참석자 1

그 계속 나중에 이제 프로젝트를 정리하라고 생각합니다.

그 있잖아 소프트웨어가 이렇게 돌아가지 않아

참석자 2

운이 좋았거든 ??

참석자 1

그렇고. 제가 했던 일은 그게. 지금도 아마 그렇게 간단하게 관심입니다.

안문 쪽은 리눅스가 들어요

참석자 2

미치겠어요

참석자 1

뒷문 쪽에 원래는 OS도 없이 그렇게 코딩이 됐었는데

내가 뒷문 쪽에 os를 만들어줄테니까

os 베이스로 일하고 os 베이스로 일한다는 게 어떤 거냐면은 각자 고민해야 될 내용을 각자 고민하면 돼 멀티태스킹이 되니까. 근데 예전에는 OS가 없으니까 이 사람이 코딩을 했다가 다 한 루프를 그냥 이렇게 도니까 이 사람이 뭘 하고 이 사람이 뭘하고 이 사람이 뭘 하는지 차례대로 돌잖아여. 이 사람이 코드를 잘못 짜서 시간을 끌면 이 뒷사람도 타이밍이 다 망가지는거야

참석자 2

철학자 문제네요?

철학자 문제는 아니고 그러니까 타이밍 문제인데 이 사람들이 그런 일들이 벌어지는 거에 대해서 여러 가지 printf을 열라게 넣어가지고 시간을 찍어서 그걸 관리하고 싸우고 막 하는 거예요. os를 쓰면은 해결될거를.

그래서 OS를 제가 거기에 만들어주고 그다음에 각자 해야 될 일을 하고 그다음에 각자 해야 될 일을 쓴 다음에 코드리뷰를 할 수 있게 코드 리뷰보다는 모여서 각자 만든 코드를 설명하고 있는 코드 소스 리딩 있잖아요.

그거를 해서 일단 거기 개발을 어떻게 해야 되는지를 정리를 해주는 일을 했었죠

그러고 나서 보니까. 학교에서 교수였으니까. 학교에서 우리 학생들은 어떻게 하나 이런걸 딱 봤더니 우리 학생들도 비슷하게 하더라고요.

그래서 그때부터 이제 소스 컨트롤하는 것도 가르치고, 소스리딩하는 것도 (코드 리뷰하는 거의 전단계) 서로 코드를 그냥 읽어보게만 하는거죠.

참석자 2

어디가 잘못되었나를 찾지 않고?

참석자 1

잘못된 거를 찾아내죠 소스 리딩을 하면은 자기가 자기 코드로 봐도 잘못된 걸 금방 찾을 수 있으니까. 그러니까 엄정한 코드 리뷰 프로세스를 거치지 않더라도 소스를 놓고 같이 읽고 그 다음에 남의 코드를 읽어주고. 그러니까 느슨한 코드리뷰를 하는 거를 수업 시간에 하고 그런 일들을 꽤 많이 했어요

참석자 1

그러고 나서 이제 보니까 소프트웨어를 어떻게 봐야 될지 어떻게 가르쳐야 될지에 대한 전반적인 이슈라는 게 생각이 많았잖아요. 그래가지고 소프트웨어를 잘 가르치는 방법에 대해서 고민도하고 예전에 nhn 넥스트에서 소프트웨어 교육 기관을 만든다니까 거기 가서 ??도 좀 하고 그런 일을 했었죠.

그리고 여기는 이제 새로운 소프트웨어 교육기관을 만든다고 그랬잖아요. 그런데 누가 만드냐가 되게 중요하잖아요.

정부가 만들면 성공적이 아닐 가능성이 매우 높죠. 그래서 사실은 이런 거를 정부가 하는 건 좀 아름답지 않다는 생각이 되게 많기는 했어요 그래서 한편으로는 이제 42시스템이든 우리가 지금 새로 교육시장 만들려고 하는 것이든 새로운 교육 시장을 만드는 거는 되게 중요하고 의미로운 일이기 때문에 그래서 ??

