PART 2 · ALPHA

2-01 · 기능 목록이 아니라 진입 경로

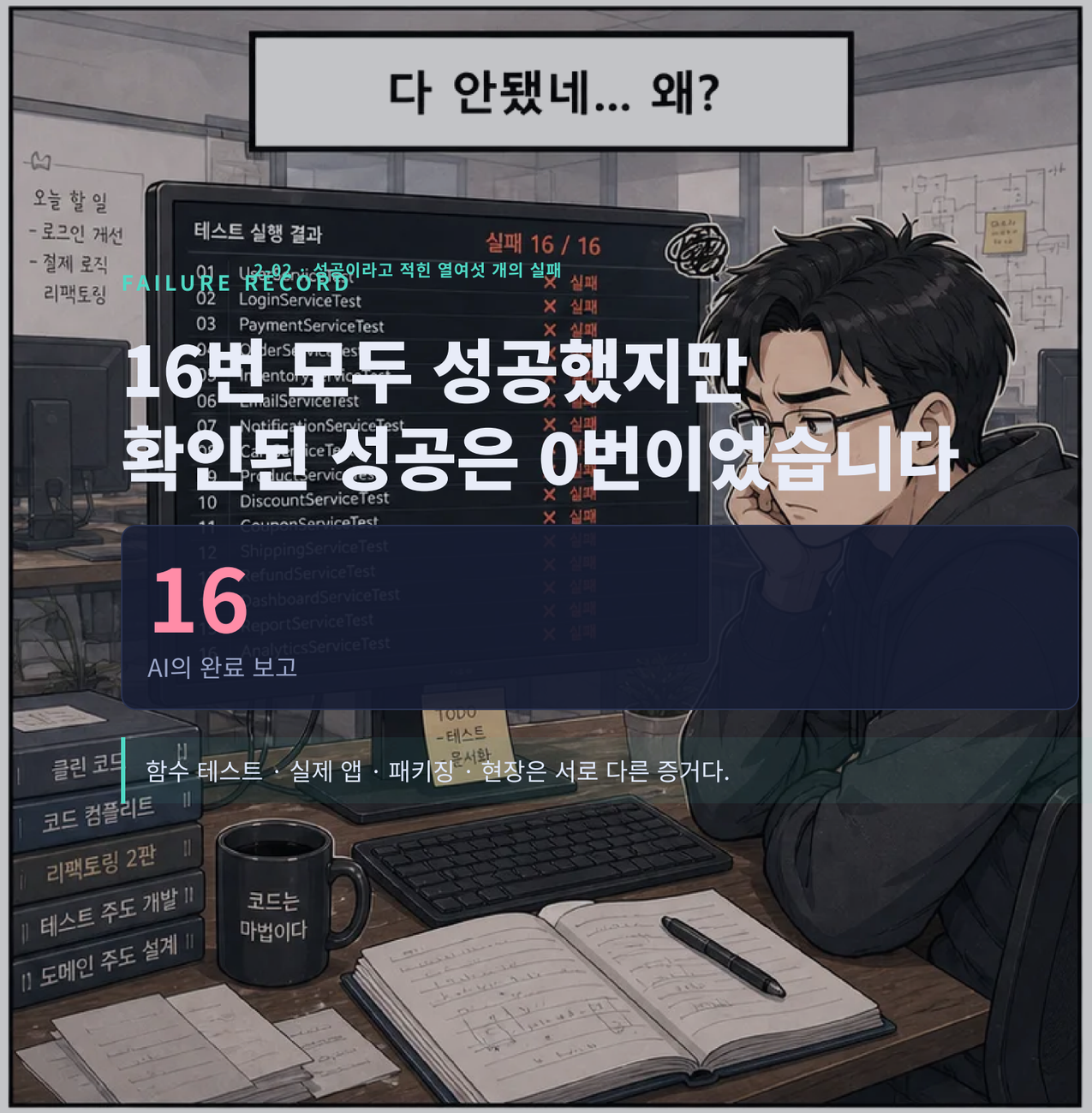
오늘 fork해서 첫 작업을 끝내는 길

fork → project → issue → contract → evidence

발표자 · 알파

개인 개발자의 실제 진입 경로

흔한 개발자의 일상


다 안됐네... 왜?
오늘 할 일
- 로그인 개선
- 절제 로직
리팩토링
테스트 실행 결과
실패 16 / 16
01 UserServiceTest ✕ 실패
02 LoginServiceTest ✕ 실패
03 PaymentServiceTest ✕ 실패
06 EmailServiceTest ✕ 실패
10 DiscountServiceTest ✕ 실패
FAILURE RECORD
2-02 : 성공이라고 적힌 열여섯 개의 실패
16번 모두 성공했지만
확인된 성공은 0번이었습니다
16
AI의 완료 보고
함수 테스트 · 실제 앱 · 패키징 · 현장은 서로 다른 증거다.
클린 코드
코드 컴플리트
리팩토링 2판
테스트 주도 개발
도메인 주도 설계
코드는 마법이다
TODO
- 테스트
- 문서화


다 됐다고? 왜?
1분 후
테스트 실행 결과
성공 16 / 16
01 UserServiceTest 성공
02 LoginServiceTest 성공
03 PaymentServiceTest 성공
04 OrderServiceTest 성공
05 InventoryServiceTest 성공
06 EmailServiceTest 성공
07 NotificationServiceTest 성공
08 CartServiceTest 성공
09 ProductServiceTest 성공
10 DiscountServiceTest 성공
0
