

누구나, 안전하게, 끝까지.

대화 한 번으로 만들고, 실제로 운영되는
앱 · 웹 서비스 플랫폼

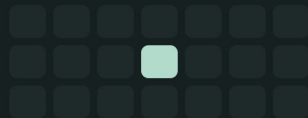
AI를 '부품'으로 쓰는 설명 가능한 End-to-End 구조

필라테스 예약 앱 만들어줘

✓ 배포 완료

필라테스 스튜디오

10월 · 수업 예약



10:00 리포머 · 잔여 3석

19:30 매트 · 잔여 5석

카카오페이로 예약

PROBLEM

AI로 앱을 만드는 시대. 그런데 왜 대부분 '데모'에서 멈출까요?



2025년 '바이브 코딩'은 콜린스 사전 올해의 단어로 선정될 만큼 대중화됐습니다. 하지만 실제 운영되는 서비스는 절반도 안됩니다.

1 어렵다

API 키 · 커넥터 · 서버 · 배포 설정, 한국형 결제와 증빙 서류까지 결국 사용자 몫. 비개발자는 여기서 포기합니다.

2 불안하다

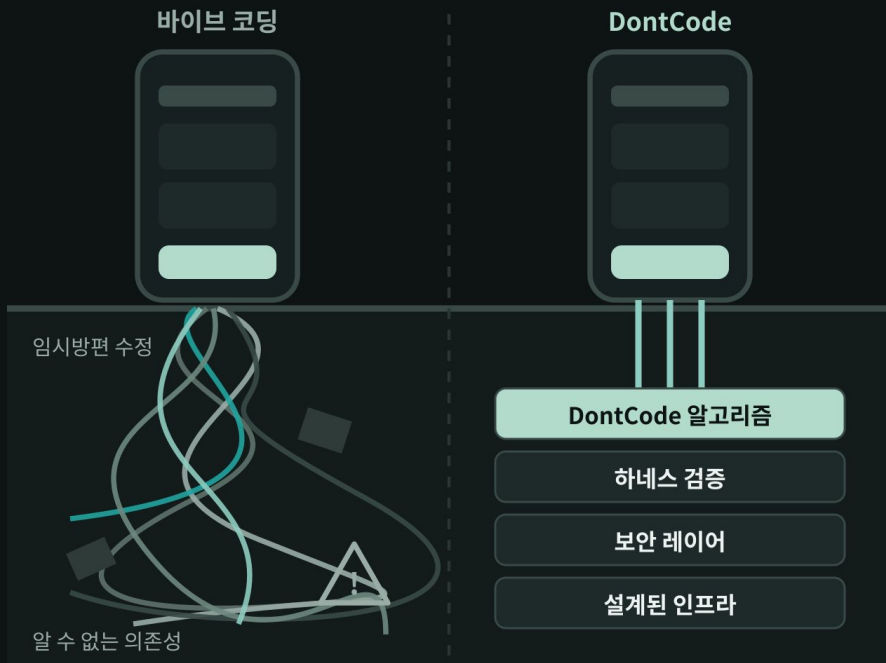
할루시네이션이 검증 없이 그대로 코드가 되고, 데이터 노출 · 인증 취약점이 남습니다.

3 오래 못 간다

작은 수정에도 전체를 다시 생성해 멀쩡하던 기능이 깨집니다. 운영할수록 비용과 불안이 쌓입니다.

→ 만들기는 쉬워졌지만, '운영'까지 건너는 다리는 비어 있습니다.

겉은 같아 보여도, 보이지 않는 곳이 다릅니다



사용자가 보는 앱 화면은 비슷해도, 서비스를 받치는 '땅 아래 구조'가 운영의 성패를 가릅니다.

바이브 코딩

지저분한 코드 · 임시방편 수정 · 알 수 없는 의존성

→ 고칠수록 엉키고, 무너질 위험이 커집니다

DontCode

알고리즘 → 하네스 검증 → 보안 → 설계된 인프라

→ 층층이 정리되어, 필요한 곳만 정확히 고칩니다

→ 이 차이는 '시를 쓰는 구조'에서 나옵니다.

