박철련

frontendtt@gmail.com

[문제를 바라보는 관점]

“왜 이 구조여야 하는가”를 먼저 묻는 개발자

개발을 이어오며 저는 기능을 구현하는 일보다,
이 기능이 어떤 맥락에서 어떤 사용자에게 왜 필요한가를 먼저 정의하는 과정을
더 중요하게 여겨 왔습니다.

처음에는 **탄탄한 구조와 적절한 기술**을 선택하는 것이
개발자의 가장 중요한 역할이라고 생각했습니다.

하지만 경험이 쌓일수록 사용자는 어떤 기술이 쓰였는지보다
어떤 흐름으로 **서비스를 경험**하고, 어떤 **상황에서 그것을 사용**하는지를
더 핵심적으로 바라본다는 사실을 깨닫게 되었습니다.

겉으로 완벽해 보이는 구조도 사용자의 실제 맥락을 반영하지 못하면
기대와 전혀 다른 결과를 만들어냅니다.

이 과정을 반복해 겪으면서, 기술 선택의 가치는 기술 자체가 아니라
사용자·환경·제약을 얼마나 **정확히 해석**했는가에 의해 결정된다는 점을 분명히 알게 됐습니다.

이후로 저는 **구조적으로 훌륭한 코드**와 **적절한 기술 선택**이 **‘정답’**이 되는 순간은
그 선택이 **사용자의 상황과 흐름을 온전히 이해**한 뒤 내려졌을 때라는 기준을 가지게 되었습니다.

개발자의 유지보수 관점, 사용자의 경험, 실제 환경의 제약을 함께 바라보고
그 **교차점**에서 기술을 선택할 때 비로소 서비스가 가장 자연스럽게 완성된다는 것을 배워왔습니다.

그래서 저는 새로운 프로젝트를 시작할 때마다 **“어떤 기술을 쓸까?”**가 아니라
“이 서비스는 어디에서 사용되고, 사용자는 무엇을 기대하며, 어떤 흐름이 가장 자연스러운가?” 라는
질문을 먼저 던지는 습관을 갖게 되었습니다.

함께 문제를 정의하고, **사용자의 패턴을 관찰**하며, **기술·환경·제약을 함께 고려**해
가장 현실적인 해법을 선택하는 과정은 저를 기능을 만드는 사람이 아니라
서비스의 맥락을 이해하고 **기술 선택의 이유**를 설명할 수 있는 개발자로 성장하게 해주었습니다.

[실사용 서비스 운영에서 배운 책임감]

“기능은 만들기보다 유지하는 것이 더 어렵다”

‘양치킹’ 프로젝트는 저에게 가장 깊은 문제해결 경험을 남긴 서비스입니다.
학교 **10곳·6천 명 이상이 실제로 사용**하는 시스템을 혼자서 개발·배포·운영까지 해야 했습니다.

처음 직면한 문제는 기능 부족이 아니라 **운영 환경의 예측 불가능함**이었습니다. 학교 단말기마다 다른 제약, 불안정한 네트워크, 다양한 사용자의 패턴등의 다양한 문제들을 풀기 위해
저는 기능 구현보다 **환경 분석 → 문제 정의 → 지속 가능한 구조 설계의 흐름에 집중**했습니다.

EC2 기반 서버 구성, PM2·GitHub Actions를 통한 배포 자동화, 학교 현장의 단말 환경에 맞춘 설치형 앱 제공까지 직접 설계하며, **“기능의 완성도”**보다 **“운영의 완성도”**가 서비스의 수명을 결정한다는 사실을 마음 깊이 체감했습니다.

이 경험은 단순히 코드를 잘 짜는 개발자가 아닌
서비스를 끝까지 책임지는 개발자로 성장하는 계기가 되었습니다.

[부딪히며 성장한 실전 사례들]

“팀 안에서 일하는 방식을 처음 알게된 경험 – Art-bonbon”
Art-bonbon은 제가 처음 팀에 들어가 협업의 흐름을 배운 프로젝트였습니다.
팀의 컨벤션을 맞추고 기획·디자인팀과 각자의 의도를 공유하며,
단순한 기능의 구현보다 팀 전체가 유지할 수 있는 구조가 더 중요하다는 사실을 배웠습니다.
PR 리뷰를 통해 코드의 방향성을 합의하는 경험을 하며,
이후 모든 프로젝트에서 문제 정의와 기준 설정의 감각을 갖게 되는 계기가 되었습니다.