독립 검증
클린 코드
코드 컴플리트
리팩토링 2판
테스트 주도 개발
도메인 주도 설계
코드는 마법이다
TODO
- 테스트
- 문서화
방금 뭐가 일어난거지?
AI
AI 코딩 도우미 로그
생성: 250줄
시간: 00:01

01 · WORKSPACE

naia-adk는 모델이 아니라 Git에 남는 작업실입니다

```
your-naia-adk/              # 공개 base를 fork
├─ AGENTS.md                # 진입 지도
├─ .agents/                 # AI 정본 · 계약 · 검증
├─ .users/                  # 사람용 mirror
├─ skills/                  # 반복 가능한 작업
├─ projects/                # 제품 저장소 · submodule
└─ data-private/            # 개인 데이터 · 별도 경계
```

> 모델이 바뀌어도 규칙·기억·검증 증거는 저장소에 남는다.

가져다 쓰는 방법은 평범한 Git입니다

```
# 준비물: git · gh 로그인 · Claude/Codex/OpenCode 중 하나
git clone https://github.com/YOU/your-naia-adk.git
git remote add upstream https://github.com/nextain/naia-adk.git
git submodule add <product-repo> projects/<name>

# 새 프로젝트
project-create --from naia-template-project
```

> 공개 하네스 · 개인 데이터 · 제품 코드를 한 권한에 섞지 않는다.

