

김건우 빌더

불편을 기술로 해소합니다.

포트폴리오

geonwoo.kim
contact@geonwoo.kim
github.com/takealook97

가치관

- AI 네이티브를 지향합니다. 특정 LLM에 종속되지 않는 벤더 중립을 중시합니다.
- 실현되지 않은 좋은 아이디어보다, 부족하지만 실현된 아이디어가 더 높은 가치를 가진다고 믿습니다.
- 혼자 빠르게 가는 것보다, 함께 멀리 가는 것에 가치를 둡니다.

경험

- 메시지 발송 서비스와 Shopify 기반 이커머스에서 백엔드 개발을 맡아 왔습니다.
- 연동 서버와 배포 자동화, 사내 업무 시스템까지 필요한 것들을 직접 만듭니다.
- Proxmox 기반 홈서버에서 나만의 서비스를 만들어가고 있습니다.

현재

- 루쿠쿠에서 여러 브랜드의 Shopify 스토어 백엔드를 맡아 ERP 연동, 커스텀 앱 등을 개발하고 있습니다.
- 디핏의 에이전트 운영 체계를 설계하고 구축하고 있습니다.
- 자체 개발 도구를 팀에 도입하고, 오픈소스로 만들고 있습니다.

개발

- Claude Code, Codex, Gemini CLI를 주력으로 멀티 에이전트 파이프라인을 구축해 개발합니다.
- tmux, Herdr 같은 터미널 멀티플렉서 환경에서 여러 에이전트를 병렬로 운영합니다.
- 작업 배정과 검토는 오케스트레이터 orch가, 여러 저장소의 변경 기록과 메모리 관리는 vat이 맡습니다.

학력

동국대학교 · 국제통상학과 학사 — 2017.03 — 2023.02

김포외국어고등학교 · 영어과 — 2013.03 — 2016.02

자격

OPIc IH · ACTFL — 2025.09

SQLD · 한국데이터산업진흥원 — 2024.04

교육

삼성 청년 SW 아카데미(SSAFY) · Java 전공 과정 — 2023.07 — 2024.06
Java, Spring Boot, MyBatis, DB, 알고리즘, Vue.js, 웹 개발 · 팀 프로젝트 5개 이상 참여, 3개에서 팀장

코드스쿼드 · 백엔드 코스 — 2023.01 — 2023.06
CS, Java, Spring, DB, 팀 프로젝트 등

활동

노션 엠버서더 — 2024.07 — 2025.07

- 글로벌 엠버서더 선정
- Notion API로 워크플로를 자동화하는 아티클 작성
- 해외 엠버서더들과 베타 테스트·QA·회의 참여

수상

프로젝트 우수상 · SSAFY — 2024.02
공통 프로젝트 1위

목차

루쿠쿠

space-I · 고객사 전용 OS 2026.10 — 현재 3
미국 법인 Shopify B2B 스토어 구축 · B2B 백엔드 2026.09 — 현재 4
루쿠쿠 OS · 회사 온톨로지와 에이전트 운영 체계 2026.09 — 현재 5
설치미술 전시를 위한 360도 영상 앱과 AI 영상 생성 서버 2026.06 — 2026.08 7
스포츠 브랜드 국내 자사몰과 ERP의 판매 대사·연동 운영 2026.04 — 현재 8
Hermes Agent 기반 사내 운영 에이전트 2026.04 — 현재 9
오프온 · 전사 업무 관리 시스템 2026.04 — 현재 11
스포츠 브랜드 국내 자사몰 · 대회 참가자 전용 우대권 주문 2026.04 — 2026.09 12
브랜드 Shopify 스토어 기능 개발과 연동 2026.04 — 2026.07 13

비즈톡

AWS MSK 3.8 마이그레이션 2025.08 — 2025.09 14
BGMS 로그 마스킹 모듈 2025.07 14
채권 미납 안내 RPA 2025.06 — 2025.07 15
BGMS 메시지 발송 시스템 개발과 운영 2025.05 — 2025.10 16
배포 파이프라인 구축 및 사내 GitLab 마이그레이션 2025.05 — 2025.10 16
서버 설정 관리와 배포 자동화 스크립트 2025.05 — 2025.10 17
카카오톡 발송·템플릿 통계 수집 서버 2025.05 — 2025.06 17

디핏

에이전트가 사업을 운영하는 AI 네이티브 커머스 2026.07 — 현재 18
광고 운영 2026.10 19
가격·프로모션·정산 2026.08 20
라우팅·판매 상태 제어 2026.07 — 2026.09 21
에이전트 운영·조직 기억 2026.07 — 2026.08 22
소싱 데이터 2026.07 — 2026.08 24
개발·인프라 운영 2026.07 — 2026.10 25
운영 콘솔·고객 응대 2026.07 — 2026.08 26
상품 등록 2026.07 — 2026.08 27
발주 자동화 2026.07 — 2026.08 28

개인

보르테소 · 커피 기록 앱 2025.10 — 2026.03 30
글로티커 · 거시경제 지표 크롬 익스텐션 2025.03 — 2025.05 31
언더더키 · 커스텀 키보드 플랫폼 2024.08 — 2024.11 33

SSAFY

서울팝 · 위치 기반 역사 탐방 2024.04 — 2024.05 36
리릭잇 · 노래 가사 기반 실시간 게임 2024.02 — 2024.04 38
명절 잔소리 영수증 2024.01 — 2024.02 40
뉴집스 · 부동산 모의투자 2023.11 42
트리플랫 · 여행 금융 다이어리 2023.09 44

도구와 오픈소스

herdr-cockpit · orch 작업을 보는 Herdr 레이아웃과 모드 2026.11 — 현재 46
wiki-island · 벤더 중립 AI 메모리 2026.11 — 현재 47
orch · 멀티 에이전트 오케스트레이터 2026.09 — 현재 48
hermes-agent · Hermes Agent 오픈소스 컨트리뷰션 2026.08 — 현재 49
vat · 상위 폴더 하나에서 여러 저장소를 개발하는 작업 공간 CLI 2026.08 — 현재 50
offon · 출퇴근·휴가를 관리하는 오픈소스 슬랙 앱 2026.04 — 2026.08 51
sshift · 점프 서버 접속과 비밀번호 저장을 묶은 SSH CLI 2025.07 — 2025.08 52

ᓚᘏᗢ 루쿠쿠

space-I · 고객사 전용 OS

기간	2026.10 — 현재
팀	4명 서버 담당, 프론트엔드 공동
역할	구조 설계　Go API 서버　배포 체계
기술	Next.js · TypeScript · Go · Vercel · AWS EKS · AWS S3

해외 3개 시장의 광고 성과와 일별 매출을 한 화면에

진행 상황을 문서와 메신저로 따로 전하던 일을 한곳에 모았습니다. 서버와 배포 체계를 맡았습니다.

기능

- 고객사는 아이디·비밀번호, 사내는 슬랙 로그인으로 들어오는 두 경로
- 현지 통화 금액과 원화 참고액 함께 표시
- 일정 타임라인과 자료 공유
- 사내 CRM과 문의 접수

고객 데이터와 사내 데이터

문제	고객사 화면과 사내 CRM을 한 앱에서 다루더라도 고객사 데이터는 서로 섞이면 안 됐습니다.
해결	화면은 Next.js, 인증·DB·파일·메일은 Go API 서버로 나눴습니다. 고객사 담당자는 자기 회사 화면만 봅니다. 파일은 고객별 저장소와 그 저장소만 쓰는 권한으로 격리했습니다.

고객사마다 쌓이는 분기

문제	고객사별 요구를 코드에 그대로 넣으면 고객사가 늘 때마다 분기가 쌓입니다.
해결	공통 기능과 고객사별 업무 모듈을 나눴습니다. 데이터는 고객, 브랜드, 프로젝트·캠페인, 업무 순의 계층으로 묶고, 고객별 하드코딩은 규칙으로 막았습니다. 서버 업무를 공통 기능과 업무 모듈 11개로 나눴습니다.

개발자를 거쳐야 했던 화면 수정

문제	화면 수정을 반영할 때마다 PM이 개발자에게 병합과 배포를 부탁해야 했습니다.
해결	사내 배포기가 브랜치를 2분마다 확인해 검사 뒤 배포합니다. PM이 라벨을 붙이면 검사를 통과한 PR만 병합합니다. PM이 개발자 도움 없이 화면 수정을 반영하고, 배포 토큰은 저장소에서 걷어 냈습니다.

ᓚᘏᗢ 루쿠쿠

미국 법인 Shopify B2B 스토어 구축 · B2B 백엔드

기간	2026.09 — 현재
팀	루쿠쿠 프로젝트 팀 B2B 백엔드 담당
역할	B2B 백엔드　ERP 연동 서버　연동 인터페이스 검토
기술	Go · chi · pgx · sqlc · goose · OpenAPI · Shopify B2B

거래처별 카탈로그·가격을 제공하고, 주문 송신과 부분 출고까지 잇는 B2B 서버

글로벌 스포츠 브랜드 미국 법인이 거래처 주문을 받는 Shopify B2B 스토어입니다. 백엔드를 맡아 B2B 서버와 ERP 연동 서버를 만들었습니다.

기능

- 거래처·사업장, 상품·재고 동기화
- 거래처별 주문 수량 규칙
- 서명을 검증한 웹훅 저장, 실패 재시도, 잘못된 메시지 격리

ERP와 통신하는 서버를 하나로

문제　B2B와 B2C 서버가 각자 ERP 클라이언트를 두면 인증 정보와 연동 규칙이 여러 곳에 흩어집니다. ERP 쪽이 바뀔 때마다 서버마다 고쳐야 합니다.

해결　ERP와 통신하는 창구를 연동 서버 하나로 모았습니다. B2B·B2C 서버는 연동 서버의 내부 API와 이벤트만 씁니다. ERP 쪽 변경은 연동 서버 한 곳에서만 고치면 됩니다.

요청한 개발 범위의 전제부터 확인

문제　고객이 요청한 개발 범위는 ERP 연동 커넥터가 이미 있다는 전제로 정해져 있었습니다.

해결　요청 항목을 재사용·수정·신규·외부 확인으로 나눴습니다. 커넥터가 없다는 점과 별도 계약이 필요한 범위를 개발 전에 짚었습니다.

세 가지 주문 방식, 하나의 검증

문제　거래처별 접근 권한, 카탈로그 노출, 대량 주문, 선불·후불 구분 요구가 한꺼번에 들어왔습니다. 주문도 매트릭스, 카탈로그, 품번·CSV 세 방식으로 받아야 했습니다.

해결　정책마다 기술 검토를 따로 열고, 고객사가 정할 항목과 기술로 확인할 항목을 나눠 질문지 세 종류로 회신했습니다. 세 주문 방식은 모두 같은 장바구니로 모인다고 보고, 주문 검증 API를 하나로 설계했습니다.

바뀐 주문 기준과 확정 정책

문제　회의에서 B2B 주문 기준이 생산 발주에서 현재 재고와 입고 예정 수량으로 바뀌었습니다. 이미 개발한 부분과 부딪히는 변경이었습니다.

해결　변경 요청과 확정된 정책을 나눠 기록하고, 기존 구현과 부딪히는 부분을 따로 정리했습니다. 변경은 개발한 내역을 백업하고, 변경 목록을 만든 뒤, 범위와 일정 영향을 합의하는 순서로 받았습니다. 확정 정책과 추가 확인이 필요한 요청이 섞이지 않게 됐습니다.

가짜 응답이 놓친 제약

문제　가짜 서버의 응답만으로는 실제 스토어의 제약을 알 수 없었습니다.

해결　합성 데이터를 연동 서버와 처리기를 거쳐 개발 스토어까지 보내고, 실제 가격과 재고를 확인했습니다. 식별자 형식, 거래처 수정 필드, 가격표 이름 충돌을 실제 응답에서 찾아 고쳤습니다.

ᓚᘏᗢ 루쿠쿠

루쿠쿠 OS · 회사 온톨로지와 에이전트 운영 체계

기간	2026.09 — 현재
팀	사내 에이전트 시스템 담당 2명 구조 설계·구현
역할	온톨로지 설계　지식 그래프　답변 평가　에이전트 운영
기술	Hermes Agent · MCP · PostgreSQL · AWS S3 · BGE-M3 · launchd

슬랙·메일·깃허브 등 15곳의 원문을 고객사·프로젝트 120여 개에 연결

회사 정보를 묶은 그래프를 직원과 에이전트가 함께 조회합니다. 에이전트는 사람 승인 아래에서 개발 작업을 나눠 맡습니다.

기능

- 슬랙 메시지 50만 건 이상을 보관하고 검색하는 원문 저장소

여러 곳을 뒤져야 알 수 있던 고객사

문제　한 고객사를 이해하려면 슬랙, 메일, 문서, 저장소를 따로 뒤져야 했습니다. 사람이 채우던 사내 위키는 문서 51개가 빈 채로 갱신이 멈춰 있었습니다.

해결　원천 15곳을 정해진 주기로 모아 매일 그래프를 다시 계산합니다. 원문 링크가 있는 관계만 남기고, 수집이 끊기면 마지막 정상본을 유지합니다. 2026.11 기준 대상 약 280개와 관계 약 4,150건을 담습니다. 2026.09부터 전 직원이 씁니다.

온톨로지: 무엇을 어떤 말로 잇는가

문제　이름이 비슷한 대상이나 뜻이 다른 관계를 섞으면 틀린 답이 나옵니다. 관계 이름을 정하지 않으면 같은 사실이 여러 말로 흩어졌습니다.

해결　무엇을 어떤 말로 이어도 되는지 관계 종류 20개로 정했습니다. 업무 한 건마다 누가 어떤 책임을 맡았는지도 잇게 했습니다. 업무별 책임 역할 관계 100건이 생겼습니다.

그래프만으로 답하는가

문제　그래프가 커져도 실제 질문에 답하는지는 알 수 없었습니다.

해결　원문 근거로 기대 답을 정한 고정 질문을 매일 돌리고, 못 답한 질문은 어느 단계에서 막혔는지 진단합니다. 슬랙에서 뽑은 관계는 서로 다른 계열 모델 두 곳이 동의해야 채택합니다.

첨부 문서에만 있던 거래처

문제　결제·물류·본인인증 거래처가 첨부 문서 본문에만 있고 그래프에는 없었습니다.

해결　역할을 나타내는 말 옆의 고유명사를 거래처 후보로 뽑고, 사람이 확인한 것만 거래처로 올렸습니다. 추측한 거래처는 빈 거래처보다 나쁘다고 봤습니다. 그래프의 거래처는 20개에서 24개로 늘었습니다.

되돌리기 어려운 일

문제　에이전트에게 실행을 맡기면서 되돌리기 어려운 일은 막아야 했습니다. 한 에이전트가 지시를 무시하고 고객 스레드에 답한 일도 있었습니다.

해결　권한을 주제가 아니라 조사·구현·검토 단계로 나눴습니다. 조사와 검토 에이전트는 읽기만 하고, 구현 에이전트는 커밋까지만 하며 병합과 배포는 하지 못합니다. 고객 채널을 지켜보는 에이전트에는 메시지를 쓰는 권한을 주지 않았습니다. 장애 대응을 뺀 배포, 외부 발송, 운영 데이터 쓰기는 사람이 승인합니다.

아키텍처

업무 배정 에이전트
요청 분류 · 배정 · 결과 취합

회사 온톨로지

- **수집**
  슬랙 · 메일 · 캔버스 · 첨부 · 저장소 · 업무 원장
- **슬랙 원문 저장소**
  메시지 50만 건 이상 · 전문 검색 · 첨부 원본과 목록 별도 보관
- **위키 그래프**
  매일 재계산 · 원문 링크가 붙은 관계만 남김
- **관계 규칙**
  허용 관계와 근거 등급을 한 파일에 정의
- **회사 기록**
  결정 · 확인할 일 · 기억 · 사람 검토 후 반영

조회 창구

- **오프온 MCP**
  OAuth · 요청자 권한 안에서만 쓰기
- **답변 평가**
  고정 질문을 매일 돌려 못 답한 원인 진단

개발 담당

- **조사**
  읽기 전용 원인 분석
- **구현**
  별도 작업 공간에서 커밋 · 병합·배포 불가
- **검토**
  작성자와 다른 모델로 검토

업무 담당

- **업무 관찰**
  슬랙 대화의 업무를 오프온에 올림 · 근거가 불분명하면 그대로 둠

요청 → 업무 배정 → 구현·커밋 → 다른 모델 검토 → 사람 승인 → 배포 후 확인



고객사·프로젝트·사람·저장소 등 대상 약 280개와 관계 약 4,150건을 잇습니다.

ᓚᘏᗢ 루쿠쿠

설치미술 전시를 위한 360도 영상 앱과 AI 영상 생성 서버

기간	2026.06 — 2026.08
팀	고객사 프로젝트 현장 앱·서버·대시보드 담당
역할	Swift 현장 앱　AI 생성 서버　운영 대시보드　결과 웹
기술	Swift · SwiftUI · Python · FastAPI · PostgreSQL · AWS S3 · TypeScript

카메라 SD카드를 꽂으면 영상을 이어 붙여 QR로 건네는 무인 현장 앱

작품마다 관람객의 사진·영상을 받는 방식이 달랐습니다. 작품별 현장 앱과 생성 작업은 따로 두고, 공통 서버와 운영 대시보드, 결과 웹은 함께 쓰게 만들었습니다.

기능

- 날다 · 풍선 의자 체험의 Swift 현장 앱
- 나무블럭 · 이미지와 단어로 약 15초 영상을 만드는 서버 API
- 비비히어로 · 사진으로 캐릭터 포스터를 만들고, 고른 모션으로 약 10초 영상을 만드는 서버 API
- 릴스13 · 13초씩 세 번 찍은 영상에서 하나를 골라 AI 영상으로 만드는 서버 작업
- 360도 풍선 체험 결과 웹 · 관람객이 QR로 영상을 받는 모바일 페이지
- 운영 대시보드 · 매장·장치·서비스 단위로 주문, 실패, 재시도, AI 사용료 확인

결과를 기다리는 관람객

문제　사람이 카메라 파일을 옮기고 편집하면 관람객이 결과를 오래 기다려야 했습니다.

해결　SD카드를 꽂으면 앱이 새 파일만 골라 중복을 거르고, 화면 없이 영상을 잇습니다. 현장 PC가 합성과 업로드를 함께 하면 과부하가 걸릴 수 있어, 합성은 한 번에 하나씩 돌립니다. 미리보기를 먼저 올려 QR을 띄우고, 원본은 뒤이어 올립니다.

이전 관람객의 QR

문제　다음 관람객이 이전 관람객의 QR을 읽을 수 있는 구간이 있었습니다.

해결　기존 QR은 일정 시간 뒤 지우고 대기 영상으로 바꿨습니다. 카메라를 연결하면 복사 진행률을 보여 준 뒤 새 QR을 띄웁니다.

AI 제공자와 실패 처리

문제　외부 AI 서비스는 바뀌거나 실패할 수 있었습니다. 제공자를 바꿀 때마다 서버 코드와 작업 대기열을 함께 고치는 구조는 피해야 했습니다.

해결　자체 GPU 대신 외부 서비스를 쓰기로 하고, 제공자 이름을 코드에 넣지 않고 연결부만 바꾸게 했습니다. 실패하면 최초 요청 뒤 3회까지만 다시 시도하고, 인증·설정·청구 오류는 다시 시도하지 않습니다. 이 기준을 코드, 문서, 대시보드에 같게 넣었습니다. 같은 요청이 다시 들어와도 생성 작업은 하나만 만듭니다.

ᓚᘏᗢ 루쿠쿠

스포츠 브랜드 국내 자사몰과 ERP의 판매 대사·연동 운영

기간 2026.04 — 현재
팀 루쿠쿠 운영 팀
대사·연동 서버 담당
역할 판매 대사 대사 도구 개발 연동 서버 운영 정산 보정
기술 Python · Shopify · 마이크로서비스

58일치 기록 약 39만 행을 대조해 차이를 원인별로 나눈 대사 도구

월말 판매 대사와 두 시스템을 잇는 연동 서버 운영을 맡고 있습니다. 대사 도구로 찾은 차이 가운데 실제로 빠진 판매와 정산을 바로잡습니다.

기능

- 주문·출고·반품·교환·쿠폰·포인트 기록을 키 단위로 대조
- 교환 주문의 정산 누락 유형을 나눈 보정 절차
- 우대권 가격을 ERP에서 받아 주문 항목별로 할인 계산

실제 누락과 집계 방식 차이

문제 두 시스템은 기록 방식과 시점이 달라, 시트로 대조하면 실제 누락과 집계 방식 차이가 섞였습니다.

해결 차이를 ERP에만 있음, 자사몰에만 있음, 금액 다름으로 나누고, 차이 행마다 양쪽 원천 기록을 다시 조회해 원인을 붙이는 단계를 넣었습니다. 협력사가 건수가 부풀었다고 지적한 뒤에는 분석 오류와 실제 차이를 가르는 규칙을 다듬었습니다. 차이 16,365행을 추렸습니다. 협력사가 확인하고 보정할 10,203행은 항목별 파일 49개로 넘겼습니다.

월말 누락 목록에서 시차 가르기

문제 매달 말 협력사가 보내는 누락 목록에는 목록을 뽑은 뒤 이미 전송된 판매도 섞여 있었습니다.

해결 항목마다 자사몰 송신 이력과 대조해 원인을 나눴습니다. 시차로 이미 나간 건은 이중 전송을 피하려고 보내지 않았습니다. 누락되거나 일부만 전송된 판매 데이터를 ERP 규칙에 맞춰 다시 보냈습니다.

사이즈를 바꾼 주문의 판매 데이터

문제 사이즈를 바꿔 출고한 주문은 변경 내역, 송장, 출고 상태, 출고 연동에 바뀐 사이즈가 반영됐습니다. 그런데 판매 데이터만 원래 사이즈로 ERP에 들어갔습니다.

해결 원인을 상품 변경 연동 로직으로 좁혔습니다. 판매 데이터를 ERP에 다시 보낸 뒤 로직 수정에 들어갔습니다. 해당 판매 데이터의 재전송을 마쳤습니다.

