PORTFOLIO · AN SEONGJIN

RAG 챗봇을 설계부터 스토어 출시까지, 실서비스로 증명한 엔지니어

지원 포지션 : 다우기술 디지털혁신그룹 AX개발팀 · AI 개발 (경력)

안성진 · Full-Stack / AI Backend Engineer

hello@anveloper.dev · github.com/anveloper · anveloper.dev

FIT CHECK

AX개발팀이 찾는 경험, 이렇게 채워왔습니다

공고 요구 역량	제 경험
RAG 기반 지식 검색·응답 시스템 설계/구현	직접 구현 육아밸 : pgvector 임베딩 검색 + SSE 스트리밍 RAG 챗봇을 설계·구현, **iOS·Android 양대 스토어 출시**
Vector DB 기반 지식 구조 설계	직접 설계 Prisma **39개 모델(pgvector 임베딩 포함)** 데이터 모델링, 도메인 테이블 30+
LLM 기반 Agent 개발 및 Orchestration	직접 구현 딸깍톤 : 디스코드 명령으로 호출하는 **AI 자동 심사 파이프라인**, 판단은 프롬프트·흐름 통제는 코드
정형/비정형 데이터 처리 파이프라인	운영 중 커머스 **16개 배치 스케줄러**, 외부 플랫폼 3사 연동 데이터 정합, 임베딩 생성·저장 파이프라인
REST API 개발 및 백엔드 서비스 연계	주력 API **120+ 엔드포인트** 설계·운영, OpenAPI 명세를 단일 계약으로 멀티레포 타입 동기화
Python 기반 서비스 개발·운영	전환 준비 주력은 TypeScript·NestJS. 다만 **Java에서 TS로 주력 스택을 전환해 낸 이력**이 있고, RAG·Agent 설계 개념은 언어에 독립적입니다. FastAPI·LangChain 재구현 학습으로 전환을 준비하고 있습니다

우대사항 접점 : Claude Code·Codex 멀티에이전트 협업 체계(CLAUDE.md·AGENTS.md) 직접 구축 / MCP·Tool·Skill 기반 Agent 기술 실사용 / LLM 입출력을 Zod 스키마로 검증하는 구조화 출력 설계

육아밸 : RAG 챗봇 · AI 주간 리포트 육아 서비스

2026.06 · iOS App Store(별점 4)·Google Play 양대 스토어 출시

프로젝트 배경

- 육아 기록을 기반으로 질문에 답하는 RAG 챗봇과 AI 주간 리포트를 제공하는 모바일 서비스
- Expo 네이티브 셸 + Next.js 웹뷰의 하이브리드 앱 구조

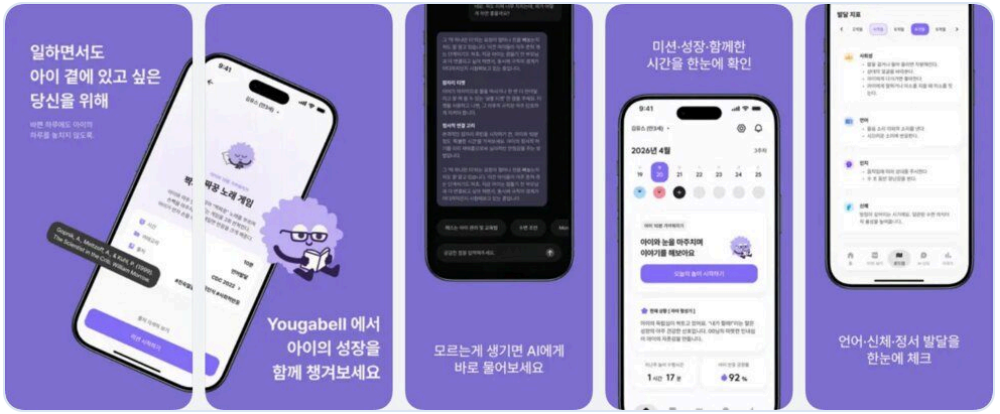
주요 역할 : API·백엔드 약 80% 기여 (커밋 약 80%)

