CareMinder

케어마인더 (2025.10 ~ Now)

| CareNote 개발 (음성 기반 간호 진술문 요약 서비스)

Next.js, FastAPI, Clova Speech, LangChain, LangSmith, Qdrant

- Clova Speech 기반 화자 분리 및 키워드 부스팅을 적용하여 진술문 생성 정확도 향상
- LangGraph 기반의 워크플로우를 설계하여 비정형 데이터를 전문 의료 문장으로 정제하는 파이프라인을 구축
- Qdrant 벡터 DB로 의료 지식 베이스를 구축하여 용어 추출 및 검색 성능을 강화
- LangSmith로 LLM 입출력을 모니터링하며 추론 품질을 실시간으로 최적화

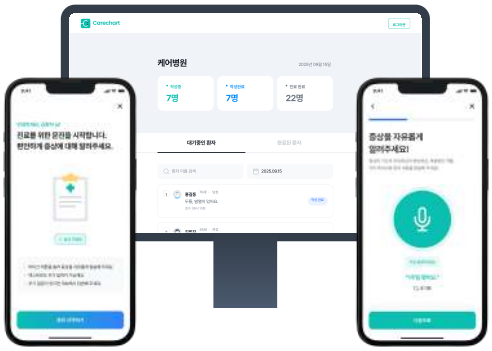
| CareAdmin 개발 (사내 관리 서비스)

Spring Boot , JPA, QueryDSL, MySQL, Redis

- OAuth 2.0 기반 네이버 웍스 인증 체계 구축
- 출결, 휴가, 일정을 통합 관리하는 CQRS 기반의 캘린더 서비스 설계 및 개발
- Facade와 Projection 패턴을 적용하여 복잡한 도메인(출결·휴가·일정)의 조회 성능을 최적화
- Transactional Outbox 패턴과 비동기 동기화를 활용하여 데이터 정합성 및 시스템 가용성 확보

| CareForm 개발(음성 기반 AI 문진 서비스)

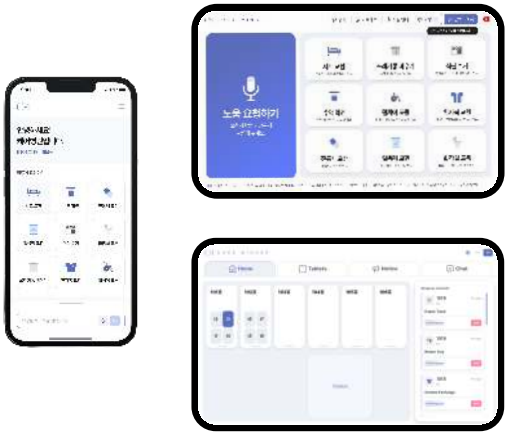
Spring Boot , JPA, QueryDSL, MySQL, MongoDB, Redis,



- 멀티 LLM(Clova, GPT, Gemini) 하이브리드 운용을 통한 추론 성능 극대화 및 비용 최적화
- Google STT 기반 실시간 음성 데이터의 의료 문진 데이터 정형화 파이프라인 구축
- 사용자 응답 시나리오에 따른 동적 문진 폼 엔진 및 검진 모듈 설계·개발
- CompletableFuture와 비동기 논블로킹 통신(RestClient)을 결합하여, 다중 섹션으로 구성된 AI 문진 분석 로직을 병렬화함으로써 전체 응답 시간을 약 3배 단축하고 동시 처리 성능 개선
- 빈번한 데이터 수정이 발생하는 문진 답변 시스템에서 Bulk Upsert 로직을 설계하여 데이터베이스 네트워크 오버헤드를 최소화하고, AI 분석 결과와 사용자 입력을 효율적으로 병합

| CareFlow 개선 (스마트 베드 & 그룹웨어)