매달 맞춰 보는 정산

문제 자사몰 주문과 ERP 정산이 매달 맞는지 확인해야 했습니다. 정산 데이터가 잘못 만들어진 주문을 그대로 다시 보내면 중복 전송이 될 수 있었습니다.

해결 9월분은 정산 데이터가 잘못 만들어진 주문 41건과 기준 정보가 빠진 건을 목록으로 정리했습니다. 10월분은 다시 뽑은 72행에서 제외 대상 2행을 빼고 보냈습니다. 중복 위험이 있는 추가 전송은 보류했습니다. 9월분과 10월분 대사를 마쳤습니다.

ᓚᘏᗢ 루쿠쿠

Hermes Agent 기반 사내 운영 에이전트

기간	2026.04 — 현재
팀	사내 에이전트 시스템 설계·구현·운영 담당
역할	에이전트 권한 설계　운영 서버 구성　장애 자동 처리　모델 평가
기술	Hermes Agent · Python · macOS · launchd · OpenViking · BGE-M3 · Slack

슬랙 요청을 역할별 에이전트 9개에 나눠 맡기고, 오류는 배포와 회복 확인까지 처리하는 에이전트

직원이 슬랙과 텔레그램에서 부르는 사내 운영 에이전트입니다. 사무실 맥미니 한 대에서 Nous Research의 Hermes Agent로 돌아갑니다. 요청을 나눠 맡을 에이전트에 배정하고, 결과를 모아 같은 스레드에 답합니다. 설계부터 구현과 운영까지 맡았습니다.

기능

- 업무 조회·등록과 연차·반차 신청을 슬랙 스레드에서 처리
- 고객 채널의 요청을 업무로 올리는 관찰 에이전트 · 고객 채널에 쓰는 권한은 없음
- 재직자마다 슬랙 대화로 매일 새로 고치는 개인 에이전트
- 맥미니 한 대에서 launchd 작업 33개로 상시 실행 · 5분마다 설치본과 소스 대조

이름만 셋이던 예비 모델

문제　대화가 약 141k 토큰으로 커지자 응답이 최대 1,221초까지 늦어졌습니다. 예비 모델 세 자리 중 하나는 인증이 만료됐고, 둘은 주 모델과 같은 공급자였습니다.

해결　원인은 커진 대화의 압축이 300초 제한에 걸려 실패한 것이었습니다. API 키로 결제하지 않는다는 원칙에 맞춰, 구독 계정으로 Claude를 부르는 로컬 어댑터를 만들었습니다. 이 어댑터를 주 모델과 다른 공급자의 예비 경로로 붙였습니다.

오류마다 기다리던 승인

문제　처음에는 오프온 오류만 에이전트가 고쳤고, 수정과 배포에 사람 승인이 필요했습니다.

해결　모든 프로젝트의 오류를 원인 조사부터 배포와 회복 확인까지 맡겼습니다. 오탐이면 알림을 끄지 않고 감지 조건을 고칩니다. 외부 발송, 거래·환불, 원문 삭제, 권한 변경, 회사 기록 반영 다섯 가지만 사람이 승인합니다.

지어낸 링크로 답하는 개인 에이전트

문제　직원 대신 답하는 에이전트가 도구가 찾지 않은 링크를 근거로 대면 안 됐습니다.

해결　질문 9개로 모델을 비교하고, 도구가 돌려주지 않은 링크를 하나라도 쓰면 0점으로 매겼습니다. 첫 실제 대화에서는 위키 도구를 한 번도 부르지 못했습니다. 도구를 뒤로 미루지 않게 하고, 조회 순서를 지침에 적어 고쳤습니다. 5점 만점에 2.86–4.54점이 나왔고, 지어낸 링크는 모든 후보에서 0건이었습니다.

하루 늦은 회사 기록

문제　회사 결정 기록을 새벽에 한 번만 받아 와서, 에이전트가 그날 정한 결정을 찾지 못했습니다.

해결　10분마다 도는 수집 작업에 기록 동기화를 넣었습니다. 같은 질문에 그날의 결정으로 답하는 것을 확인했습니다.

아키텍처

총괄 에이전트
요청 해석 · 계획 · 배정 · 인계 · 결과 검증 · 최종 판단 · 결과는 실제 원천과 대조

지식과 기억

- **회사 기록**
  목표 · 결정 · 운영 기준 · 사람 승인 없이 반영하지 않음
- **운영 설정 저장소**
  에이전트 설정 · 실행 규칙 · 회사 기록과 역할 분리
- **Hermes Agent 내장 기억**
  에이전트마다 기억을 따로 둠
- **OpenViking 검색**
  지식 · 대화 검색 · 검색 결과를 회사 기록으로 보지 않음

실행과 통제

- **총괄이 맡는 시스템**
  오프온 · 회사 기록 · 설정 저장소 · 자격 증명 · 작업자 보고만으로 완료 처리하지 않음
- **단계 인계 원장**
  업무 · 완료 조건 · 담당 · 검토 · 장기 실행 · 감사 · 재시도
- **범위와 승인**
  역할마다 실행 범위 제한 · 작업자는 일을 다시 맡기지 않음 · 외부 발송은 별도 승인

개발 담당

- **조사** 읽기 전용
  원인 · 기술 근거 조사 · 제품 코드 수정 없음
- **구현** 격리
  별도 작업 공간에서 구현 · 로컬 테스트 · 최종 커밋 · 푸시 · 병합 · 배포 없음
- **검증** 독립 QA
  테스트 근거 확인 · 통과 · 반려 판정 · 최종 반영은 총괄

운영 담당

- **슬랙 관찰**
  공개 채널 메시지 수집 · 요청을 오프온에 반영 · 판단은 총괄로 반환
- **슬랙 원문 저장소** 저장소
  메시지 50만 건 이상 · 비공개 방과 웹훅 방을 뺀 공개 채널을 매일 수집
- **직원 정보**
  HR 정보 정리 · HR 범위 안에서만 실행
- **개인 업무**
  개인 오프온 업무 · 연차 · 반차 · 승인 판단은 총괄
- **직원 대응**
  오프온 읽기 전용 · 직원에게 한 답변은 회사 승인이 아님

운영 알림

- **오류 알림 전달**
  오류 알림 채널의 슬랙 웹훅 알림을 총괄 에이전트에 전달 · 작업은 실행하지 않음

역할 수행 → 결과 · 근거 · 위험 반환 → 총괄이 원천과 대조 → 승인 · 검증 후 반영 → 다시 조회해 완료 확인

ᓚᘏᗢ 루쿠쿠

오프온 · 전사 업무 관리 시스템

기간	2026.04 — 현재
팀	사내 제품 설계·구현·운영 담당
역할	설계 풀스택 개발 MCP 서버 운영
기술	Next.js 16 · React 19 · TypeScript · Prisma · PostgreSQL · Vercel · MCP · OAuth 2.1
링크	github · 근태 기능만 담은 저장소 github.com/takealook97/offon

근태부터 CRM까지 슬랙 안에서 다루고, AI 에이전트의 MCP 창구도 맡는 시스템

루쿠쿠의 사내 제품으로, 설계부터 구현과 운영까지 맡았습니다.

기능

- 슬랙 명령으로 출퇴근·휴식 기록, 정정 요청과 승인
- 자정을 넘긴 근무, 휴일 제외, 연차 부여와 이월 자동 계산
- 표·칸반·캘린더·간트로 보는 업무와 담당자별 상태
- 넘긴 업무의 다음 확인일에 담당자에게 묻는 알림
- 회의실 예약과 회의 10분 전 알림
- 슬랙 대화 약 39만 줄을 근거로 찾는 에이전트용 검색

다시 옮겨 적던 요청

문제 슬랙과 메일로 온 요청을 PM과 개발자가 업무 도구에 다시 옮겨 적었습니다. 확인이 필요한 요청은 대부분 메일이었고, 절반은 알림 메일이었습니다.

해결 요청을 원문·첨부와 함께 업무로 바꾸거나 기존 업무에 잇는 수신함을 만들었습니다. 같은 스레드는 이미 만든 업무에 붙고, 자동 발신 메일은 빠집니다. 규칙을 처음 적용했을 때 대기 메일 59건이 사람 손 없이 정리됐습니다. 사람이 다른 스레드에서 두 번 이상 버린 주소는 다음부터 걸러집니다.

에이전트에게 권한을 어디까지 줄 것인가

문제 에이전트마다 DB와 슬랙 권한을 따로 주면 관리가 흩어집니다. 요청자를 헤더 값으로 받으면 위조할 수도 있었습니다.

해결 오프온이 OAuth 2.1 인가 서버를 겸하고, 신원은 토큰 안에서만 받습니다. 로그인은 기존 슬랙 로그인을 쓰고, 쓰기는 요청한 사람의 권한 안에서만 됩니다. 직원은 설치나 API 키 없이 로그인 한 번으로 도구 21개를 씁니다.

동시에 들어온 연차 요청

문제 에이전트가 슬랙에서 연차를 대신 처리하면서 같은 요청이 겹칠 수 있었습니다. 같은 휴가를 동시에 두 번 취소하면 둘 다 성공해 연차 사용량이 1에서 −1이 됐습니다.

해결 연차가 하루 남았을 때 같은 날 신청 두 건이 모두 통과하는 것도 재현했습니다. 겹친 요청은 하나만 처리하고 나머지는 충돌로 거절하게 고쳤습니다. 실제 PostgreSQL에 붙여 API 경로를 지나는 검사 9건을 더했습니다. 같은 재현에서 두 번째 요청은 409로 거절됐습니다.

화면에서 찾을 수 없던 기능

문제 업무 배정 규칙을 서버에만 넣어, 사용자는 화면에서 기능을 찾지 못하고 오류만 봤습니다.

해결 규칙을 화면에도 반영했습니다. 찾을 수 없는 기능은 없는 기능으로 본다는 원칙을 남겼습니다. 제보받은 날 고쳤습니다.

ᓚᘏᗢ 루쿠쿠

스포츠 브랜드 국내 자사몰 · 대회 참가자 전용 우대권 주문

기간	2026.04 — 2026.09
팀	3명 백엔드 담당
역할	우대권 주문 API　주문 검증　중복 결제 방지　우대권 종료 운영
기술	Python · Shopify

우대권 가격으로는 대상 상품만 주문받고, 같은 회원의 중복 결제를 막는 주문 백엔드

브랜드가 여는 테니스 동호회 대회의 참가자는 회원 전용 페이지에서 우대권으로 할인받아 구매할 수 있었습니다. 대회 전 약 일주일 동안 이를 위한 우대권 주문 백엔드를 만들었습니다.

기능

- 우대권 가격을 적용할 수 있는 상품만 돌려주는 API
- 주문 상품의 스타일·색상을 우대권 품목과 대조해 대상이 아니면 막는 주문 검증
- 스테이징 테스트에서 판매 불가로 보이던 묶음 상품을 Shopify 관리자 상태와 맞춤

여럿이 함께 쓰는 우대권 코드

문제　우대권 코드는 참가자 여럿이 함께 써서, 중복 결제를 가리는 키로 쓸 수 없었습니다.

해결　우대권 코드와 회원 ID를 합친 해시를 장바구니 키로 정했습니다. 같은 회원이 같은 우대권으로 들어오면 늘 같은 장바구니를 쓰고, 중복 결제는 거절됩니다.

우대권 기간 종료와 보류된 주문

문제　우대권 기간에 들어온 주문은 ERP로 보내지 않고 보류해 두었습니다. 기간이 끝나면 보류된 주문을 빠짐없이 넘겨야 했습니다.

해결　기간이 끝난 뒤 우대권을 비활성으로 바꾸고, 주문 보류를 풀어 ERP 연동을 시작했습니다. 보류됐던 주문 13건을 약 2초 만에 모두 ERP로 보냈고, 남은 보류 주문은 없었습니다.

ᓚᘏᗢ 루쿠쿠

브랜드 Shopify 스토어 기능 개발과 연동

기간	2026.04 — 2026.07
팀	고객사 유지보수 기능 개발 담당
역할	배치 Shopify 앱·확장 연동 API
기술	Shopify · Shopify Checkout UI extensions · React · 카카오 알림톡

쓰지 않은 쿠폰을 만료 7일 전에 알리는 알림톡 배치부터 판매처 주문 연동까지

고객사 Shopify 스토어를 유지보수하며, 브랜드마다 요청한 기능과 외부 연동을 만들었습니다.

기능

- 스포츠 그룹 한국 법인의 브랜드 두 곳 · 쿠폰 만료 알림톡 배치
- 언더웨어 브랜드 글로벌 스토어 · 체크아웃 확장과 언어별 문구
- 유럽 수면 브랜드 한국 법인 · 국내 판매처 주문을 Shopify로 모으는 백엔드 설계와 시험 환경
- 케이스 브랜드 · 쇼핑몰 상품 옵션과 ERP 품목을 짝짓는 조회 API
- 뷰티 브랜드 · 수량 할인 앱의 앱 스토어 제출 작업과 새 테마 장바구니 대응

이미 쓴 쿠폰에는 안내하지 않기

문제	배치가 대상을 고른 뒤 발송하기 전까지 고객이 쿠폰을 쓸 수 있습니다.
해결	발송 직전에 쿠폰 사용 여부를 다시 조회하고, 이미 쓴 쿠폰은 건너뛰게 했습니다. 두 브랜드의 운영 시작 시점이 달라, 브랜드별로 켜고 끄는 설정을 두고 한 곳부터 켰습니다. 만료 안내 템플릿 2종은 카카오 검수 등록본과 본문·버튼 주소를 글자 단위로 맞췄습니다. 두 브랜드 모두 운영에 적용했습니다.

체크아웃 정책 동의 저장

문제	언더웨어 브랜드 글로벌 스토어의 체크아웃에서 정책 동의를 받아 저장해야 했습니다.
해결	Shopify 체크아웃 UI 확장으로 동의 체크박스를 넣고, 동의 값을 저장할 때 생기던 경쟁 상태를 고쳤습니다.

대량 수집의 시간 초과

문제	상품 옵션을 한 번에 대량으로 가져오면 요청 시간이 초과됐습니다.
해결	수집을 나눠 저장하게 바꿔 시간 초과를 피했습니다.

비즈톡

AWS MSK 3.8 마이그레이션

기간 2025.08 — 2025.09
팀 1인
역할 아키텍처 설계 개발·운영 클러스터 KRaft 도입 SASL 인증 추가 토픽·복제본 재구성 발송 테스트 Kafka UI 모니터링
기술 AWS MSK 3.8 · Kafka KRaft · SASL/SCRAM · AWS Secrets Manager · Kafka UI

토픽 약 160개의 설정을 새 클러스터에 자동으로 재생성

운영 중인 MSK 3.5의 지원이 2025.10에 끝날 예정이었습니다. 3.8 신규 클러스터를 KRaft 메타데이터로 구성하고 기존 토픽을 다시 만들었습니다.

인증 방식 선정

문제 신규 클러스터에는 서버 인증을 새로 추가해야 했습니다. MSK는 IAM, TLS, SASL 등을 지원해, 사내 환경과의 호환성과 운영 복잡도를 함께 따져야 했습니다.

해결 검토 끝에 SASL/SCRAM 방식을 택했습니다. 사용자 계정은 AWS Secrets Manager에 저장했습니다. 각 서버에 Secrets Manager 정보를 연동해, 서버가 계정을 읽어 인증하게 했습니다.

토픽 설정 재생성 자동화

문제 기존 클러스터에서 운영하던 토픽 전체를 새 클러스터에 같은 설정으로 만들어야 했습니다. 토픽마다 파티션 수, 복제본, 옵션이 달라 손으로 옮기면 누락과 오류의 위험이 컸습니다.

해결 기존 클러스터의 토픽 설정을 CSV 파일로 일괄 추출하고, 수정이 필요한 토픽은 복제본 구성을 다시 잡았습니다. 이 CSV를 읽어 신규 클러스터에 토픽 전체를 만드는 스크립트를 썼습니다. 토픽명, 파티션 수, 복제본, 옵션이 그대로 반영되어 손으로 옮기는 시간을 크게 줄였습니다.

비즈톡

BGMS 로그 마스킹 모듈

기간 2025.07
팀 1인
역할 기획 모듈 설계 Redis 병목 개선 개발·테스트 패치
기술 Java 8 · Spring Boot 2.7 · Redis · Gson · JaCoCo

로그마다 25회 넘게 하던 Redis 조회를 로컬 캐시로 건너뛰는 구조

SK텔레콤 해킹 사고 이후 전화번호 암호화 외에 로그 마스킹까지 요구하는 고객사가 생겼습니다. MSA 서버 14종에 공통으로 적용해야 해서, 어느 서버에나 이식할 수 있는 모듈로 만들었습니다.

기능

- 민감 고객사인지 먼저 판단하고, 로그 객체를 넘기기 전에 감싸 모듈 안에서 마스킹

Redis 병목

문제 민감 고객사인지 판단하려고 로그를 남길 때마다 Redis를 조회했습니다. 발송 로그 하나에 Redis 접근이 평균 25회를 넘어 병목이 될 가능성이 커졌습니다.

해결 모듈 안에 로컬 캐시를 두고, 캐시에 있으면 Redis 조회를 건너뛰게 했습니다. 민감 고객사 목록은 매일 로컬로 동기화해 최신 상태를 유지했습니다.

중첩 객체의 필드 탐색

문제 로그로 넘어오는 값은 단순 문자열이 아니라 객체·목록·JSON이 중첩된 형태였습니다. 마스킹할 필드를 찾으려면 모든 계층을 살펴야 했고, get/set 접근으로는 이를 할 수 없었습니다.

해결 로그 객체를 Gson으로 직렬화한 뒤 DFS로 JSON 트리를 순회하게 했습니다. 순회 중 bsid로 민감 고객사인지 판단하고, 그 아래의 recipient, phone_number, message 같은 필드를 가렸습니다. 객체의 구조와 관계없이 같은 방식으로 필드를 찾게 되어 어느 서버에나 붙일 수 있었습니다.

비즈톡

채권 미납 안내 RPA

기간	2025.06 — 2025.07
팀	1인
역할	기획 DB·플로우 설계 PDF 생성 자동화 메일·알림톡 고지 자동화 디자인 클라이언트 페이지
기술	Java 17 · Spring Boot 3.5 · MyBatis · MySQL · poi-ooxml · LibreOffice · JavaMailSender · Next.js 15 · React 19

2일 넘게 걸리던 미납 안내를 10분으로

기획팀은 매달 초 미납 고객사에 메일과 알림톡을 보내며, 엑셀 수기 입력과 PDF 직접 생성을 반복했습니다. 이 과정을 한 페이지로 옮기는 RPA를 기획부터 맡았습니다.

기능

- 연체 고객사 조회부터 채권 등록·삭제까지 한 페이지에서 관리
- 미납 고객사별 미납액과 납부기한을 채운 PDF 자동 생성
- 메일·알림톡 고지를 버튼 하나로 발송

고객사마다 다른 PDF와 메일 본문

문제 매달 100곳이 넘는 미납 고객사마다 다른 미납액과 납부기한을 엑셀 양식에 고쳐 넣어 PDF로 뽑고, 메일 본문도 손으로 적었습니다.

해결 엑셀 양식에서 바뀌는 셀(문서 번호, 생성일, 수신자, 청구월, 미납액, 납부기한)의 위치를 상수로 정리했습니다. poi-ooxml로 그 셀에 DB 값을 채우고, LibreOffice로 PDF를 만들었습니다. 메일 제목과 본문도 고객사에 맞게 채워 JavaMailSender로 PDF와 함께 보냈습니다. 2일 넘게 걸리던 미납 안내 업무가 10분으로 줄었습니다.

설치 권한이 없는 서버에서 PDF 변환

문제 처음에는 Aspose로 엑셀을 PDF로 바꿨지만, 무료 라이선스에서는 PDF 상단에 빨간 문구가 찍혔습니다. 운영 서버에는 프로그램을 설치할 권한도 없었습니다.

해결 LibreOffice로 바꿔 변환이 정상 동작하는 것을 확인했습니다. 설치 대신 AppImage로 올려 실행하고, WAS에서 headless 옵션으로 변환 프로세스를 띄웠습니다. 변환한 PDF의 한글이 깨져, 서버에 없던 맑은 고딕 폰트도 함께 올렸습니다.

같은 문서의 반복 생성

문제 같은 조건으로 문서를 다시 요청하면 같은 파일을 또 만들어 자원을 낭비할 수 있었습니다. IDC 서버라 S3를 쓸 수 없어 파일을 서버 디스크에 저장해야 했습니다.

해결 고객사 ID, 문서 생성일, 청구월, 납부기한, 미납액으로 16자리 해시를 만들었습니다. 이 해시와 확장자를 복합키로 하는 문서 생성 로그 테이블을 두고, 파일은 해시별 디렉터리에 저장했습니다. 로그에 있는 문서는 다시 쓰고, 없을 때만 새로 만들었습니다.

대량 발송 중 진행 상황

문제 버튼 하나로 여러 고객사에 보내면 화면에서 진행 상황을 알 수 없었습니다.

해결 발송 건별 로그를 저장하고 SSE로 진행 이벤트를 흘렸습니다. 메일·알림톡은 병렬(core 4, max 8)로 보냈습니다.

비즈톡

BGMS 메시지 발송 시스템 개발과 운영

기간	2025.05 — 2025.10
팀	SD 파트 BGMS 개발·운영 담당
역할	브랜드메시지 기능 개발 MSK Realtime 토픽 구축 관련 로직 리팩터링 서비스 운영
기술	Java · Spring Boot · AWS MSK · Redis

월 약 4억 건을 보내는 메시지 발송 시스템의 기능 개발과 운영

BGMS는 카카오 알림톡·브랜드메시지를 보내는 메시지 발송 시스템입니다. Kafka 기반 MSA 서버 14종이 인스턴스 54대에서 돌아갑니다. 월 평균 약 4억 건을 보내고, 피크 때 초당 2,000건을 처리합니다. 비즈톡에 있는 동안 이 시스템의 기능 개발과 운영을 맡았습니다.