- 모든 LLM 호출을 NestJS API 한 곳으로 모으고, 임베딩 검색(RAG) 기반 챗봇을 **SSE 스트리밍 UI**로 구현
- Zod 스키마(Output.object)로 **LLM 응답 구조를 강제**하고 토큰 사용량까지 DB에 기록. AI 실패는 명시적 실패로 처리하고 **forceRegenerate 재실행**으로 복구하는 관측 가능한 AI 파이프라인
- OpenAPI(Swagger) 명세를 단일 계약으로 삼아 멀티레포(API·웹·어드민·모바일) 타입 동기화
- Prisma **39개 모델**(pgvector 임베딩 포함) 데이터 설계

핵심 성과

- 기획·디자인·개발이 나뉜 팀에서 **짧은 기간에 양대 스토어 출시까지 도달**
- 스트리밍 응답·구조 검증 등 AI 기능을 실사용 품질로 다듬는 경험 확보

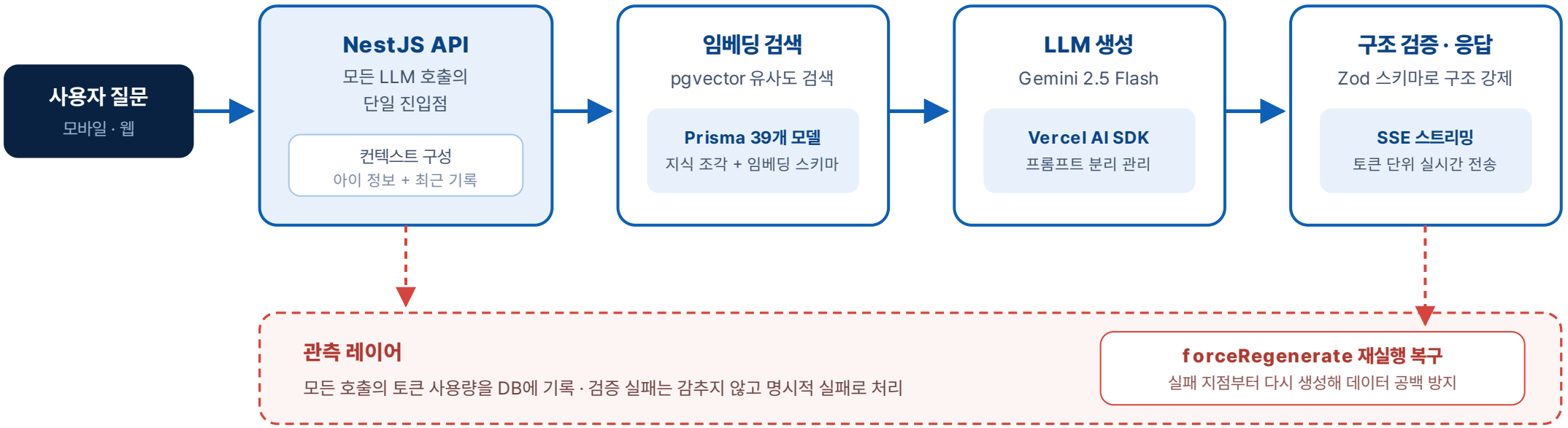
App Store 등록 카드 · 별점 4



공식 스크린샷 : 기록 · RAG 챗봇 · 발달 리포트

Stack : NestJS 11 · Prisma 7 · Next.js 16 · Expo · Vercel AI SDK · Gemini 2.5 Flash · Supabase(pgvector)

실패까지 설계한, 관측 가능한 RAG 파이프라인



생성 품질은 흔들릴 수 있다는 전제로, **구조 검증 · 사용량 기록 · 재실행 복구**를 파이프라인 안에 함께 설계했습니다

동일한 설계를 Python · FastAPI · LangChain 스택 위에서도 그대로 재현할 수 있습니다

딸깍톤 : AI 해커톤 운영 · 자동 심사 플랫폼

2026.02 ~ 2026.04 (약 6주) · ttalkkakthon.vibecodingclub.kr 운영

백엔드·AI 파이프라인 담당 · 개발 커밋 약 85%

LLM 기반 자동 심사 Agent 파이프라인

- 운영자가 디스코드에서 명령 한 줄로 호출하면, 서버가 프로젝트 정보를 조회하고 **GitHub 코드와 제출 정보를 검토**해 점수·피드백을 생성, 파싱해 DB에 저장한 뒤 디스코드로 회신
- **심사 규칙을 코드가 아닌 프롬프트로 분리** : 기준이 바뀌어도 서버 코드 무수정. 한 팀이 여러 프로젝트를 낸 경우까지 처리하도록 확장
- 판단은 AI가 하되 조회·저장·회신 흐름은 코드가 통제, 행사 도중에도 결과가 흐트러지지 않고 축적
- 심사 결과를 마크다운 타이핑 효과 + TTS 음성으로 현장 표시