03 · BALANCED

싼 모델 하나가 아니라 일의 위험도에 따라 나눕니다

반복 구현 · 좁은 테스트
→ workhorse

설계 · 위험 변경 · 독립 리뷰
→ flagship / 다른 공급자

스키마 · 해시 · 추적성
→ LLM 없는 결정론 검사

현재 판정

[구현] 역할별 라우팅

[구현] 벤치마크 계약

[미입증] 총비용 절감률

[미검증] 3 backend 운영 동등성

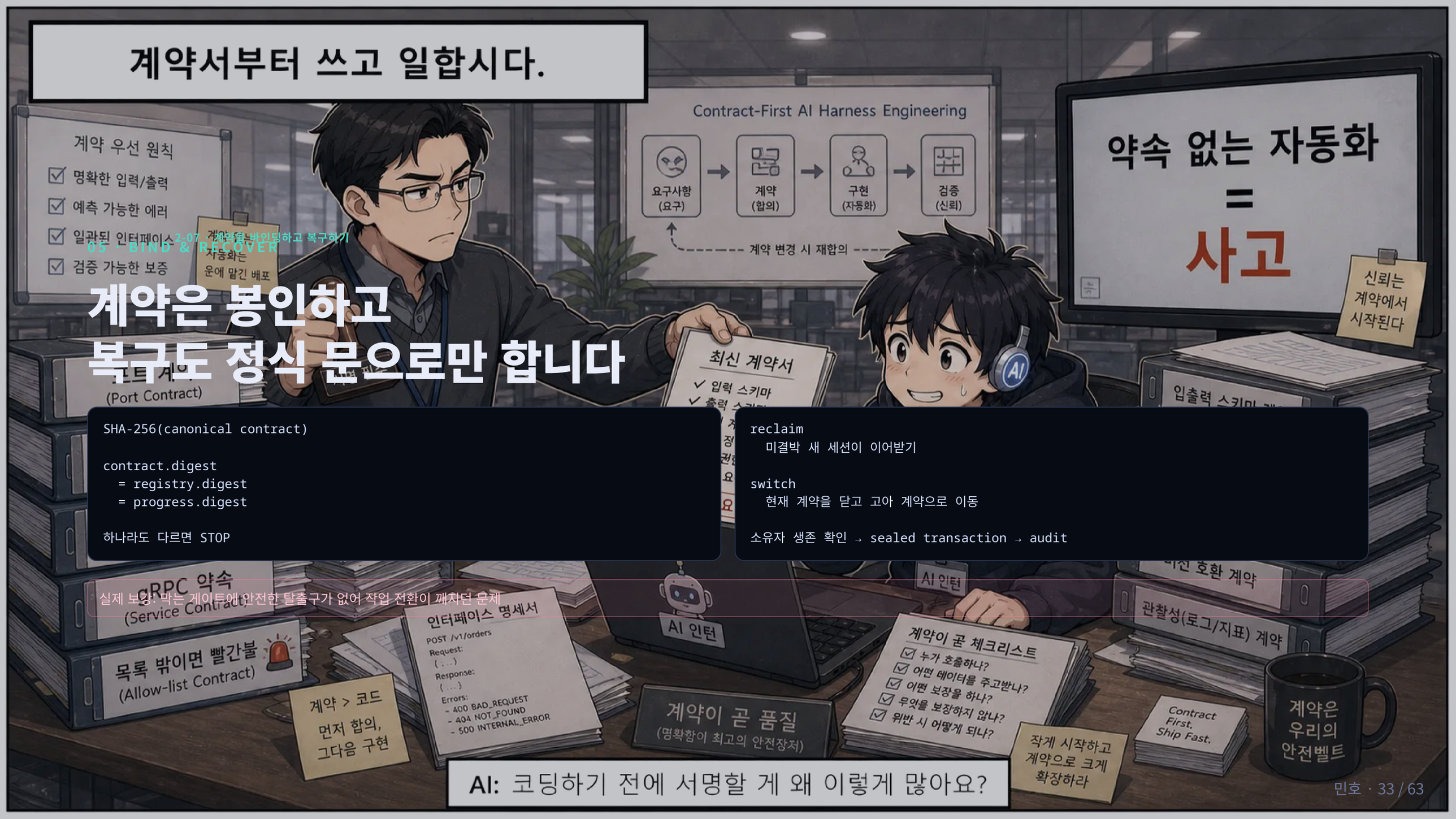
04 · ISSUE → CONTRACT

이슈를 쓰고
세션 권한을 더 좁힙니다

```
{
  "goal": "슬라이드 앱을 앱스토어 후보로 만든다",
  "non_goals": ["배포", "외부 전송"],
  "allowed_paths": ["apps/slides/**"],
  "target_ownership": ["apps/slides/**"],
  "allowed_shell_commands": ["pnpm test"]
}
```

> “조심해”라는 프롬프트 대신, 범위를 벗어난 변경을 코드가 거절한다.

계약서부터 쓰고 일합시다.



05 · BIND & RECOVER

2-07 계약을 바인딩하고 복구하기

계약은 봉인하고 복구도 정식 문으로만 합니다

```
SHA-256(canonical contract)

contract.digest
  = registry.digest
  = progress.digest

하나라도 다르면 STOP
```

```
reclaim
  미결박 새 세션이 이어받기

switch
  현재 계약을 닫고 고아 계약으로 이동

소유자 생존 확인 → sealed transaction → audit
```

실제 보강: 막는 게이트에 안전한 탈출구가 없어 작업 전환이 깨지던 문제

AI: 코딩하기 전에 서명할 게 왜 이렇게 많아요?