기능

- 카카오톡 브랜드메시지 차단 기능
- 카카오톡 브랜드메시지 타겟팅(N, I) 재처리 로직
- MSK 안의 Realtime 토픽 구축과 관련 로직 리팩터링

비즈톡

배포 파이프라인 구축 및 사내 GitLab 마이그레이션

기간	2025.05 — 2025.10
팀	SD 파트 배포 파이프라인·QA 서버 담당
역할	Jenkins 파이프라인 AWS CodeDeploy 배포 ELB 이중화 QA 서버 구성 GitLab 마이그레이션
기술	Jenkins · AWS CodeDeploy · AWS ELB · Nginx Proxy Manager · Portainer · GitLab

QA·운영 환경의 배포 파이프라인 17종

MSA QA 서버와 커머스 신사업의 배포를 Jenkins 파이프라인으로 만들었습니다. 신사업 서버는 ELB로 이중화했습니다.

기능

- MSA QA 서버 13종, 커머스 신사업 QA·운영 4종(AWS CodeDeploy)으로 나눈 파이프라인
- QA 서버의 Nginx Proxy Manager·Portainer 구성
- 사내 GitLab 서버 마이그레이션, 버전 11에서 18로 업그레이드

비즈톡

서버 설정 관리와 배포 자동화 스크립트

기간	2025.05 — 2025.10
팀	SD 파트 서버 설정·배포 스크립트 담당
역할	설정 표준화　패치 자동화　배포 스크립트　설정 암복호화 도구
기술	Shell script · scp

운영 서버 50대 이상의 설정을 표준화하고 패치를 자동화

BGMS 운영 서버의 설정과 배포를 스크립트로 관리하게 바꿨습니다. 설정 파일의 민감 정보를 암호화하는 도구도 만들어 전사에 도입했습니다.

기능

- 설정 버전 관리
- 설정 암복호화 라이브러리와 스크립트
- 사내 암복호화 API 서버

손으로 하던 IDC 서버 배포

문제	IDC 서버의 일부 프로젝트는 jar 파일을 FTP 프로그램으로 올리고 서버 스크립트로 실행해 배포했습니다. 순서를 사람이 기억해야 했고, 프로그램이 자주 느려져 디렉터리를 계속 새로고침해야 했습니다.
해결	빌드, 전송, 기존 프로세스 종료, 재기동, 헬스체크를 scp와 셸 스크립트 하나로 묶었습니다. 이전 jar와 로그는 시각별 tar.gz로 백업하고, 환경 변수 파일은 서버의 conf 디렉터리에서 읽게 했습니다. 재기동 뒤에는 /ping 응답을 최대 20번 확인해 성공이나 실패를 푸시 알림으로 보냈습니다.

비즈톡

카카오톡 발송·템플릿 통계 수집 서버

기간	2025.05 — 2025.06
팀	1인
역할	DB·플로우 설계　배치 스케줄러　집계 API　재수집 API　데이터 재가공 스케줄러
기술	Java 17 · Spring Boot 3.5 · Spring Batch · MyBatis · MySQL

월 800만 건 이상의 발송 통계 수집

알림톡·친구톡·브랜드메시지 세 종류를 다룹니다. 통계 테이블은 사용자 ID 기준으로 역정규화해 조회 쿼리 성능을 높였습니다.

경과일마다 다른 통계

문제	카카오 API의 읽음·클릭 통계는 경과일(0·1·2일)별로 따로 집계되어 들어왔습니다. 한 번에 합치면 값이 맞지 않아 데이터 정합성이 깨졌습니다.
해결	매일 카카오 API에서 집계된 통계 전체를 받아오게 했습니다. 읽음·클릭 값은 경과일별 3단계로 나눠 누적했습니다. 배치는 500건씩 DB에 써서 쓰기 성능을 높였습니다.

재수집 시 중복 적재

문제	배치 집계가 빠지는 경우에 대비해 수동 재수집 API를 만들자, 같은 데이터를 여러 번 누적할 수 있는 구조가 되었습니다.
해결	역정규화한 최종 테이블에서 변동 컬럼을 뺀 주요 컬럼으로 복합키를 구성했습니다. elapsed_mask 컬럼을 추가해 경과일 0·1·2를 비트 1·2·4로 표시했습니다. 누적할 때 이 비트를 확인해 같은 경과일 값을 두 번 더하지 않게 했습니다. 같은 데이터를 다시 수집해도 행이 중복되지 않아 통계의 정합성이 유지되었습니다.

— 디핏

에이전트가 사업을 운영하는 AI 네이티브 커머스

기간	2026.07 — 현재
팀	제품·에이전트 운영 체계 설계와 구현
역할	공동창업자 · 빌더　제품 설계　에이전트 운영 모델　백엔드　운영 도구 개발
기술	Go · Next.js · PostgreSQL · Hermes Agent
링크	deepeet.com

자동화를 기반으로 3개월간 누적 주문 700건 이상, 누적 매출 1천만 원 이상을 돌파했습니다.

디핏은 도매 공급사의 과일을 판매채널에서 파는 위탁판매로 시작했습니다. 가장 먼저 만든 것은 판매채널 주문을 도매 공급사 발주로 잇는 발주 자동화입니다. 그 위에 라우팅·가격·정산·소싱·상품 등록·고객 응대·광고를 하나씩 직접 만들었습니다. AI 에이전트가 근거를 보고 판단하면, 커머스 시스템이 권한과 안전 조건을 확인한 뒤 실행합니다. 실행 결과는 판매채널과 공급사의 공식 상태로 다시 확인합니다. 중요한 코드 변경은 작성한 모델과 다른 모델이 검증합니다.

보고와 실제 상태의 차이

문제	에이전트의 완료 보고와 실제 시스템 상태가 다를 수 있습니다.
해결	총괄이 보고를 검토한 뒤, vat에 정한 규칙을 통과한 절차만 시스템이 실행합니다. 실행 뒤에는 실제 상태를 다시 확인합니다. 하위 저장소 19곳이 같은 규칙을 따릅니다.

아키텍처

사업 운영 총괄 에이전트
유일한 사용자 창구 · 작업 배정 · 결과 확인 · 최종 판단

지식·기억
- **사업 기록**
  변경 전 검토
- **총괄 기록**
  원칙 · 업무 절차 · 운영 일지
- **대화 기억**
  대화에서 요약한 기억과 사용자 정보
- **검색 색인**
  원본 기록에서 만든 검색 자료 · 근거로 쓰지 않음

실행·통제
- **커머스 시스템**
  발주 자동화 · 라우팅 · 가격·정산 · 소싱 · 상품 등록 · 운영 콘솔 · 광고
- **작업 보드**
  오래 걸리거나 인계·재시도가 필요한 작업
- **배포 절차**
  저장소 19곳의 변경 · 바뀐 범위 검증 · 배포 뒤 확인
- **실행 권한**
  L0 조회·분류·초안 · L1 정해진 절차 실행
- **무인 운영**
  메신저 연결 감시 · 재부팅 뒤 자동 기동 · 암호화 백업

개발 담당
- **조사** L0
  원인·기술 근거 조사
- **구현** L0
  배정된 티켓 범위 구현
- **검토** L0
  품질·회귀·보안 검토

사업 업무 담당
- **상품** L1
  상품 기획·상세페이지·등록·검증
- **고객 응대** L0
  문자 문의 분류·답변 초안
- **재무** L0
  원천 조회·대사·보고
- **마케팅** L0
  광고·마케팅 판단
- **공급처 확인** L0
  공급처 공지 확인·정리

담당 에이전트 작업 → 결과 보고 → 총괄 검토 → 권한 확인 뒤 시스템 실행 → 실제 상태 확인

— 디핏

광고 운영

기간	2026.10
팀	1인
역할	광고 연동　성과·변경 이력 수집　광고비 원장　광고 유입 주문 표시
기술	Go · PostgreSQL · TypeScript · Next.js

토스쇼핑 광고의 성과와 변경을 한 화면에서 다루고, 광고비와 광고로 들어온 주문을 손익에 반영하는 체계

광고는 채널마다 계정·캠페인·청구 방식이 다릅니다. 토스쇼핑 광고는 상점별 격리된 실행기로 일별 성과, 캠페인·광고그룹·광고, 채널 변경 이력을 읽습니다. 광고로 들어온 주문은 공식 주문 혜택으로 확인해 표시합니다. 쿠팡 광고도 같은 콘솔 메뉴와 계정 구조로 다룹니다. 유료 청구와 광고비를 나눠 기록합니다.

기능

- 토스쇼핑 광고의 일별 성과, 캠페인·광고그룹·광고, 채널 변경 이력
- 광고로 들어온 주문의 표시와, 그 주문에 적용된 수수료율을 쓴 예상 손익
- 광고비를 반영한 스토어 손익
- 변경 작업의 원장과 중복 방지, 결과를 모르는 변경은 다시 보내지 않는 처리
- 광고 충전을 비용으로 보지 않는 광고비 원장
- 쿠팡 광고 메뉴와 계정 구조

광고 충전과 광고비의 혼동

문제　광고 지갑 충전을 그대로 비용으로 잡으면, 실제로 쓴 광고비와 손익이 어긋납니다.

해결　지갑 충전, 청구 원천, 실제 비용을 나눠 기록하는 원장을 두었습니다. 스토어 손익에는 실제로 발생한 광고비만 뺍니다.

결과를 모르는 광고 변경

문제　캠페인 변경 요청의 응답을 잃으면, 다시 보냈을 때 같은 변경이 두 번 적용될 수 있습니다.

해결　변경 작업은 지워지지 않는 원장에 남기고, 같은 요청은 같은 작업으로 묶습니다. 작업마다 실행권과 처리 순서를 정해 한 실행기만 제출하게 합니다. 결과를 모르는 변경은 다시 보내지 않고 공식 조회로 확인합니다.

광고로 들어온 주문의 수수료

문제　토스쇼핑은 광고로 들어온 주문에 상품 판매수수료를 다르게 적용합니다. 이를 구분하지 않으면 예상 손익이 실제와 어긋납니다.

해결　공식 주문 조회의 혜택 근거로 광고 유입 품목을 확인하고, 주문 내역과 발주 추적에 표시합니다. 예상 손익은 그 품목에 적용된 수수료율을 씁니다. 운영 주문 화면과 발주 추적에 광고 유입 표시가 나타났습니다.

공식 조회 전 연결 표시

문제　공식 조회가 성공하기 전에 연결됨으로 보이면, 빈 화면이나 다른 계정의 성과를 실제 값처럼 볼 수 있습니다.

해결　공식 조회가 성공한 계정만 연결됨으로 표시하고, 조회 결과가 없는 채널은 빈 값으로 둡니다. 계정이 바뀌면 연결을 끊고, 이전 성과를 새 계정에 붙이지 않습니다.

— 디핏

가격·프로모션·정산

기간	2026.08
팀	1인
역할	판매가 정책　프로모션 실행　정산·환불 원장　자금 원장　손익 검산
기술	Go · PostgreSQL · TypeScript · Next.js

공급가와 수수료를 반영한 판매가를 채널에 적용하고, 정산과 실제 자금으로 손익을 맞추는 원장

신선식품은 공급가가 자주 바뀌고, 채널마다 정상가·쿠폰·기간 판매가가 다릅니다. 공급가·수수료·부가세를 반영해 목표 마진 판매가를 계산합니다. 채널에 적용한 뒤에는 공식 조회로 다시 확인합니다. 일반 판매가와 프로모션 고객가는 따로 관리합니다. 채널 정산·환불·은행 거래를 각각 원장으로 남겨 예상 손익과 실제 손익을 맞춥니다.

기능

- 공급가·채널 수수료·부가세로 목표 마진 판매가를 계산하고, 적용 뒤 공식 조회로 확인하는 가격 반영
- 일반 판매가와 프로모션 고객가를 나누고, 쿠팡 쿠폰·즉시할인과 토스쇼핑 기간 판매가를 관리하는 가격 정책
- 등록 판매가를 고정하고 프로모션 고객가의 범위를 정해 두는 가격 고정
- 쿠팡·토스쇼핑 공식 정산 이력은 보존하고 새 기록만 추가하며, 공식 환불만 주문 손익에서 한 번 차감하는 정산 원장
- 은행 거래내역 파일과 공급사 예치금을 위치별로 나눈 자금 원장
- 상품·옵션별 손익과 배분되지 않은 비용을 보여 주는 손익 검산

정책과 다른 채널 가격

문제　가격 정책을 바꿔도 채널에 실제 반영된 가격이 다를 수 있고, 지난 작업이 새 작업과 겹쳤습니다.

해결　정책 의도와 채널 적용 결과를 나눠 기록하고, 최신 평가가 같은 정책의 지난 작업을 끝내게 했습니다. 적용 직후에는 공식 조회로 확인하고, 필요하면 다시 적용합니다.

요청 제한과 일반 거절의 구분

문제　가격을 연속으로 바꾸면 채널이 요청 제한으로 거절했는데, 이를 일반 거절로 닫고 원인을 버렸습니다.

해결　판매 계정마다 가격 변경 최소 간격을 두고, 확정된 요청 제한만 다시 예약했습니다. 거절 코드와 사유는 기록에 남깁니다. 일시적인 제한에서는 회복하고, 결과 불명이나 일반 거절은 무조건 다시 보내지 않습니다.

채널 계산과 어긋나는 고객가

문제　채널마다 할인 금액을 반올림하는 방식과 정률 쿠폰의 할인 상한이 달라, 계산한 고객가가 실제와 어긋났습니다.

해결　채널의 실제 반올림 방식으로 고객가를 계산하고, 정률 쿠폰의 최대 할인액에 별도 상한을 두었습니다.

반려된 은행 연동

문제　은행 거래내역을 직접 받으려던 오픈뱅킹 이용 신청이 반려됐습니다.

해결　은행에서 내려받은 암호화 거래내역 파일을 올려 가져오게 바꿨습니다. 같은 시각·같은 금액의 거래를 구분하고, 사람이 고친 분류는 보존합니다.

정산 손익과 회계 결산의 혼동

문제　채널 정산으로 계산한 손익을 최종 회계 결과로 읽을 수 있었습니다.

해결　정산 반영 손익과 최종 회계 결산을 나눕니다. 비용 근거가 없으면 0으로 채우지 않습니다. 화면에는 정산 기준으로 계산한 손익임을 표시합니다.

— 디핏

라우팅·판매 상태 제어

기간	2026.07 — 2026.09
팀	1인
역할	라우팅 정책 설계　상품 동일성 판정　판매 상태 자동 제어　재고 유지
기술	Go · PostgreSQL · TypeScript · Next.js

판매 옵션마다 규격이 정확히 맞는 공급사를 고르고, 발주 경로가 사라지면 판매를 멈추는 제어

같은 과일도 공급사마다 품종·중량·입수·등급 표기가 다릅니다. 판매 옵션마다 현재 발주 경로와 대안을 보여 주고, 운영자가 고정·선호·정지를 정합니다. 상품명이 비슷한 후보가 아니라 규격이 정확히 맞는 상품만 고릅니다. 발주 경로가 사라진 옵션은 판매를 멈추고, 경로가 돌아오면 다시 엽니다.

기능

- 자동 선택 위에 고정(PIN)·선호(PREFER)·정지(HOLD)와 후보 제외를 두는 라우팅 정책
- 품목·품종·중량·입수·등급·크기와 채널 옵션 식별자로 같은 상품을 판정하는 매칭
- 기간이 끝난 선호 경로는 남은 적격 후보로 넘기는 처리
- 발주 경로를 잃은 옵션의 판매 자동 중지와 경로 회복 뒤 재개
- 채널별 재등록 기준과 운영자가 직접 켠 옵션의 재고 유지

상품명으로는 알 수 없는 같은 상품

문제　채널의 중량 표기와 공급사의 입수 기준이 다르고, 품종 별칭과 등급 이름도 공급사마다 달랐습니다.

해결　채널 중량과 발주 입수 기준을 나눠 저장하고, 품종 별칭과 등급 통합 규칙을 두었습니다. 고객이 주문한 상품의 규격을 기준으로 경로를 고릅니다. 이름만 비슷한 공급사 상품으로 발주하지 않습니다.

흩어진 과거 주문의 옵션

문제　과거 쿠팡 주문에 임시 옵션이 남아, 같은 판매 옵션이 여러 개로 보였습니다.

해결　임시 옵션을 쿠팡 옵션의 변하지 않는 식별자로 묶어 옵션 목록을 정리했습니다.

공급이 끊겨도 계속 팔리는 옵션

문제　공급사 품절로 발주 경로가 없는 옵션이 판매채널에서 계속 팔릴 수 있었습니다.

해결　발주 경로를 잃은 옵션은 판매를 자동으로 멈추고, 경로가 돌아오면 다시 엽니다. 판매 상태는 운영자의 조치나 경로 상실이 있을 때만 바꿉니다.

다시 설명할 수 없는 선택

문제　후보 목록은 계속 바뀌어, 지난 주문에서 왜 그 공급사를 골랐는지 나중에 설명할 수 없었습니다.

해결　선택 당시의 근거는 바꾸지 않는 기록으로 남기고, 현재 후보는 따로 저장했습니다.

— 디핏

에이전트 운영·조직 기억

기간	2026.07 — 2026.08
팀	1인
역할	역할·권한 설계　무인 운영과 복구　장애 기록과 알림　조직 기억
기술	Hermes Agent · Codex CLI · Python · Shell · Telegram · Discord · launchd · OpenViking · Git

총괄 에이전트가 역할별 에이전트에 일을 나누고, 장애와 사업 기록을 재시작 뒤에도 남도록 관리하는 체계

총괄 에이전트는 Hermes Agent 위에서 상시 운영되며 인프라 관측·장애 조사·운영 판단을 맡습니다. 회계·개발·상품 등록·고객 응대·광고 판단은 역할별 에이전트가 나눠 맡습니다. 역할 설명이 아니라 실제 도구 권한과 완료 조건으로 실행을 통제합니다. 장애는 재시작에도 남는 기록으로 관리하고, 검토된 사업 사실은 Git 문서로 남깁니다.

기능

- 역할마다 할 수 있는 일과 금지 행동, 결과·재시도 규칙을 기계로 읽는 권한 목록
- 재무 역할이 정해진 조회 스크립트와 읽기 도구만 쓰도록 도구 호출 전에 막는 훅
- 장애를 원인별 고정 ID로 기록하고, 재시작을 새 장애로 세지 않는 장애 기록과 알림
- 정전·재부팅·자동 업데이트 뒤 자동으로 기동하고 메신저 연결까지 확인하는 무인 운영
- 검토된 사업 사실을 담은 Git 문서와, 그 문서에서 다시 만드는 대화 기억·검색 색인
- 현재를 묻는 질문은 현재 상태 문서를 먼저 찾고, 과거 결정을 현재 사실로 올리지 않는 검색

재배포마다 다시 울리던 같은 장애

문제　재시작이나 재배포 때마다 같은 공급 수집 실패가 새 경보로 다시 왔습니다.

해결　장애를 공급 범위와 원인별 고정 ID로 기록하고, 최근 성공·연속 실패 시작·발생 횟수·알림 여부를 보존했습니다. 로그가 더 보이지 않는 것은 복구로 보지 않고, 공식 근거가 있어야 복구로 판단합니다. 같은 장애는 한 번만 알리고, 처음 알린 스레드에서 종결합니다.

자동 업데이트 뒤 끊긴 메신저 연결

문제　자동 업데이트 뒤 에이전트가 메신저 연결 서비스를 직접 다시 시작하다가, 서비스 등록이 빠진 채 연결이 완전히 끊겼습니다.

해결　업데이트는 별도 실행기가 맡고, 서비스 등록과 실행 중인 프로세스가 함께 확인될 때만 성공으로 기록합니다. 독립 감시가 상태를 보고 복구합니다. 일부러 서비스를 내린 시험에서 감시가 새 프로세스로 복구했고, Telegram과 Discord의 재연결을 확인했습니다.

정전과 재부팅 뒤의 복구

문제　정전이나 재부팅 뒤에는 사람이 로그인하기 전까지 에이전트와 검색 서버가 멈춰 있었습니다.

해결　전원이 돌아오면 자동 부팅하고, 자동 로그인 뒤 에이전트·검색 서버·작업 워커를 띄우게 했습니다. 부팅 뒤 검증 스크립트가 프로세스, 메신저 연결, 검색 서버 상태를 확인합니다. 실제 재부팅에서 부팅 뒤 검사를 모두 통과했습니다.

과거 결정이 현재 사실처럼 보이는 검색

문제　검색 결과에는 과거 결정과 현재 결정이 섞여 나옵니다.

해결　현재를 묻는 질문은 현재 상태 문서 범위를 먼저 검색해 지난 기록보다 앞에 둡니다. 중요한 판단은 원문을 다시 읽고, 보관 폴더는 기본 검색에서 뺐습니다.

