2026.

Back-End · AI Agent Developer.

문제를 구조로 만들고 반복되는 구조를 에이전트로 자동화하며
백엔드·인프라까지 운영하는 개발자 김재환입니다.

cbkjh0225@gmail.com · github.com/Jaeboong

#에이전트 오케스트레이션 #백엔드 #인프라 운영

2026.

에이전트부터 인프라까지.

서비스의 필요성과 구현 가능성을 함께 따지며,
멀티에이전트 오케스트레이션과 LLM기반 서비스를 설계하고
백엔드·인프라까지 운영하는 개발자입니다.

AGENT ORCHESTRATION.

- 여러 AI 에이전트의 역할을 나누고 협업을 설계·운영

BACKEND.

- API·데이터 모델·트랜잭션 정합성을 우선해 서버를 설계

INFRA OPS.

- 컨테이너 기반으로 배포하고 운영·개선

CORE SKLLS.

#에이전트 오케스트레이션 #백엔드 #인프라 운영

Back-End Developer / 2026

Contents.

#멀티에이전트 #자소서 검토 오케스트레이터 # 개인 프로젝트 #1인 개발

자소전.

멀티 AI 에이전트를 활용한 자기소개서 첨삭 서비스.

PROBLEM.

1. 도구 분산
   자소서·회사·공고 따로
2. 단일 LLM의 한계
   경험·사실·요건 동시 미흡

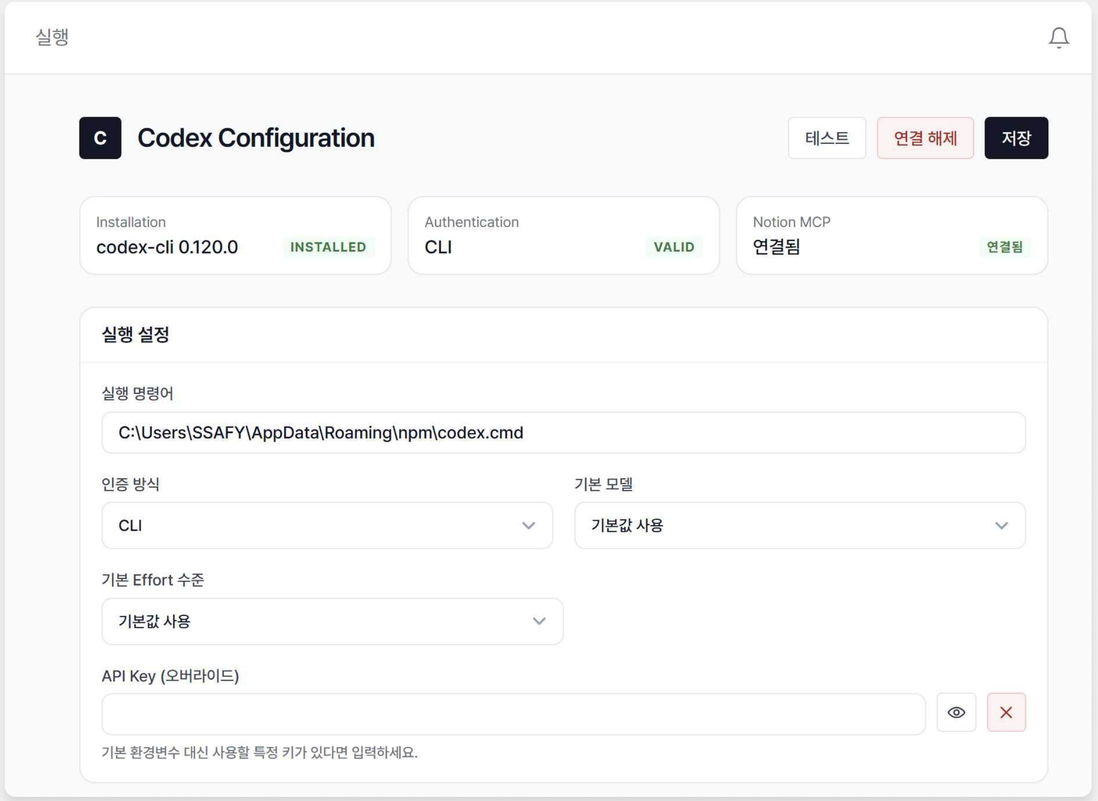
APPROACH.

1. 정보 분석 → Notion MCP 사전 패스로 개인 경험 brief 주입
2. 역할별 멀티 에이전트 오케스트레이션

TECH STACK.

#TypeScript #React #Vite #Fastify #PostgreSQL #Redis #Nginx #Docker #Nginx #Docker

REPRESENTATIVE SCREENS.



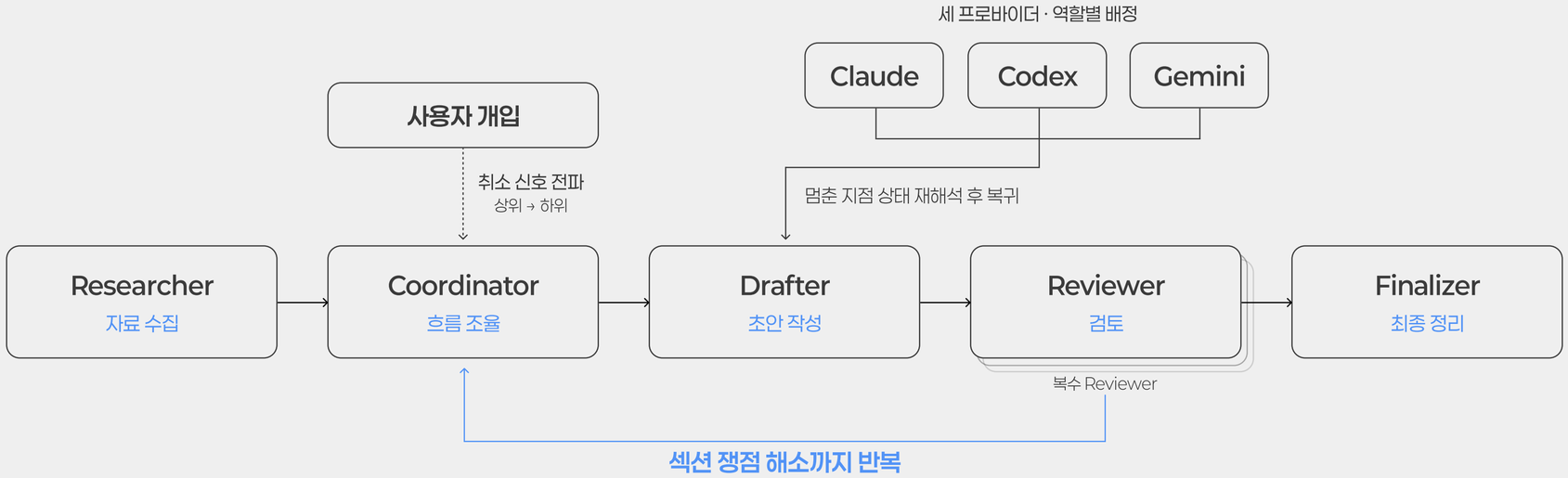
역할 / 결과.