42 시스템 자체는 되게 괜찮은 것 같아요 그러니까 거기서 놓치고 있는 것들이 좀 보이기는 하지만 뭐냐면 학생들이 아까 기본적인 이론을 공부하는 거를 푸쉬하는 시스템이 없어요. 그거를 배우기를 바라는 건데 되게 오랫동안 필드에 있으면 결국 배우게 돼요. 왜냐면은 자기가 답답하거든. 그러니까 이론을 배우면 할 수 있는 생각할 수 있는 범위가 훨씬 넓어지기 때문에 언젠가는 배울 수 있게 돼요. 그리고 자기의 의지로 배우게 돼요 그런데 그 시간은 유럽에서는 기다릴 수 있을지 모르겠는데요. 우리나라는 기다릴 수가 없어요 학생들이 당장 취업해야 되고 면접 가서 깨지고 나면은 그다음에 현타도 오고 인생이 이런것인가 그럴텐데

심각하게 깨져, 그러니까 인생에서 시험은 떨어져도 내가 인생을 실패했다고 느끼는 경험이 없잖아요. 모든 학생들이. 그런데 면접에서 떨어지면 인생이 실패했다는 느낌이 들거든 되게 데미지가 크거든요. 그런데 그 이유가 이제 그런 것들이 부족한 거죠. 그래서 그런 걸 어떻게 얘기해 줄 수 있을까. 그런 고민도 좀 있고. 그래서 개선할 포인트도 꽤 많이 있고.

그거를 개선해서 우리가 새로운 교육 시스템도 만들어야 되겠다.

그다음에 사실 이제 우리가 취업을 할 때 항상 네카라쿠배만 생각하는데 그런 데는 신입 잘 안뽑거든요.

알잖아요 거기서 뽑는 신입사원 스펙을 딱 보세요. 그러면은 이게 학교에서 경험할 수 있는 스펙이 아니야.

서버를 운영해서. 내가 서버를 집에 있는 서버도 없는데 서버를 언제. 서버에서 벌어지는 일을 내가 어떻게 경험하겠어요.

그러니까 사실 보통 학생들은 경험할 수 없는 것들이 신입사원 스펙이 그냥 있어요. 잡디스크립션(JD)에. 그래서 사실 그런 데는 정말로 학생들 중에 액티브하게 자기 프로젝트도 많이 하고 현실적인 문제를 다뤄본 친구들만 갈 수 있는거에요.

우리가 그거를 자기 프로젝트 하라고 계속 얘기하잖아요. 그게 이제 그거를 쌓으라는 거고 그러고 나서 딱 보면은 나머지 영역을 생각해 보면은..

그런 방법들을 어떻게든 implement를 해야지 학생들이 성장을 하잖아요. 그게 그런 툴을 만들 기회가 그래도 조금 있지 않을까라는 생각이 되게 여기서 많이 있는 것 같아요. 멘토님들도.. 개발자들은요 똑같은 고민을 해요. 내가 조금 더 잘 배웠으면 지금 이러고 있지 않을텐데.. 우리가 보는 선수 개발자들 있잖아요. 그리고 우리 멘토님들이 시중에서 보면 완전 만렙 끝판왕인 분들이거든요.

이 사람들이 늘 느끼는 게 자기가 이제 새로운 프로젝트라고 하면서 야 내가 소프트웨어를 좀 더 제대로 배웠으면 지금보다 훨씬 더 빨리 더 훌륭하게 했을텐데 라는 생각을 해요. 그래서 다들 배우는 과정이 아쉬움이 있어요.

그래서 이렇게 했으면 좋겠어 저렇게 했으면 좋겠어라는 의견들이 다 있거든요. 그거를 이제 여기서 우리가 해볼 기회가 생기는 거예요.

