OrchestrAI

1. Backend Document

 # OrchestrAI 백엔드 문서

1. 시스템 개요
OrchestrAI는 사용자 친화적인 GUI를 통해 사용자가 여러 AI 에이전트를 생성, 관리 및 조정할 수 있는
웹 앱. 백엔드 시스템은 에이전트 생성, 관리, 통신 및 작업 실행을 처리합니다.

2. 기술 스택
- Language : Python 3.10+ (CrewAI dependency)
- Web Framework : FastAPI
- DB : MongoDB (NoSQL) 사용 여부 체크, 데이터 JSON으로 직접 전달 (MVP 제작 목표)
- Server : Google Cloud Run or AWS
- ORM:
- Main Library : crewAI, LangChain
- 인증 : JWT
- 작업 대기열 : Redis가 있는 Celery
- 테스트 : pytest

3. 아키텍처 설계
OrchestrAI는 마이크로서비스 아키텍처를 따릅니다.
- API 게이트웨이: 들어오는 요청을 처리하고 적절한 서비스로 라우팅합니다.
- 에이전트 서비스: 에이전트 생성 및 구성을 관리합니다.
- 워크플로 서비스 (Crew): 에이전트 워크플로의 생성 및 실행을 처리합니다.

- 최종 결과 : Streamlit or API or Code 생성

(** 핵심 : 워크플로우 컴파일(코드생성), 컴포넌트를 JSON, Yaml 형식으로 표현 **)
(** 데이터 플로우도 위 형식으로 맞춤**)
(** GUI 생성된 구성을 중간표현(IR)형태로 변환 **)
(** IR을 Python code로 변환하는 엔진 개발 **)

- 작업 실행 서비스: 장기 실행 작업을 관리하고 실시간 업데이트를 제공합니다.

- 인증 서비스: 사용자 인증 및 권한 부여를 처리합니다.

4. API 사양

엔드포인트 (현재 예시) :
- `/agents`
- POST: 새 에이전트를 만듭니다.
- GET: 모든 에이전트를 나열합니다.
- `/agents/{agent_id}`
- GET: 에이전트 세부 정보 검색
- PUT: 에이전트 구성 업데이트
- DELETE: 에이전트 삭제
- `/workflows`
- POST: 새 워크플로 생성
- GET: 모든 워크플로 나열
- `/workflows/{workflow_id}`
- GET: 워크플로 세부 정보 검색
- PUT: 워크플로 업데이트
- DELETE: 워크플로 삭제
- `/workflows/{workflow_id}/execute`
- POST: 워크플로 실행

5. 데이터베이스 스키마 (사용할지 판단 필요, MVP는 없어도 되지 않나?)
Main Entity :
- 사용자
- 에이전트
- 워크플로
- 작업
- 결과

6. 인프라 설정
- 컨테이너화: Docker
- 오케스트레이션: Kubernetes
- CI/CD: GitHub Actions
- 클라우드 공급자: AWS, GCR

7. 보안 고려 사항 (추후 고려, 우선 MVP 제작 목표)

- JWT를 사용한 API 인증
- 모든 통신에 HTTPS 사용
- 입력 검증 및 살균
- 속도 제한
- 정기적인 보안 감사

8. 확장성 및 성능
- DALL-E 등 생성 AI 기능을 추가 (실제 적용 해봤음)
- 서비스의 수평적 확장
- Redis를 사용한 캐싱 계층
- 데이터베이스 인덱싱 및 쿼리 최적화
- Celery를 사용한 비동기 작업 처리

9. 테스트 전략
- 개별 구성 요소에 대한 단위 테스트
- 서비스 상호 작용에 대한 통합 테스트
- 중요한 워크플로에 대한 종단 간 테스트
- 고부하 시나리오에 대한 성능 테스트

10. 배포 프로세스
- Firebase Hosting
- GitHub Actions를 사용한 자동화된 배포
- Blue-green 배포 전략
- 롤백 절차
- Prometheus 및 Grafana를 사용한 모니터링 및 알림 설정