원인은 AI의 성능이 아니라, AI를 쓰는 '구조'에 있습니다

기존 AI 빌더
AI = 두뇌

프롬프트



AI가 전부 생성 (블랙박스)



결과물

결과를 설명할 수 없고, 검증할 지점이 없으며, 부분만 고칠 수도 없습니다.

DontCode
AI = 부품

대화



자체 알고리즘이 지휘 · AI는 플러그인



검증 · 보안 후 배포

모든 단계가 정의되어 있어 설명 · 검증 · 부분 수정이 가능합니다.

→ 이 구조 하나가 DontCode의 세 가지 약속을 가능하게 합니다.

DontCode의 세 가지 약속

1

누구나

대화 한 번이면 운영 가능한 서비스까지. 한국형 결제 · 서류도 기본 탑재.

2

안전하게

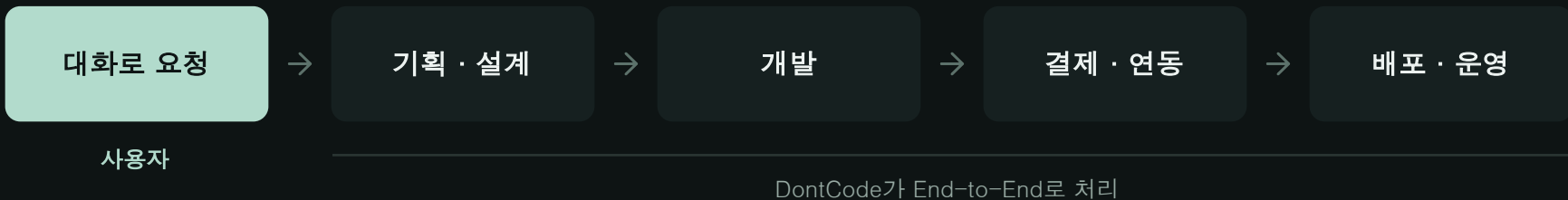
AI는 부품, 통제는 알고리즘.
하네스 검증과 보안이 구조에 내장.

3

끝까지

필요한 부분만 정밀하게 고치는 사후관리. AI가 발전할수록 함께 좋아지는 서비스.

대화 한 번이면, 운영 가능한 서비스까지



✓ 이미 익숙한 방식

ChatGPT · Claude · Gemini처럼
대화하듯 입력하면 끝. 새로 배워야 할
툴이 없습니다.

✓ 연동 걱정 제로

컨넥터 · 플러그인 · 서버 · 도메인
설정을 사용자가 신경 쓸 필요가
없습니다.

✓ 데모가 아닌 실제 서비스

결과물은 체험판이 아니라, 바로
고객을 받을 수 있는 앱과 웹입니다.

→ 개발을 모르는 소상공인 · 1인 창업자 · 남녀노소 누구나 고객이 됩니다.

한국 비즈니스에 필요한 결제와 서류까지, 처음부터

한국형 결제 전면 지원

고객이 원하는 방식으로 바로 결제

- 신용 · 체크카드
- 카카오페이 · 네이버페이 · 토스페이
- 계좌이체 · 가상계좌
- 휴대폰 결제

서류 업무 자동화

확장 기획 — 사업 운영에 필요한 행정까지

- 세금계산서 발행
- 현금영수증
- 거래 · 정산 내역
- 약관 · 개인정보처리방침

→ 글로벌 AI 빌더가 풀지 못하는 로컬 장벽이, 곧 DontCode의 진입장벽이 됩니다.

