**프로그래밍을 시작한 경위와 본격적으로 투신하고 생업으로 삼기로 결정한 이야기를 해주세요.**

[시작]

제 시작은 매우 평범하고 재미가 없습니다.

성적에 맞춰 대학교와 학과를 결정했고, 그렇게 흘러온 컴퓨터과학과에서 프로그래밍을 처음 시작하였습니다.

프로그래밍은 재미없지도, 열렬히 재미있지도 않았습니다. 어떤 것이 프로그래밍 역량에 도움이 되는지는 고민하지 않고, 전공과목 이론 공부에만 충실하여 좋은 성적을 받는 것이 목표인 학부생활이었습니다.

[회고]

프로그래머가 되고 싶다는 것을 처음 자각한 것은 모순적이게도 첫 직장에 취업한 이후입니다. 저는 졸업 이후 대기업 SI 업체에 취업했고 1년간 소프트웨어 운영 및 유지보수 업무를 담당하였습니다.

이 회사에서는 납기에 맞춘 빠른 기능 구현만이 최고의 가치였고, 가장 높은 우선순위였습니다. 효율적인 로직이나 클린 코드와 같은 작업에 대한 의식과 니즈가 없었습니다.

핵심 로직 변경을 피하고자 수차례 덧붙여진 코드는 점점 크기가 커지고, 복잡도 역시 증가하였습니다. 하지만 요구사항 및 코드 분석 이전에 먼저 정해져 내려오는 납기를 준수하기 위해서는 저 역시도 기능 구현에만 초점을 맞춰 개발해야 했습니다.

본능적이게도 이러한 업무를 매우 싫어하였습니다. 쓰레기더미 위에 제 손으로 직접 쓰레기를 쌓는 기분이었습니다.

[진짜 시작]

‘만약 개발 역량이 뛰어나다면, 이러한 환경일지라도 주어진 납기 안에 효율적인 코드로 리팩토링 가능하지 않을까.’ 하는 생각으로 시간을 쪼개어 공부하기 시작했습니다. 대학 시절에는 심심했던 공부가 흥미롭고 재미있게 다가왔습니다.

점차 코드의 품질을 생각하는 진짜 개발자가 되는 것을 꿈꾸고 있었고, 현재 회사가 되려 목표 달성에 저해된다고 판단하여 퇴사를 결심하였습니다.

퇴사 이후 현재 ‘코드스쿼드’라는 교육기관에서 교육을 받고 있습니다. 이곳에서 하는 개발은 저를 몰입하게 하였습니다. 객체 지향적 설계, 가독성 높은 코드를 작성하기 위한 욕심이 생겼습니다.

현재 제 목표는 저를 몰입하게 하는 개발을 하며, 높은 품질에 대한 욕심으로 끊임없이 리팩토링하는 ‘진짜 개발자’가 되는 것입니다.

**좋은 개발자가 되기 위해 갖추어야 한다고 생각하는 덕목 셋을 고르고 그 이유를 말해주세요. (예를들어 책임감, 꼼꼼함, 유연성 등등)**

첫째, ‘이해력’ 입니다.

좋은 개발자가 되기 위해서는 새로운 기술을 이해하고 습득하여 적용하는 능력이 필수적으로 수반되어야 한다고 생각합니다.

웹 애플리케이션 자체가 최근 10년 정도 사이에 새롭게 확산된 것이고, 개발 기법도 계속 발전하고 있기 때문에, 이해력이 곧 성장동력이라고 생각합니다.

둘째, ‘지속성’ 입니다.

‘Slow and steady wins the race’

달리기에서 빠른 것은 중요하지만 승리하기 위한 필수조건은 아닙니다. 승리하기 위한 필수조건은 완주하는 것입니다.

위 문단에서 작성하였듯, 새로움이 끊이지 않는 업계의 특성상 IT 개발자는 평생 공부해야 하는 직업이라고 생각합니다. 오랜 기간 좋은 개발자로 남기 위하여, 습관처럼 공부할 수 있는 지속성이 중요하다고 생각합니다.

셋째, ‘나태’입니다.

> Perl의 설계자인 Larry Wall은 프로그래머가 가져야 할 3가지 자질을 나태, 조바심, 자만심이라고 소개했다.

‘코딩을 지탱하는 기술’이라는 책에서 위와 같은 문장을 읽었습니다. 이 3가지 자질 중 저는 ‘나태’라는 자질에 크게 공감하였습니다.

나태한 개발자는 노동력을 줄이기 위해 프로그램을 만들게 되고, 이 프로그램은 다른 사람들도 사용하게 되고 편리함을 제공합니다.

또한, 나태한 개발자가 그 프로그램에 관한 질문에 일일이 답하는 수고를 덜기 위해 문서를 만들게 되고, 이는 모든 사용자를 편리하게 합니다.

이러한 편리함을 추구하기 위하여, ‘나태’라는 덕목을 갖추어야 한다고 생각합니다

**지금까지 읽었던 개발서 중 가장 중요하다고 생각하는 책 3 권을 골라 선정한 이유를 말해주세요**

[훌륭한 프로그래머 되는 법]

이 책은 ‘개발자용 자기계발서’라고 생각합니다. 선배가 새로 시작하는 개발자에게 조언하는 듯한 느낌을 많이 받았습니다.