멀티 에이전트 오케스트레이션·Notion MCP 연동·DART/웹/공고 컨텍스트 합성 구현과 출처 기반 정합성 처리, Docker·OCI 기반 self-host 인프라를 구축, Notion 동기화→회사·공고 분석→섹션 초안→리뷰 라운드로 이어지는 실행 흐름과 출처 우선순위 기반 사실 통합, **역할 기반 에이전트 조정 구조를 설계**했습니다

#멀티에이전트 #자소서 검토 오케스트레이터 # 개인 프로젝트 #1인 개발

여러 AI 에이전트가 역할을 나눠 자기소개서를 검토하는 멀티에이전트 오케스트레이터.

REVIEW PIPELINE.



TECH STACK.

#TypeScript #Node.js #React #WebSocket #Claude / Codex / Gemini

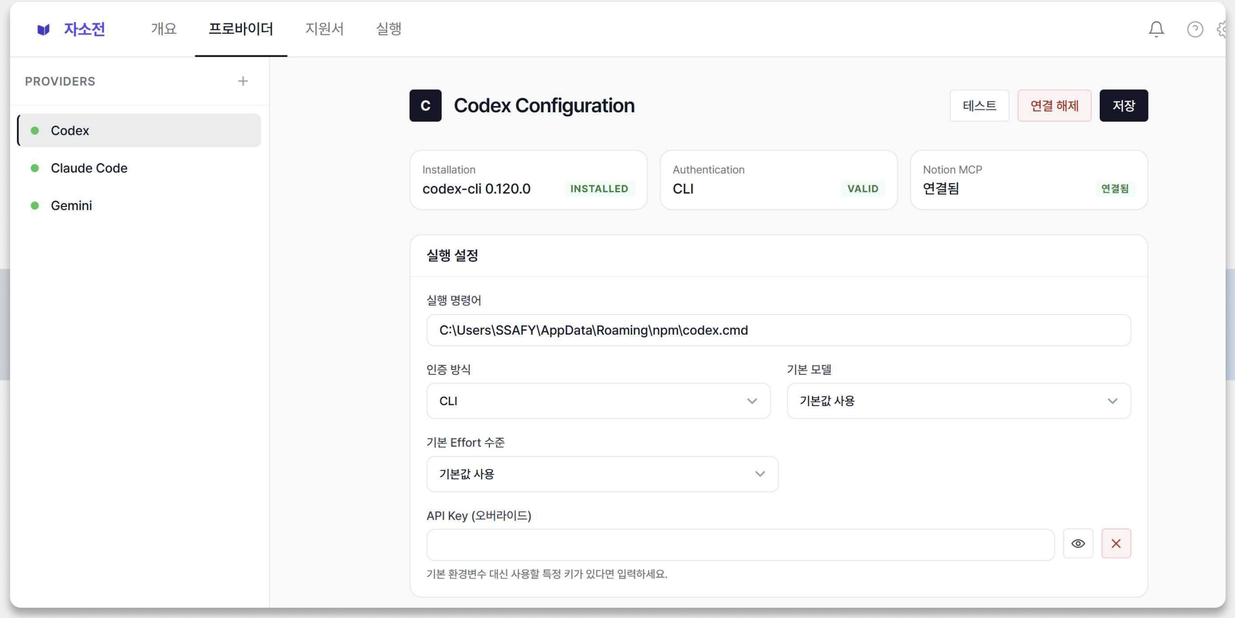
역할 / 결과.

단일 LLM 호출이 아닌 **역할 기반 멀티에이전트 구조**로 검토의 품질과 일관성을 확보했고, 현재 운영하며 개선하고 있습니다.

핵심 기능 흐름 소개

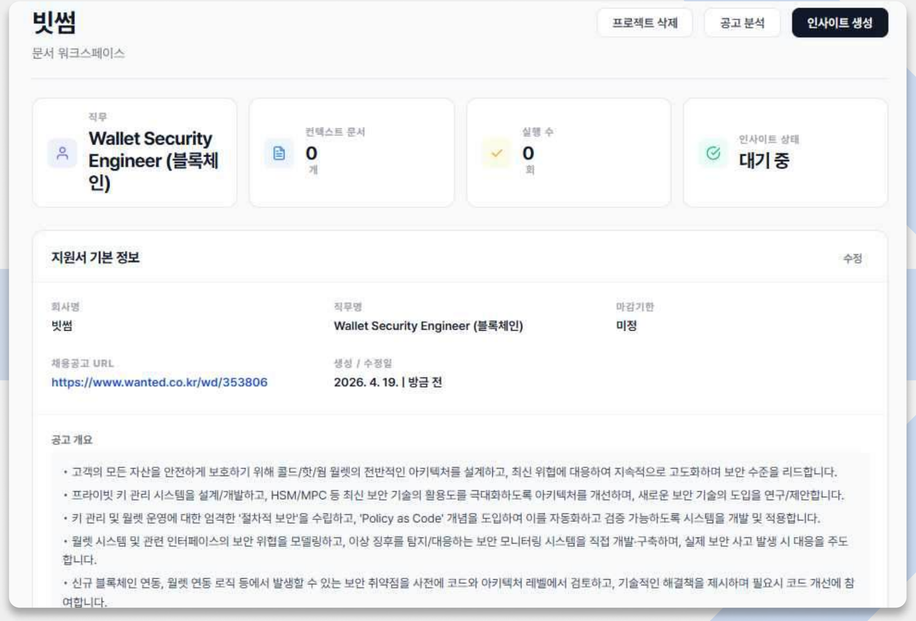
Core Function Flow.