Spring Boot , JPA, QueryDSL, MySQL, MongoDB, Redis



- 태블릿 상태 변화 실시간 감지 및 푸시 알림 체계 구축을 통한 현장 장애 대응 속도 80% 개선
- 실시간 서버 에러 모니터링 및 즉각적인 장애 전파 시스템 설계를 통한 서비스 가용성 및 안정성 확보
- Microsoft Teams Webhook 연동을 통한 의료진 간 협업 알림 워크플로우 자동화
- SRP와 OCP 위배 및 의존성 해결을 위한 전략 패턴 도입을 통한 확장 가능한 인증 로직 설계
  - 문제: 모든 역할(Role)의 인증 로직이 하나의 서비스에 집중되어 클래스가 비대해지고, 역할 추가나 로직 변경 시마다 연관된 모든 매니저와 기존 코드를 전수 수정해야 하는 높은 결합도 및 유지보수 비효율 발생
  - 해결: BaseAuthManager 인터페이스를 필두로 한 전략 패턴을 도입 하여 공통 로직과 각 역할별 독립적인 인증 로직을 구현체로 분리 하고 AuthManagerFactory를 구현 하여 런타임 시점에 필요한 매니저만 동적으로 주입 받는 구조로 리팩토링
  - 성과: 의존성 전이 최소화 및 단일 책임 원칙(SRP) 준수하고 인증 방식이 변경 시 기존 코드 수정 없이 클래스 추가 및 수정만으로 대응이 가능한 유연한 구조를 확보하고, 인증 로직 간의 결합도를 낮추어 코드 가독성 및 단위 테스트 용이성 대폭 개선

| 인프라 개선

NCP, Docker, Docker-Compose, GitHub Actions, Prometheus, Grafana, Loki

- 인프라 운영 비용 및 자원 절감을 위해 Jenkins를 GitHub Actions로 전환
- Docker Compose 사용 및 배포 스크립트 분리 설계로 공통 코드 수정 시 모든 서비스에 일괄 반영되는 배포 일관성 확보
- Docker Hub에서 NCP Container Registry로 도커 이미지 이전 보안 강화
- Matrix 전략 기반의 모노레포 환경에서 다중 모듈 CI/CD 파이프라인 구축
  - 이슈: 모노레포 내 다수 마이크로서비스의 복잡한 배포 의존성으로 인해 변경 모듈 추적이 어렵고, 매번 전체 시스템을 재배포하는 비효율적인 프로세스와 수동 배포에 따른 환경 설정 오류 위험 상존
  - 해결: GitHub Actions의 Matrix 전략과 Path Filter를 결합하여, 변경된 모듈만 선택적으로 빌드하고, 공통 모듈 변경 시에는 Matrix와 Docker-Compose를 통해 연관 서비스들을 병렬로 동시 배포하도록 자동화 설계
  - 성과: 전체 빌드 및 배포 시간을 기존 대비 60% 이상 단축하고, 공통 로직 변경 시 발생할 수 있는 서비스 간 버전 불일치 문제를 원천 차단하여 MSA 운영 안정성 확보

YouTIL

GitHub: youtil-cloud
Docs: youtil-cloud-wiki

| 설명

GitHub 커밋 내역을 분석하여 학습 기록(TIL)을 자동으로 생성하고, TIL을 기반으로 맞춤형 면접 질문을 추출해 개발자가 기록에 들이는 공수를 줄이고 자신의 코드를 복기하며 기술 면접까지 대비할 수 있는 서비스입니다.