이 경험이 말하는 것

- 공고의 **"LLM 기반 Agent 개발 및 Orchestration"**과 같은 문제 : 도구 호출·평가·저장·회신을 잇는 워크플로우를 직접 설계·운영
- 실제 행사에서 실시간으로 돌아간 시스템, 데모가 아닌 운영 검증

AI 심사 결과 : 항목별 점수·피드백 자동 생성

행사 라이브 사이트 (ttalkkakthon.vibecodingclub.kr)

Stack : Next.js 16 · TypeScript · Prisma 7 · Neon Postgres · Vercel · Discord API

날별 : 자체 사주 엔진 + AI 운세 서비스

2026.04 · 1인 기획·개발 (천문우주과학 전공의 천체 계산 흥미에서 출발)

설계 원칙 : 정확해야 하는 계산은 코드, 유연한 판단은 AI

- 사주 계산은 절기·율리우스적일 기반 **결정론적 TypeScript 엔진**이 전담, AI는 결과 풀이만 담당. 모델을 교체해도 계산값이 달라지지 않는 구조
- 일간·오행·성별·절기 **4차원 버킷팅 + 캐싱**으로 유사 사주의 AI 호출을 재사용, 호출 비용 절감
- **Gemini → GPT-4o-mini → Mock 3단 폴백 체인**으로 모델 장애에도 서비스 지속
- OAuth 3종 + refresh token rotation, pnpm·Turborepo 모노레포로 웹·모바일이 엔진 공유

이 경험이 말하는 것

- 모델 추론 구조를 **비용·장애·재현성 관점에서 설계**해 본 경험 : 공고의 "모델 추론 구조 설계 및 성능 개선"과 닿아 있는 문제의식

결정론 엔진 (코드)

절기 · 율리우스적일 계산

항상 같은 입력 = 같은 결과

TypeScript 자체 구현

→

AI 풀이 (LLM)

계산 결과를 자연어로 해석

모델 교체 자유

역할이 겹치지 않는 경계

버킷팅 캐시

일간 · 오행 · 성별 · 절기 4차원 키로 풀이를 캐싱

유사 사주는 AI 호출 없이 재사용, 호출 비용 절감

3단 폴백 체인

Gemini → GPT-4o-mini → Mock

모델 장애 상황에도 응답이 끊기지 않는 구조

Stack : Next.js 16 · React 19 · Expo · TypeScript · Prisma 7 · PostgreSQL · Gemini

멀티테넌트 커머스 플랫폼 : 설계부터 운영까지

기술연구소 연구2팀 팀장(과장) · PM 겸 개발 리드 · 팀 2~7명 리드

데이터 · 파이프라인

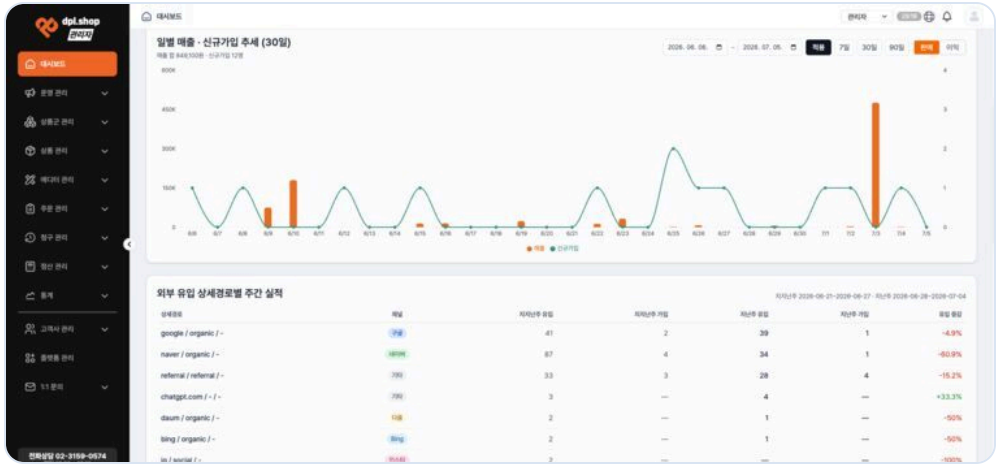
- 멀티테넌트 SaaS 데이터 모델 **40+ 테이블**, API **120+ 엔드포인트** 직접 설계 (DPS Store, 코어 설계·백엔드·인프라 단독 + 프론트엔드 2인 협업)
- 주문 승인·결제·배송 추적·통계 등 **16개 배치 스케줄러**로 정형 데이터 파이프라인 자동화 (DPS, 관리자·파트너·스토어·크리에이터 4개 역할군)
- 네이버 스마트스토어·Cafe24·Shopify 등 **외부 플랫폼 연동을 단독 구현**, 주문·재고·정산 데이터 정합 유지

