향후 개선이 필요한 부분[1006-퀄리저널 최신 개발 방향]

중복 코드 및 설정 통합 - engine_core.py와 orchestrator.py의 중복 유틸리티(키워드 토큰화, 점수 계산, DEFAULT_CFG 등)를 공통 모듈로 추출해 단일한 소스에서 관리하도록 리팩터링을 권장한다. 이는 추후 기능 추가나 버그 수정 시 일관성을 유지하는 데 도움이 된다.

비동기화 및 작업 큐 - README에서 권장하는 대로 asyncio.create_subprocess_exec 또는 FastAPI BackgroundTasks를 사용해 오케스트레이터 실행을 완전히 비동기화하면 API 응답 지연을 최소화할 수 있다. 또한 작업 큐 라이브러리(예: Celery, RQ)를 도입하면 장기 실행 작업에 대한 모니터링·취소 기능을 강화할 수 있다.

구성 스키마 표준화 - config.json, engine_core.DEFAULT_CFG, orchestrator.DEFAULT_CFG 간 불일치를 해소하고, JSON 스키마를 정의해 각 설정 항목과 자료형을 문서화할 필요가 있다. 이렇게 하면 운영자가 파라미터를 잘못 입력했을 때 즉시 검증하고 경고할 수 있다.

보안 강화 - 외부 배포를 준비하려면 JWT/OAuth2를 통한 인증 및 HTTPS 적용이 필수이며, 중요한 API 호출(예: 발행)은 2단계 확인과 감사 로그를 추가해야 한다. FastAPI의 종속성 주입(Dependencies)을 활용해 인증 로직을 중앙화하는 것이 바람직하다.

로그·모니터링 - server와 orchestrator 모두 표준 로깅으로 작업 정보를 남기고, /api/tasks뿐 아니라 UI에서도 로그를 열람하거나 다운로드할 수 있도록 기능을 확장하는 것이 좋다. 장애 대응 시 원인을 파악하는 데 도움이 된다.

테스트 자동화 - 현재 코드에는 단위테스트/통합테스트가 포함되어 있지 않다. 설정 변경이나 리팩터링 이후에도 기능이 유지되는지 보장하기 위해 pytest 기반 테스트 스위트와 GitHub Actions 같은 CI 파이프라인을 도입하는 것이 권장된다.

결론

핵심 파일 간의 연결은 전반적으로 논리적이며 README에서 정의한 목표를 잘 충족하고 있다. 그러나 코드 중복, 동기 실행으로 인한 응답 지연, 보안 미흡 등 기술적 부채가 남아 있어 **리팩터링과 고도화 작업을 진행하는 것이 바람직하다**. 위에서 제시한 개선 과제(공통 모듈 추출, 완전한 비동기화, 설정 스키마 통합, 인증·모니터링 강화)를 계획적으로 도입하면 유지보수성·확장성·보안성을 모두 높일 수 있다.