현장에서 얻은 것 No. 23

나만의 AI 에이전트 필살기 II – 코드를 이해하는 기획자, 비개발자의 바이브 코딩 입문기

> "거인의 어깨 위에 올라서서 더 넓은 세상을 바라보라."
> – 아이작 뉴턴

AI라는 거대한 변화의 파도는 우리 삶 곳곳을 흔들고 있었다. 이는 단순히 새로운 기술의 등장이 아니라, 사고방식과 일하는 방식, 나아가 사회 전체의 구조를 바꾸는 흐름이었다. 필자는 지난 8개월 동안 이 변화의 흐름 속에서 매일 배우고 실험하며 자신만의 여정을 이어갔다. 이 시간 동안 AI를 단순한 도구로만 보지 않게 되었는데, 그것은 업무, 창작, 학습, 그리고 삶 전반을 통해 스스로를 끊임없이 자극하는 동반자였다. AI를 맹목적으로 신뢰하기보다는 신중하게 거리를 두고, 동시에 적극적으로 받아들이는 태도를 통해 자신만의 '필살기'를 다듬어왔다.

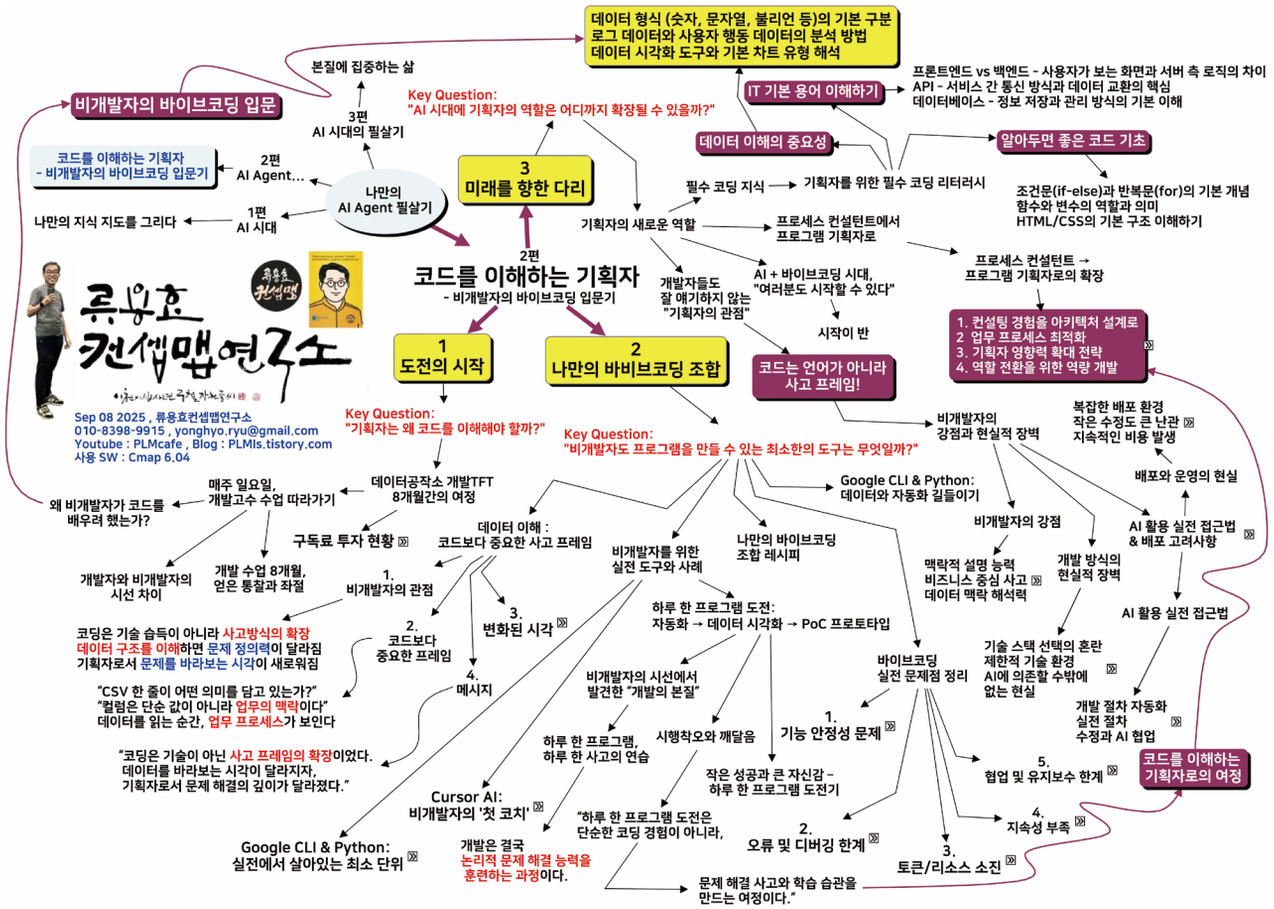
필자의 학습법은 눈으로 익힌 것이 70%, 손으로 부딪히며 체득한 것이 30%로 다소 독특했다. 이러한 비율을 받아들인 이유는 필자의 경험이 개발자의 삶이 아니었기 때문이었다. '바이브 코딩(vibe coding)'을 통해 비개발자도 개발을 할 수 있다고 광고했지만, 실제로는 한계가 있음을 이해했다. 커서 AI(Cursor AI)로 회사 홈페이지를 만들고, 리플릿(Replit) 프로그램으로 MBTI 판별 프로그램을 바이브 코딩으로 시도하며, 만들고 수정하는 것도 가능했다.