좋은 코드를 작성하는 방법부터, 팀원들과 협업하는 방법까지. 단순히 개발뿐 아니라 인간관계에 이르기까지 훌륭한 프로그래머가 되기 위한 비법들을 담았습니다.

가장 인상 깊었던 부분은 ‘주석’에 대한 부분이었습니다.

> 모든 주석이 코드에 가치를 더하지 않는다는 것을 명심하자.

> 주석 처리로 코드를 제거하지 마라.

[프로가 되기 위한 웹 기술 입문]

컴퓨터과학과를 전공하였지만, 웹 기술에 대해서는 제대로 교육받지 못했다고 생각했었습니다.

이 책은 웹 기술을 배우기 어려운 이유에 대한 글로 시작합니다. 이 시작은 매우 논리적이고 공감되었습니다. 이후 웹의 역사와 용어 전반에 대해 알기 쉽게 설명되어 있습니다. 필요한 지식이 방대한 웹 개발자의 기본을 닦는 중요한 책이라고 생각합니다.

[코딩을 지탱하는 기술]

위 ‘프로가 되기 위한 웹 기술 입문’이 웹의 역사를 다룬다면, 이 책은 주로 코딩의 역사를 다루고 있습니다. 존재하는 기술이 왜 존재하는지에 대한 물음을 해결하는 데에 초점이 맞춰져 있습니다. 이 책의 시작 부분에 학습에는 중요한 세 가지 포인트가 있다고 서술되어 있습니다.

> 1. 비교를 통한 학습.

> 2. 역사를 통한 학습.

> 3. 만드는 것을 통한 학습.

‘코딩을 지탱하는 기술’ 책을 통해 역사를 통한 학습의 깊이를 더했다고 생각합니다.

**기술 경향을 파악하고 업무 역량을 강화하기 위한 본인의 학습 방법을 구체적으로 설명해주세요.**

가장 좋은 방법은 ‘코드 리뷰’ 라고 생각합니다.

타인의 코드를 보는 것도, 자신의 코드를 타인에게 보여주는 것도 모두 큰 도움이 된다고 생각합니다.

저는 현재 ‘코드스쿼드’ 라는 교육기관에서 ‘javajigi’라는 필명으로 유명하신 박재성님께 코드리뷰를 통한 피드백을 받아 좋은 코드를 작성하기 위한 노력을 꾸준히 하고 있습니다. 왜 이러한 피드백을 주셨는지에 대해 고민하고, 수정하였을 때 이전의 형상에 비해 어떤 부분이 좋아졌는지를 고민합니다.

또한 같은 요구사항을 다르게 구현한 함께 공부하는 학생들의 코드를 열람하며, 어떤 부분이 나의 코드와 다르고 왜 다른지에 대해 탐구합니다. 코드 작성자에게 그 의도를 물어보고, 더 좋은 방법을 찾고자 토론하기도 합니다.

이에 더하여 ‘인디언 포커’라는 주제로 페어 프로그래밍을 진행하고 있습니다. 혼자 개발할 때보다 더욱 코딩 컨벤션을 잘 지키게 되고, 타인의 생각에 다양한 인사이트를 얻곤 합니다. 한 사람이 코드를 작성하는 동안 다른 사람은 눈 코딩을 하며 의견을 취합하게 됩니다. 하나의 프로그램을 완성하는 시간은 혼자보다 더욱 많이 소요되지만, 서로의 코딩 역량이 단기간에 향상되는 좋은 방법이라고 생각합니다.

**지난 개발 프로젝트들의 아쉬운 점이 있었다면 무엇이고 개선 방안은 뭘까요?**

제가 경험했던 모든 프로젝트의 공통되는 아쉬운 점은 ‘의사소통과 공감’이었습니다.

각자 자신의 포지션의 이해관계를 중심으로 타협점을 도출하고자 하기에, 함께 하나의 목소리를 내는 일이 어려웠습니다.

이러한 아쉬운 점의 개선 방안은 ‘타 포지션에 대한 이해’ 라고 생각합니다.

위와 같은 역량을 기르고자 기업에서는 직무 순환 제도와 같은 시스템을 도입하기도 하는 것으로 알고 있습니다.

저는 앞으로의 프로젝트에서는 이러한 아쉬움을 줄일 자신이 있습니다. 다양한 포지션에서의 협업 경험이 있기 때문입니다.

QA로서 개발자와 이슈 회의를 해보기도 하였고, 시스템 장애 상황에서 Front-end 개발자 위치에서 Back-end 개발자와 이슈를 해결해 본 경험이 있습니다. 또한, 개발자로서 업무 담당자와 요구사항을 협의하여 기능을 개발해본 경험도 있으며, 사내 프로젝트 관리자로 외주 개발자와 함께 일한 경험도 있습니다.

다양한 경험들을 바탕으로 앞으로의 프로젝트상황에서 다른 직무의 의견을 공감하고 존중할 수 있으며, 마찰을 최소화하고 소통할 수 있을 것이라 기대합니다.