1. AI 에이전트 연동.



- Claude·Codex·Gemini CLI 통합 실행 파이프라인 연결
- 로컬 시크릿 저장 기반 키 처리와 Notion MCP 연동
- 프로바이더 자동 설치·테스트 모듈로 셋업 효율 개선

2. 공고 분석.

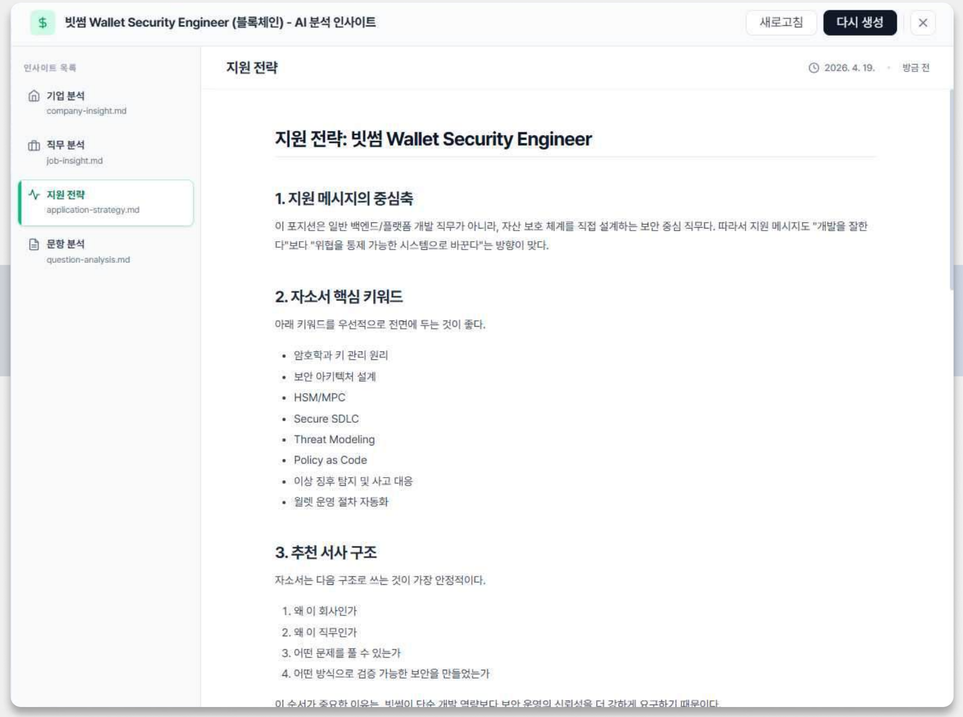


- JSON-LD → 정형 데이터 → HTML 3단 파서 연결
- 출처 등급 스키마·ATS 블랙리스트 기반 정합성 처리
- 교차 검증 게이트로 회사명 오인식률 개선

핵심 기능 흐름 소개

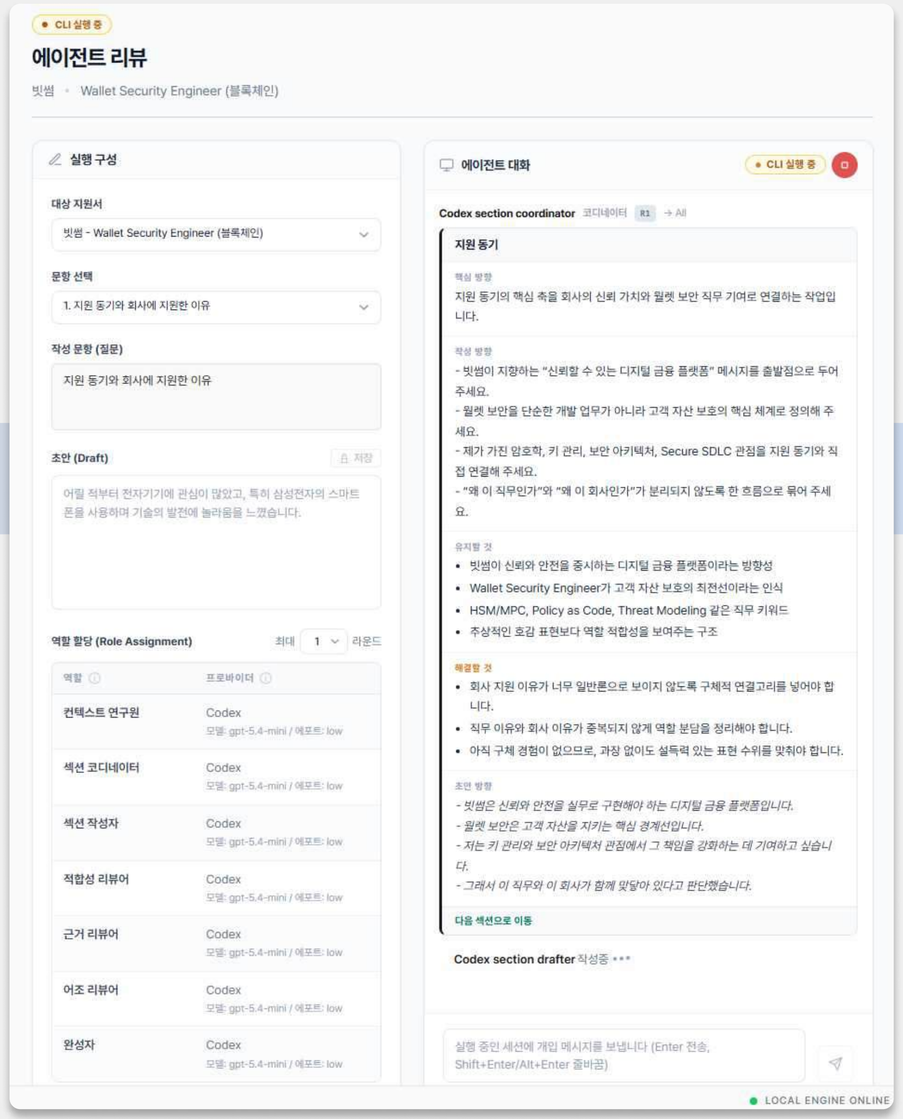
Core Function Flow.

3. 인사이트 생성.



- DART 공시 → 재무 → 컨텍스트 합성 파이프라인 연결
- 공시 > 웹 > 공고 3-tier 출처 우선순위 처리
- 회사 단위 캐시 모듈로 분석 일관성·비용 개선

4. 멀티 에이전트 첨삭.