| 프로젝트 정보

팀 구성: FE(1), BE(2), AI(2), DevOps(2)
역할 : Lead DevOps Engineer

| 기술 스택

FE
- TypeScript, Next.js
- SCSS + Dart Sass
- TanStack Query

BE
- Java, Spring Boot
- JPA, QueryDSL
- MySQL, Redis, Kafka

AI
- Python, Pytorch
- vLLM, FastAPI
- Gemma3, Mistral
- LangChain
- Qdrant

DevOps
- AWS, GCP
- Docker, Kubernetes
- Terraform, Flyway
- Nginx, OpenVPN
- GitHub Actions, Helm, ArgoCD
- SigNoz, Prometheus, Loki, Tempo, Grafana
- k6, Playwright, Artillery

| 주요 기여 사항

- v1: 수동 빅뱅 배포
  - 배경: 프로젝트 초기, 복잡한 인프라 설정보다 빠르게 MVP를 출시하여 비즈니스 가치를 검증
  - 방식: 배포 스크립트 실행 시 서버에서 직접 Git Pull로 소스를 갱신하고, 빌드 후 PM2를 활용해 프로세스를 일괄 재시작하는 방식 채택
  - 성과: 익숙한 도구를 통해 배포 환경을 빠르게 구축했으나, 서비스 규모 확장에 따른 빌드 시간 정체 및 전체 서비스 재시작으로 인한 배포 중단 시간 발생 확인
- v2: GitHub Actions CI/CD 구축
  - 배경: FE, BE, AI의 병렬 개발이 진행되면서 코드 병합 후 빌드 오류 발견, 팀원 간 스타일 불일치, 테스트 누락 등 통합 안정성 저하 문제 발생
  - 방식: Jenkins/GitLab CI 대비 낮은 러닝커브와 GitHub 연동성이 뛰어난 GitHub Actions를 최종 도구로 선정
  - 성과: Required Status Check를 통한 결함 코드 병합 원천 차단 및 빌드 배지/Discord 알림으로 즉각적인 피드백 루프 구축.
- v3: AWS CodeDeploy CD 자동화
  - 배경: 단순 스크립트 실행 방식은 배포 성공 여부 판단이 어렵고, 장애 발생 시 수동 롤백으로 인한 서비스 다운타임 발생 우려가 큼.
  - 방식: AWS CodeDeploy를 연동하여 배포 가시성을 확보하고, 배포 설정(AppSpec)을 통한 안정적인 배포 프로세스 수립.
  - 성과: 배포 이력 관리 및 실패 시 즉각 롤백 기반을 마련하였으며, 인스턴스 그룹에 대한 자동화된 배포 제어권 확보로 배포 안정성 극대화.
- v4: Kubeadm 클러스터 자가 구축
  - 배경: 단일 서버 기반 운영의 한계를 극복하고, 서비스 자가 치유 및 무중단 배포가 가능한 클러스터 환경의 필요성 대두.
  - 방식: Kubeadm을 활용하여 쿠버네티스 클러스터를 직접 구축하고, Rolling Update 및 Self-healing 환경 구현.
  - 성과: 고가용성(HA) 환경 구축 능력을 검증하였으며, 운영 효율성의 비약적 도약 달성.
- v5: Amazon EKS 전환
  - 배경: 직접 구축한 클러스터의 마스터 노드 관리 부담을 줄이고, AWS의 강력한 보안 및 네트워크 생태계를 활용하고자 함.
  - 방식: AWS EKS 기반 완전 관리형 쿠버네티스로 전환하고, ALB Ingress, IAM, EFS 등을 유기적으로 통합.
  - 성과: 컨트롤 플레인 관리 공수를 최소화하는 동시에, 대규모 트래픽에도 유연하게 대응 가능한 엔터프라이즈급 클라우드 네이티브 환경 완성.