아키텍처

총괄 에이전트
유일한 요청 창구 · 최종 판단

해석 → 계획 → 배정 → 실행 → 검증 → 판단

지식·기억

- **사업 기록**
  목표 · 결정 · 현황 · 로드맵
  정해진 절차 없이 자동으로 바꾸지 않음
- **에이전트 규칙**
  원칙 · 워크플로 · 운영 일지를 Git으로 관리
  사업 기록과 섞지 않음
- **런타임 기억**
  Hermes Agent 내장 기억 · 검색 계층이 멈추면 대신 사용
- **파생 검색**
  OpenViking · Git 문서와 대화 기억을 다시 색인
  기준 기록으로 쓰지 않음

실행·통제

- **업무별 실행 시스템**
  커머스 백엔드 · 운영 콘솔 · 소싱 수집기 · 인프라 · DB · 실제 실행은 각 시스템의 권한 확인을 따름
  역할 에이전트의 보고만으로 완료 처리하지 않음
- **작업 원장**
  장기 · 인계 · 재시도 · 감사가 필요한 작업만 기록
- **권한 등급** L0 · L1
  L0 조회 · 분류 · 초안 / L1 정해진 워크플로 단계 실행
  등급 자체는 외부 쓰기 권한이 아님
- **운영 알림**
  장애 알림은 총괄 에이전트를 깨우는 신호
  알림이 판단을 대신하지 않음

개발 전용 역할

- **원인 조사** L0
  코드 · 로그 · 공식 기술 문서를 읽고 근거 정리
  수정 · 상태 변경 없음
- **티켓 구현** L0
  티켓 범위 안에서 구현 · 테스트 · 커밋 · 로컬 검증
  병합 · 배포 없음
- **QA·보안 판정** L0
  완성된 변경의 회귀 · 보안 판정
  구현 · 배포 결정 없음

도메인 역할

- **상품 등록** L1
  상품 기획 · 상세페이지 · 등록 후 공식 재조회 검증
  DB · 판매채널 직접 쓰기와 발주 없음
- **고객 문의** L0
  고객 문자 분류 · 사실 정리 · 답변 초안
  발송 · 환불 · 보상 없음
- **재무 조회** L0
  매출 · 원가 · 정산 · 현금 흐름 원천 조회와 대사
  송금 · 발주 없음 · 연결되지 않은 원천을 0으로 채우지 않음
- **광고 판단** L0
  확인된 근거로 광고 · 마케팅 판단
  집행 · 예산 · 입찰 변경 없음
- **공급 공지** L0
  공급사 공지 관측 · 정규화 · 출처 보존
  가격 · 재고 · 발주 변경 없음

결과 흐름

역할 실행 → 정해진 형식의 결과 · 근거·검증·위험 → 총괄 검증 · 실제 원천 대조 → 업무 시스템이 권한 확인 후 실행 → 공식 재조회로 완료 확인

— 디핏

소싱 데이터

기간	2026.07 — 2026.08
팀	1인
역할	공급가 수집　시장·키워드 분석　소싱 추천　브라우저 확장　공급사 공지 수집
기술	Python · Playwright · PostgreSQL · MinIO · NumPy · TypeScript · Chrome Extensions

도매 공급사 가격과 검색 수요·경쟁을 모아, 공급할 수 있는 과일 가운데 무엇을 팔지 고르는 데이터

발주할 공급사를 고르려면 공급사별 상품과 가격부터 알아야 했습니다. 도매 공급사 가격 수집으로 시작해 네이버 검색 수요, 경쟁 지표, 키워드 연관어를 더했습니다. 이 자료로 실제 공급할 수 있는 품목 가운데 다음에 팔 품목을 추천합니다. 공급사 단체방 공지도 모아 공급 중단과 가격 변동에 대응합니다.

기능

- 공급사 공식 엑셀을 먼저 읽고, 실패하면 API·웹 화면, 그다음 직전 성공본 순서로 가격을 남기는 수집
- 품목·품종·중량·입수·등급으로 정규화하고 원문 상품명과 가격도 보존하는 공급가 원장
- 네이버 월별 인기 키워드와 일별 트렌드, 키워드 온톨로지를 나눠 보관하는 시장 분석
- 공급 가능성·수요·경쟁·시기 근거를 함께 보여 주는 소싱 추천
- 토스쇼핑 인기 상품에서 품목·옵션별 대표 가격을 뽑는 경쟁 가격 관측
- 사람이 연 쿠팡 검색·상품 화면만 구조화해 저장하는 Chrome 확장
- 공급사 카카오톡 공지 원문을 보존하고, 반품·교환 제한을 공급사 상품과 발주 후보 판정에 잇는 공지 수집

잘못된 가격 한 줄이 막던 가격표

문제　공급가 파일에 빈 값이나 잘못된 가격이 한 줄만 있어도 정상 가격표 전체가 실패했습니다.

해결　파일 구조 검사와 행 단위 격리를 나누고 정상 행만 갱신했습니다. 불량 행은 이전 정상 가격과 관측 시각을 유지합니다. 정상 데이터를 잃지 않고, 확인하지 않은 가격이 새것처럼 보이는 일도 막았습니다.

갑자기 비어 버린 카탈로그

문제　공급사 카탈로그가 비거나 크게 줄어든 채 수집되면, 정상 가격표를 덮어쓸 수 있었습니다.

해결　빈 카탈로그와 급감한 카탈로그는 부분 자료로 보고, 마지막 정상본을 보존합니다. 일시적인 수집 이상이 발주 경로를 지우지 않습니다.

중량과 포장이 차지한 추천 순위

문제　추천 후보 상위를 품목이 아니라 중량·포장·등급 차이가 차지했습니다.

해결　중량·포장·등급을 품목 후보가 아닌 옵션으로 내렸습니다. 계절성과 수요 흐름도 따로 계산합니다.

늦게 그려지는 리뷰와 사라지는 관측

문제　상품명·가격보다 리뷰·평점이 늦게 그려져 빈 값이 저장되고, 전송에 실패한 관측은 사라졌습니다.

해결　카드 변화를 지켜보다가 화면이 안정되면 한 번 읽게 했습니다. 전송에 실패한 관측은 보관함에 남겨 다시 보냅니다. 무한 대기나 스크롤 조작 없이 늦게 그려지는 화면을 읽습니다.

— 디핏

개발·인프라 운영

기간	2026.07 — 2026.10
팀	1인
역할	저장소 간 연동 규약　영향 기반 검증　배포 절차　서버 구성　백업과 복원
기술	Git · Python · Shell · Proxmox · Docker · Jenkins · PostgreSQL · MinIO · Cloudflare R2 · SOPS · Prometheus · Vercel

저장소 19곳의 변경을 vat로 묶고, 배포 뒤 실제 상태와 사업 기록까지 확인하는 절차

백엔드·콘솔·수집기·에이전트 런타임이 저장소 19곳에 나뉘어 있습니다. vat로 여러 저장소에 걸친 변경과 저장소 사이 연동 규약을 관리하고, 바뀐 범위만 검증합니다. 서버는 Proxmox 한 대에 API·PostgreSQL·MinIO·Jenkins를 나눠 두고, 콘솔은 Vercel에 둡니다. 백업은 Cloudflare R2에 암호화해 보관하고 복원까지 확인합니다. 자격증명은 SOPS로 암호화해 관리합니다.

기능

- 여러 저장소에 걸친 변경을 참여 커밋·검증·되돌릴 지점과 함께 묶는 기록
- 저장소 사이 연동 규약을 선언하고 기계로 검사하는 계약
- 기준 커밋과 현재 파일의 해시, 의존 관계로 검사를 고르고 영향이 불명확하면 막는 검증
- 코드 검증·푸시·배포·운영 재조회·사업 기록 반영까지 통과해야 끝나는 배포 절차
- 검증, 이미지 전송, 마이그레이션, 서비스 기동, 실행 중인 커밋 대조로 이어지는 Jenkins 배포
- PostgreSQL 덤프와 필요한 객체를 암호화해 Cloudflare R2에 보관하고, 격리된 환경에 복원해 확인하는 백업
- SOPS로 암호화해 Git에 두는 자격증명

다른 저장소가 읽는 테이블의 변경

문제　수집기가 백엔드 테이블을 직접 읽는 SQL은 백엔드 마이그레이션이 바뀌면 조용히 깨질 수 있습니다.

해결　수집기 SQL의 테이블·컬럼·타입·허용 값을 마이그레이션과 대조하고, 모양을 알 수 없는 동적 SQL은 막았습니다. 실제 DB 상태 검증과 구조 대조는 따로 돌립니다.

제품만 배포되고 남지 않는 기록

문제　제품만 배포되고 사업 기록이 로컬에 남으면, 다음 작업자는 실제 배포 상태를 알 수 없습니다.

해결　코드 검증과 푸시, 허용된 범위의 배포와 운영 재조회, 사업 기록 반영까지 통과해야 작업이 끝나게 했습니다.

복호화해 보지 않은 백업

문제　백업 파일이 있어도 실제로 풀어 복원할 수 있는지는 알 수 없습니다.

해결　백업은 오프사이트 사본을 만들고 복호화해 격리된 환경에 복원한 뒤에야 완료로 봅니다.

브라우저 좀비 프로세스가 다 쓴 PID

문제　브라우저 자동화가 남긴 Chromium 하위 프로세스를 거두지 않아, 좀비 프로세스가 PID 한도를 다 썼습니다. 헬스 체크와 요청 처리 스레드를 만들 수 없었습니다.

해결　컨테이너에 init 프로세스를 켜고, 배포 때 실제 설정에 켜져 있는지 확인하게 했습니다.

— 디핏

운영 콘솔·고객 응대

기간	2026.07 — 2026.08
팀	1인
역할	운영 화면 설계　서버 중계 계층　권한·사업자 경계　고객 문자　클레임 케이스
기술	Next.js 16 · React 19 · TypeScript · Tailwind CSS 4 · Vercel · Go · PostgreSQL

주문·발주·가격·정산을 한곳에서 처리하고, 고객 문자와 클레임을 주문 단위로 잇는 운영 웹

주문·발주·판매가·상품 등록·고객 문의·환불·정산·자금 상태를 41개 업무 화면에서 보고 처리합니다. 화면은 금액이나 단계를 직접 계산하지 않고 백엔드 응답만 그립니다. 고객 문자는 직접 운영하는 회선 단말로 주고받습니다. 문의·반품·교환·환불은 주문별 클레임 케이스로 묶습니다.

기능

- 메뉴 권한, 관리자 전용 조치, 허용한 경로만 넘기는 서버 중계 계층으로 나눈 3중 권한
- 여러 사업자를 한 콘솔에서 전환하고, 주문·계정·공급사 실행은 사업자별로 격리
- 연락처·주문번호를 화면에서 가리는 민감정보 숨김 모드와 좁은 모바일 화면용 배치
- 직접 운영하는 회선 단말로 고객 문자를 주고받고 주문에 자동으로 잇는 문자 게이트웨이
- 주문자 정보로 새 주문을 보류하는 블랙리스트
- 문자·반품·교환·환불을 주문 단위 케이스로 묶고, 개인정보를 가린 입력으로 AI 분류 제안을 받는 클레임 처리

해외 문자 서비스로는 안 되던 양방향 문자

문제　해외 메시징 플랫폼은 한국 번호의 양방향 문자를 지원하지 않아, 고객 회신을 받을 수 없었습니다.

해결　직접 운영하는 회선의 단말을 게이트웨이로 써서 문자와 이미지 문자를 주고받게 했습니다. 모든 발신 주소는 원문 그대로 보존하고, 차단한 발신자는 저장 전에 버립니다.

장애 종류를 지키는 중계 계층

문제　중계 계층이 오류를 하나로 뭉치면 시간 초과와 사용자 취소, 전송 실패를 구분할 수 없습니다.

해결　시간 초과는 504, 사용자 취소는 499, 그 밖의 전송 실패는 502로 나눴습니다. 요청 ID로 백엔드 기록과 잇되, URL·본문·헤더는 로그에 남기지 않습니다. 네트워크 오류는 로그아웃으로 처리하지 않습니다.

취소와 엇갈린 사진 요청 문자

문제　사진 요청을 예약한 뒤 고객이 회신하거나 취소가 들어오면, 필요 없는 문자가 나갈 수 있었습니다.

해결　보내기 직전에 케이스와 요청을 차례로 잠그고 회신·환불·처리 경로를 다시 확인합니다. 취소가 먼저 커밋되면 보내지 않고, 발송이 먼저 확정되면 남은 요청만 취소합니다. 두 순서를 실제 PostgreSQL 동시성 테스트로 고정했습니다.

금액 형식 변화로 멈춘 반품 수집

문제　채널 반품 응답의 배송비가 숫자가 아닌 금액 객체로 와서, 목록 전체를 읽지 못하고 수집이 멈췄습니다.

해결　객체·숫자·숫자 문자열을 모두 받게 하고, 실패는 수집 방식별로 격리했습니다. 진단에는 필드 이름과 타입만 남깁니다. 배포 뒤 수집이 다시 이어졌고, 빠진 이벤트를 채우면서 알림을 다시 보내지 않았습니다.

— 디핏

상품 등록

기간	2026.07 — 2026.08
팀	1인
역할	상품 등록 에이전트 설계　등록 상태 흐름　상세페이지 생성과 검수　채널 게시와 확인
기술	Go · PostgreSQL · MinIO · Hermes Agent · Node.js · Playwright · JSON Schema

상품 등록 에이전트가 기획 초안과 상세페이지 자산을 만들고, 확인을 통과한 상품을 두 채널에 게시한 뒤 공식 조회로 확인하는 흐름

상품 등록 에이전트가 상품 하나를 기획부터 채널 게시 확인까지 맡습니다. 실제 상품 사진과 백엔드의 가격 계산을 근거로 기획 초안과 상세페이지 자산을 만듭니다. 기획 초안에는 상품명·옵션·검색어를 담습니다. 정해진 절차를 통과하면 백엔드가 쿠팡·토스쇼핑에 등록합니다. 에이전트는 채널 공식 조회로 실제 등록 상태를 다시 확인합니다. 첫 실제 상품을 두 채널에 등록하고 게시까지 확인했습니다.

기능

- 기획 초안·상세페이지 자산·두 채널 게시·공식 조회 확인을 같은 등록 건으로 잇는 상태 흐름
- 서로 다른 실제 상품 사진 세 장을 근거로 삼고, 맞지 않는 사진은 다른 후보로 바꾸는 기획
- 자동 등록 전에 옵션마다 발주 경로를 2개 이상 확보하게 한 선행 조건
- 정해진 절차 우회와 DB·판매채널 직접 쓰기, 공급사 발주를 막은 에이전트 권한
- 운영자가 고른 사진과 등록 입력을 해시와 함께 보관하는 수동 등록 자료

서로 다른 상품처럼 보이던 상세 이미지

문제　상세페이지 이미지는 한 장씩 따로 생성돼, 같은 과일이 장마다 다른 품종처럼 보일 수 있었습니다.

해결　조사한 품종·과피·외형을 모든 이미지의 생성 지시에 똑같이 넣었습니다. 생성된 이미지는 한 장씩 비전 모델로 검사해 같은 상품인지 확인했습니다. 모든 이미지가 검사를 통과해야 작업이 완료됐습니다.

근거 없이 시작될 수 있던 이미지 작업

문제　실제 사진 근거가 부족하면 다른 품종이나 합성 이미지를 기준으로 생성될 위험이 있었습니다.

해결　백엔드가 이미지 작업을 큐에 넣기 전에 같은 등록 건의 서로 다른 실제 사진이 세 장 이상인지 확인했습니다. 합성·일반 이미지이거나 조사 내용이 바뀌었으면 작업을 멈췄습니다. 근거가 맞지 않는 작업은 유료 생성 전에 멈췄습니다.

결과를 모르는 유료 재시도

문제　작업이 끝났는데 완료 응답이 사라지면, 같은 이미지를 다시 생성해 비용이 두 번 들 수 있었습니다.

해결　재시도 전에 백엔드의 작업 상태와 보관된 결과를 먼저 확인했습니다. 결과가 있으면 완료 보고만 다시 보내고, 결과를 알 수 없으면 유료 생성을 자동으로 다시 실행하지 않았습니다. 보관된 결과가 있는 작업은 다시 생성하지 않았습니다.

손대지 않은 이미지까지 바뀌던 수정

문제　문구만 고쳐 달라는 요청에도 이미지와 구도가 함께 바뀔 수 있었습니다. 다시 생성하면 비용도 다시 들었습니다.

해결　수정을 지정 이미지, 자동 선택, 문구 전용으로 나눴습니다. 선택하지 않은 이미지는 기존 결과에서 그대로 복사하고, 실패한 이미지만 다시 생성했습니다. 문구 수정은 이미지 생성 없이 마쳤고, 원본 이미지의 해시가 유지되는지 확인했습니다.

— 디핏

발주 자동화

기간	2026.07 — 2026.08
팀	1인
역할	업무 정의와 도메인 설계　판매채널·공급사 연동　무인 발주 실행과 복구　발주 추적
기술	Go · chi · PostgreSQL · pgx · sqlc · goose · OpenAPI · MinIO · Xvfb

디핏이 가장 먼저 만든 기능입니다. 고객 결제부터 공급사 발주·송장·배송 완료 확인까지 잇습니다.

디핏은 도매 공급사의 과일을 쿠팡·토스쇼핑에서 파는 위탁판매로 시작했습니다. 주문이 들어오면 공급사를 골라 발주·결제하고, 공급사 송장을 판매채널에 등록해야 합니다. 이 일을 가장 먼저 자동화했습니다. 첫 실제 결제 주문을 자동으로 발주했고, 이어서 배송완료 확인까지 이었습니다.

기능

- 판매채널 주문을 중복 없이 받아 공통 모델로 저장하는 주문 수집
- 주문 품목을 공급사 상품에 연결하고, 실제 발주 없이 후보와 비용을 보는 모의 발주
- 공급사별 엑셀 양식에 맞춘 발주 파일
- 작업 큐와 중복 발주 방지를 갖춘 무인 발주·결제·송장 회수
- 판매채널의 공식 배송완료까지 확인하는 흐름
- 주문·옵션 단위로 자동 발주와 결제를 멈추고 다시 켜는 중지·재개

응답을 잃은 발주와 결제

문제　공급사에 발주나 결제를 보낸 뒤 응답을 잃으면, 실제로 처리됐는지 알 수 없습니다.

해결　결과를 모르는 요청은 결과 불명으로 두고, 공급사 공식 조회로 확정하기 전에는 다시 보내지 않습니다. 다른 공급사로 다시 제출하지도 않습니다. 제출에 결제가 포함되는 곳과 결제 API가 따로 있는 곳도 나눠 기록합니다. 재시도 때문에 같은 주문을 두 번 발주하거나 결제하지 않습니다.

정전과 네트워크 단절 뒤 빠진 인계

문제　공급사에서 송장을 읽은 뒤 채널에 등록하기 전에 서버가 멈추면, 다음 단계가 생기지 않았습니다.

해결　외부 조회 결과와 다음 처리 단계를 함께 기록했습니다. 서버가 멈추면 DB에 저장한 마지막 처리 지점부터 다시 시작하게 했습니다. 장애가 길어져도 남은 발주·송장 인계를 이어서 처리합니다.

손해를 보는 주문

문제 판매가를 정한 뒤 공급가가 오르면, 주문이 들어올수록 손해가 커질 수 있습니다.

해결 제출과 결제 직전에 실제 결제액에서 수수료·부가세·공급 원가를 빼서 다시 계산합니다. 기여이익이 음수이거나 근거를 확인할 수 없으면 발주를 막습니다.

공급사 로그인의 봇 차단

문제 공급사 사이트가 로그인 단계에서 자동화 접속을 막으면, 발주와 송장 회수가 멈췄습니다.

해결 서버의 가상 화면에서 브라우저를 띄워 인증을 복구하고, 얻은 세션을 백엔드에 안전하게 넘깁니다. 운영자에게 로컬 브라우저 작업을 요구하지 않습니다.

실행 중인 발주와 부딪친 중지 버튼

문제 옵션 중지 저장이 공급사 재평가 작업과 잠금을 다투다가 HTTP 500으로 실패했습니다.

해결 공용 잠금은 공유 모드로 바꿔 독립 작업을 병렬로 돌리고, 실제 발주와 중지는 같은 잠금으로 차례를 세웠습니다. 오래 기다리면 저장 전에 409 충돌로 돌려줍니다. 옵션 전체를 멈춘 상태에서도 주문 한 건만 재개할 수 있습니다.

아키텍처

판매채널
- **주문 수집**
  쿠팡 · 토스쇼핑 · 마지막 성공 지점부터 재개
- **송장 등록**
  공급사 송장을 채널에 등록 · 공식 조회로 확인

발주 실행
- **작업 큐**
  일정 시간 동안 한 실행기만 처리 · 재시도 · 중복 발주 방지
- **공급사 연동**
  제출 · 결제 · 송장 회수 · 공급사별 결제 방식
- **서버 브라우저**
  공급사 로그인 인증 복구

기록
- **단계별 처리 기록**
  외부 조회 결과와 다음 처리 단계를 함께 기록
- **감사 이력**
  기존 이력은 보존하고 새 이벤트만 추가 · 결과를 모르면 보류

주문 접수 → 후보 선정 → 상품 준비 → 공급사 제출 → 공급사 결제 → 출고·송장 확인 → 채널 송장 등록 → 배송 중 확인 → 배송 완료 확인

● 개인

보르테소 · 커피 기록 앱

기간	2025.10 — 2026.03
팀	1인
역할	기획 Flutter 앱 Spring Boot API 홈서버 배포 CI/CD 모니터링
기술	Flutter · Dart · Java 17 · Spring Boot 3.5 · Spring Security · JPA · MariaDB · Redis · MinIO · Google Cloud Vision · OpenAI API · Docker · Jenkins · Proxmox
링크	github github.com/vortesso

커피 카드 사진에서 원두 정보를 읽어 기록을 채우는 앱

마신 커피의 맛과 원두를 기록하는 앱입니다. Flutter 앱과 Spring Boot API를 만들었습니다.

기능

- 핫·아이스, 컵 노트, 메모를 남기는 컵 기록
- 맛 감각 평가와 기록 분석
- 기록 달력과 연속 기록
- Google 로그인

카드 정보를 손으로 옮겨 적는 일

문제 커피 카드에 적힌 산지, 농장, 고도, 가공 방식, 품종, 컵 노트 같은 원두 정보를 사용자가 직접 입력해야 했습니다.

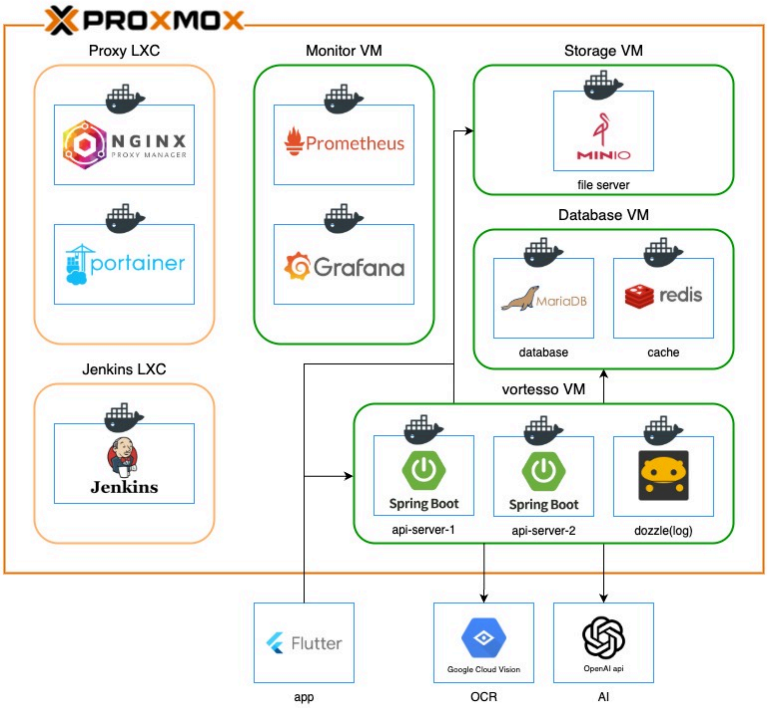
해결 카드 사진을 MinIO에 올리고 Google Cloud Vision으로 글자를 읽었습니다. 읽은 글자는 OpenAI API에 넘겨 원두 정보 항목별 JSON으로 정리했습니다. 커피와 관계없는 글자면 커피 카드가 아니라는 오류를 돌려줬습니다. 카드 사진은 임시 경로에 두었다가 사용자가 기록을 저장할 때 정식 경로로 옮겼습니다.