- 조사 → 초안 → 리뷰 → 마감 다중 역할 파이프라인 연결
- WebSocket 기반 백엔드 ↔ 러너 세션 처리와 MCP 연동
- 결정론적 하네스·회귀 게이트로 출력 일관성 개선

#오픈소스 런타임 확장 #크로스-런타임 턴 교대

인프라 어시스턴트 봇.

지도 기반 학교 커뮤니티 · 현장 이슈 공유와 게시판 기반 정보 축적을 결합한 서비스.

PROBLEM.

1. 인적 리소스 공백
   부재 중 대응 공백
2. 시스템 한계와 속도
   장애 대응 수동 모니터링

APPROACH.

1. AI 기반 통합 관제
2. 장애 대응 자동화
3. 실시간 담당자 호출

TECH STACK.

#TypeScript #Node.js #SQLite #Discord.js #Docker

역할 / 결과.

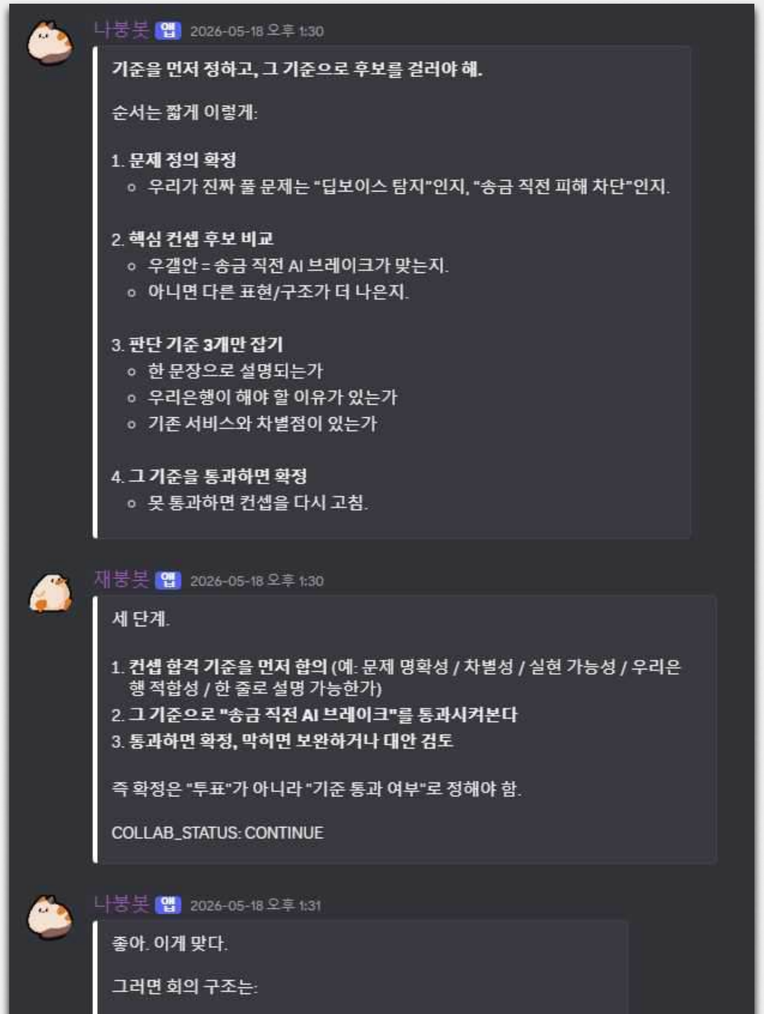
두 오픈소스 런타임이 공유 상태 파일과 봇 메시지로 **턴을 교대**하는 크로스-런타임 오케스트레이션을 베이스 위에 구축·운영했습니다.

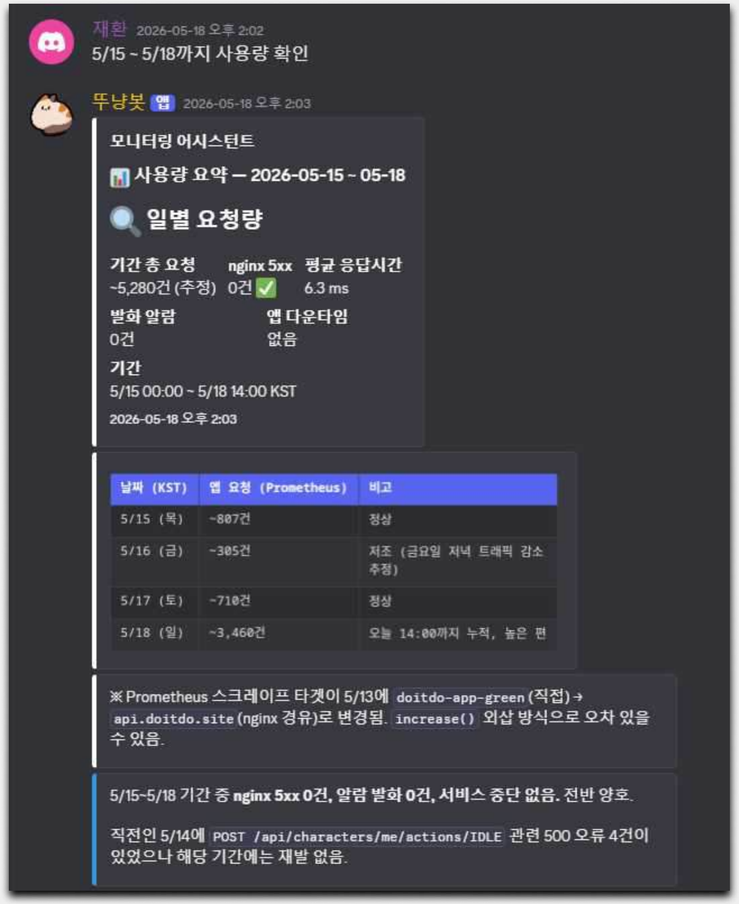
#오픈소스 런타임 확장 #크로스-런타임 턴 교대

인프라 어시스턴트 봇.

지도 기반 학교 커뮤니티 · 현장 이슈 공유와 게시판 기반 정보 축적을 결합한 서비스.

REPRESENTATIVE SCREENS.