“운영 흐름을 방해하던 반복 업무를 자동화한 경험 – Ecole-Admin”
관리자들이 매일 엑셀을 수작업으로 입력하는 불편함을 해결하기 위해
업무 패턴과 오류 지점을 먼저 분석했습니다.
CSV 파싱, Zod 검증, Server Action을 결합해 등록 시간을 크게 줄였고
운영 흐름을 방해하던 반복 업무를 실질적으로 개선하는 해결책을 만들었습니다.

“레거시 환경의 불안정함을 구조적으로 해소한 경험 – Tool-Manager”
Vue2 레거시 환경에서는 상태 불일치와 UI 동기 문제가 반복되고 있었습니다.
저는 오류의 근본 원인을 찾기 위해 기존 흐름을 추적하며 문제를 재정의했고,
React 기반으로 구조를 다시 설계했습니다.
Dnd-Kit 인터랙션, Zustand 타입 정합성, 프리셋·색상·레이어 구조를 안정화하며
레거시 문제를 구조적으로 해소하는 방식을 체득한 프로젝트이기도 했습니다.

“기존 서비스를 맡아 개선 포인트를 찾아낸 경험 – School-Teacher”

School-Teacher는 제가 처음부터 개발한 서비스가 아니라,
운영 중인 프로젝트를 넘겨받아 유지·개선해야 하는 과제였습니다.
처음 코드를 인수받았을 때, 학교 Wi-Fi 환경의 불안정함 때문에 무한스크롤 방식이 자주 흐름을 끊고 있다는 사용자 불편이 눈에 들어왔습니다. 저는 기술적인 원인을 먼저 추적하기보다 “사용자가 어떤 상황에서 불편을 느끼고, 무엇을 더 편하게 만들 수 있을까”를 기준으로 개선 포인트를 정의했습니다.

그 관점에서 Alive Ping + Dead Count 방식의 네트워크 감지 구조를 추가하고,
데이터 요청 방식을 페이지네이션으로 전환해 환경 변화에도 안정적인 조회 흐름을 만들었습니다.

기존 코드를 그대로 유지하는 것이 목적이 아니라, 사용자가 실제로 겪는 문제를 중심에 두고 서비스의 약한 지점을 찾아 개선하는 경험이었고, 프론트엔드가 사용자 경험을 가장 가까이에서 지켜내는 역할이라는 사실을 다시 확인한 프로젝트였습니다.

[앞으로의 방향]

“문제를 정의하고 기준을 세울 줄 아는 개발자“

여러 프로젝트를 거치며 저는 문제 해결은 코드를 쓰는 순간이 아니라,
기준을 세우는 순간부터 시작된다는 결론에 도달했습니다.

사용자의 경험, 운영 흐름, 네트워크 환경, 팀의 작업 방식 등 여러 요소를 함께 이해해야 기능 하나라도 “올바른 방식”으로 구현할 수 있다는 점을 몸으로 배웠습니다.

그 과정에서 저는 아직 더 성장해야 할 부분이 많다고 느끼고 있습니다. 기술적인 선택에 대한 근거를 명확하게 설명하기 위해 JavaScript의 코어 개념, React·Next.js의 등장 배경과 내부 구조 같은 기술의 역사와 철학을 공부하며 **제가 사용하는 기술**이 **어떤 문제를 해결하기 위해 만들어졌는지**를 꾸준히 파고들고 있습니다.

이런 공부는 단순히 기술을 더 잘 쓰기 위한 목적이 아니라, **기술이 탄생한 맥락을 이해**하는 개발자만이 **타인이 읽기 편한 코드**와 **일관된 구조**를 만들고, **사용자 입장에서 근거 있는 기술 선택**을 할 수 있다고 믿기 때문입니다.

또한 저는 개인의 스타일보다 **팀이 합의한 기준**이 더 큰 가치를 갖는다는 점을 깨닫게 되었습니다.
프로젝트를 진행하며 기획자·디자이너·운영자와 **함께 문제를 정의**하고, **모두가 같은 맥락에서 이해할 수 있는 구조**를 만들어가는 과정, 그리고 PR 리뷰를 통해 서로 더 나은 방향을 찾기 위해 의견을 조율하는 과정은 저에게 개발 자체만큼 중요한 **협업의 근본**이라고 느껴졌습니다.

앞으로도 저는 **문제를 정확히 정의**하고 **기준**을 세우며,
팀이 함께 유지할 수 있는 구조를 만들어가는 개발자로 성장하고 싶습니다.

그리고 **사용자와 팀 모두가 신뢰할 수 있는 개발자**로 남고 싶습니다.