장비 한 대에서 서비스 운영

문제 API, 데이터베이스, 파일 저장소, CI, 모니터링을 집에 둔 서버 한 대에서 돌려야 했습니다.

해결 Proxmox 위에 프록시, CI, 모니터링, 파일 저장소, 데이터베이스, API를 VM·LXC 6개로 나눴습니다. API 서버는 2대로 띄우고 컨테이너 로그는 Dozzle로 모아 봤습니다. Jenkins는 Proxmox가 설치된 서버를 거쳐 API VM에 SSH로 접속해 새 이미지를 받아 띄웠습니다. 배포 뒤에는 상태 확인 주소를 최대 20번 호출하고 성공·실패를 푸시 알림으로 보냈습니다.

아키텍처



홈서버의 Proxmox 위에 서비스를 VM·LXC 6개로 나누고 API 서버 2대를 띄운 구성입니다.

● 개인

글로티커 · 거시경제 지표 크롬 익스텐션

기간	2025.03 — 2025.05
팀	1인
역할	기획·디자인 FastAPI 수집 서버 Spring Boot 스트리밍 서버 프론트엔드 홈서버 CI/CD 모니터링
기술	Java 17 · Spring Boot · Python · FastAPI · Redis 7.2 · React · Proxmox · Docker · Nginx Proxy Manager · Jenkins · Prometheus · Grafana
링크	github github.com/gloticker

7개국 이상에서 누적 사용자 300명 이상

금융 지표를 확인할 때마다 4~5개 사이트를 오가야 해서 만들었습니다. 수집 서버(FastAPI)와 스트리밍 서버(Spring Boot)를 나눠 홈서버에서 운영했습니다.

기능

- 지수 실시간 시세: 나스닥, S&P 500, 다우존스, 러셀 2000, 국채 20년, VIX, 공포탐욕 지수
- 주식 시세: Magnificent 7(애플, 엔비디아, 마이크로소프트, 아마존, 구글, 메타, 테슬라)
- 암호화폐 시세: 비트코인, 이더리움, 솔라나, 도미넌스
- 환율: 달러 지수, 원화 기준 달러·유로·위안·엔
- Redis 메시지 큐 기반 SSE 실시간 갱신
- ChatGPT API 시황 분석
- 30일 일봉 차트
- 숨김·테마·언어 설정

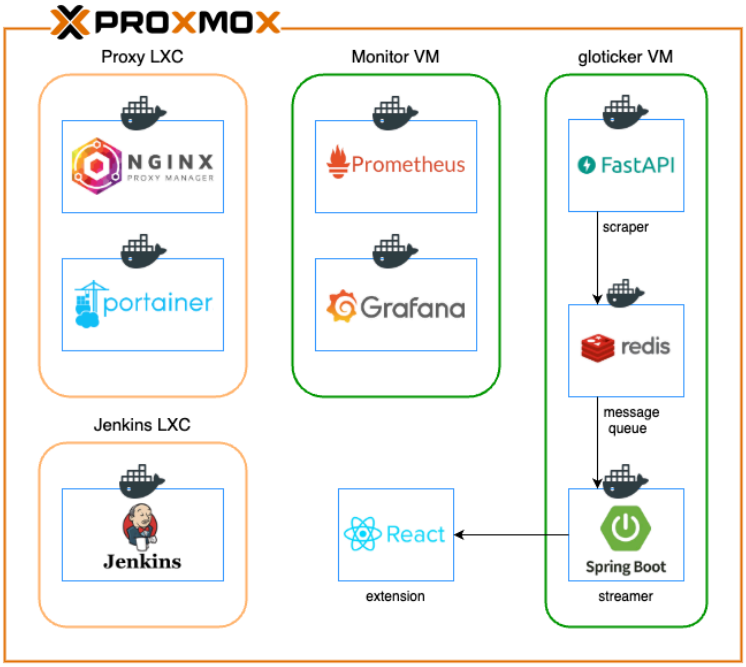
스크래핑 차단

문제	yfinance 호출이 많아지자 429가 나고, 일부 서버는 bot으로 보고 401을 돌려줬습니다. 막히지 않으면서 시세를 자주 받을 수 있는 호출 빈도를 찾아야 했습니다.
해결	여러 포럼을 참고해 초당 1회가 적당하다고 보고, 호출 간격을 1~1.2초 사이 랜덤으로 두었습니다. 호출마다 유효한 랜덤 헤더를 붙여 bot으로 인식되는 문제를 피했습니다. 공포탐욕 지수·도미넌스·환율처럼 짧은 시간에 크게 변하지 않는 값은 1분마다 갱신했습니다.

팝업마다 몰리는 API 요청

문제	익스텐션 팝업을 열 때마다 종목별 가격·변동률, 차트, SSE 구독, 분석 결과 API가 한꺼번에 호출됐습니다. 홈서버라 부하를 미리 막아야 했습니다.
해결	Redis에 최신 데이터 스냅샷을 캐시해 한 번에 보내고, 익스텐션 로컬 스토리지에 최신 데이터와 수신 시각을 남겼습니다. 수집 1회에 평균 23초가 걸리므로, 마지막 수신이 20초 안이면 SSE 구독만 보냈습니다. 차트는 30일 일봉이라 장 종료 때 한 번만 받고, Bucket4j로 클라이언트당 10초 6회로 제한했습니다.

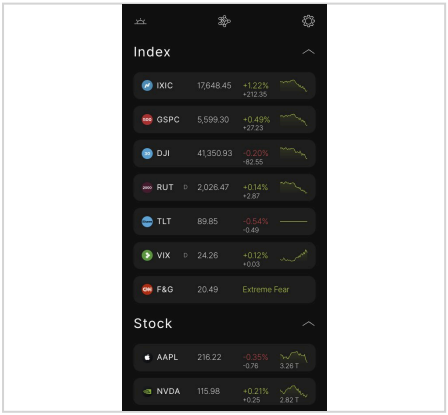
아키텍처



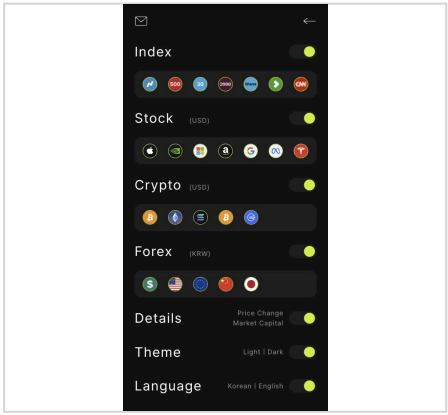
시세 수집과 스트리밍 서버를 Proxmox에서 운영합니다.



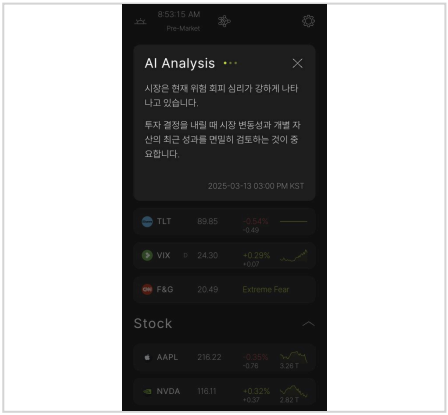
글로티커 로고 커버



지수별 등락률과 추세선을 한 화면에 모았습니다.



관심 종목과 자산 항목을 설정합니다.



시장 분석을 시세와 함께 확인합니다.

● 개인

언더더키 · 커스텀 키보드 플랫폼

기간	2024.08 — 2024.11
팀	1인
역할	기획·디자인 스위치 녹음·믹싱 랜덤 문장 공개 API MSA 코어 모듈 프론트엔드 홈서버 CI/CD 모니터링
기술	Java 17 · Spring Boot 3.3.3 · Spring Data JPA · Querydsl · MariaDB · Redis 7.2 · MinIO · Next.js · Web Audio API · Proxmox · Docker · Nginx Proxy Manager · Jenkins
링크	github github.com/underthekey sentence-api github.com/underthekey/sentence-api

2일 만에 사용자 500명

커스텀 키보드 빌드를 한곳에서 다루려고 만든 플랫폼입니다. 실제 타건음을 들으며 타자를 연습합니다. 스위치 소리는 직접 녹음하고 믹싱했습니다.

기능

- 체리 축 샘플 기반 스위치 소스 13종
- 키보드 배열별 입력 테스트
- imager: 사운드 공간감 시각화
- 볼륨 조절과 언어 변경
- 한국어 속담 약 5,000개와 영문 격언 약 300개를 언어·카테고리별로 내주는 공개 API
- 요청당 최대 20문장 응답

배열마다 늘어나는 음원 파일

문제 처음에는 DAW 템플릿으로 키 배열(약 110개)별 소스를 만들어, 스위치마다 믹스된 파일 110개를 내려보냈습니다. 내려받는 파일이 많아 성능 문제가 있었고, 이 과정을 자동화하면서 줄일 방법이 필요했습니다.

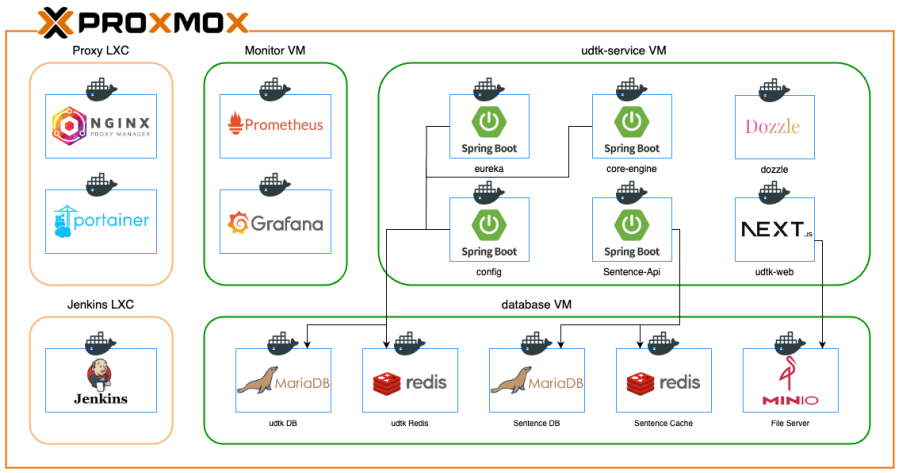
해결 믹스를 DAW 대신 클라이언트에서 Web Audio API로 하게 바꿨습니다. 8KB 소스 하나에 좌우 패닝, 상하 주파수, 키캡 크기별 필터를 걸어 모든 배열에 대응하는 템플릿을 만들었습니다. 내려받는 파일이 110개에서 1개로 줄었습니다.

공개 API 과다 요청

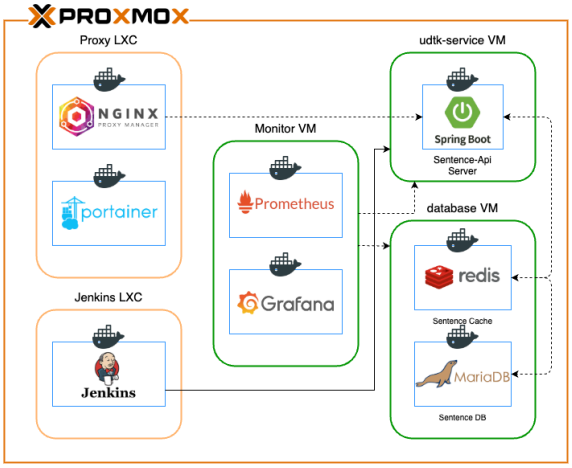
문제 랜덤 문장 API는 공개 API라 요청이 몰릴 수 있었고, 서비스 핵심 서버라 멈추면 안 됐습니다. Remote IP를 직접 받아 세면 동시성 처리 부담이 컸고, 세부 규칙도 정할 수 있어야 했습니다.

해결 정확성과 Redis TTL 연동을 따져 Bucket4j를 도입했습니다. 5분에 20회까지만 받도록 토큰을 제한해, 과도한 요청이 서버 부하로 이어지지 않게 했습니다.

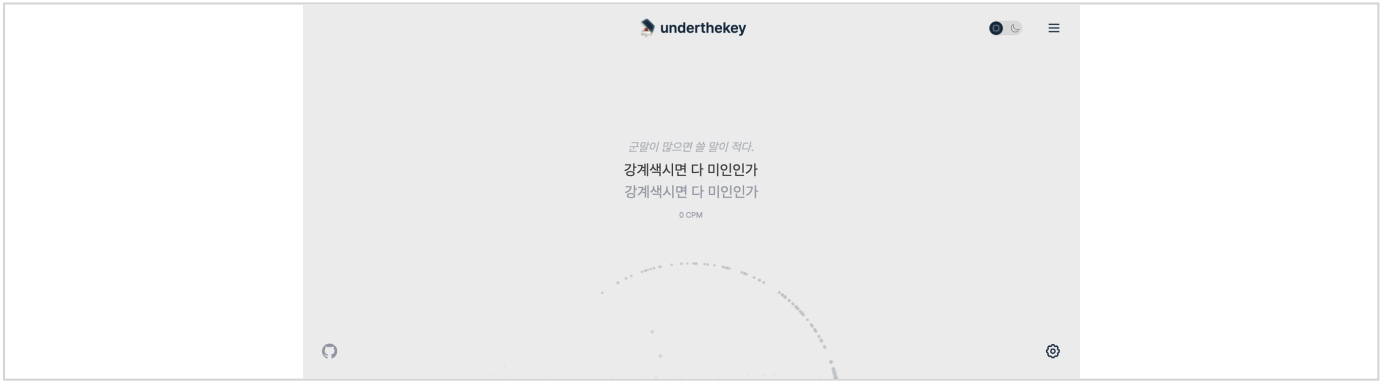
아키텍처



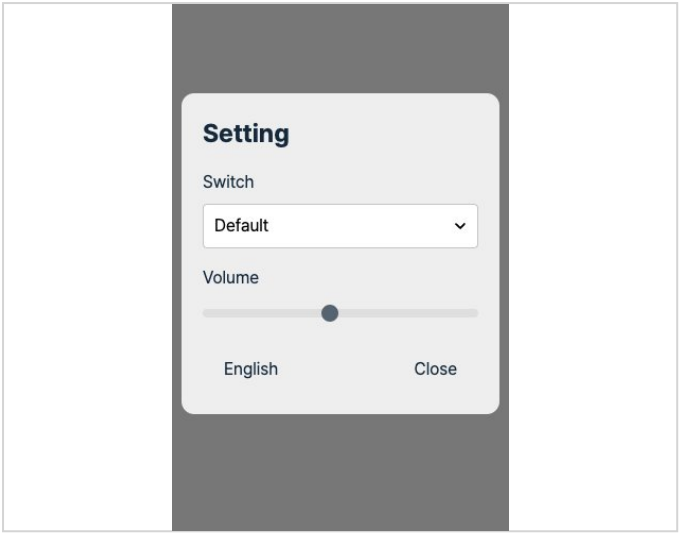
Proxmox 한 대 위에 서비스, 데이터베이스, 모니터링, CI를 나눈 구성입니다.



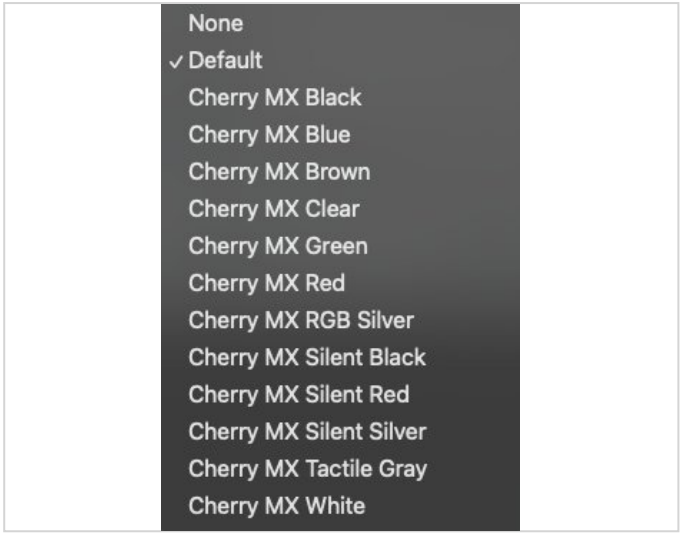
문장 API 서버와 캐시, 데이터베이스를 VM별로 나눈 구성입니다.



문장 입력과 타이핑 효과가 보이는 언더더키 화면



타건음의 볼륨과 스위치 종류를 설정합니다.



스위치별 타건음을 선택합니다.



입력에 맞춰 입자 효과를 보여 줍니다.



누른 키를 가상 키보드에 표시합니다.

SSAFY

서울팝 · 위치 기반 역사 탐방

기간	2024.04 — 2024.05
팀	6명 백엔드 3, 프론트엔드 3
역할	팀장 백엔드 리드 서버·CI/CD 회원 API 문화재·사적지 조회
기술	Java 17 · Spring Boot 3.2.5 · Spring Data JPA · Spring Security · Querydsl · MySQL 8.0 · Redis 7.2 · AWS S3 · AWS EC2 · Nginx Proxy Manager · Jenkins · Docker · Uber H3
링크	github github.com/seoulpop/seoulpop-be

팀장과 백엔드 리드로 회원·역사 조회 API와 인프라 담당

사용자 위치를 따라 서울의 문화재와 사적지를 탐방하는 서비스입니다. 서버와 CI/CD는 혼자 구축했습니다.

기능

- 서울 역사 지도
- GPS 기반 주변 문화재·사적지 알림
- 카메라 AR과 실시간 콘텐츠 렌더링
- 위치·경로 안내
- 크롤링한 데이터로 만든 역사 상세 정보
- 방문 도감 스탬프

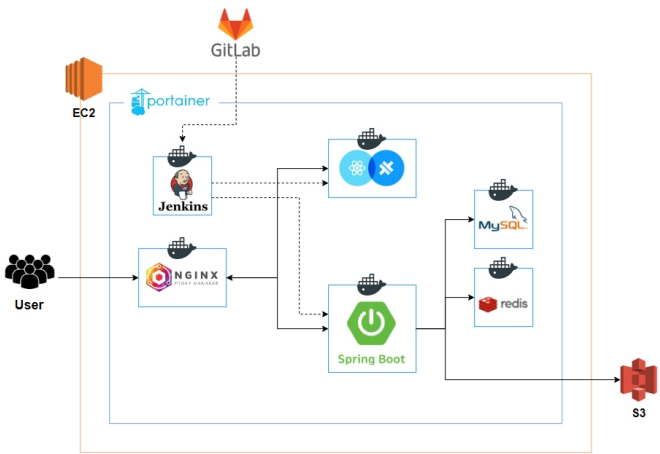
실시간 위치에 따른 거리 계산

문제	사용자가 움직일 때마다 주변 문화재·사적지 좌표를 다시 계산해야 했습니다. 최단 거리를 매번 구하면 계산 비용이 빠르게 커져, 단순 탐색보다 나은 방식이 필요했습니다.
해결	팀에서 사용자 좌표 기반 부분 탐색, GeoHash, Uber H3를 비교했습니다. 실시간 근접성 계산과 정확성, 편의성을 따져 H3를 쓰기로 했습니다. 여러 스케일을 실험해 약 200m 범위로 맞췄고, 거리 계산 구현은 팀원이 맡았습니다.