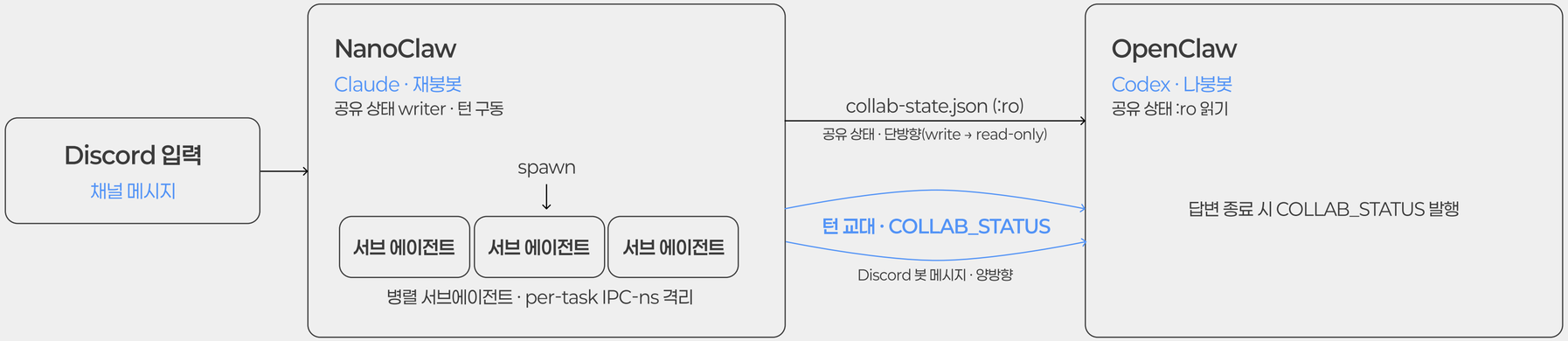




#오픈소스 런타임 확장 #크로스-런타임 턴 교대

오픈소스 에이전트 런타임을 확장해, 두 런타임이 턴을 교대하는 개인 멀티-에이전트 환경을 구축·운영.

ISOLATED MULTI-RUNTIME.



오픈소스 런타임(NanoClaw·OpenClaw) 컨트리뷰터 · 개인 포크 운영 · 공유 상태와 봇 메시지로 두 런타임 턴 교대를 조율

역할 / 결과.

OSS 베이스 (업스트림) — 컨테이너 격리 실행 · OneCLI 시크릿 게이트 · 그룹 큐 · SQLite
Jaeboong 보강 — docker.sock/SSH passthrough · NANO_UMASK · per-task IPC-ns · 로그 redaction · 관측(Prometheus/Grafana/Loki)

#접근성 #음성 인터페이스 #음성 의도분류

HearBe.

시각장애인을 위한 음성 기반 웹 쇼핑 지원 서비스.

PROBLEM.

1. 이미지 중심 쇼핑 환경
   스크린 리더로 읽을 수 없는 한계
2. 복잡한 결제/인증 절차
   혼자서는 완료하기 힘든 온라인 결제 과정

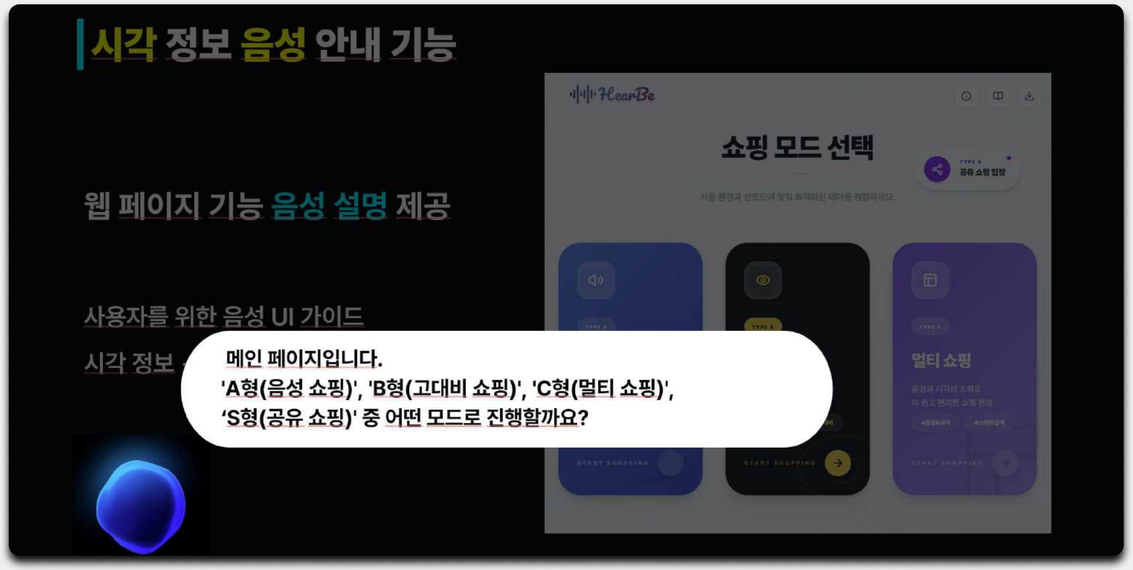
APPROACH.

1. OCR 기반 정보 재구성
2. 상태 기반 쇼핑 자동화
3. 하이브리드 NLU 설계

TECH STACK.

#Python #PyTorch #KoELECTRA #HuggingFace

REPRESENTATIVE SCREENS.



역할 / 결과.

KoELECTRA 기반 음성 의도분류를 평균 **23ms**로 처리해 끊김 없는 실시간 음성 쇼핑 흐름을 확보했습니다.

#접근성 #음성 인터페이스 #음성 의도분류

AI 파트담당.

ISOLATED MULTI-RUNTIME.

CAMPUNG

우리학교 지도기반 커뮤니티

캠 펑!

빛이 나는 SOL로

해커톤 대상 #백엔드 / 인프라

CAMPUNG!

지도 기반 학교 커뮤니티 · 현장 이슈 공유와 게시판 기반 정보 축적을 결합한 서비스.

PROBLEM.

