기술

캔버스에 표 하나 그렸을 뿐인데: 노코드 제조 플랫폼이 DB·API·화면을 알아서 만드는 이야기

방동걸 (주)웨이스 대표이사
2026년 6월 21일

제조 시스템 프로젝트를 몇 번 해 보면 어느 순간 기시감이 옵니다. 새 고객사 미팅에 들어가 요구사항을 듣는데, 머릿속에서 이미 같은 그림이 그려져요. 또 테이블을 설계하겠구나, 또 CRUD API를 짜겠구나, 또 입력 화면이랑 조회 화면을 만들겠구나. 솔직히 개발자 입장에서 이게 제일 김 빠지는 순간입니다. 새로운 문제를 푸는 게 아니라, 지난번에 했던 걸 살짝 모양만 바꿔서 또 하는 거니까요. VEXPLOR를 만드는 웨이스 개발자들도 제조 시스템을 구축하며 똑같이 느껴 온 순간입니다.

저희가 던진 질문은 그래서 좀 삐딱했습니다. "이 반복, 매번 손으로 하는 게 맞나?" 새 솔루션을 만들 때마다 테이블 설계하고, 그 위에 API 얹고, 그 위에 화면 올리는 이 일은 사실 패턴이 거의 똑같거든요. 다음 고객, 다음 모듈에서 또 같은 일을 합니다. 그리고 이 반복이 그냥 시간 낭비로만 끝나면 차라리 다행인데, 시스템마다 구조가 조금씩 어긋나게 쌓이면서 결국 데이터 사일로가 됩니다. 나중에 이걸 다시 잇느라 또 비용이 들고요.

오늘은 그 얘기를 해 보려고 합니다. 저희가 이 반복을 어떻게 제품으로 접어 넣었는지, 그러니까 SI(매번 처음부터 다시)가 아니라 OS(한 번 잘 만든 토대 위에서 확장)로 어떻게 방향을 틀었는지요. 결론부터 말하면, 캔버스에 스키마 하나를 그리는 것으로 그 아래가 줄줄이 따라 나오게 만들었습니다.

01

사람이 코드를 쓰는 대신, 모델 하나만 그리게 했습니다

VEXPLOR 노코드 플랫폼은 이 OS의 커널에 해당합니다. 그리고 이 커널이 하는 핵심 일은 한 문장으로 요약됩니다. **캔버스에서 스키마를 설계하면 DDL·API·UI가 자동 생성**됩니다.

좀 더 풀어 볼게요. 이건 메타데이터 주도(model-driven) 개발이라고 부르는 방식의 한 형태입니다. 발상이 뒤집혀 있어요. 보통은 사람이 코드를 직접 쓰고, 그 코드가 곧 시스템이 됩니다. 메타데이터 주도에서는 사람이 모델, 그러니까 스키마를 단 하나의 원천으로 두고, 거기서 산출물을 뽑아냅니다. 코드는 결과물이지 원본이 아닌 거죠.

이게 왜 좋으냐면, 손으로 짠 시스템에서 제일 흔하게 터지는 사고를 원천에서 막아 주기 때문입니다. 한번 떠올려 보세요. 테이블에 컬럼 하나 추가했는데, API 고치는 걸 깜빡하고, 화면은 또 다른 사람이 건드려서

셋이 서로 안 맞는 상황. 다들 한 번씩 겪어 보셨을 겁니다. 모델이 단일 원천이면 이런 일이 안 생깁니다. 모델을 바꾸면 DB도, API도, 화면도 같이 바뀌니까요. 세 곳을 따로 고치다 어긋나는 문제 자체가 사라지는 셈입니다.

여기에 한 가지 더해진 장치가 있습니다. 모델링에 쓰는 어휘를 표준화해 뒀어요. **노드 타입 37종, 데이터 타입 12종**으로만 스키마를 구성합니다. 빌딩블록을 한정해 둔 건데, 이게 답답해 보일 수 있지만 사실 정반대입니다. 정해진 블록으로만 짓기 때문에 누가 그리든 생성 결과가 들쭉날쭉하지 않고 일관됩니다. 레고가 아무 모양이나 다 만들 수 있는 건 오히려 블록 종류가 정해져 있어서인 것과 비슷해요.