| 트러블 슈팅

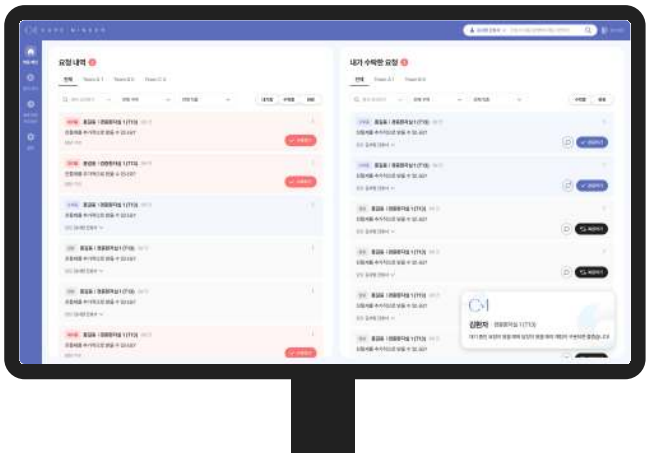
- Terraform 기반의 IaC 도입 및 인프라 관리
  - 문제: 제한된 리소스와 인프라 비용 최적화를 위해 클라우드 서비스(AWS, GCP) 간의 빈번한 서버 이전 및 환경 재구축이 발생하며, 이 과정에서 수동 설정으로 인한 리소스 누락과 환경 불일치 문제 심화
  - 해결: Terraform을 활용한 IaC를 도입하여 인프라 구성을 코드화하고, 클라우드에 구애받지 않는 표준화된 프로비저닝 파이프라인 구축
  - 성과: 서버 이전 및 환경 구축 시간을 기존 대비 80% 이상 단축하고, 형상 관리를 통해 환경 간 일관성을 100% 확보함으로써 인프라 운영의 안정성 및 비용 효율성 극대화

CareVoice & CareLink

케어마인더 - 인턴 (2024.09~2024.12)

| 설명

음성 인식 기반 통합 AI 스마트 병원 솔루션입니다. 음성인식 기술로 고령 환자의 병상 요청을 실시간 전달하는 CareVoice와 의료진의 원격 관리를 돕는 CareLink로 구성되어 있습니다.



| 프로젝트 정보

팀 구성: WEB(3), APP(2), BE(3)
역할 : Backend & DevOps Engineer

| 기술 스택

WEB
- TypeScript
- Next.js, Vite
- TanStack Query, Recoil
- Sentry

APP
- TypeScript
- React Native, Expo
- TanStack Query
- Zustand & Jotai

BE
- Java, Spring Boot
- JPA, QueryDSL
- MySQL, Redis, MongoDB

DevOps
- NCP
- Docker
- Nginx
- Jenkins
- Prometheus, Loki, Grafana

| 주요 기여 사항

- 병원·병동·구역·환자 간의 관계를 최적화한 계층적 도메인 모델을 설계하여 공간 기반의 데이터 관리 및 권한 제어 체계 구축
- JWT 기반 인증 체계에 RTR(Refresh Token Rotation) 전략을 도입하여 무상태 인증의 보안 취약점을 개선하고 토큰 탈취 위험을 최소화함
- MongoDB의 유연한 스키마를 활용하여 문항 변경 및 추가에 유연하게 대응할 수 있는 설문 기능을 구현
- STOMP 프로토콜과 Channel Interceptor를 결합하여 간호사와 환자 간의 실시간 채팅을 구현
- 1인 DevOps 환경으로 NCP를 활용하여 개발 및 운영 환경을 설계하고, Jenkins를 통해 빌드·테스트·배포 전 과정을 자동화한 CI/CD 아키텍처를 구축함