하지만 PLM을 기업에 구축하는 PM으로서 경험한 바로는, 비개발자가 프로그램을 만드는 데에는 한계가 있었다. 취미로 만드는 것은 환영하지만 프로그램이 론칭된 이후 발생하는 많은 이슈를 경험하며, 개발자와의 협업이 더 효율적이라는 자신만의 학습 공식을 터득했다. 강의와 책, 스터디에서 얻은 지식이 토대가 되었고, 실습과 시행착오가 그 지식을 현실과 연결해 주었다.

이부일 대표의 강의를 들으며 챗GPT를 활용한 파이썬 코드를 직접 따라가던 순간, AI가 단순한 언어 모델이 아니라 강력한 실무 도구라는 사실을 처음 체감했다. 첫날은 잘 따라갔지만 둘째 날 노트북 배터리가 나가 낭패를 본 기억도 생생했는데, 이러한 경험조차도 학습 과정의 일부가 되었다. AI 학습은 지식을 머리에 담는 것뿐만 아니라 삶과 환경 속에서 몸으로 받아들이는 과정임을 깨달았다. 실패와 해프닝도 자산이 되어 필자의 학습 지도 위에 하나씩 좌표가 찍혀갔다. 중요한 것은 속도가 아니라, 끊임 없이 배우고 기록하고 다시 활용하는 과정이 훨씬 값지다는 것이었다.

류용효

디원의 상무이며 페이스북 그룹 '컨셉맵연구소'의 리더로 활동하고 있다. 현업의 관점으로 컨설팅, 디자인 싱킹으로 기업 프로세스를 정리하는 데 도움을 주며, 1장의 빅 사이즈로 콘셉트 맵을 만드는데 관심이 많다.
이메일 Yonghyo.ryu@gmail.com
블로그 PLMIs.tistory.com



▲ 코드를 이해하는 기획자, 비개발자의 바이브 코딩 입문(Map by 류용효)

> "미래는 예측하는 것이 아니라 상상하는 것이다."
> – 앨런 케이

비개발자가 코드를 배우려 했던 이유

필자가 비개발자로서 코드를 배우기 시작한 동기는 개인적인 필요에서 비롯되었다. PLM 구축 PM으로서 개발자와 같은 언어로 소통하고 싶었고, 프로세스 컨설팅을 수행하며 시스템/프로세스 흐름을 실제 코드 레벨에서 검증하고 싶었다. 또한 콘셉트맵과 AI를 접목하여 아이디어를 프로토타입 코드로 구현하고, 데이터 및 AI 기반으로 확장하고자 했다. 바이브 코딩을 통해 손쉽게 프로토타입을 직접 만들어 아이디어를 빠르게 실험하고 싶었던 것도 큰 동기였다.