그중에 당연히 무조건 하면 안 되는 일도 여러 가지 있고, 가르치려고 드는 거 있잖아요. 옛날 방법을 요즘 학생들한테 가르치려는 거 요즘에 더 좋은 방법이 있음에도 불구하고. 그런 거를 구분하는 역량이 이제 잘 갖추어진 분들도 있고 아닌 분들도 있겠지만 그런 거를 구분하면서 학생들이 갈 수 있는 방향을 조금 더 잘 해주면 되죠. 그리고 예전처럼 다 가르쳐줘야되나 이럴 필요도 없어요. 지금은 유튜브에 다 있잖아. 유튜브 인프런 ~~~ 커세라 이런 곳에 다 있기 때문에 이런 거를 배워야 되지 않겠냐 하면은 그거를 뭘 배워야 될지만 알려주면 그다음에 그것이 찾는거는 일도 아니거든.

유튜브에서 조회수 제일 많은 걸 그냥..

훌륭하거든. 보통 대학에 어떤 교수님보다 잘 가르치는 동영상이 있는 것 같아. 그래서 그거를 시스템화하는 일이 되게 중요한 거죠. 그러니까 우리가 교육을 할 때 훌륭한 선생님이 있으면 훌륭한 선생님이 나와요. 근데 훌륭한 선생님 있을 때 훌륭한 학생을 많이 키울 수는 없어. 힘들거든. 우리가 교육에 희생이라는 단어라고 덧붙이잖아요. 희생을 통해서는 교육이 스케일러블하게 큰 규모로 이루어질 방법이 없어요. 내가 희생을 해야 하는데 희생할 사람은 한 명밖에 없잖아요. 그래서 이거를 시스템화하는 게 되게 중요한 문제예요.그거를 이제 이노베이션 아카데미라는 곳에서 하는거죠.

(그럼. 그 시스템이라는 건 이제 사실이 별개인 것이겠네요?)

예, 42시스템도 시스템이잖아요. 그러니까 지금 있는 이 42시스템도 괜찮은 애고요. 또 새로운 시스템은 또 그 시스템대로 괜찮은 게 만들어지겠죠. 코드리뷰를 조금 더 효율적으로 할 수 있게 한다거나 또 현실에는 정해진 코스가 이렇게 쭉 있잖아요 처음부터 끝까지. 근데 모든 학생들한테 이런 처음부터 끝까지 하는 게 필요하지는 않을 수 있잖아요.

중간에 이만큼 잘라서 할 수 있는 것도.. 그러니까 이제 다른 시스템을 만드는 거죠.

(가칭이 있나요?)

우리가 프로젝트X 라고 이름을 붙여서 가고있는데, 이름이 없어서. (웃음) 모든 걸 다 아는 이런 게 아니라 이름이 없다.

---식사---

(질문은 잘림. 아우터서클을 진행할 가치가 있는지? 이런 질문이었던 듯)

아우터 서클에 있는 문제 수준의 현실적인 문제를 만나거나 만들 수 있다 그러면, 아우터 서클보다 그게 훨씬 더 중요하다고 보고 있고요. 그 기회를 만들기 위해서 회사들을 학생들이랑 만나게 하려고 하는 거고. 그게 없으면 어설픈 문제 어설픈, 진도 나가기 공부 이런 것보다는 거기 문제들이 또 쉽지 않거든요. 훨씬 좋죠. 이너서클에 있는 것 보다 시간도 더 많이 걸릴 거고, 문제의 규모도 크고 그래서 아우터 서클이 현실적인 현장 문제를 접근 가능한 몇몇 친구들한테는 아우터 서클보다 그게 더 좋을 수도 있어요.

(위 질문을 적어준 친구의 의견인 것 같은데요. 본과정을 등한시하는 문화가 있다고 생각을 하나 봅니다. 이런 문화에 대해서 어떻게 생각을 하느냐. 라는 질문이 있었는데..)

학생들이 얼마나 교육에 대한 관점이 있는지 모르지만, 본과정 자체 문제를 문제가 왜 있고 그것이 뭘 목표로 하느냐.