02

스키마 한 장에서 세 층이 동시에 나옵니다

그래서 캔버스에 스키마를 하나 완성하면, 그 아래로 세 층이 한꺼번에 떨어집니다.

- **DDL** — 표준 구조의 테이블이 생성됩니다. 그게 쌓이고 쌓여 지금은 **834+ 표준 테이블**이 됐습니다.
- **API** — 표준 CRUD와 로직 엔드포인트가 같이 나옵니다.
- **UI** — 입력 화면과 조회 화면이 만들어집니다. 플랫폼 전체로 보면 **900+ UI 스크린**, 그리고 현장 작업자가 직접 쓰는 **POP 화면 82개**가 이런 식으로 구성됩니다.

여기서 끝이 아닙니다. **워크플로우 145+**, **문서 템플릿 96종**이 표준 자산으로 같이 딸려 옵니다. 그래서 새 솔루션을 시작할 때 "맨땅"에서 출발하지 않아요. 이미 깔려 있는 이 표준 위에서 필요한 걸 조립하는 식으로 일이 진행됩니다. 0에서 1을 만드는 게 아니라 80에서 100을 채우는 느낌에 가깝습니다.

03

만들고 끝이 아니라, 현장 시스템까지 굴러갑니다

여기서 "생성까지는 알겠는데, 그래서 그게 실제로 돌아가는 시스템이 되긴 하는 거야?"라는 의심이 드실 수 있습니다. 당연한 반론이에요. 코드가 자동으로 나온다고 해서 그게 곧장 현장에서 쓰이는 건 또 다른 얘기니까요. 생성과 배포 사이에는 보통 긴 골짜기가 있습니다.

그래서 VEXPLOR는 생성 다음 단계까지 자동화해 뒀습니다. **6단계 자동 배포 파이프라인**이 설계 결과를 실제로 동작하는 시스템까지 자동으로 내보냅니다. 스키마를 바꾸면 그게 코드 생성으로, 검증으로, 배포로 쭉 이어지는 흐름인데요. 이 발상 자체는 사실 새롭지 않습니다. 애플리케이션 개발에서 익숙한 CI/CD와 같은 원리거든요. 다만 그걸 애플리케이션 코드 레벨이 아니라 스키마·메타데이터 레벨에 적용했다는 게 포인트입니다.

한 흐름으로 따라가 볼까요. 현장 엔지니어가 캔버스를 열고 설비 점검 이력을 다룰 스키마를 그립니다. 노드 타입에서 적당한 걸 골라 점검 항목, 설비, 작업자를 배치하고 관계를 잇습니다. 저장을 누르는 순간, 표준

구조의 점검 이력 테이블이 DDL로 떨어지고, 그 테이블을 읽고 쓰는 CRUD API가 같이 생기고, 점검 결과를 입력하는 화면과 이력을 조회하는 화면이 만들어집니다. 여기까지가 한 호흡이에요. 그리고 이어서 6단계 파이프라인이 이 결과를 검증하고 배포해서, 다음 날 현장 태블릿에서 그 점검 화면이 그냥 떠 있습니다. 캔버스에서 그린 모델이 며칠 안에 현장에서 쓰는 시스템이 되는 거죠. 중간에 누가 테이블을 짜고, API를 붙이고, 화면을 코딩하는 그 긴 구간이 통째로 접혀 들어간 겁니다.

04

이게 왜 표준화로, 또 AI로 이어지나

노코드로 만든 결과물이 그냥 "돌아가는 시스템"인 것만으로는 사실 절반의 얘기입니다. 진짜 핵심은 그 결과물이 곧 **표준 구조**라는 데 있어요. 아무렇게나 생긴 시스템이 아니라, 정해진 빌딩블록으로 정해진 패턴을 따라 생성된 구조라는 거죠. 그래서 재미있는 일이 벌어집니다.

- 시스템을 도입하는 순간 **Day 1 Ready** 상태가 됩니다. L1~L3에 해당하는 AI 토대가 자동으로 깔려요.
- 그 위에 온톨로지(관계)와 4-Way RAG(지능)가 올라탑니다. 그림으로 그리면 이렇습니다. 커널(노코드)이 표준 데이터를 깔고, 파일시스템(온톨로지)이 그 데이터들 사이를 관계로 잇고, 인텔리전스(RAG)가 그 위에서 추론합니다. 이게 VEXPLOR라는 OS의 3개 층입니다.