AI는 부품으로, 통제는 알고리즘이

사용자 요청

DontCode 오케스트레이션 알고리즘

AI 플러그인 (교체 가능)

하네스 검증

데이터 보안 레이어

운영 서비스

✓ AI 발전을 그대로 흡수

더 좋은 모델이 나오면 해당 플러그인만 교체. 빅테크의 모델 경쟁이 우리에게선 비용이 아닌 성장 동력입니다.

✓ 할루시네이션의 구조적 통제

단계마다 입력 · 출력이 정의되어 있어 검증이 가능하고, 잘못된 결과는 다음 단계로 넘어가기 전에 차단됩니다.

✓ 추적 가능한 유지보수

워크플로우 전체가 알고리즘으로 정의되어 있어, 업데이트와 수정 과정을 논리적으로 설명할 수 있습니다.

고객에게는 즉시, 공격자에게는 어렵게



하네스 엔지니어링

- ✓ AI 출력을 검증 · 제약하는 정교한 안전장치
- ✓ 정해진 설계 틀 안에서만 코드 생성
- ✓ 누가 만들어도 일관된 코드 품질

데이터 보안 엔지니어링

- ✓ 필요한 정보만, 필요한 사람에게 즉시
- ✓ 설계 단계부터 보안 내장 (Security by Design)
- ✓ 공격 표면을 최소화한 구조

45%

AI 생성 코드의 보안 테스트 실패율 — 더 크고 새로운 모델에서도 개선되지 않았습니다.

보안은 모델 성능이 아니라 '구조'로 해결해야 합니다. DontCode가 AI 바깥에 검증 · 보안 레이어를 둔 이유입니다.

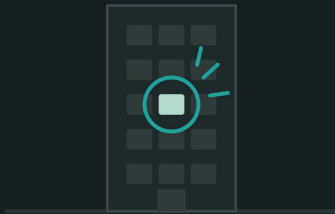
만든 다음이 진짜 시작 — 정밀한 사후관리



일반 AI 빌더

창문 하나 고치려다, 건물 전체를 다시 짓습니다

→ 멀쩡하던 기능까지 깨지고, 처음부터 재검증



DontCode

고장 난 창문 하나만 정확히 교체합니다

→ 기존 구조 그대로, 즉시 반영

✓ 핀포인트 수정

부족한 기능 · 고객 불편사항만 정확히

✓ 구조 유지 리모델링

틀을 유지한 채 전체 UI 리뉴얼도 가능

✓ 개발자 노하우 알고리즘

체험판이 아닌 장기 운영 전제의 설계

AI가 발전할수록, 모든 고객의 서비스가 함께 좋아집니다

새로운 AI 모델 출시



해당 플러그인만 교체



운영 중인 모든 서비스 품질 상승

그리고 이것이 만드는 선순환

알고리즘 기반 구조



높은 품질 · 보안



운영 고객의 신뢰



반복되는 사후관리 매출



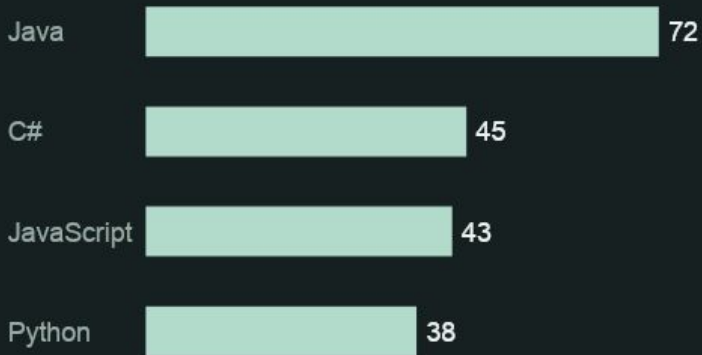
운영 노하우 축적

노하우가 다시 알고리즘을 강화

→ 경쟁사가 모델 성능에 기댈 때, DontCode는 모델의 발전을 자산으로 흡수합니다.