일반적인 경우에도 비개발자가 코드를 배우는 다양한 이유가 있었다. 반복적이고 단순한 작업을 효율화하여 업무를 자동화하고, 데이터 구조를 직접 다루어 인사이트를 도출하며 데이터 이해력을 강화하는 것이었다. 개발자와의 협업 과정에서 기술적 언어를 이해하여 소통을 원활하게 하고, 아이디어를 직접 테스트하고 시각화하여 창의적 문제 해결 능력을 키우는 데에도 코딩이 필요했다. 또한 디지털 리터러시와 융합 역량을 확보하여 커리어를 확장하고, AI 툴 활용의 전제 조건인 코드 이해를 통해 AI 시대에 적응하고자 했다.

결론적으로, 비개발자가 코드를 배우는 이유는 개발자가 되기 위해서가 아니라 아이디어를 직접 다루고, 빠르게 실험하며, 더 나은 협업자이자 창의적 문제 해결자가 되기 위함이었다. 개발자와 비개발자의 시선 차이는 명확했는데, 개발자는 '코드와 로직을 어떻게 짤까'에 집중하고 성능, 안정성, 기술적 가능성에 관심을 두는 반면, 비개발자는 '왜 이게 필요한 걸까'에 집중하며 사용성, 효율, 비즈니스 가치를 중요하게 생각했다. 예를 들어, 같은 CSV 데이터를 보더라도 개발자는 데이터의 구조와 처리 방법을, 비개발자는 그 데이터가 무엇을 말해주고 경영 의사결정에 어떻게 쓰일지에 대한 의미와 활용 방법을 보았다.

> "가장 현명한 사람은 계속해서 배우는 사람이다."
> – 소크라테스

나만의 바이브 코딩 조합 : 작은 성공에서 배운 것들

AI와 바이브 코딩 시대에 기획자의 새로운 역할이 중요하게 부각되었다. 바이브 코딩은 2025년 2월 안드레이 카르파티가 처음 언급한 개념으로, 코드 작성보다는 '원하는 결과물의 느낌(바이브)'을 AI에게 자연어로 설명하여 프로그래밍하는 방식이었다. 이는 코드 작성 능력이 창의력과 기획 능력으로 전환되는 트렌드를 반영했다. 비개발자를 위한 AI 개발 방법론은 문제 정의, PRD(제품 요구 문서) 작성, AI 프롬프팅, 그리고 결과 검증의 단계로 이루어졌다. 기획자는 문제 정의와 사용자 경험에 집중하고, AI와 대화하며 요구사항을 구체화하고 결과물을 정제하며, 빠른 프로토타입으로 아이디어를 시각화하고 개선점을 파악하는 데 주력했다.

필자는 8개월간의 여정 속에서 자신만의 AI 활용법, 즉 '필살기'를 만들어갔다. 이는 단순히 나열된 여러 갈래의 길이 아니라, 하나의 지도 위에 유기적으로 연결되어 있었다. AI는 단순히 도구가 아니라 이 지도를 함께 그려가는 협력자가 되었다. 필자의 AI 필살기는 다음과 같았다.

■ **커서 AI :** 비개발자의 '첫 코치' 역할을 했다. 코딩의 벽을 낮춰주는 동반자로, 복잡한 문법, 오류, 환경 설정의 두려움을 덜어주었다. 커서 AI는 단순한 코드 자동 생성이 아니라 필자의 의도를 코드로 번역하여 작은 실험과 반복을 가능하게 했고, 바이브 코딩 학습을 지원했다. GPT-4 기반의 AI 코드 에디터로 비주얼 스튜디오 코드(VS Code)와 호환되며, 자연어로 코딩하고, 즉각적인 에러 수정, 단계별 설명, 코드 리팩토링 기능을 제공했다.

■ **구글 CLI(Google CLI) :** 데이터와 시스템을 다루는 새로운 무기였다. 클릭 대신 명령어로 반복 작업을 자동화하여 속도와 효율성을 극대화했다. 가상머신(VM), 스토리지(Storage), 데이터베이스(DB) 등 클라우드 리소스를 제어하고, 데이터를 핸들링하며, API를 직접 호출하여 서비스 통합을 용이하게 했다. 이는 GUI의 한계를 넘어서는 전문가의 무기가 되었다.

■ **파이썬(Python) :** 실전에서 가장 유용한 최소 단위였다. 쉽고 직관적인 문법, 방대한 라이브러리, 빠른 프로토타이핑이 강점이었다. 데이터 읽기/쓰기 한 줄, 간단한 자동화 스크립트 등 작은 코드로도 큰 효과를 낼 수 있었고, CSV 분석 및 시각화, 업무 자동화, AI·ML 모델 실험 등에 활용되었다. 커서 AI와 제미나이(Gemini)가 내장되어 더 쉽게 사용할 수 있었다.