이걸 우리가 안 가르쳐준 탓이라고 생각을 해요. 그니까 본과정이라는 게 c언어를 배우기 때문에 이제 네카라쿠배에서 원하는 그런 역량이, 표면적인 역량이 안 키워지잖아요? 그런데 거기 가서 면접을 딱 하고 나면은 내가 뭘 놓치고 있었는지 다 알게 돼요. 어느 게 문제라는 걸 다 알게 돼요. 그러니까 그 언어가 문제가 아니라는 걸 알게 되는 그 이면에 있는 내용들을 본 과정에서 배우라고 이 코스가 만들어져 있는데, 그거를 안 배웠기 때문에 본과정이 의미가 있는지를 잘 모르는 거야. 그리고 학생들이 (우리 뿐만 아니라 대학생도 비슷한건데) 회사를 너무 몰라. 좀 심각해요. 진짜 개발자를 만나본 적이 없어요. 진짜 개발자들이 뭘 고민하고 문제를 어떻게 해결하고 그런 얘기를 잘 들어본 적이 없는 거예요. 우리가 개발자를 만났을 때 한 얘기는 다들 한풀이를 많이 들어. 회사에 이런 xx가 있었는데.. 이런 그 얘기를 많이 듣잖아요. 우선 그러니까 기성 개발자를 많이 만나는 게 되게 중요하거든요. 여러 명을 만나는 게 아니라 한 사람을 몇 번 만나는 게 되게 중요해요. 한풀이 시간이 끝나고 나서 진짜 문제에 대해서 자기가 어떻게 생각하고 접근했는지를 들을 기회가 있어요. 학생들이 지금은 이제 코로나 때문에 예전만큼 오프라인 커뮤니티가 활성화되지는 않았지만 커뮤니티에 들어가서. 그 선수들을 만나는 걸 조금 해야해요. 나의 롤모델이 지금 어떤 고민을 하고 있는지 알아야 되잖아요. 그런데 그런 노력 없이, 내가 갈 수 있는 회사가 있고, 그 다음에 가고 싶은 회사가 이렇게 있고, 가고 싶지는 않지만 그냥 존재하는 회사도 있잖아요. 이런 회사들이 있는데 이 회사들이 안에서 어떤 일이 벌어지고, 그 회사들은 어떤 고민을 가지고 갖고 있고, 저 회사 가면 내가 누구를 만날 수 있고, 이런 걸 알아보는 과정이 되게 많이 필요한 거죠. 회사들이 와서 가끔 이제 온라인도 오프라인도 톡을??하잖아요. 그러면 그걸 열심히 듣고 질문을 하는 과정을 통해서, 저 회사는 저런 회사구나. 저 정도 규모 회사가 저 정도는 되는구나. 라는 거를 아는 게 되게 중요해요.

우리가 다들 느끼고 있듯이 신문에 나왔던 그 많은 회사들 있잖아요. 연봉 얼마씩 준다는. 그런 회사들은 사실 신입사원은 거의 안 뽑거든요. 현실적으로 들어가기가 거의 불가능해. 그리고 들어간 친구들 보면 쟤가 왜 들어갔는지 잘 몰라.

운발도 되게 많이 작용해요. 사람이 사람을 선택한 과정이기 때문에. 그래서 사실 그런 회사는 진짜 재수가 조금 들어가거나 경력으로 들어가면 되거든요. 그러고 나서 그 회사들을 좀 제외하고 나면 다른 회사들은. 회사들을 너무 모르는거야. 이 회사가 이름은 조금 들어본 것 같은데 이런 회사들이 되게 많아요. 이 회사의 업의 본질이 뭔지, 과연 예산은 매출이 얼마인지 그런 것도 좀 알아야되고요. 회사가 실제 소프트웨어가 얼마나 중요시되고 있는 회사인지, 소프트 개발이라는 게 어떤 과정으로 이루어지고 그 안에 있는 개발자들이 과연 성장을 할 수 있는 곳인지, 그런 거를 알아보는 과정이 되게 많이 필요하거든요. 그거는 아무도 안 가르쳐주잖아요. 코스피나 이런 데 보면 DART공시에 안나와 있는 영역이라고. 재무재표도 공개가 안되고(웃음).그래서 사실은 물어봐야 되는 거예요. 그 언저리에 있는 사람들한테. 이 회사는 어때? 거기 가서 어떻게 일했어? 뭐 배웠어? 이거를 물어보고, 그 사람들도 스스로도 후배가 들어오면 어떻게 할 준비가 돼 있는지, 회사에 그런 프로토콜이 있는지, 그런 거에 대해서 물어보는 과정이 있어야 되거든요. 그거를 이제 커뮤니티 가서 해도 되고, 아니면 요즘에 전시회를 조금씩 하기 시작했잖아요? 전시회 같은 데 가서 모든 회사 소프트웨어를 사용하니까 내가 소프트웨어를 배우고 있는데 그 회사는 소프트웨어를 어떻게 개발하는지 미친척하고 그냥 물어보면 돼요.