데이터로 확인한 '만들기만 하는 AI'의 한계

AI 생성 코드의 보안 테스트 실패율 (언어별)



전체 평균 45% · 모델이 커져도 개선되지 않음 [1]

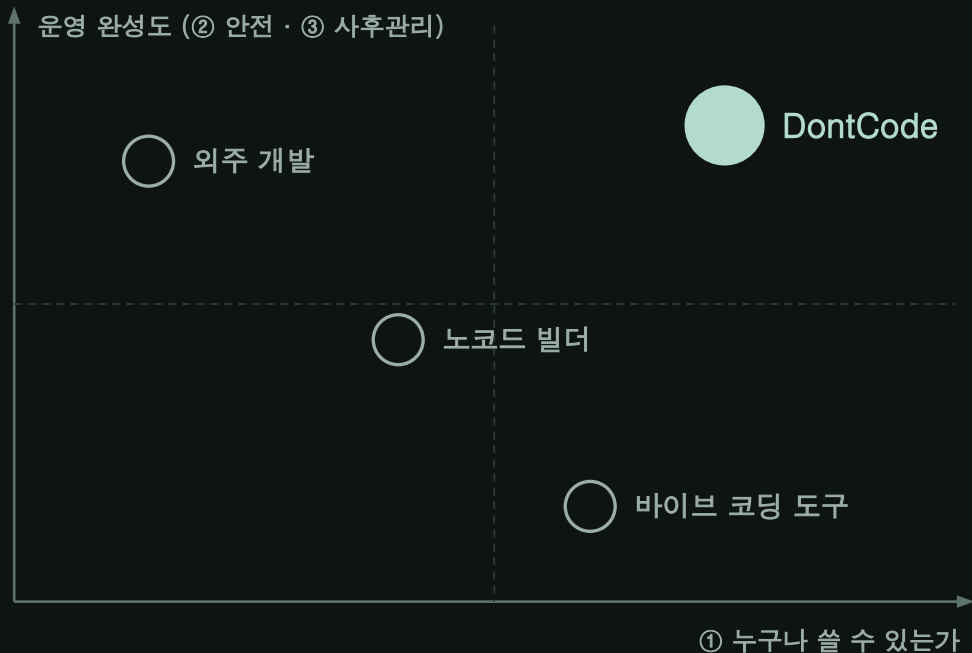
170 / 1,645

한 바이브 코딩 플랫폼으로 만든 앱 가운데, 사용자 데이터가 노출될 수 있는 취약점이 발견된 앱 수 [2]

66%

개발자가 꼽은 AI 도구 최대 불만: '거의 맞지만, 완전히 맞지는 않은' 결과 [3]

한눈에 보는 차이



외주 개발

완성도는 높지만 긴 기간 · 높은 비용

노코드 빌더

템플릿 범위 안에서, 연동은 사용자가 직접 조합

바이브 코딩 도구

보안 테스트 실패 45% · 데이터 노출 사례 (p.12)

DontCode

대화만으로 시작 + 운영 가능한 완성도

시장은 이미 움직이고 있습니다

70%

2025년까지 기업이 새로 개발하는 앱의 70%가 로우코드 · 노코드 기술을 사용할 것 (2020년 25% 미만)

출처: Gartner 전망 — '코드 없이 만드는' 수요는 이미 주류가 되었습니다.

DontCode 성과 (실제 수치 입력 필요)

[____]

가입자 수

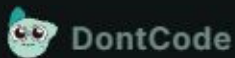
[____]

제작된 앱 · 웹

[____ %]

사후관리 유지율

수익 구조 제작 → 사후관리 구독 → 결제 · 서류 수수료 (반복 매출) 시장 규모 TAM · SAM · SOM [____]



AI로 '만들어 주는' 회사는 많습니다.
DontCode는 '운영하게 해 주는' 회사입니다.

1 누구나

2 안전하게

3 끝까지

투자 요청

[____ 억 원]

자금 용도

[알고리즘 고도화 · 보안 인증 ·
고객 확보]

목표

[18개월 내 ____]