이러한 도구들을 조합하여 데이터 분석 자동화 시나리오와 업무 자동화 봇 구축 시나리오를 구현할 수 있었다. 예를 들어, 커서 AI로 데이터 수집 스크립트를 작성하고, 파이썬으로 데이터 정제 및 시각화를 하며, 구글 CLI로 정기적 실행을 스케줄링했다.

무엇보다 데이터 이해는 코드보다 중요한 사고 프레임이었다. 코딩은 기술 습득이 아니라 사고방식의 확장임을 깨달았다. 데이터 구조를 이해하면 문제 정의력이 달라지고, 기획자로서 문제를 바라보는 시각이 새로워졌다. CSV 한 줄이 어떤 의미를 담고 있는지, 칼럼이 단순한 값이 아니라 업무의 맥락임을 이해하게 되면서, 데이터를 읽는 순간 업무 프로세스가 보이기 시작했다. 이러한 변화된 시각은 단순 결과물이 아닌 흐름과 원인을 질문하게 했고, 개발자와 같은 언어로 협업 및 설계를 가능하게 하며, 데이터 기반의 빠른 실험과 검증으로 이어졌다.

필자는 매일 새로운 프로그램에 도전하는 '하루 한 프로그램 도전기'를 통해 작은 성공을 쌓아갔다. 완벽함보다는 경험과 시행착오를 통한 학습을 강조했고, 개발의 본질이 사고의 연습임을 깨달았다. 즉, 코드는 도구일 뿐 핵심은 문제를 정확히 이해하고 구조화하는 능력이며, 실패는 학습이고 작은 성공이 쌓여 성장 곡선을 만든다는 것이었다. 끊임없이 배우고 기록하고 다시 활용하는 과정이 훨씬 값지다는 것을 체감했다.

그러나 바이브 코딩에는 현실적인 문제점도 있었다. 새로운 기능을 추가할 때 기존 기능이 손상되는 회귀 테스트 부재 문제, AI가 전체 맥락을 충분히 기억하지 못해 발생하는 기능 안정성 문제가 있었다. 무한루프나 잘못된 로직 생성, 에러 메시지 오해 등으로 인한 오류 및 디버깅 한계, 그리고 수정 과정에서 토큰/리소스를 과다하게 소비하는 문제도 발생했다. 세션이 바뀌거나 컨텍스트가 길어지면 AI가 이전 코드의 세부 흐름을 잊어버리는 지속성 부족 문제와, AI에 의해 산발적으로 작성된 코드가 구조화가 부족하여 협업 및 유지보수가 어렵다는 한계도 있었다. 이러한 문제를 경험하며 코드를 이해하거나 개발자와 협업하는 것이 필수라는 결론에 도달했다.

"성공의 비결은 기회를 잡기 위해 준비하는 것이다."
– 벤저민 디즈레일리

미래를 향한 다리 : 기획자의 새로운 역할

AI 시대에 기획자의 역할은 크게 확장될 수 있었다. 비개발자의 강점은 데이터 맥락 해석력, 비즈니스 중심 사고, 그리고 맥락적 설명 능력에 있었고, 이는 CSV 데이터 컬럼의 의미와 관계를 명확하

게 설명하고, 로직보다 비즈니스 가치와 목적에 집중하며, 기술적 디테일보다 전체적인 흐름과 맥락을 설명하는 커뮤니케이션 역량을 제공했다.

프로세스 컨설턴트에서 프로그램 기획자로의 역량 확장이 필요했다. 컨설팅 경험을 시스템 아키텍처 설계에 적용하고, 업무 분석 능력을 시스템 요구사항으로 전환하며, 사용자 관점과 시스템 관점의 통합을 통해 더 나은 UX(사용자 경험)를 설계하는 것이었다. 현업 부서와 IT 부서 간의 가교 역할을 수행하고, 업무 프로세스 최적화를 통해 비효율 지점을 발견하고, 시스템 병목 현상을 데이터 흐름 관점에서 해결하는 역량이 중요했다. 컨설팅 산출물을 소프트웨어 명세서로 변환하고 워크플로 시뮬레이션으로 최적화를 검증하는 방법이 요구되었다.