1. 텍스트 게시판.
   실시간 현장 공유 한계
2. 현장 정보.
   기록 구조 간 연결 부족

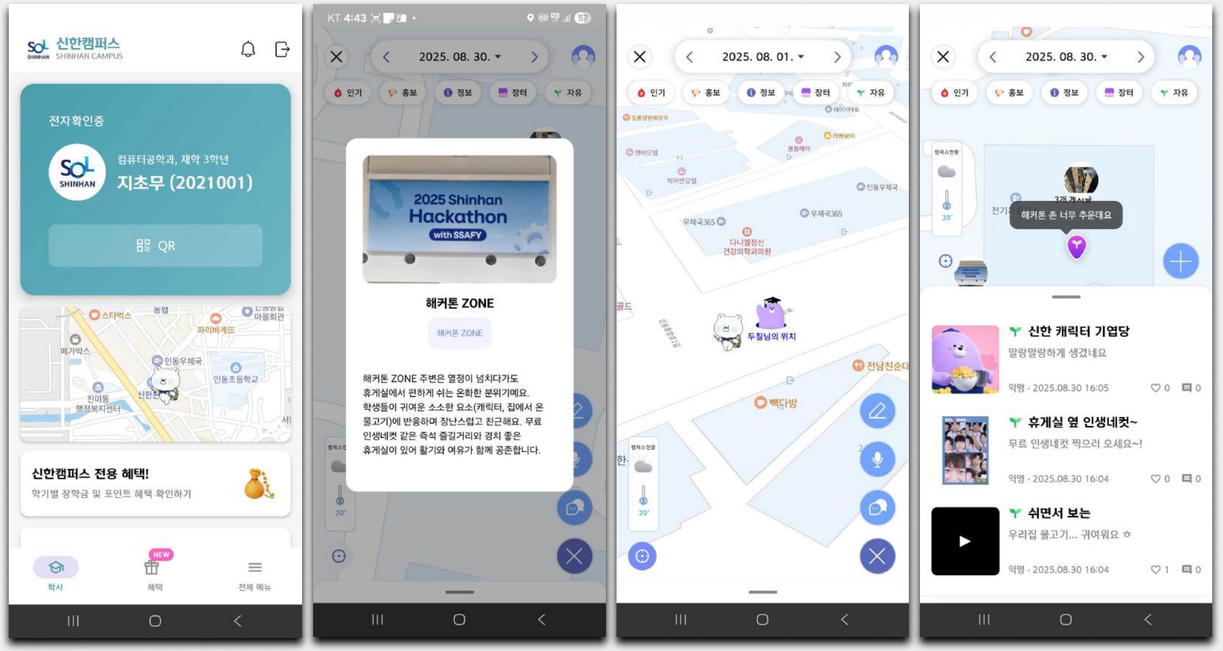
APPROACH.

1. 지도·게시판 데이터 구조 통합.
2. 감정 분석 → 캠퍼스 온도 시각화.
3. 파일 유형별 저장, 경로 분리.

TECH STACK.

#Spring Boot #MariaDB #Redis #Nginx #AWS S3 #Docker #GitHub Actions #WebFlux #OpenAI

REPRESENTATIVE SCREENS.



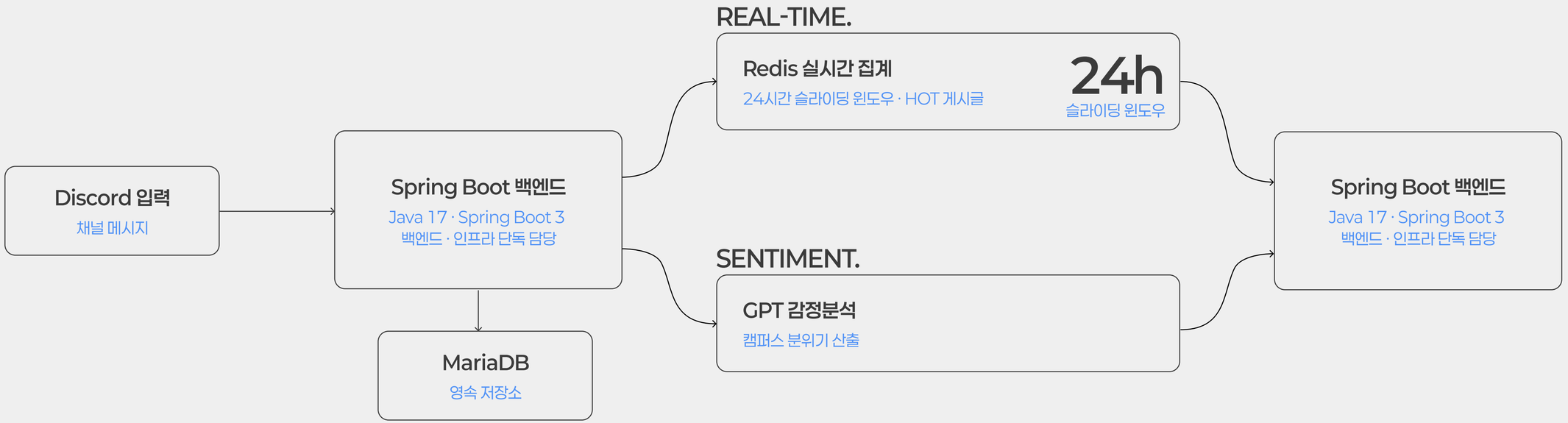
역할 / 결과.

백엔드 전반(API·온도 집계·파일 구조·CI/CD)을 설계·구현하여 캠퍼스 분위기를 **직관적으로 파악하는 서비스 구조**를 구축하고, 운영 효율성을 개선했습니다.
해당 프로젝트로 **신한 해커톤 대상**을 수상했습니다.

#대상 #백엔드 / 인프라 단독 #실시간 집계

실시간 집계와 GPT 감정분석을 시각화한 금융 서비스. 백엔드·인프라를 단독으로 담당.

ISOLATED MULTI-RUNTIME.



INFRA. Docker · GitHub Actions · SSH 자동 배포 · 헬스체크

TECH STACK.

#Spring Boot #MariaDB #Redis #Java #Docker #GitHub Actions

AI 감정 분석, 온도 지표, 저장 구조 분리

캠퍼스 온도 집계 설계.

게시글 목록만으로는 분위기를 직관적으로
전달하기 어려웠기 때문에,
활동량과 감정 분석 결과를 연결한
온도 지표를 설계했다.

```
private double
calculateBidirectionalAdjustment(
double currentTemp, int postCount,
int hour) {
PostActivityLevel activityLevel =
postActivityAnalyzer.analyzeCurren
tAct
ivity(postCount);
double baseRate =
activityLevel.getAdjustmentRate();
double protection =
getTemperatureProtectionFactor(
currentTemp,
activityLevel.isIncreasing());
TemperatureGuideline guideline =
guidelineConfig.getGuideline(hour)
;
double timeMult =
activityLevel.isIncreasing()
?
guideline.getIncreaseMultiplier()
:
guideline.getDecreaseMultiplier();
return currentTemp * (1 + baseRate
*
protection * timeMult);
}
```

설계 포인트.