저희가 입버릇처럼 "표준화 없이 AI 없다"고 말하는데, 그 첫 단추가 바로 이 자동 생성 엔진입니다. AI를 잘 만드는 게 어려운 게 아니라, AI가 올라설 만큼 깨끗하고 표준화된 데이터를 까는 게 어렵거든요. 보통은 그걸 사람이 한참 정리해야 하는데, VEXPLOR는 시스템을 만드는 행위 자체가 표준 구조를 까는 행위라서 그 단계를 건너뜁니다. 그래서 AI가 바로 올라설 토대가 생기는 거예요.

05

그럼 남들이 그냥 따라 만들면 되지 않나

여기까지 들으면 "스키마 그려서 코드 뽑는 거, 다른 데서도 만들면 되는 거 아냐?" 싶으실 수 있습니다. 충분히 나올 법한 생각이에요. 그래서 솔직하게 답해 보겠습니다. 어디까지는 따라 할 수 있고, 어디부터는 어려운가.

겉으로 보이는 UI는 비교적 빨리 흉내 낼 수 있습니다. 그런데 아래로 내려갈수록 깊이가 확 달라져요. 저희가 6-Layer Data Moat로 정리해 둔 걸 보면 이렇습니다.

- **L1 UI/UX** — 3개월
- **L2 API/로직** — 6개월
- **L3 데이터 구조(834+ 테이블)** — 1년+

화면 몇 개를 베끼는 거랑, 표준 데이터 구조와 그걸 찍어 내는 생성 엔진 전체를 갖추는 건 완전히 다른 일입니다. 진짜 자산은 위쪽 화면이 아니라 아래쪽 구조에 있어요. 그리고 이 구조는 사 올 데가 없습니다. 시간을 들여 쌓는 수밖에 없거든요. 이게 "SI 아닌 OS"라는 말의 실질적인 의미입니다. 반복되던 일을 제품으로 접어서, 그 위에서 계속 확장 가능하게 만들어 둔 것.

06

그래서, 같이 풀 사람을 찾습니다

노코드 생성 엔진을 만드는 일에는 의외로 머리 아프고 재미있는 난제가 잔뜩 깔려 있습니다. 모델에서 코드를 뽑을 때 일관성을 어떻게 보장할지, 스키마가 진화하면(마이그레이션 말이죠) 기존 데이터를 어떻게 안 깨고 따라가게 할지, 자동 생성된 결과물의 성능과 품질을 어디까지 책임질지, 6단계 파이프라인을 라인 안 세우고 무중단으로 배포하려면 뭐가 필요한지. 그리고 37종 노드와 12종 데이터 타입이라는 표준 빌딩블록을, 표현력은 충분하면서도 너무 복잡해지지 않게 어떻게 균형 잡을지. 하나하나가 단순한 구현 문제가 아니라 설계 철학이 걸린 문제입니다.

VEXPLOR는 제조 AI OS를 만드는 (주)웨이스의 제품입니다. 위 얘기 읽으면서 "이거 한번 붙어 보고 싶은데" 싶으셨다면, 아마 잘 맞으실 겁니다.

- 관심 분야: 플랫폼/메타데이터 주도 엔진 / 코드 생성(DSL·codegen) / 데브옵스(자동 배포 파이프라인) / 풀스택
- 더 알아보기: 채용·회사 소개 https://wace.me · 기술 문의 contact@wace.me

WACE(웨이스)는 공장이 스스로 판단하고 멈추지 않게 하는 제조 AI 운영체제 VEXPLOR를 만드는 회사다.

원문 blog.vexplor.com/post/nocode-schema-autogen
발행 2026년 6월 21일 · 방동걸 (주)웨이스 대표이사

회사 주식회사 웨이스 (WACE Inc.) · 대표이사 방동걸 · 사업자등록번호 439-87-02852
주소 (18462) 경기 화성시 동탄영천로 150, 에이동(A동) 1702-1708호 (영천동, 실리콘앨리동탄)
연락처 전화 070-4730-0629 · contact@wace.me
홈페이지 wace.me · linkedin.com/company/wace-inc

원문 홈페이지 링크드인