기획자는 기술 이해도를 바탕으로 개발팀과의 협상력을 강화하고, 데이터 기반의 의사결정 모델을 구축하며, 비즈니스와 기술을 잇는 통합적 관점을 제시하고, 프로토타입으로 아이디어를 구체화하는 능력을 확보해야 했다. 이를 위한 역량 개발로는 시스템 사고, 기술 리터러시(API, DB 구조, 클라우드 서비스 기본 개념), 애자일 방법론, 그리고 지라(Jira), 피그마(Figma), 미로(Miro)와 같은 협업 도구 활용 능력이 있었다. 기획자와 개발자의 경계를 허물고 함께 문제를 정의하고 해결하는 통합적 협업 체계를 구축하는 것이 중요했다.

> "나는 똑똑한 것이 아니다. 단지 문제와 더 오래 씨름할 뿐이다."
> – 알베르트 아인슈타인

AI의 본질은 '주체'가 아니라 '도움'이었다. AI는 망설임 없이 실행하지만, 그것이 옳은 방향인지 판단하는 것은 인간의 몫이었다. 필자는 회의록 요약 같은 업무를 AI에 맡겼다가 보안 문제와 인간 역량 퇴화의 위험성을 깨달았다. 편리함이 언제나 효율을 의미하는 것은 아니며, 잘못된 의존은 인간의 중요한 능력을 잃게 만들 수 있었다. 그래서 필자는 AI의 답변을 최소 세 번 이상 검증했는데, 빠른 실행보다 올바른 방향 설정이 중요했기 때문이었다. AI가 주는 답은 끝이 아니라 출발점이었다.

필자가 AI와 함께한 여정은 자신을 끊임없이 질문하게 했다. AI는 인간을 대체하는 기계가 아니라, 인간이 더 깊은 사고와 창조의 세계로 들어가도록 돕는 동반자였다. 필자가 찾은 필살기는 바로 이것이었다. AI 덕분에 자신의 본질(core)에 더 많은 시간을 쏟을 수 있게 된 것이었다. 단순 반복 업무를 대신해 주는 AI 덕분에, 필자는 사고하고 기획하고 판단하는 인간 고유의 역량에 집중할 수 있었다. AI는 더 이상 선택이 아닌 필수 도구이자 협력자였다.

중요한 것은 이 강력한 도구를 어떻게 나의 본질과 연결하여, 나만의 고유한 가치를 창출하고 미래를 만들어갈 것인가에 대한 깊은 고민과 끊임없는 실행이었다. AI는 재능은 있지만 한계에 부딪힌 사람에게 '도움'이 되어 AI 가수, AI 영화감독, AI 작가, AI 프로그래머가 될 수 있는 길을 열어주었다. 효율만을 쫓기보다는 본질에 집중하고, 변화의 흐름을 읽으면서도 자신만의 '필살기'를 계속해서 갈고 닦아야 했다. 미래를 향한 첫걸음은 지금 바로 도전하는 것이었다.

바이브 코딩은 기획 의도와 개발 실행 사이의 간극을 해소하고, AI 시대 기획자의 역할 확장과 가능성을 발견하게 해주었다. 업무 자동화로 반복 작업에서 벗어나 창의적 업무에 시간을 활용하고, 데이터 기반의 의사결정과 인사이트 도출 능력을 강화할 수 있었다. 하루 30분, 한 프로그램 만들기로 시작하는 것이 중요했고, 완벽함보다는 시작하는 용기가 중요했다. 하지만 잊지 말아야 할 것은, 바이브 코딩의 장단점을 잘 파악하여 적용해야 한다. 특히 개인적인 사용의 간단한 프로그램은 괜찮으나, 대외적인 서비스를 하는 프로그램 개발의 경우, 반드시 고급 개발자의 코드리뷰를 거쳐서 보안상의 문제, 데이터 유출 등이 없도록 해야 한다. AI는 명확하게 정의된 문제를 푸는 데 능숙하지만, 복잡하고 모호한 비즈니스 요구사항을 해석하여 견고한 시스템을 설계하는 것은 못하는 것을 명심해야 한다.

> "코딩은 기술이 아닌 사고 프레임의 확장이다."