| 트러블 슈팅

- 인증 아키텍처 보안 고도화 (JWT & RTR)
  - 문제: 무상태(Stateless) 인증 방식에서 발생할 수 있는 토큰 탈취 및 재사용 공격에 대한 방어 기제 부재
  - 해결: RTR(Refresh Token Rotation) 전략을 도입하여 토큰의 생명 주기를 엄격히 관리하고 보안 취약점을 보완
  - 성과: 토큰 탈취 위험을 최소화하고 무상태 인증의 장점을 유지하면서도 세션 관리 수준의 높은 보안성 확보
- Redis 기반 통합 인증 보안 레이어 설계
  - 문제: 무차별 대입 공격(Brute-force Attack) 시, 매번 발생하는 패스워드 해싱 연산과 RDB 조회로 인한 서버 리소스 고갈 및 DB 병목 현상 발생
  - 해결: RDB 조회 전 단계에 Redis 캐시 계층을 배치하여, 로그인 실패 횟수와 잠금 상태를 선제적으로 검증하는 Early Return 구조 설계
  - 성과: 보안 임계치 초과 요청에 대한 서버 리소스 소모를 90% 이상 차단하여 시스템 가용성을 확보하고, Redis TTL을 활용한 자동 잠금 해제(Self-Healing)로 운영 효율성 및 UX 개선
- 보안 강화형 실시간 메시징 아키텍처 설계
  - 문제: WebSocket 표준의 Custom Header 제약으로 인한 인증 토큰 노출 위험 및 실시간 인가 처리의 어려움
  - 해결: STOMP 프로토콜 및 Channel Interceptor를 도입하여 메시지 프레임 단위의 JWT 인증/인가 레이어 구축
  - 성과: 안전한 의료진-환자 간 통신 환경 확보 및 비정상적인 소켓 연결에 대한 선제적 차단 체계 수립
- Docker 이미지 누적 방지를 위한 인프라 최적화
  - 문제: CI/CD 반복 빌드 시 발생하는 <none> 태그 이미지(Dangling Images) 누적으로 인한 스토리지 고갈 문제 발생
  - 해결: Jenkins 파이프라인 내 docker image prune 자동화 및 주기적 클린업 스케줄링 도입
  - 성과: 인프라 자원 관리 효율성 제고 및 빌드 실패율 최소화

HELPT

졸업 프로젝트 (2024.03~2024.6)

GitHub: helpt-backend
Docs: HELPT-최종보고서

| 설명

AI 기반 실시간 운동 코칭 및 통합 헬스케어 플랫폼으로 인공지능이 사용자의 운동 자세를 실시간으로 분석해 교정 가이드를 제공하고, 헬스장 운영에 필요한 관리 기능을 하나로 통합한 서비스입니다.

| 프로젝트 정보

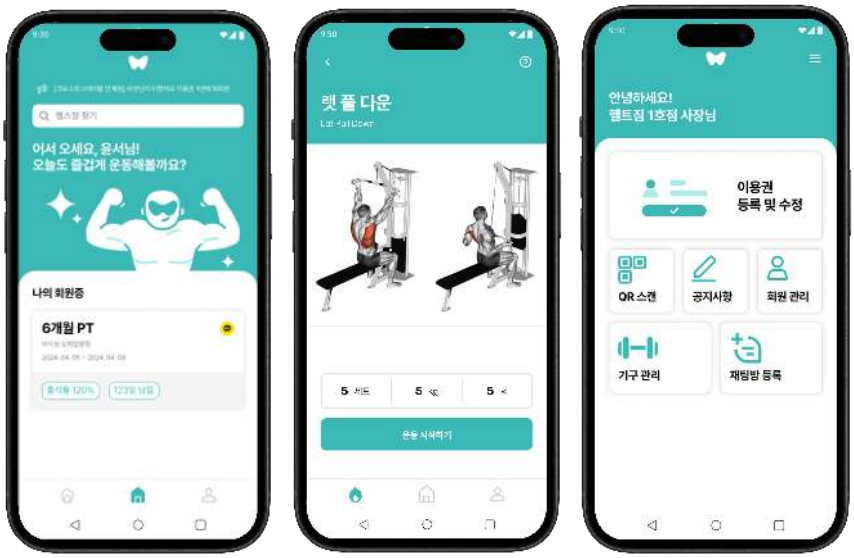
팀 구성: FE(2), BE(2), AI(1)
역할 : Team Leader & Lead Backend Developer

