추적해 하나로 통합) 셋으로 나눠 갱신에 적용했습니다.



레이어 선택 연도별 도시재생 사업지역 레이어를 골라 띄운 화면입니다 (부산진구 일대). 이렇게 고르는 조건이 걸리는 컬럼에 인덱스를 줍니다.



활성화지역진단 부산광역시 읍면동을 2024년 시행령 기준으로 진단한 화면 (2026.10 캡처)입니다. 표의 지표 다섯 개를 연례 갱신에서 가공했습니다.

이 일에서 배운 것

기준을 먼저

반복되는 데이터 작업일수록 처리 기준을 먼저 정해 문서로 남겨야 매번 같은 결과가 나옵니다.

모르는 분야

GIS처럼 처음 보는 분야도 문서와 커뮤니티로 익혀 실무에 적용했습니다. 이 경험이 이후 축구 데이터, 철강 공정처럼 낯선 도메인의 프로젝트를 시작할 때 기준이 됐습니다.

팀 프로젝트

코리아IT아카데미

2024.04 – 2024.05

3인, 팀장

Java 11, Spring Boot 2.6,
MyBatis, MySQL, React 18,
Docker, Nginx, GitHub
Actions, AWS EC2와 S3



과외나무 LessonTree

학생과 선생님을 이어 주는 과외 매칭 웹 서비스입니다. 팀장으로 이슈와 PR 흐름을 운영했고, 조회·검색 API와 무중단 배포 파이프라인을 직접 만들었습니다.

github.com/test-2024-0402-team4

담당

학생 공고와 선생님 프로필 조회·검색, 과외 신청과 프로필 메일, 관리자·신고·계정 비활성화, CI/CD

협업

이슈 → 브랜치 → PR → 팀장 병합 흐름을 운영했습니다. 두 저장소 PR 172개 중 65개를 작성하고 152개를 병합했습니다.

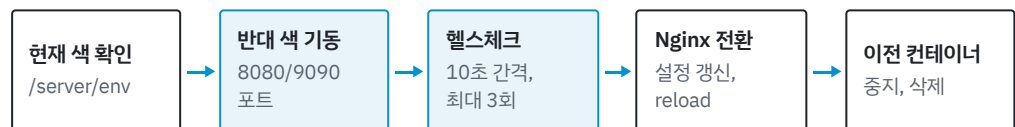
무중단 배포

main에 반영되면 GitHub Actions가 빌드한 이미지를 운영 중이 아닌 쪽 포트에 띄우고 헬스체크를 통과하면 Nginx 대상을 바꿉니다. 적용 하루 전에 별도 저장소에서 같은 구조를 먼저 검증했습니다.

GitHub Actions에서 빌드



AWS EC2에 배포



다중 조건 검색

지역, 과목, 요일, 학생 유형 조건을 MyBatis 동적 SQL로 조합해 공고 ID만 먼저 뽑고 그 ID로 과목·요일 같은 1:N 상세를 한 번에 조회했습니다. 조건 필터가 상세 목록을 잘라내지 않고 비활성 회원의 공고는 빠지게 했습니다.

돌아보면

신고와 계정 비활성화를 기획 단계에서 빠뜨려 개발 중에 테이블과 컬럼을 추가하고 계정 기능을 고쳐야 했습니다. 그 뒤로는 기능 목록을 먼저 확정하고 DB를 설계합니다. 배포 워크플로는 PR 이벤트에서도 배포 단계가 돌도록 되어 있었는데, 지금이라면 push 이벤트로만 제한하겠습니다.