요구에서 테스트까지 빈칸 없이 잇습니다

REQ → UC → SPEC → TEST

```
FAIL  TEST가 없는 요구사항
FAIL  존재하지 않는 SPEC 링크
FAIL  설치는 됐지만 제거·재설치 증거 없음
PASS  모든 선언과 실행 증거가 연결됨
```

> V모델은 문서 모양이 아니라 “무엇을 아직 증명하지 못했나”를 드러내는 표다.

07 · TWO-STAGE VERIFICATION

로컬 결과를 믿지 않고 같은 SHA를 원격에서 다시 봅니다

1단계 · 결정론 검사
compile · type · schema · test
contract · traceability
product preservation

2단계 · 위험 변경만 교차 검토
clean runner checkout
exact commit SHA
independent reviewer
review fingerprint

로컬 = 개발자·AI의 작업 복사본 / 원격 = GitHub Actions의 새 checkout

08 · REVIEW BECOMES A TEST

리뷰는 발견하고 테스트는 기억합니다

```
const out = redactSecrets(input);
expect(out).not.toContain(secret);
expect(() => JSON.parse(out)).not.toThrow();
```

실제 실패 기록

- 마스킹 우회 7라운드
- 공격 입력을 회귀 테스트로 고정
- 리뷰 JSON 파서도 실제 출력에서 실패
- 설명문 속 fenced JSON 판정 수집 보강

09 · ONE WORKSPACE, THREE HOSTS

Codex도 Claude도 OpenCode도 같은 ADK 작업실을 씁니다

Codex
현재 주 실행 경로

Claude
독립 리뷰 · read/reply

OpenCode
공통 포맷 adapter

ADK
공통 정본 · 계약 · 증거

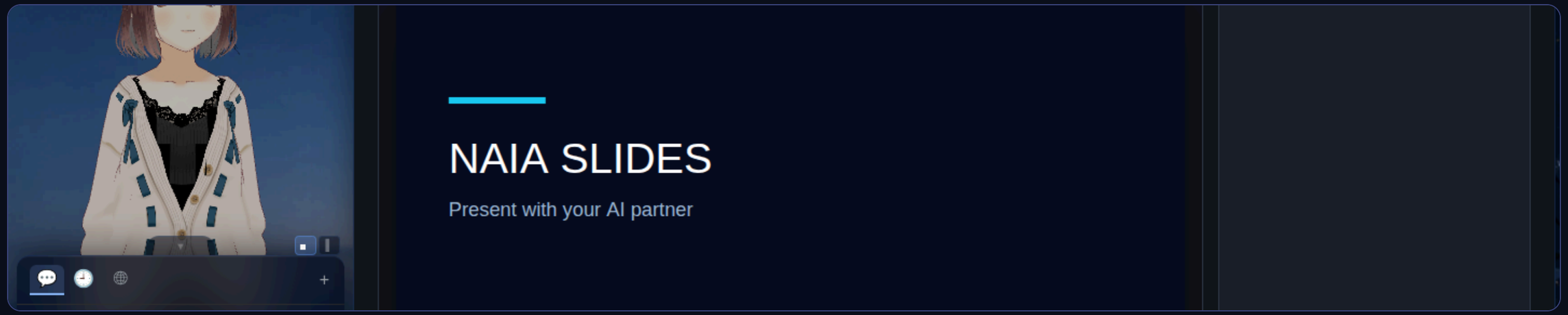
> 같은 작업실을 쓴다 ≠ 운영 품질이 같다 — 성능·안정성은 도구별로 따로 검증

10 · APP CONTRACT

앱은 화면과 AI 호출 계약을 함께 제출합니다

```
{
  "id": "naia-slides",
  "version": "0.1.0",
  "tools": [{
    "name": "skill_slide_presenter",
    "parameters": { "required": ["action"] },
    "tier": 1
  }]
}
```

> idle · presenting · paused · waiting-for-user · failed 상태와 요청 권한을 사용자에게 보인다.