- 활동량 분석으로 상승/하강 방향 판단
- 0도, 100도에 가까울수록 보호 계수로 변화 폭 축소
- 시간대별 가이드라인으로 낮, 밤 보정 배율 분리
- 지표가 급변하지 않도록 사용성 중심으로 완충

S3 업로드 경로 자동 분류.

이미지와 녹음 파일을 한 경로에 두면
운영과 검색이 복잡해지기 때문에,
Content-Type 기반
폴더 분리 전략을 적용했다.

```
public String
uploadFile(MultipartFile file)
{
String contentType =
file.getContentType();
String folder =
determineFolder(contentType);
String key = folder +
"/contents/"
+
LocalDate.now().format(DateTime
Formatter.ofPattern("yyyy/MM/dd
"))
+ "/" +
generateFileName(file.getOrigin
alFilename());
PutObjectRequest req =
PutObjectRequest.builder()
.bucket(bucketName).key(key).co
ntentType(contentType).build();
s3Client.putObject(req,
RequestBody.fromInputStream(fil
e.getInputStream(),
file.getSize()));
return baseUrl + key;
}
```

저장 계층 분리.

- MariaDB: 영속 데이터
- Redis: 캐시와 실시간 조회
- S3: 이미지, 녹음 파일 저장소
- image/*, audio/*를 서로 다른 폴 더로 라우팅

RESULT.

- 홈 화면에서 캠퍼스 분위기를 즉시 읽을 수 있는 서비스 지표를 확보
- 게시글이 적은 시간대에도 지표가 흔들리지 않도록 안정성 확보

- S3 key만으로 파일 유형과 위치를 빠르게 추적할 수 있게 됨
- 장애 원인 분석 범위를 저장 계층별로 좁혀 운영성을 높임

팀장 #백엔드 / 인프라

모아(MOA).

금융 Open API 기반 목표, 예산 관리 서비스 · 계좌 동기화, 소비 분석, 목표 계획 수립을 하나의 흐름 으로 연결.

PROBLEM.

1. 분석 중심 UI
   행동 변화 유도 한계
2. 외부 API–DB 기준 불일치.
   기록 구조 간 연결 부족

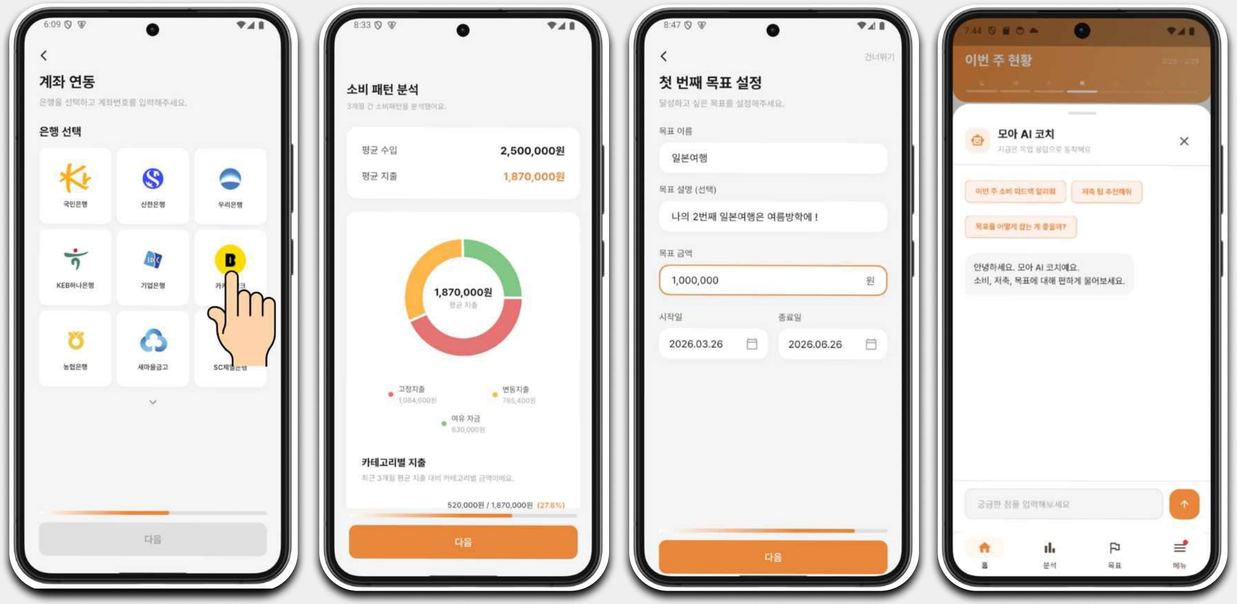
APPROACH.

1. 거래 분석 → 목표·주간 예산 실행 흐름 설계.
2. API–DB 기준점 통합(reconcile)
3. 수입 기반 예산·목표 우선순위 재조정

TECH STACK.

#Spring Boot #Docker #FastAPI #PostgreSQL #Ubuntu #SSAFY 금융 Open API

REPRESENTATIVE SCREENS.



역할 / 결과.

계좌 연동·예산 로직·거래 동기화(API) 구현과 **데이터 정합성 처리,** Docker 기반 **인프라를 구축**하고,
분석→목표→주간 예산→자유소비로 이어지는 실행 흐름과 계좌 기준 통합, 목표 우선순위 기반 조정 구조를 설계했습니다.

분석 결과를 실행 흐름으로 연결한 금융 서비스 구조

EXECUTION FLOW.

1. SYNC

계좌 연동 후 외부 금융
API 기준으로 계좌와
거래 데이터를 동기화.

2. ANALYZE

최근 거래 흐름을 집계,
소비 패턴을 해석해
사용자 상태를 보여준다.

3. PLAN

목표 설정과 우선순위
입력을 받아 예산과
저축 계획을 계산.

4. ACT

주간 예산과 자유소비를
홈에 연결해
실제 소비 행동으로 이어지도록 설계.

WHY THIS STRUCTURE.

Spring Boot로 금융 도메인 트랜잭션을 통합 관리하고,
FastAPI로 분석 로직 분리.

PostgreSQL·Docker 기반으로 데이터 관계와 실행 환경의 안정성을 확보.

WHAT I LEARNED.

계산식보다 데이터 기준 일관성이 핵심.

계좌 동기화·예산 로직으로 기준점 고정.

외부 API 동기화와 데이터 기준점 통일

1. 계좌 데이터 정합성 문제.