거기 있는 사람들은 다들 하루 종일 전시장에 서 있잖아요. 누군가 와서 질문하기를 고대하고 있는 사람들이에요. 그래서 질문을 잘 모르니까 그냥 처음에는 막연하게 질문하는 거에요. 이런 거 만들 때 소프트웨어 필요하지 않아요? 뭘로 개발하세요? 이런걸 물어봐요. 그러면 그쪽에서 오랫동안 기다려왔던 질문이기 때문에 자기네 영업 비밀까지 다 얘기한단 말이에요(웃음). 그리고 옆집에 가면 비슷한 업종의 회사가 또 있잖아요. 여기서 들은 얘기를 하는 거예요. 어떤 이런 가게들은 되게 이런 걸 개발하는데 여기는 어떻게 하세요. 딱 그러면 엄청 있어 보이잖아요. 이쪽에서는 자기네가 하고 있는 얘기를 또 해 주는(웃음). 제가 소프트웨어 배우고 인턴을 가고 싶은데 제가 프로그램 짤 줄 알거든요? 필요하면 연락주세요. 하고 명함을 주고 오는거에요. 그런 걸 해야 돼. 장사를 해야 되잖아요.

(혹시나 이건 경험담이신가요?)

그럼요. 학생들한테도 다 그렇게 하고, 어릴 때부터 명함을 만들어서 다니면서 사람들한테 그런 얘기할 때 다시 연락하시라고 줘요.

그게 사실 내가 할 수 있는 거를 써먹을 기회를 바깥에서 찾는 거는 되게 중요하잖아요. 소문이 나면 저절로 많이 일이 오죠. 처음에는 그게 아니라 내가 뭘 할 수 있는지 내가 뭘 관심 있는지 계속 얘기하고 다니고 그런 영역에 있는 사람들을 만나는 노력을 해야죠. 그게 이제 막 되게 소셜하면은 좋겠지만 저는 주는 소셜한 편이 아니어서 그냥 조용히 뒤에서 글이라도 올리고 이런 거라도 하는 거야. 전시회 같은 데 가면 소셜하고 안하고에 상관 없이 그런걸 할 수 있어. 되게 편해. 저 사람하고 나밖에 없거든 그냥 들이대면 돼요. 그리고 다시 만날 사람이 아니야. 걱정할 필요가 필요가 없어. 다시 만날 사람이면 내가 조심도 하고 그래야 되는데. 저 사람은 딱 보고 연이 아닌 것 같으면 다시 볼 사람이 아니거든요. 걱정하지 말고. 하다가보면 내가 이제 거짓말도 늘고 주워들은게 좀 생기잖아요? 그러고 나면은 내가 뭘 모르는지도 이제 조금씩 알게돼. 아 저런 곳에서 일하려면 이런걸 좀 하면 좋겠구나. 그런 거는 사실 검증이 필요하죠. 다른 사람들한테.

여기도 마찬가지인데 소셜한 친구들이 동료 학습을 더 잘 할 가능성이 매우 높잖아요? 누가봐도 사실인 것 같고. 사람들이 소셜하게 되면 좋지만, 그거는 타고난 성향일 수도 있고 여태까지 훈련받은 결과일 수도 있으니까. 그거를 이제 우리가 빨리 시험해서 제 이 팀들이 그거를 끄집어내는 역할을 해주면 좋겠다고 생각하고 있는데, 생각보다 잘 되고 있는 것 같지는 않아요.