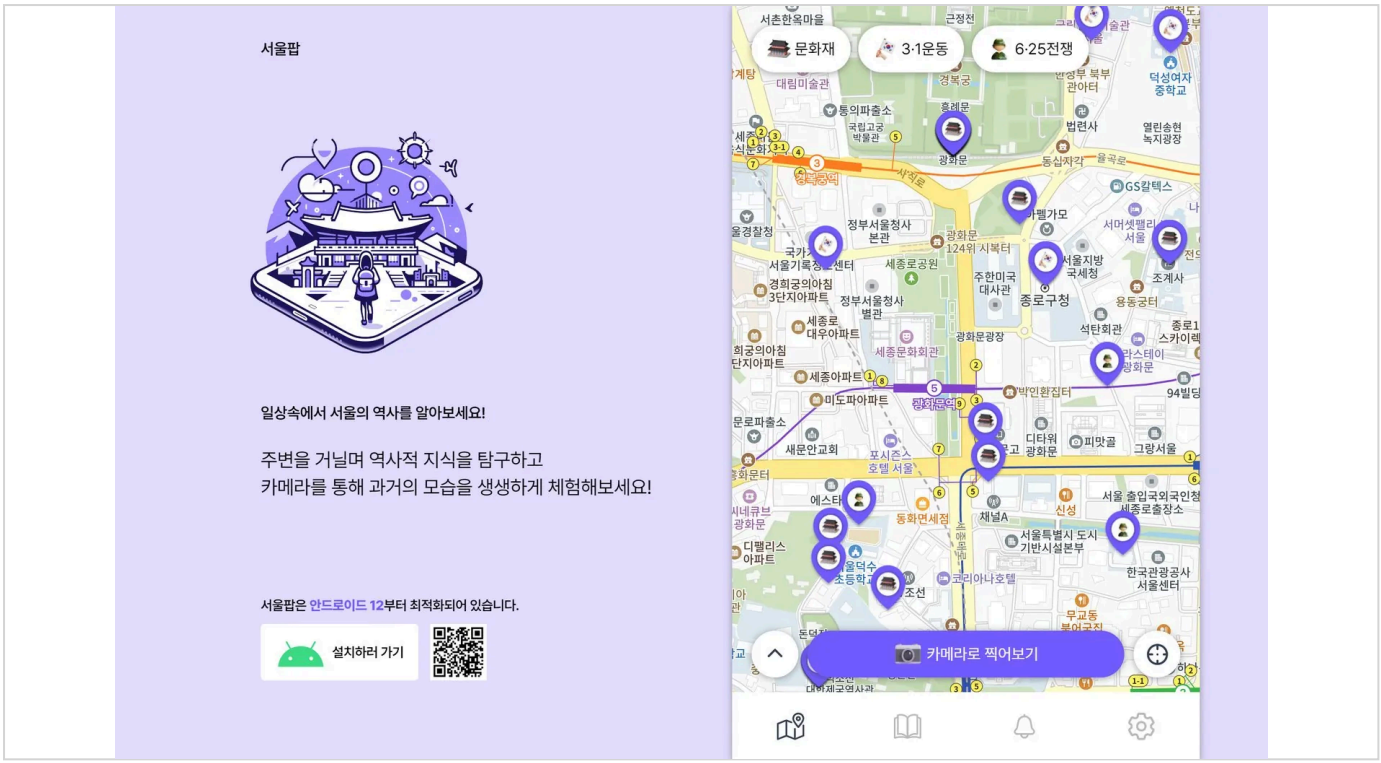
컨테이너 로그 추적

문제	Portainer로 컨테이너 로그를 보다 보니 컨테이너가 늘수록 추적이 어려워졌습니다. Portainer 콘솔의 스크롤 문제 때문에 이전 로그를 찾기도 불편했습니다.
해결	여러 도구를 살펴보고, 간편하면서 여러 컨테이너 로그를 함께 보는 Dozzle을 도입했습니다. 트랜잭션별 로그를 한 화면에서 볼 수 있게 됐고, 팀원 모두가 같은 화면에서 로그를 확인했습니다.

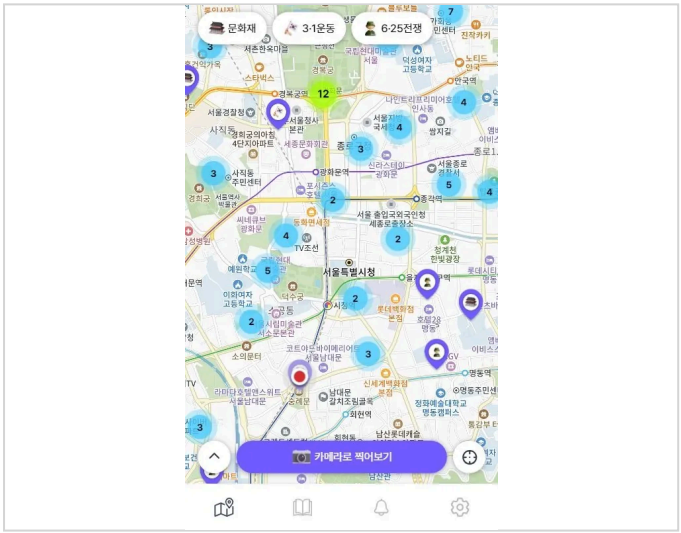
아키텍처



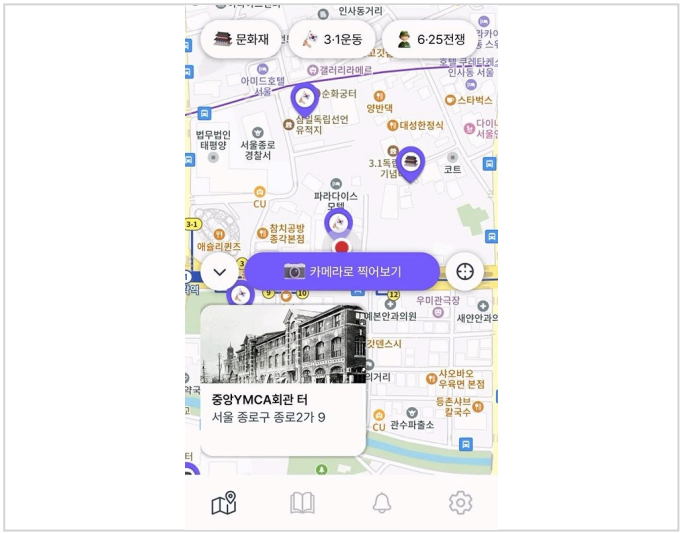
코드 푸시부터 배포까지의 흐름입니다.



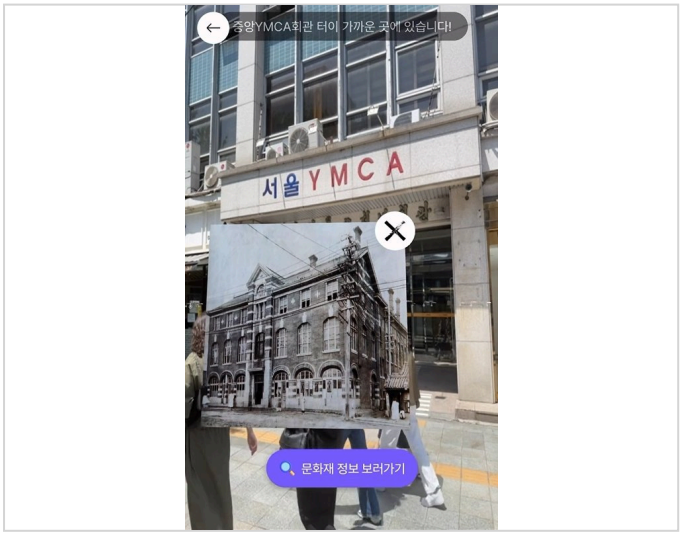
서비스 소개와 지도를 담은 서울팝 커버



지도에서 주변 사적지를 찾습니다.



지도에서 고른 사적지를 확인합니다.



과거 사진과 현재 풍경을 비교합니다.



사적지의 요약과 설명을 읽습니다.

SSAFY

리릭잇 · 노래 가사 기반 실시간 게임

기간	2024.02 — 2024.04
팀	6명 백엔드 2, 프론트엔드 2, 데이터 2
역할	백엔드 리드 CI/CD·인프라 총괄 회원 API WebSocket 통신 채팅 방 생성·게임 참여 API 게임 플로우 리팩터링 게임 타이머 스케줄링
기술	Java 17 · Spring Boot 3.2.1 · Spring Data JPA · WebSocket (STOMP) · MySQL 8.0.35 · Redis 7.2 · AWS EC2 · Jenkins · Docker
링크	github github.com/lyricit/lyricit-be

백엔드 리드로 실시간 채팅·게임 통신과 인프라 담당

라운드마다 나오는 가사 키워드를 보고 채팅으로 노래를 맞히는 실시간 게임입니다. 크롤링하고 분산 처리한 역대 인기 가요 가사 데이터를 씁니다. 백엔드 작업 대부분을 맡았고, 서버와 CI/CD는 혼자 구축했습니다. Jenkins로 배포 시나리오를 여러 가지 구성했고, QA를 이어 가며 사용성을 다듬었습니다.

기능

- 실시간 방 정보 갱신과 채팅
- 방 생성, 게임 준비·시작, 진행 방식 설정
- 채팅으로 가장 먼저 정답을 맞힌 사람이 이기는 라운드
- 가사 데이터 검색

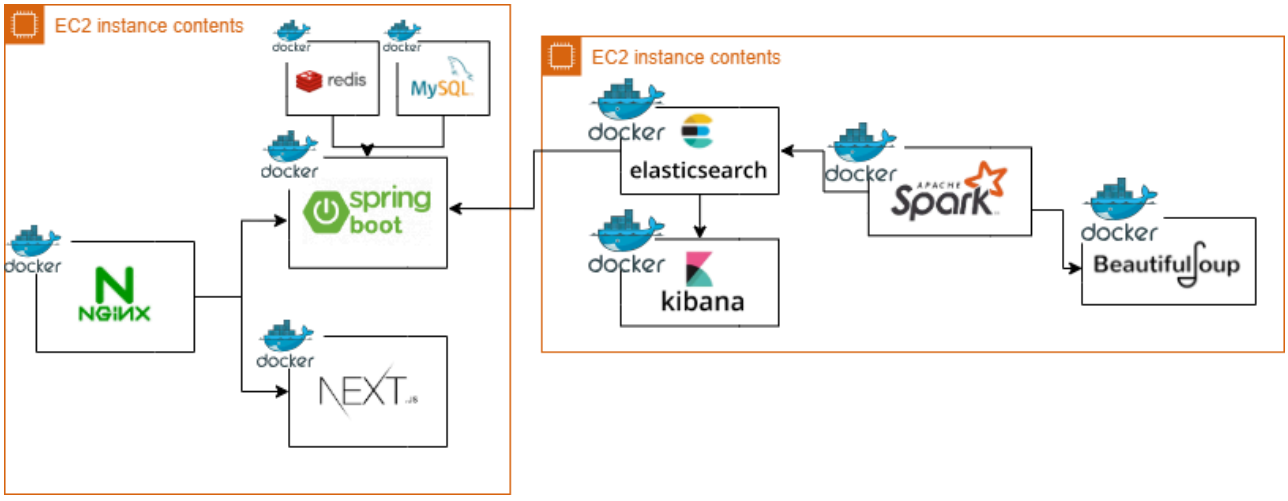
페이지별 실시간 메시지 채널

문제	게임의 메시지 채널이 많아 리소스 부담과 채널 관리, 개발 비용을 함께 줄여야 했습니다. 컴포넌트별 채널과 페이지별 채널 중 하나를 골라야 했고, 개발 비용을 줄이려면 직관적인 방식이어야 했습니다.
해결	페이지마다 메시지 채널을 하나로 합쳤습니다. 응답 형식은 통일하고 메시지 종류로 상황을 나눠, 채널 관리 복잡도를 줄였습니다. 리소스 부담을 줄였고, 기능은 메시지 종류를 더하는 방식으로 늘렸습니다.

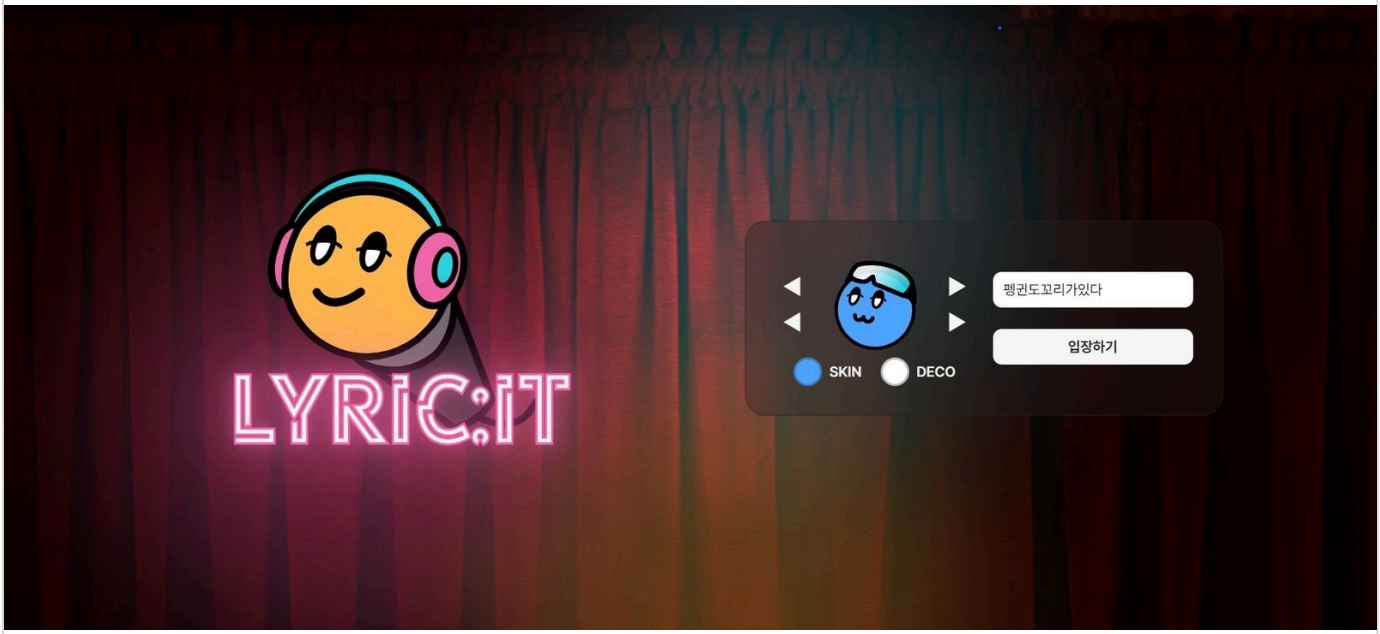
세션 끊김

문제	로그아웃이 아니라 소켓 세션 단절로 오프라인을 판단했고, Redis의 방·게임 상태도 함께 갱신해야 했습니다. 타임아웃으로 연결이 끊기는 문제도 있었습니다.
해결	인터셉터에서 STOMP 명령을 확인해 연결 상태별로 처리하고, 게임 중 한 명이 끊기면 그 게임과 방 데이터를 지우고 갱신했습니다. 채널 인터셉터는 이벤트 리스너로 처리를 넘겨 순환 참조를 피했습니다. heartbeat로 일정 간격마다 ping·pong 프레임을 주고받아 타임아웃 끊김을 막았습니다.

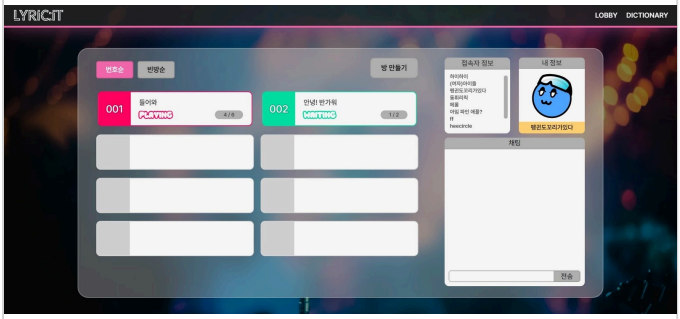
아키텍처



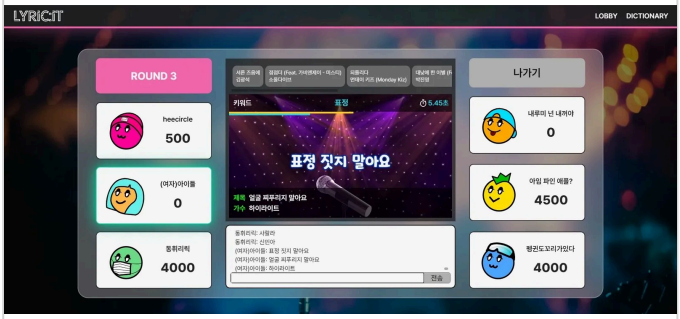
서비스 서버와 가사 검색·수집 구성을 나눈 구조입니다.



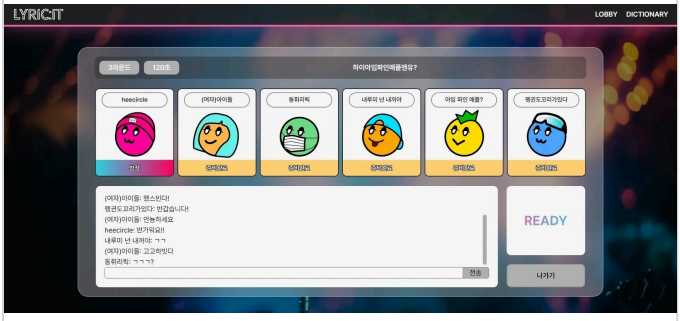
로고와 로그인 창이 보이는 리릭잇 시작 화면



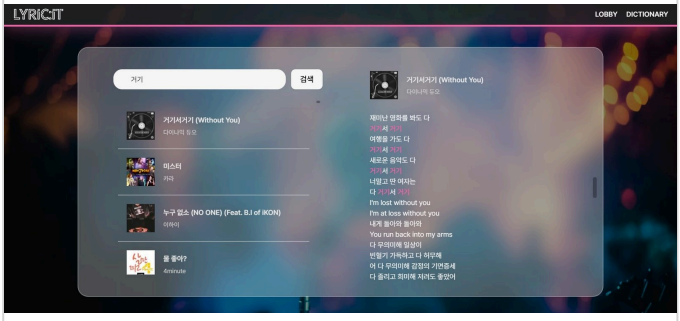
로비에서 참여할 게임 방을 찾습니다.



가사 키워드로 게임을 진행하고 순위를 확인합니다.



대기실에서 참가자와 게임 설정을 확인합니다.



노래를 검색하고 가사를 확인합니다.

SSAFY

명절 잔소리 영수증

기간	2024.01 — 2024.02
팀	6명 백엔드 3, 프론트엔드 3
역할	팀장　백엔드 리드　인프라·기술 총괄　잔소리·좋아요·팔로우·영수증 API　배치
기술	Java 17 · Spring Boot 3.2.1 · Spring Data JPA · Spring Security · Spring Batch · Querydsl · MySQL 8.0.35 · Redis 7.2 · AWS EC2 · GitLab Runner · Docker
링크	github github.com/jansorry/jansorry-be

설 연휴 동안 회원 약 200명

명절 잔소리로 쌓이는 젊은 세대의 스트레스와 세대 간 단절에서 출발했습니다. 잔소리와 대처 경험을 공유하고, 영수증과 통계로 만들어 나눕니다. 300명 이상이 참여한 사전 설문으로 명절 스트레스 데이터를 모았고, SSAFY 공통 프로젝트 1위로 우수상을 받았습니다. 백엔드 작업의 절반 이상을 맡았고, 서버와 CI/CD는 혼자 구축했습니다.

기능

- 잔소리·대처 경험 피드
- 좋아요·팔로우
- 영수증 출력과 공유
- 마이페이지
- 통계 데이터 시각화
- GitLab Runner와 Docker Hub를 거친 EC2 배포
- Nginx 뒤 Next.js·Spring Boot·MySQL·Redis 컨테이너 구성

Redis에서 팔로우 양방향 관계

문제　비관계형 저장소에서 팔로우 관계를 표현해야 했습니다. 단방향인 좋아요와 달리 팔로우는 N:M 관계라, 양방향 저장은 적재량이 늘고 단방향 저장은 조회가 느려집니다.

해결　와일드카드 탐색은 최대한 피해야 한다고 보고 양방향 저장을 택했습니다. 키에 following·follower 접두사를 붙여 한 사용자의 팔로잉과 팔로워를 각각 바로 읽게 했습니다. 팔로워·팔로잉 목록을 조회할 때 탐색 시간을 크게 줄였습니다.

배치 동기화의 전체 탐색

문제　매일 Redis 데이터를 MySQL로 옮기는 배치가 전체 데이터를 훑었습니다. 바뀐 데이터만 옮길 방법이 필요했습니다.

해결　변경된 데이터의 키를 Redis 정렬 집합에 넣고, 점수를 갱신 시각으로 두었습니다. 배치는 직전 배치 이후 바뀐 데이터만 MySQL에 반영하고, 끝나면 변경 목록을 비워 다음 변경만 담게 했습니다. 좋아요·팔로우 취소로 값이 빈 항목은 테이블에서도 지워 공간을 확보했습니다.

가격 상위 5개 영수증

문제　최종 통계에 가격이 가장 높은 영수증 5개가 필요했습니다. 모든 사용자의 모든 영수증에서 한 번에 찾으면 비용이 컸습니다.

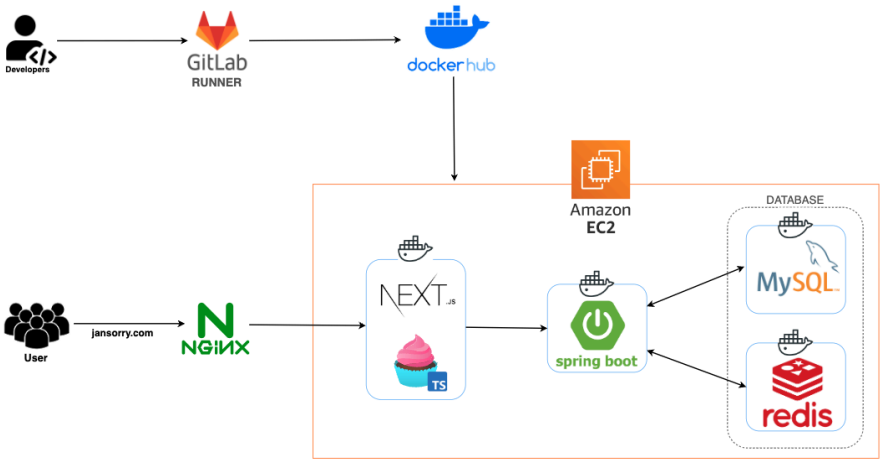
해결　배치 동기화에 쓴 정렬 집합 방식을 응용했습니다. 영수증을 저장할 때 가격을 점수로 상위 5개 집합에 넣고, 6위로 밀린 영수증은 바로 지웠습니다. 한 시점에 몰리던 탐색 부하를 저장 시점으로 나눠, 전체 영수증을 한꺼번에 검색하지 않게 했습니다.