로컬 DB의 계좌 정보와 외부 금융 API 데이터가 어긋나면, 분석과 예산, 목표 화면 전체가 서로 다른 기준을 보게 됨.

WHY IT MATTERS.

- 계산보다 데이터 기준 정의가 핵심
- 동기화 불일치 → 사용자 신뢰 저하

SERVICE IMPACT.

- 계좌 정합성 → 추천·예산 로직의 기준점
- 기준 통일 → 분석·계획 흐름 일관성 확보

```
@Transactional
public List execute(Integer userId) {
List linked = accountRepo.findAllLinkedByUserId(userId);
User user = findUserUseCase.execute(userId);
Map external =
apiClient.inquireAccounts(user.getUserKey()).accounts().stream()
.collect(Collectors.toMap(AccountItem :accountNo, identity()));
return linked.stream()
.filter(acc > reconcile(acc, external))
.toList();
}
private boolean reconcile(Account acc, Map external) {
AccountItem ext = external.get(acc.getAccountNo());
if (ext = null) {
acc.markInactive();
return false;
}
acc.sync(ext);
acc.markActive();
return true;
}
```

RESULT.

- 외부 API 기준으로 Map을 구성해 계좌 비교를 O(1)로 처리
- 비활성 처리 → API 오류 시 안정적 복구
- @Transactional → 동기화 안정성 확보
- 분석, 예산, 목표가 모두 같은 계좌 기준점을 보도록 보장

월간 수입을 주간 실행 계획으로 바꾸는 예산 배분 로직

2. 예산 배분 알고리즘

단순 지출 분석만으로는 사용자가 행동으로 옮기기 어렵기 때문에, 월간 수입과 목표를 주 단위 실행 계획으 로 바꾸는 로직이 필요.

```
1   public AllocationResult allocateBudget(
2   User user, LocalDate start, Long monthlyIncome,
3   List essentials, List activePlans) {
4   long income = normalizeAmount(monthlyIncome);
5   List weeks = buildWeekFrames(start, allocationEnd);
6   Map incomeByWeek =
7   allocateMonthlyIncome(start, end, income, weeks);
8   List essentialAllocs =
9   buildEssentialAllocations( .);
10  applyShortageAdjustments(
11  weeks, budgetWeeks, incomeByWeek, essentialByWeek, planAllocs);
12  return new AllocationResult( .);
13  }
14
```

LOGIC STRUCTURE.

- 월간 수입 정규화
- 주 프레임 생성과 주차별 수입 분배
- 고정비 우선 차감
- 목표 저축 계획 반영
- 부족분 발생 시 우선순위가 낮은 목표부터 자동 조정

KEY JUDGMENT.

- 분석 → 계획으로 이어지는 구조 필요
- 목표 우선순위 설명 → 사용자 납득 강화

RETROSPECTIVE.

- 데이터 기준·동기화 설계 → 서비스 안정성 핵심
- 성과 지표 수집 → 추천 로직 정량화 계획

RESULT.

- 분석 → 예산·목표·소비로 이어지는 실행 흐름 구축
- 부족분 자동 조정 + 금액 변화 근거 확보

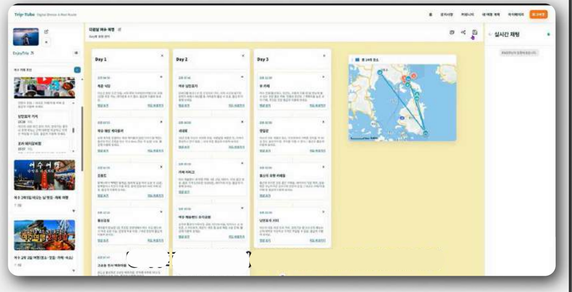
기타 프로젝트 및 경험

Other Projects.

Trip-Tube.

풀스택

유튜브 여행 영상 자막을 AI로 구조화하고
시맨틱 검색을 제공하는 **여행 계획** 플랫폼



- YouTube 자막 추출 → GPT 구조화 → 벡터 임베딩 파이프라인 구축
- 한국어 특화 임베딩 기반 시맨틱 검색 API 설계와 구현
- Spring Boot와 AI 서버를 분리해 검색, 추천 기능을 서비스 API와 연결

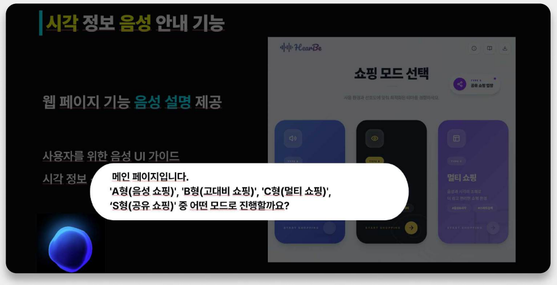
#Spring Boot #FastAPI #Vector DB #GPT

SSAFY 관통 프로젝트 · 2025.11 - 2025.12.26

Sequence.

백엔드

대학 간 PM, 디자이너, 개발자 교류 및 **프로젝트 매칭** 플랫폼



- 프로젝트 아카이빙과 팀원 평가 기능 중심의 백엔드 개발
- QueryDSL 기반 동적 필터링과 복합 조회 조건 처리
- JPA 연관관계 설계와 soft-delete 패턴 적용

#Spring Boot #JPA #QueryDSL #Docker

대학 연합 프로젝트 · 2024.10 - 2025.06

블루로봇 인턴.

프로젝트형 인턴

고정밀 균형 유지 **하드웨어**의 실시간 상태를
시각적으로 표현하는 **2D 시각화 모듈** 개발

- Kotlin 기반 2D 시각화 모듈을 단독 개발
- 하드웨어 센서 데이터를 수신해 균형 상태를 그래픽으로 렌더링
- 실시간 상태 표현 로직과 시각 인터페이스를 함께 설계

#Kotlin #2D Visualization

기업 인턴십 · 2025.06 - 2025.08

COMMON THREAD.

실시간성, 데이터 해석, 사용자 행동 을 연결하는 서버 중심 설계를 다양 한 도메인에서 반복적으로 경험.

2026.

End Of Portfolio.

정합성과 유지보수성을 우선하는 백엔드·AI 에이전트 개발자, 김재환입니다.

cbkjh0225@gmail.com · github.com/Jaeboong

Back-End · AI Agent Developer.

#에이전트 오케스트레이션 #백엔드 #인프라 운영