숨어 있는 친구들을 끄집어내서 그 친구들을 데리고 같이 공부를 해야 해요. 그 친구는 분명히 내가 가지고 있지 않은 많은 역량을 가지고 있거든요. 그렇게 친구를 기다리고 있고 공부할 의지가 있으면 기다리고 있을 거죠. 그러니까 이제 가서 데리고 나와야 돼. 그러니까 동료 학습이라는 게, 고등학교 때 이제 독서실에서 하던 동료 학습 있잖아요. 옆에 친구가 자고 있으면 뒷통수를 때려서 깨워주는 거야. 그거는 동료학습이 아니잖아요. 그런데 지금 온라인 상황에서는 그 동작도 조금 필요한 거예요. 숨어 있는 애를 깨워가지고 데리고 나와야 돼. 데리고 나오면 여기 보면 다른 사람이 있으니까 버티다 버티다 다른 사람한테도 물어보겠죠. 그게 이제 경험이 쌓이면은 어느 순간에 아 이게 별거 아구나라는 걸 알게되는데, 우리가 피신 때 그 과정을 다들 겪었다고 생각을 했었는데, 피신이 딱 끝난 다음에 온라인으로 딱 가면서 이 동료학습의 텐션이라는게 딱 떨어져 버린거죠. 그거를 이제 의도적으로 스스로도 복원해야 되고 친구들끼리도 그걸로 도와줘야지 되는 거죠.

혼자서 하려면은 뭐하러 여기 왔어요. 그냥 문제집 풀지. 혼자서 새로운 프로젝트하고 인프런이나 생활코딩이나 이런거 보면서 공부해도 혼자 해도 시간을 들이면 할 수 있어요. 근데 혼자 가는 거는 빨리 가는 방법이 전혀 아니죠. 혼자해도 많은 걸 배울 수 있지만, 혼자해서 버틸 수 있는, 많은 동료 학습을 해서 얻을 수 있는 많은 것을 혼자 하면 놓칠 가능성이 매우 높아요. 그러니까 옛날에는, 제가 배울 때는 그런 게 비교적 용서가 잘 됐어요. 책에 있는거 진도 나가고 그다음에 언어는 그때 쓸 수 있는 언어라는 게 잘해야 c, 베이직, 파스칼. 파스칼은 금방 없어졌고. 그래서 사실 옵션이 별로 없었기 때문에. 그때는 혼자 하나 여럿이 하나 별 차이 없었어요. 그런데 지금은 전혀 그렇지 않거든요. 배워야 될 게 너무 많아. 그래서 같이 배우는 게 훨씬 효율적이고 좋은거죠.

그리고 점점 올라갈수록 다른 사람의 관점으로 배우는게 너무나 중요하고. 선수가 되면은 나중에 보면 내가 고민했던 방식이 다른 사람이 고민했던 방식이랑 결국 비슷했구나라는 걸 알게 되는 포인트도 있어요. 패턴이라고 들어 봤어요? 설계패턴. 설계 패턴이라는 게 있어요. 이런 문제에 대해서는 이런 방식으로 접근하는 게 제일 수월하다는 거를. 우리가 어떤 문제의 답을, 알고리즘을 패턴화 해서 전형적으로 푸는 거야. 동기화 문제가 있을 때는, 이렇게 하면 돼. 그런 패턴이라는 책이 나왔어요. 95년? 4명이 지은 훌륭한 책이있는데, 설계 패턴에 관한 초창기 책이 있는데, 95년쯤에 나왔거든요. 사람들이 95년 전까지는 사람들이 세상의 모든 문제를 보면은. 문제를 해결하는 이제 유형이라는 게 있잖아요?

이 유형에 대해서 다들 비슷하게 접근하고 있다가 여기서 이제 패턴을 다 정해줬어. 내가 했던 게 여기 딱 정리돼 있네.