한 저장소에서 프론트엔드·백엔드 배포

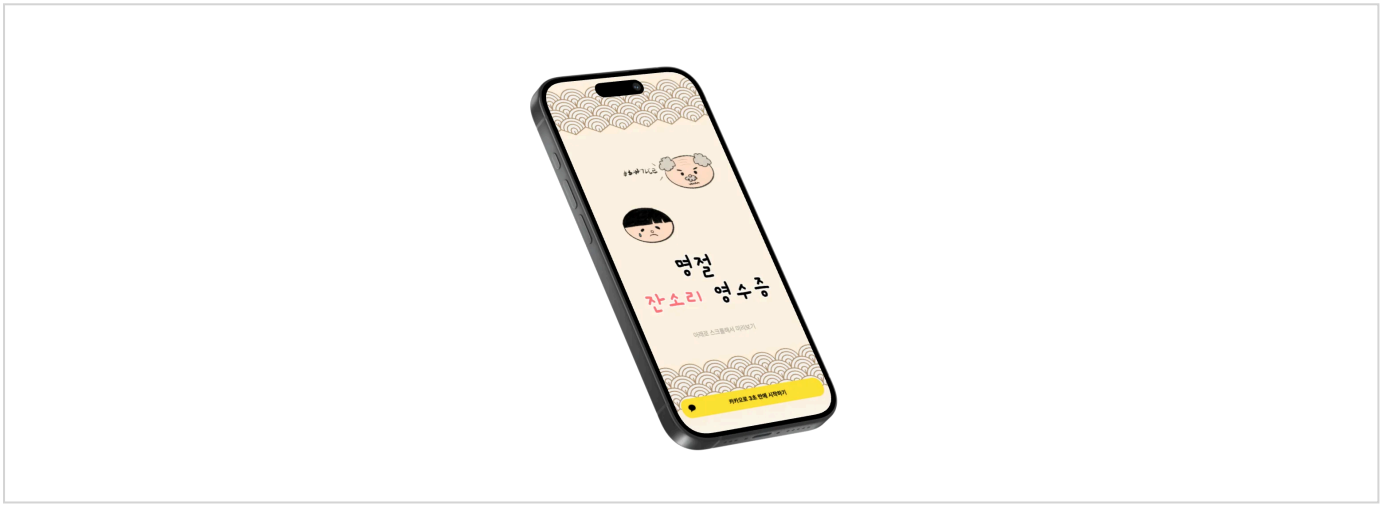
문제　프로젝트에서 저장소를 하나만 쓸 수 있어, 프론트엔드와 백엔드 코드를 한 저장소에 두고 배포를 자동화해야 했습니다. Jenkins는 초기 설정 비용이 크고 서버에서 차지하는 자원도 많았습니다.

해결　플러그인은 적지만 가볍게 파이프라인을 꾸릴 수 있는 GitLab Runner를 택했습니다. 파이프라인은 환경변수 파일 생성, 빌드, Docker 이미지 빌드·Docker Hub 업로드, 서버 실행의 4단계로 나눴습니다. DB 접속 정보 같은 비밀 값은 GitLab CI/CD 변수에 두고 배포 때 .env 파일로 만들었습니다.

아키텍처



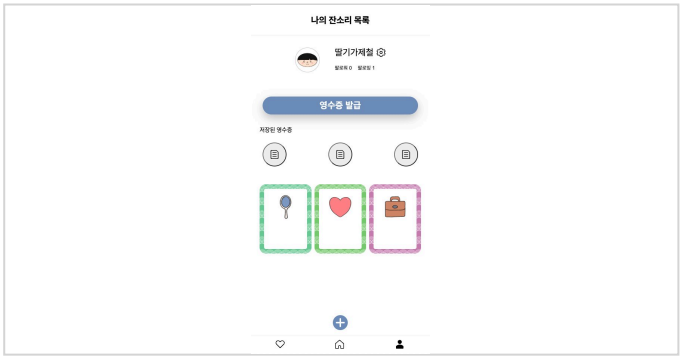
GitLab Runner로 이미지를 빌드해 EC2의 Docker 컨테이너로 배포하는 구성입니다.



명절 잔소리 영수증 시작 화면을 담은 스마트폰 목업



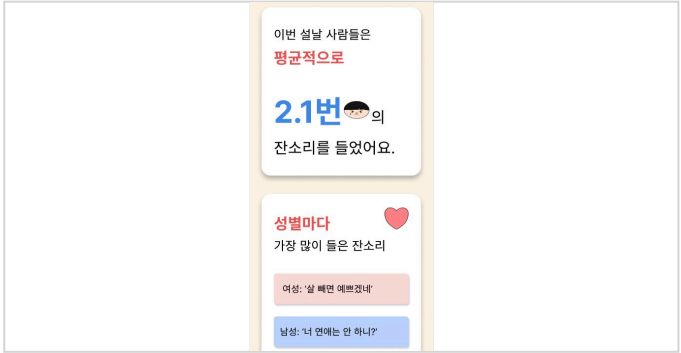
해시태그로 잔소리 목록을 골라 봅니다.



저장된 잔소리를 영수증으로 만듭니다.



잔소리 내역을 영수증 형태로 출력합니다.



잔소리 횟수와 성별 통계를 확인합니다.

SSAFY

뉴집스 · 부동산 모의투자

기간	2023.11
팀	2명 백엔드 1, 프론트엔드 1
역할	팀장 백엔드 단독 서버·CI/CD 모의투자·랭킹·회원·공지 뉴스 크롤러 스케줄링
기술	Java 8 · Spring Boot 2.7.16 · Spring Data JPA · Spring Security · MariaDB · Redis · AWS S3 · AWS EC2 · Cloudtype · Docker · GitHub Actions
링크	github github.com/NewJibs/newjibs-be

거래 데이터 400만 건을 약 6만 건으로 축소

가상 예산으로 부동산에 투자하고 2년 뒤 수익률을 확인하는 모의투자 서비스입니다. 코로나 시기 가파르게 오른 부동산 가격에서 출발했습니다. 백엔드와 서버·CI/CD를 혼자 맡았습니다.

기능

- 카카오 지도의 위·경도 클러스터 렌더링
- 부동산 정보 조회
- 랭킹
- 공지사항과 부동산 뉴스
- JWT 기반 회원 인증
- Vue.js 프론트엔드의 Netlify 배포
- GitHub Actions로 Cloudtype·EC2에 올린 Spring Boot·MariaDB·Redis 컨테이너

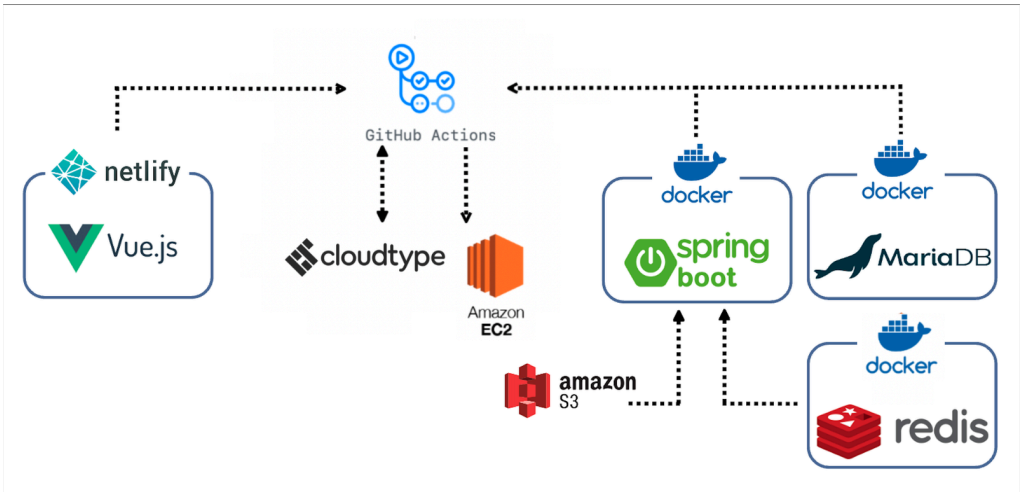
거래 데이터 축소

문제	2016년 이후 매매 데이터가 400만 건이 넘어 렌더링이 느려졌고, 모의투자에는 전체가 필요하지 않았습니다. 코로나19가 시작된 2020년 이후 가격이 크게 오른 점을 기준으로 데이터를 줄이기로 했습니다.
해결	2020년과 2022년 거래 중 층수와 평수가 같은 가구를 1:1로 매핑해 같은 집의 가격 변화만 남겼습니다. 같은 가구의 매매 데이터를 써야 투자 결과를 더 믿을 수 있다고 봤습니다. 400만 건이 넘던 거래 데이터가 약 6만 건으로 줄었고, 렌더링도 빨라졌습니다.

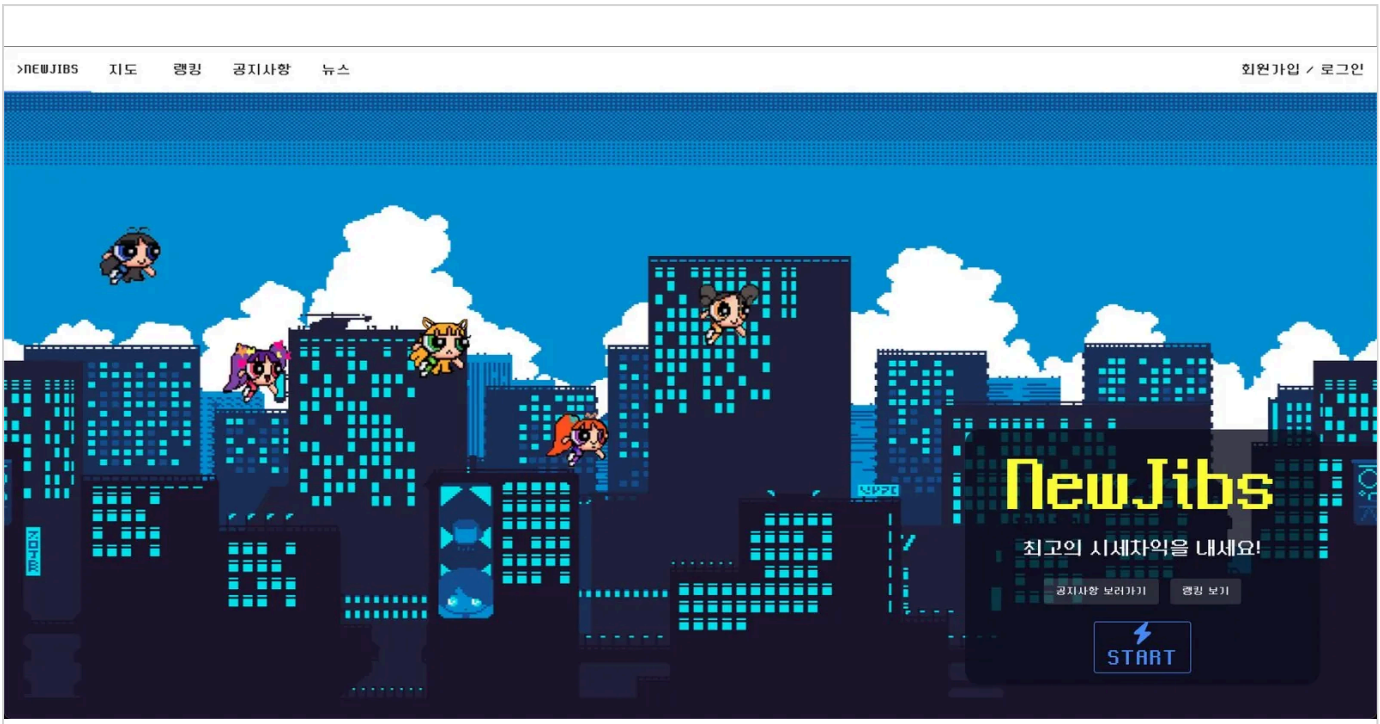
뉴스 크롤링 차단

문제	jsoup으로 매 정각 부동산 뉴스를 수집하자 bot으로 인식돼 서버 IP가 매번 차단됐습니다. Selenium을 도입하기보다 크롤러 안에서 해결하려 했습니다.
해결	정각마다 호출하는 시점과 요청 헤더 때문에 bot으로 인식된다는 점을 찾았습니다. 크롤러에서 요청 헤더를 바꾸고 호출 시점에 랜덤 지연을 넣었습니다. 호출 패턴이 덜 드러나 차단을 피할 수 있었습니다.

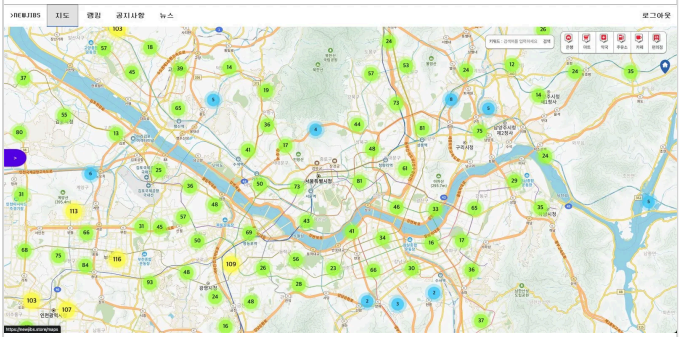
아키텍처



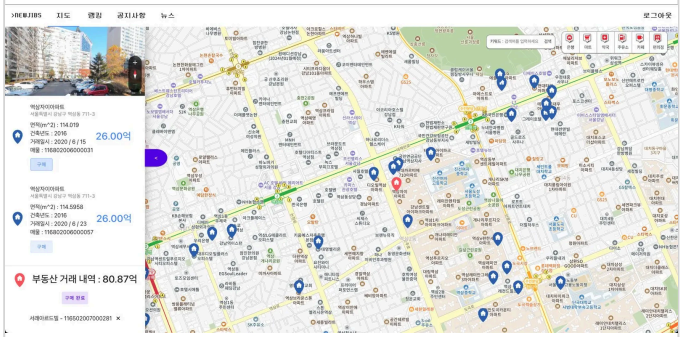
GitHub Actions를 중심으로 Netlify, Cloudtype, EC2에 배포하는 구성입니다.



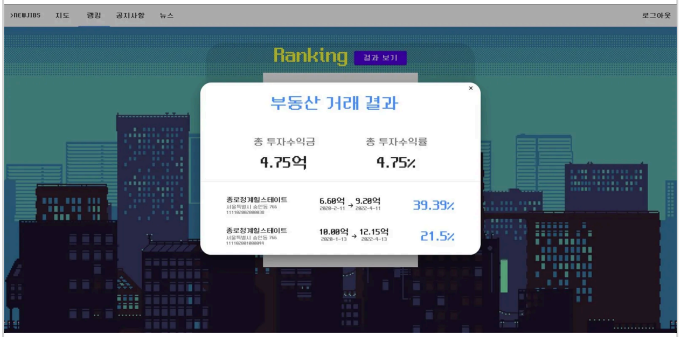
도시 일러스트를 배경으로 한 뉴집스 시작 화면



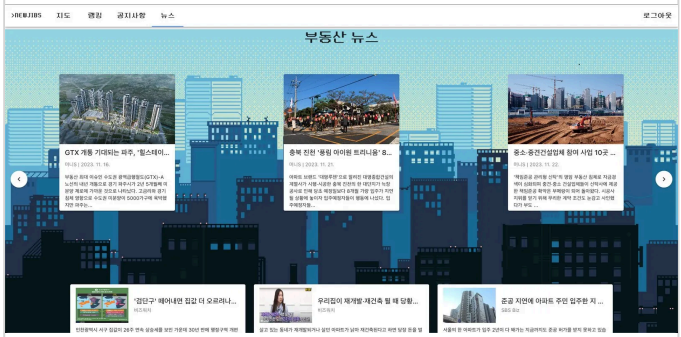
지도 위에 매물 위치를 모아 보여 줍니다.



지도에서 선택한 매물의 거래 내역을 확인합니다.



모의투자의 수익 결과를 확인합니다.



부동산 뉴스를 카드 목록으로 보여 줍니다.

SSAFY

트리플랫 · 여행 금융 다이어리

기간	2023.09
팀	4명 백엔드 3, 프론트엔드 1
역할	백엔드 API(회원, 여행, 데일리, 이미지 업로드) 서버·CI/CD
기술	Java 11 · Spring Boot 2.7.10 · Spring Data JPA · MySQL 8.0 · AWS S3 · AWS EC2 · Docker · GitHub Actions
링크	github github.com/Triplet-Shinhan/Triplet

8:1 경쟁을 거쳐 신한은행 해커톤 진출

환전 현금과 카드 외화 지출을 함께 기록하고, 환율을 반영해 원화로 자동 환산하는 여행 금융 다이어리입니다. SSAFY 과정 중 참가한 신한은행 해커톤에서 만들었습니다. JPA를 처음 쓰고 처음 배포까지 해 본 프로젝트입니다. 서버와 CI/CD 구축 작업의 절반을 맡았습니다.

기능

- 여행 생성과 예산·보유 현금 설정
- 환전 신청
- 달력형 금융 다이어리
- 이미지 업로드

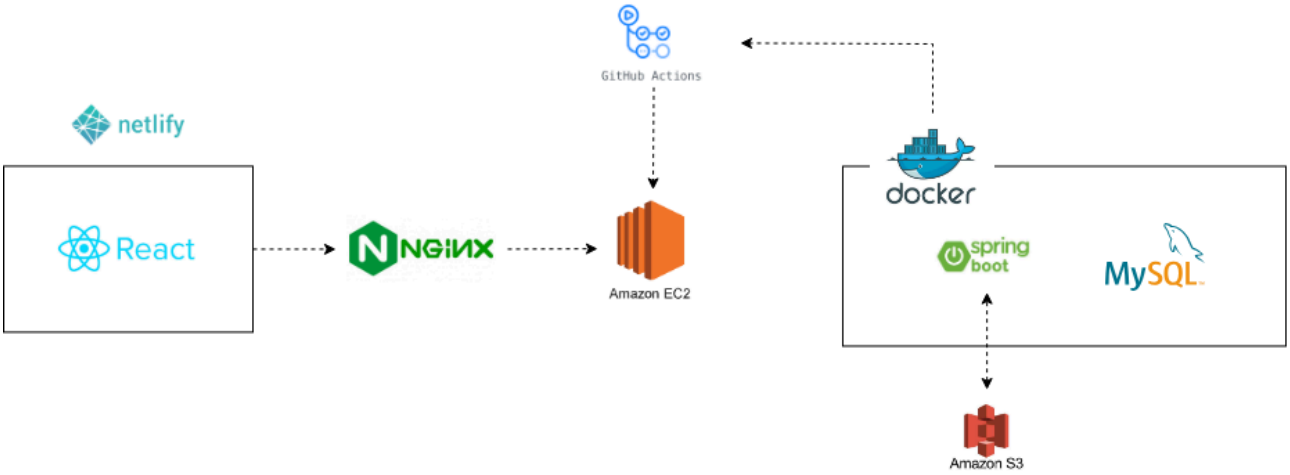
현실과 다른 API

문제	실제 은행 API는 보안과 기술 문제로 제공되지 않아, 해커톤에서는 더미 데이터 API만 쓸 수 있었습니다. 거래 로직과 환율 정보를 직접 파악해 서비스를 만들어야 했습니다.
해결	지출과 환율 정보 등 여러 상황을 가정해 추가 API와 더미 데이터를 만들었습니다. 이 데이터로 사용자 시나리오별 상황을 재현하며 기능을 만들었습니다.

인스턴스 자원 부족

문제	비용 없이 운영하려 AWS 프리티어 인스턴스 하나에 모든 컨테이너를 올렸습니다. 사용량이 계속 치솟다가 인스턴스가 종료됐습니다.
해결	유료 사양으로 올리는 방법도 검토했지만, RAM 1GB 인스턴스에 메모리 스왑을 설정해 해결했습니다. 첫 배포라 원인을 찾기까지 오래 걸렸습니다.

아키텍처



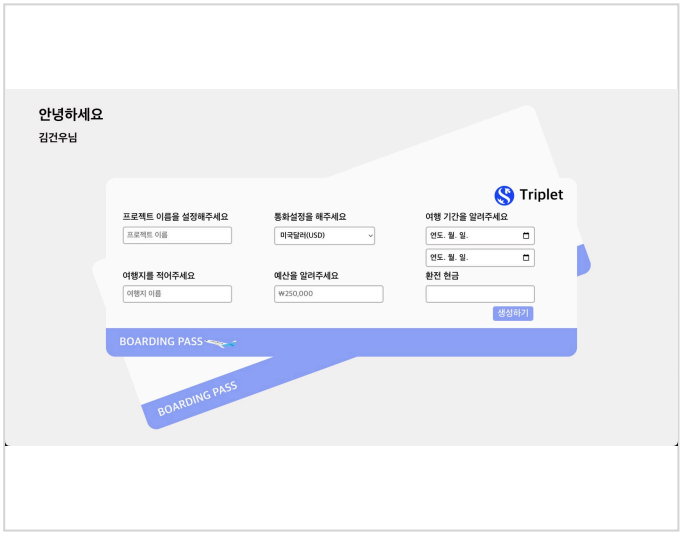
GitHub Actions로 EC2에 배포하는 구성입니다.



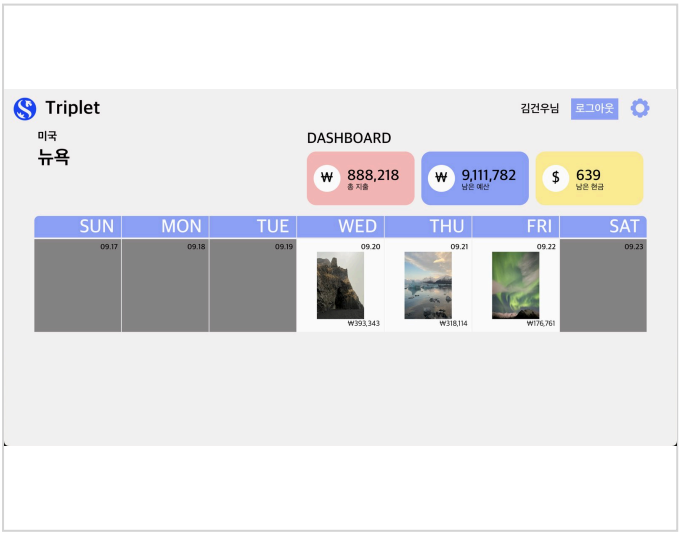
비행기 일러스트와 로그인 창이 있는 트리플랫 시작 화면



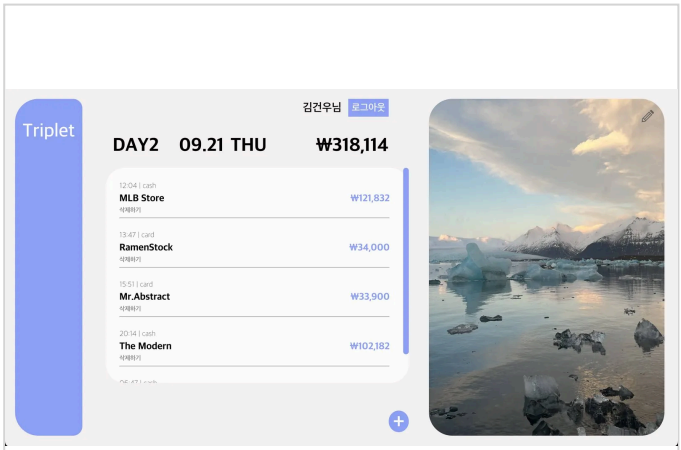
여행별 프로젝트를 목록으로 관리합니다.