| 기술 스택

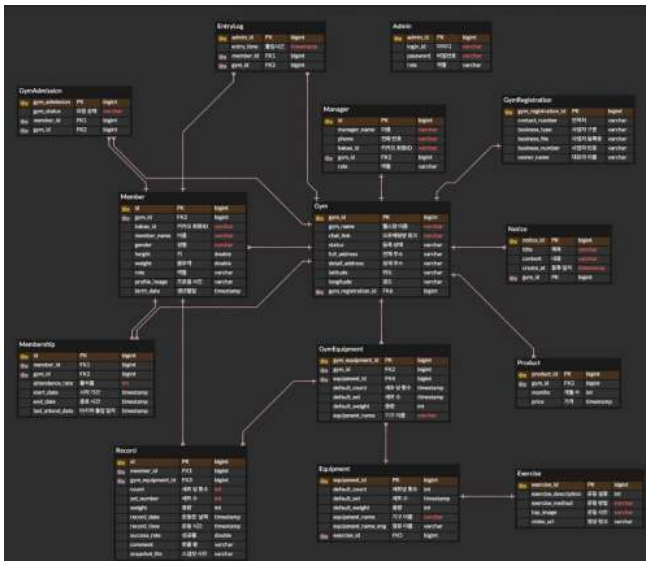
FE
- Kotlin
- Android

BE
- Java
- Spring Boot
- JPA
- QueryDSL

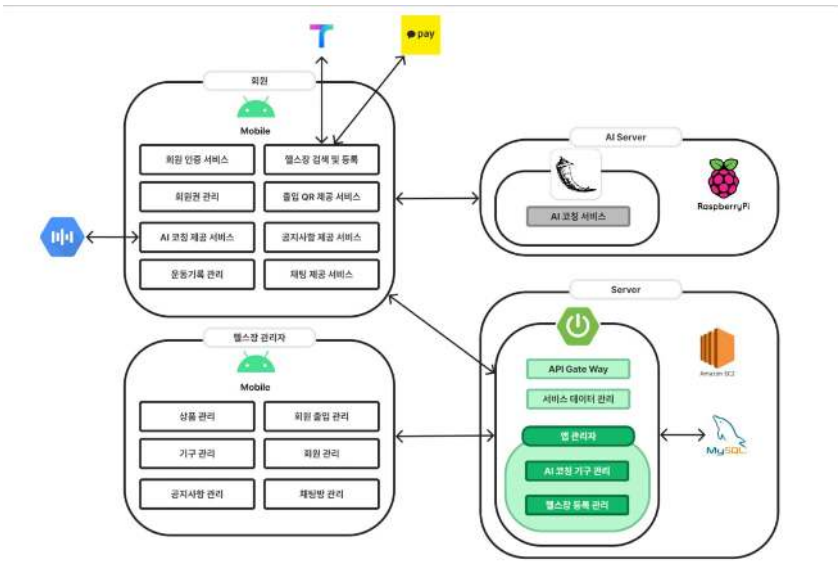
AI
- Python, PyTorch, OpenCV, MediaPipe, Flask



| 설계



[ERD]



[Architecture]

| 주요 기여 사항

- Kakao 소셜 로그인 구현을 통한 Oauth 2.0 프로토콜 이해 및 적용
  - 동일 이메일 사용 시 사용자 계정 통합 로직 설계로 중복 가입 방지
- Spring Security와 JWT를 활용한 무상태(Stateless) 인증 체계 구축
  - 권한 기반 접근 제어 구현
  - 회원/입점 업체/관리자 구분 및 페이지 접근제어
- Querydsl을 활용한 다중 조건 기반의 회원 검색 및 정보 조회 구축으로 출석률, 회원권 잔여 기간 등 복잡한 관리자용 검색 및 필터링 기능 구현
- KakaoPay API의 샌드박스 환경을 활용하여 실무 수준의 결제 프로세스 구현
- Thymeleaf를 활용한 서버 사이드 렌더링(SSR) 기반 관리자 대시보드 구현