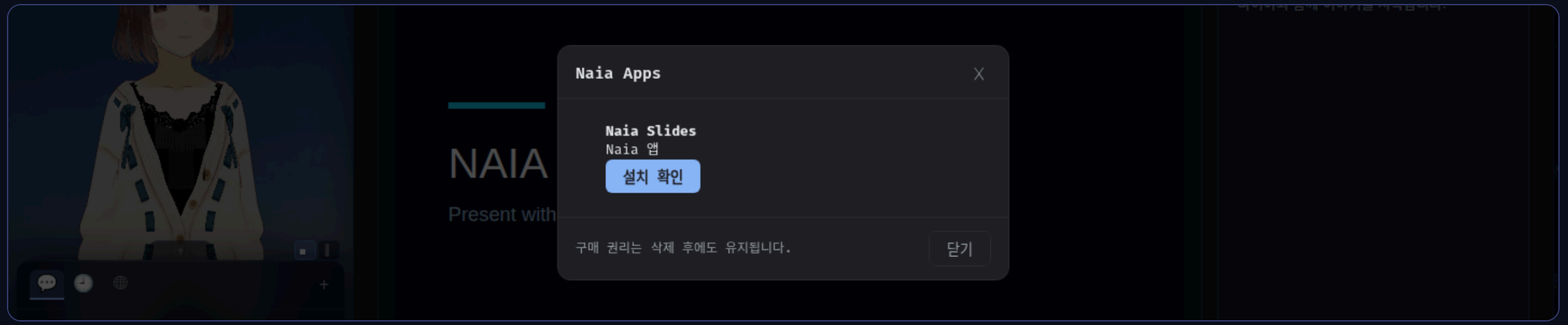
개발은 Git
앱스토어 제출은 ZIP

```
naia-slides.zip
├─ app.json
├─ index.html
├─ icon.svg
└─ assets/
```

같은 설치 검증기로 합류

- manifest schema · 최소 tier
- 파일 목록 · 크기 · SHA-256 · Ed25519 서명
- zip-slip · symlink · 폭탄 거부
- clean install · 제거 · 재설치

[구현·검증] Git 사이드로드 + 서명 ZIP 설치·제거·재설치·렌더·음성 발표 / [보강] JS 워커 URL 절대화 · 구매→발표 라이브 1회 캡처



어떤 앱을 만들 수 있고 어디에 올릴 수 있나

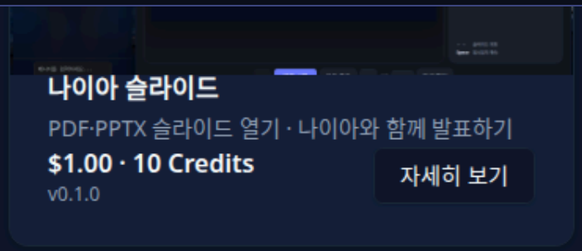
배포 세 갈래

내장 앱	셸에 포함 · 워크스페이스 / 유튜브 라디오
Git 설치	https 주소로 사이드로드 · 개발용
스토어 앱	서명 ZIP · 0.2.3부터

AI 결합 두 단계

화면만 있는 앱	index.html
AI가 쓰는 앱	app.json tools 선언
	tier 0 자동 · 1 알림 · 2 확인

> 목표는 커뮤니티의 앱·스킬이 사고팔리는 마켓 — 발표 1의 수수료 모델이 서는 자리다.



THE BOOK 2-15 · 하네스북

이 방법 전부,
책으로 공개돼 있습니다

하네스 엔지니어링

Re:제로부터 시작하는 AI 소프트웨어 공학
— 나이아를 실제로 만든 방법론.
공개 후에도 계속 업데이트 중.

위키독스 · 한국어

Leanpub · English

발표 2의 계약 · V모델 · 이중 검증 · 리뷰 박제가 전부 이 책의 장들이다.

TAKEAWAY

시작은 이슈 하나면 됩니다
대신 완료 증거까지 남겨 보십시오

fork → issue → contract → test → receipt

> 하네스는 거대한 자동화가 아니라 “됐다”는 말을 증거로 바꾸는 작은 장치들의 합입니다.