여행지와 일정으로 새 프로젝트를 만듭니다.



달력에서 일별 지출을 확인합니다.



날짜별로 지출과 여행 사진을 함께 봅니다.

도구와 오픈소스

herdr-cockpit · orch 작업을 보는 Herdr 레이아웃과 모드

기간	2026.11 — 현재
팀	1인
역할	설계 Herdr 모드 개발 Go 수집 프로그램
기술	Go · JavaScript · Swift · Herdr.app · gopsutil
링크	github github.com/takealook97/herdr-cockpit

계정·머신별 세션과 사용량을 한 화면에서 확인

orch로 여러 에이전트를 돌릴 때 진행 상황이 잘 보이도록 만든 Herdr 레이아웃과 모드입니다. 지휘하는 메인 세션, 계정과 머신에 흩어진 작업 에이전트 세션, 제공자별 남은 사용량을 한 화면에서 봅니다. orch 없이 일반 CLI 세션에도 쓸 수 있습니다.

기능

- Codex·Claude·Gemini 남은 사용량과 초기화 시각, CPU·RAM을 보여 주는 상단 막대
- 머신·스페이스·탭·창 트리와 메인 세션별 작업 에이전트 수를 보여 주는 에이전트 목록
- 이 컴퓨터의 메인 세션 터미널을 격자로 모아 바로 입력하는 hub
- 말이나 글로 한 지시를 알맞은 메인 세션에 전달하는 nexus와 한국어 음성 입력
- 연결이 끊긴 원격 머신의 탭 숨기기와 복구
- 원격 머신마다 사용량과 창 정보를 모으는 Go 수집 프로그램

제공자마다 다른 로그인과 호출 제한

문제 Codex, Claude, Gemini의 사용량을 읽으려면 제공자마다 다른 로그인과 호출 제한을 다뤄야 했습니다.

해결 인증, 요청, 캐시, 재시도 간격을 머신마다 도는 Go 수집 프로그램에 모았습니다. 화면은 JSON만 받아 표시하고 인증 정보를 다루지 않습니다.

머신을 바꾸는 사이 늦게 온 응답

문제 머신을 바꾸는 동안 이전 머신의 응답이 늦게 도착하면 지금 보는 머신의 사용량을 덮어쓸 수 있었습니다.

해결 응답을 머신별로 따로 보관하고, 지금 고른 머신이 아닌 응답은 화면에 반영하지 않게 했습니다. 머신을 빠르게 바꿔도 다른 머신의 사용량이 섞여 보이지 않습니다.

엉뚱한 세션으로 갈 수 있던 지시

문제 음성이나 짧은 글로 지시하면 여러 메인 세션 중 어디로 보낼지 모호할 수 있었습니다.

해결 모델이 이 컴퓨터의 메인 세션 목록만 보고 대상을 고릅니다. 확신도가 0.6보다 낮으면 후보를 보여 주며 묻고, 실패하거나 20초를 넘기면 아무것도 보내지 않습니다.

겹쳐 들어온 화면 배치 변경

문제 여러 메인 세션을 한 격자에 모아 보는 화면에서, 세션 목록과 배치 변경이 비동기로 겹칠 수 있었습니다. 늦게 끝난 변경이 이미 지난 화면을 다시 띄울 수 있었습니다.

해결 레이아웃 쓰기를 한 줄로 세워 차례로 처리하고, 열린 화면 상태를 한곳에서 공유하게 했습니다. 세션 목록이 바뀌면 500ms를 기다렸다가 한 번에 반영합니다. 늦게 도착한 쓰기가 지난 화면을 다시 고르지 않습니다.

도구와 오픈소스

wiki-island · 벤더 중립 AI 메모리

기간	2026.11 — 현재
팀	1인
역할	설계　Rust 개발
기술	Rust · SQLite FTS5 · MCP · Tokio · Git · Markdown
링크	github github.com/takealook97/wiki-island

모델 호출 없이 MCP 도구 15개로 개발 도구 10곳에 연결

AI 도구마다 기억을 따로 쌓지 않고, 특정 회사에 묶이지 않는 기억 하나를 여러 도구가 함께 쓰도록 만든 오픈소스입니다. 기억은 내 컴퓨터의 Markdown 폴더에 저장됩니다. 저장·검색·읽기에는 모델도 네트워크도 쓰지 않습니다.

기능

- MCP 도구 15개로 하는 검색·읽기·기록·수정·보관·복원
- wi setup 한 번으로 Claude Code·Codex·Cursor 등 AI 앱 10종 자동 연결
- 띄어쓰기 없는 두 글자 한국어도 찾는 SQLite FTS5 검색
- Git으로 남기는 변경 이력과 이전 버전 복원
- Obsidian 폴더와 함께 쓰기, PDF·Word·PowerPoint·CSV·이미지 가져오기
- 백그라운드 서비스가 맡는 편집 반영, 색인 복구와 정리

망가질 수 있는 색인

문제　검색 색인을 원본처럼 다루면 색인이 손상되거나 기기 사이에서 동기화될 때 기억을 잃을 수 있습니다.

해결　Markdown 파일을 원본으로 두고, SQLite 색인은 기기마다 따로 두는 캐시로 분리했습니다. 색인이 없거나 손상되면 파일을 직접 훑어 찾고, 색인은 다시 만듭니다. 색인을 지워도 기억은 그대로 남습니다.

저장 중에 끊기거나 겹친 편집

문제　여러 AI 도구가 같은 노트를 동시에 고치거나 저장 중에 프로세스가 끊길 수 있었습니다. 그러면 오래된 내용으로 덮어쓰거나 저장한 내용을 잃을 수 있었습니다.

해결　쓰기마다 호출자가 기대한 체크섬을 확인해 오래된 쓰기는 거부합니다. 임시 파일을 디스크에 쓴 뒤 시작 기록을 남기고 파일을 바꿔 넣고, 끝 기록을 남깁니다. 중간에 끊긴 작업은 이 기록으로 복구합니다. 파일을 원자적으로 바꿔 넣을 수 있는 플랫폼에서는 동시에 들어온 편집을 충돌 사본으로 남깁니다.

두 글자 한국어 검색

문제　띄어쓰기로 단어를 나누는 기본 검색으로는 문장 속 '결제', '환불' 같은 두 글자 한국어를 찾기 어려웠습니다.

해결　기본 토크나이저는 그대로 두고, 띄어쓰기가 없는 문자를 두 글자씩 끊어 담는 색인 열을 더했습니다. 여러 낱말 검색은 모든 낱말이 맞는 결과를 먼저 보이고, 남은 자리를 순위 매긴 부분 일치로 채웁니다. 결제, 환불 같은 두 글자 검색과 한영이 섞인 검색으로도 의도한 노트를 찾습니다.

동기화 폴더 속 Git 이력

문제　iCloud Drive 같은 동기화 서비스는 파일을 하나씩 보냅니다. 폴더 안의 Git 이력이 일부만 또는 순서가 뒤바뀐 채 전달될 수 있습니다.

해결　동기화 폴더 안에서는 Git 이력을 기기별 외부 폴더에 두고, 노트 폴더에는 Markdown만 남깁니다.

도구와 오픈소스

orch · 멀티 에이전트 오케스트레이터

기간	2026.09 — 현재
팀	1인
역할	설계 Go CLI 개발
기술	Go · YAML · Herdr · Claude Code · Codex · Antigravity
링크	github github.com/takealook97/orch

작성한 모델과 다른 모델이 검증해야 끝나는 작업 흐름

에이전트 세션 하나에서 모든 일을 처리하지 않고, 판단만 하는 메인 세션을 따로 두려고 만들었습니다. 메인 세션은 계획을 세우고 작업을 배정하고 결과를 판단합니다. 실행은 계약 파일을 받은 작업 에이전트가 맡고, 결과는 작성한 모델과 다른 계열의 모델이 검토합니다.

기능

- Codex·Claude·Gemini 계열 7개 모델 키 중에서 고르는 작업 배정
- 작업 범위와 결과 형식을 적은 계약 파일, 결과 파일과 상태 기록
- 창을 열지 않고 끝내는 단발 조사·수정, 실패만 요약하는 테스트 실행
- 작성한 모델과 다른 계열에 맡기는 독립 검토
- 의존 관계에 따른 순차·병렬 실행과 중단 뒤 이어서 실행
- Herdr 작업 창의 배치와 정리, 사용자 화면 포커스 보호

지침만으로는 지켜지지 않던 역할

문제 규칙을 지침 문서에만 두면 자체 위임 기능을 쓰는 명령에서는 지침이 적용되지 않았습니다. 작업 에이전트도 같은 지침을 읽고 다시 위임할 수 있었습니다.

해결 규칙을 공통 지침, Go 실행기, 우회 경로를 막는 hook, 작업 유형별 YAML 절차의 네 층으로 나눴습니다. 작업 에이전트는 작업자 표시를 달고 실행되므로 다시 위임하지 못합니다. 모델 이름이나 정리 절차 같은 규칙은 모델이 매번 다시 적지 않아도 실행기가 지킵니다.

작업 중 옮겨 가던 화면

문제 작업 창을 조작할 때 사용자가 보던 화면이 작업 창으로 옮겨 갔습니다. 서버 로그에서 확대 해제 요청 때 55건, 입력 전달 때 78건의 포커스 이동을 확인했습니다.

해결 실제로 확대된 창에만 해제를 요청하고, 입력은 한 줄 전송과 Enter 키 입력으로 나눠 보내도록 바꿨습니다. 수정 뒤 측정에서 orch 요청으로 생긴 포커스 이동은 0건이었습니다.

작은 작업에는 느렸던 병렬 실행

문제 작은 구현 작업을 창 3개로 나눈 평가에서 병렬 실행이 평균 84초로, 단일 실행 49초보다 오래 걸렸습니다.

해결 짧은 단발 작업은 창 없이 바로 실행하는 ask·do로 넘기고, 테스트는 모델 없이 실패 요약만 돌려받게 했습니다. 위임 기준도 걸리는 시간에서 메인 세션이 쓰는 토큰으로 바꿨습니다.

도구와 오픈소스

hermes-agent · Hermes Agent 오픈소스 컨트리뷰션

기간	2026.08 — 현재
팀	컨트리뷰션
역할	버그 수정　테스트
기술	Python · ripgrep · macOS
링크	hermes-agent github.com/NousResearch/hermes-agent/commit/5fd6811dfe6fb18c81a510da354bf5356eb2449b

홈 폴더 검색 중 뜨던 macOS 권한 팝업을 막은 수정

Hermes Agent는 Nous Research의 오픈소스 AI 에이전트입니다. 에이전트가 홈 폴더처럼 넓은 경로를 검색하면, 지켜보는 사람이 없는 실행 중에도 macOS 개인정보 권한 팝업이 떴습니다. 검색을 시작하기 전에 보호 폴더를 빼는 수정을 컨트리뷰션했습니다.

기능

- 홈이나 /Users 같은 상위 경로 검색에서 Desktop·Documents·Downloads 등 보호 폴더 7곳 제외
- 파일명 검색과 파일 내용 검색, ripgrep·grep·find 경로 모두에 같은 제외 적용
- 보호 폴더를 직접 지정한 검색은 그대로 허용
- 건너뛴 폴더와 직접 검색하는 방법을 알리는 경고
- macOS가 아닌 환경에서는 적용하지 않음

결과를 거르기 전에 뜨는 팝업

문제　검색 결과에서 보호 폴더를 걸러 내도, 폴더에 들어가는 순간 이미 권한 요청이 발생했습니다.

해결　ripgrep 제외 패턴과 find의 -prune으로 탐색을 시작하기 전에 보호 폴더를 뺐습니다. 원본 PR에 관련 테스트 61개의 통과 결과와 실제 macOS에서 넓은 경로를 검색한 결과를 함께 올렸습니다.

말없이 줄어든 검색 결과

문제　폴더를 빼면 사용자는 결과가 일부 빠졌다는 사실을 알 수 없었습니다.

해결　생략한 폴더와 직접 검색하는 방법을 경고로 알리고, 보호 폴더를 직접 지정한 검색에서는 제외하지 않습니다. 사용자는 무엇이 빠졌는지 알고, 필요한 보호 폴더만 직접 지정해 검색할 수 있습니다.

이름만 같은 폴더와 원격 실행 환경

문제　처음 구현은 폴더 이름으로 제외해, 같은 이름의 평범한 하위 폴더까지 빠질 수 있었습니다. 원격 실행 환경에서도 macOS 보호 폴더 규칙을 적용할 수 있었습니다.

해결　통합 PR의 후속 수정에서 이름 대신 경로를 기준으로 탐색을 막았습니다. 실행 환경을 판별해 원격 환경에서는 제외하지 않게 했습니다. 같은 이름의 일반 폴더와 원격 환경의 검색 결과가 불필요하게 빠지지 않습니다.

도구와 오픈소스

vat · 상위 폴더 하나에서 여러 저장소를 개발하는 작업 공간 CLI

기간	2026.08 — 현재
팀	1인
역할	설계 Go CLI 개발
기술	Go · YAML · Git
링크	github github.com/takealook97/vat

다음 작업이 이어받도록 출처와 함께 남기는 변경과 결정

여러 저장소를 상위 폴더 하나에서 한꺼번에 개발하려고 만들었습니다. Git이 저장소마다 이력을 남긴다면, vat는 저장소 사이에서 일어난 변경을 한곳에 기록합니다. 개발하는 동안 에이전트가 읽을 결정과 기록도 출처와 확인 날짜를 붙여 관리합니다.

기능

- 상위 폴더의 vat.yaml 하나로 선언하는 저장소 목록과 저장소별 수정 범위
- 로컬 변경을 건드리지 않고 안전할 때만 fast-forward하는 일괄 동기화
- 여러 저장소에 걸친 변경의 복귀 지점, 검사 결과, 함께 검증된 커밋 기록
- 출처 커밋과 확인 날짜가 붙은 결정 기록, 오래된 기록의 인용 중지
- 역할 정의 하나에서 만드는 Claude Code·Codex용 에이전트 지침과 불일치 검사
- 설정과 환경 문제를 진단하는 lint와 doctor

함께 검증된 커밋 조합

문제 API 변경이 여러 저장소에 걸치면 저장소별 git log만으로는 어떤 커밋들이 함께 검증됐는지 알 수 없었습니다. 문제가 생겼을 때 돌아갈 지점도 따로 남지 않았습니다.

해결 변경을 시작할 때 저장소마다 돌아갈 커밋을 기록하고, 검사 결과를 검증한 커밋에 연결했습니다. 배포 브랜치에 포함됐는지는 Git 조상 관계로 확인합니다. 함께 검증된 구성과 복귀 지점을 변경 하나의 기록에서 찾을 수 있습니다.

성공으로 보고되던 일괄 pull

문제 저장소마다 git pull을 반복하면 한 저장소의 작업이 stash돼도 전체는 성공으로 보고됐습니다.

해결 동기화 전에 저장소 상태를 먼저 판단해 안전한 fast-forward만 실행합니다. 로컬 변경, 브랜치 이탈, 원격 불일치, 분기는 저장소별로 따로 보고합니다.

확인한 지 오래된 사실

문제 문서에 적힌 현재 상태는 아무도 다시 확인하지 않아도 사람과 에이전트가 계속 사실로 인용했습니다.

해결 기록마다 출처 저장소, 커밋, 확인 날짜를 남깁니다. 기간이 지나도록 다시 확인하지 않은 기록은 근거로 인용할 수 없는 상태로 낮춥니다.

도구와 오픈소스

offon · 출퇴근·휴가를 관리하는 오픈소스 슬랙 앱

기간	2026.04 — 2026.08
팀	1인
역할	설계　풀스택 개발
기술	Next.js 16 · React 19 · TypeScript · Tailwind CSS 4 · shadcn/ui · PostgreSQL · Prisma · Slack Web API · Vercel Cron
링크	github github.com/takealook97/offon

사내 업무 시스템에서 떼어 낸, MIT 라이선스의 직접 설치형 앱

루쿠쿠에 합류했을 때 근태 관리가 제대로 되지 않아, 퇴근 후 시간에 만든 시스템에서 출발했습니다. 이 시스템은 이후 회사 온톨로지까지 붙은 전사 업무 시스템으로 커졌습니다. 그중 근태·휴가 기능만 떼어 오픈소스 슬랙 앱 offon을 만들었습니다.

기능

- 자정을 넘긴 근무와 하루 여러 번의 출퇴근, 근무·휴식·초과근무 시간 자동 계산
- 시작할 때 끝나는 시각을 정하고 복귀 알림을 보내는 60분 식사
- 주말·공휴일을 뺀 종일·반차 휴가 신청과 남은 연차 추적
- 달력에서 하는 근태 정정 신청과 승인 뒤 시간 재계산
- 휴가와 근태 정정을 한곳에서 처리하는 관리자 승인함
- 슬랙 일회용 코드 로그인, 슬래시 명령 5개, 출퇴근 누락 알림

동시에 들어온 출근 요청

문제　슬랙 재전송이나 두 번 클릭으로 출근 요청이 거의 동시에 오면, 두 요청 모두 사전 검사를 통과해 열린 근무가 두 개 생길 수 있었습니다.

해결　PostgreSQL 부분 유니크 인덱스로 근태 하나에 열린 근무를 하나만 허용했습니다. 두 번째 요청은 사전 검사와 같은 '이미 근무 중' 응답을 받습니다.

정정 신청 뒤에 생긴 기록

문제　정정을 신청한 뒤 실제로 퇴근하거나 휴식을 더 하면, 승인할 때 그 기록을 덮어쓸 수 있었습니다.

해결　신청 당시 값, 신청한 값, 승인 시점 값을 비교합니다. 신청에서 건드리지 않은 값은 이후 기록을 그대로 두고, 같은 항목이 양쪽에서 다르게 바뀌면 충돌로 처리합니다.

서버와 로컬의 시간대 차이

문제　UTC 서버와 한국 시간 로컬에서 날짜를 그대로 다루면 9시간이 두 번 더해지거나 요일이 틀릴 수 있었습니다.

해결　시각은 한국 시간으로 해석해 UTC로 저장합니다. 한국 시간으로 표시하는 함수와 날짜로 요일을 구하는 계산을 나눴습니다. 실행 환경이 달라도 같은 시각과 요일을 표시합니다.

슬랙의 재전송

문제　슬랙은 명령 응답이 늦으면 같은 요청을 다시 보냅니다. 그대로 처리하면 같은 기록과 알림이 두 번 생길 수 있었습니다.

해결　요청의 재전송 표시를 먼저 확인하고, 재전송이면 처리하지 않고 바로 200으로 응답합니다. 응답이 늦어도 같은 알림이 두 번 가지 않습니다.

도구와 오픈소스

sshift · 점프 서버 접속과 비밀번호 저장을 묶은 SSH CLI

기간	2025.07 — 2025.08
팀	1인
역할	설계 Go CLI 개발 Homebrew 배포
기술	Go · OpenSSH · sshpass · Homebrew
링크	github github.com/takealook97/sshift

서버 50여 대의 점프 서버 접속을 명령 하나로

비즈톡에서 서버 50여 대에 SSH로 접속해 패치 작업을 했습니다. 매번 점프 서버를 거쳐 접속하는 일이 번거로워, 점프 서버 연결과 암호화한 비밀번호 저장을 CLI 하나로 묶었습니다.

기능

- 대화형 서버 선택, ID·이름으로 바로 접속, 원격 명령 실행
- 서버와 점프 관계의 추가·수정·삭제·목록
- 점프 서버와 대상 서버의 비밀번호·키 인증 조합 처리
- 암호화한 비밀번호 저장과 사용자 지정 암호화 키
- 서버와 점프 관계의 JSON 내보내기·가져오기
- Homebrew, 설치 스크립트, 릴리스 바이너리로 설치

서로를 가리키는 점프 관계

문제 점프 관계를 자유롭게 추가하면 A가 B를 거치고 B가 다시 A를 거치는 순환이 생길 수 있었습니다.

해결 관계를 추가하기 전에 깊이 우선 탐색으로 대상에서 출발 서버로 돌아오는 경로가 있는지 검사하고, 있으면 추가를 거부합니다. 저장된 점프 관계에는 순환이 남지 않습니다.

바뀐 점프 관계 저장 형식

문제 점프 관계를 그래프로 저장하도록 형식을 바꾸자, 이전 버전에서 만든 서버 목록을 읽지 못할 수 있었습니다.

해결 새 그래프 형식으로 읽지 못하면 기존 관계 배열을 읽어 그래프로 바꿉니다. 사용자가 서버 목록을 다시 만들 필요가 없게 했습니다.

서버마다 다른 인증 방식

문제 비즈톡에서 서버 50여 대에 패치 작업을 하려면 매번 점프 서버를 거쳐 접속해야 했습니다. 점프 서버는 비밀번호로, 대상 서버는 키로 접속하는 식으로 인증 방식도 서로 달랐습니다.

해결 비밀번호와 키 조합에 따라 OpenSSH의 ProxyJump, ProxyCommand와 sshpass를 골라 명령을 만듭니다.

평문으로 남던 비밀번호

문제 서버 목록에 비밀번호를 그대로 저장하면 파일만 열어도 보였습니다.

해결 비밀번호는 무작위 IV를 쓰는 AES-CTR로 암호화하고, 설정 파일은 소유자만 읽도록 권한을 제한했습니다. 서버 목록 파일을 열어도 비밀번호는 암호문으로만 보입니다.