운영 · 인프라

- Caddy On-Demand TLS로 테넌트별 커스텀 도메인 자동 SSL, PM2 Blue-Green **무중단 배포**와 헬스체크·롤백 운영
- 15년 된 PHP 레거시 중 담당 영역 약 3만 줄을 Remix·TypeScript로 재구현, 기술부채 해소

이 경험이 말하는 것

- AI 기능이 올라설 토대인 **데이터 파이프라인과 서비스 운영**을 처음부터 끝까지 책임져 본 경험. 정부지원 프로토타입을 실제 매출 단계까지 끌고 간 0에서 1의 이력



DPS 관리자 대시보드 : 매출·유입 데이터 파이프라인

DPS Store : 테넌트별 커스텀 도메인 스토어

Stack : Next.js 16 · React 19 · TypeScript · Prisma 7 · MariaDB · NestJS · Caddy · PM2 · AWS

TRANSITION PLAN

Python은 세 번째 주력 언어가 됩니다

전환을 해낸 이력이 근거입니다

- 가장 오래 다룬 언어는 Java였습니다. 현업에서 **TypeScript·Node로 주력을 전환**해 멀티테넌트 플랫폼을 처음부터 만들었고, Remix·NestJS·Expo 등 새 스택을 프로젝트마다 학습해 실서비스에 적용해 왔습니다
- RAG·Agent 설계에서 어려운 부분은 문법이 아니라 **검색 품질, 구조화 출력, 실패 복구, 비용 통제**입니다. 이 문제들을 이미 실서비스에서 풀어봤습니다
- 입사 전까지 육아밸 RAG 구조를 **FastAPI·LangChain·LangGraph로 재구현**하는 학습을 진행해 언어 갭을 좁히겠습니다

AX개발팀의 일하는 방식과 맞닿은 지점

- "완성도보다 빠른 시도" : 정부지원 프로토타입을 전시회 출품, 실서비스, 매출까지 반복 개선으로 끌고 간 방식과 같습니다
- "초기 단계, 방향을 함께" : DPS Store 코어를 백지에서 설계했고, 팀 리뷰 문화와 협업 규칙(문서 우선 프로세스)을 직접 만들어 온 경험이 있습니다

우대사항 접점 : Agent 도구를 만들며 씁니다

- Claude Code와 Codex를 병행하는 **멀티에이전트 협업 체계**를 직접 구축 : 진입점과 파일별 작업 범위를 CLAUDE.md·AGENTS.md로 분리해 충돌 없이 동시 작업
- 반복 작업은 **스킬(Skill)로 모듈화**, 작업 단위는 plan 문서로 관리. MCP·Tool 기반 Agent 생태계를 실무 흐름 안에서 사용
- AI를 코드 생성만이 아니라 기획 검토·코드 리뷰에 투입, 누락 시나리오를 사전에 탐지

보조 근거

- 항해 플러스 백엔드 : 동시성 제어·대용량 트래픽 처리·순차 보장 설계, TDD (Java·Redis, Kafka는 교육 과정에서 경험)
- 항해 플러스 프론트엔드 5기 블랙배지(상위 1%) · SSAFY 7기 우수상 4회

기술 검토에서 멈추지 않고
실제 고객에게 닿는 AI 기능을,
AX개발팀에서 만들겠습니다.

감사합니다.

AI Backend Engineer 지원자 안성진 · hello@anveloper.dev · github.com/anveloper