이 생각이 드는 거야. 그러니까 이제 우리가 충분히 커가고 나면 이제 그런 포인트들이 생기게 되거든요. 그러고 나면 새로운 문제, 새로운 방법론을 찾고 하는 거를 이제 할 수 있게 되는 거죠.

그런데 사실은 그 전까지는 경험이 필요해. 그런 패턴이나 이런 것들을 먼저 학습을 하게 되면은 답을 먼저 보는거잖아요? 그렇기 때문에 생각을 자유롭게 할 수가 없어요. 그런 패턴을 모두가 똑같이 보면. 대학교 교육에 약간의 문제이기도 한데, 그런 걸 먼저 가르쳐줘. 생각하는 방법을 가르쳐주고 진도를 나가서 문제를 풀면 다 비슷한 답이 나오는 거죠.

그런데 그 과정을 안 가르켜주고 각자 해결하라고 그런 다음에 각자 답을 보면, 전혀 다른 방법을 해결하고 있는 거에요. 이게 이제 진도를 안 나가는 동료 학습에 가장 큰 장점인데, 그 장점을 사실은 서로 얘기를 하면서 만나서 하지 않으면 누릴 수 없는거죠.

(궁금한 게 있는데, 본 과정에서 배울 수 있는 cs 지식도 있고 에티튜드가 있다고도 말씀을 해주셨는데 그런걸 이제 본과정 거의 끝에 와 가니까 자연스럽게 알게 되거든요. 그래서 이런게 필요해서 이런 과제가 있었구나 라는걸요. 42가 원래 좀 잘 안 알려주는 듯한 컨셉인 것 같기도 한데, 이런 걸 안내해 주실 생각도 있으실까요?)

제가 이론을 봐야 된다는 거는, 그러니까 유튜브나 저나 멘토님들이나, 하여간 학생들을 만날 기회가 있을 때 마르고 닳도록 얘기하고 있을 거예요. 그게 되게 중요한데 안되고 있는걸 우리가 보니까. 그거는 학생들이 어떤 경로든 그걸 배웠으면 좋겠고. 에티튜드에 관련돼서는 훈련이 사실 잘 안 되거든요? 되게 오랫동안 우리가 살아온 흔적이 그냥 묻어 있는 것이기 때문에 훈련이 잘 안 돼. 그 에티튜드에 관해서 최근에 우리 히치하이커스 가이드라고 만들어놓은 거 있잖아요.

거기 보면 에티튜드에 관한 새로운 항목이 하나 생겼어요. 거기 동영상들이 쭉 있는데, 왜 우리가 이런 공부를 해야 되고, 이런 학습을 구성주의 철학에 의한 교육이라고 하는데. 구성주의 철학이라는 게 뭐고, 학생들이 다 어린이들이 아니잖아요? 그래서 다 공부를 할 만큼 한 사람들이기 때문에 이런 학습 방법에 대한 이론. 커뮤니케이션에 관한 것. 이런 것들을 동영상을 좀 모아놨어요. 그러니까 수보멘토님이 그걸 모아서 정리해놓은 게 히치하이커스 가이드로 올라갔어요.

그래서 그거를 다 보시면 좋을 것 같고. 그게 있다는 거를 사람들이 또 모르는게 문제지만 (웃음). 그거를 보시면 좋은 것 같고, 그 다음에 지금은 그런 에티튜에 관한, 협력이나 생각하는 법, 그다음에 우리 개발한 프로세스와 관련된 거를피신 같은 거를 만들어서 운영한 걸까 지금 생각 중이에요. 피신때 가장 텐션이 올라가잖아요. 지금 제가 우리가 레벨이 조금 올라간 친구들을 데리고 이제 애플리케이션 쪽 피신들 있잖아요? 웹을 개발한다거나. 아마 이런 피신들을 이제 종종 열라고요. 학생들이 C만 가지고 이제 고민하는 친구들이 많으니까 다른 영역을 경험할 수 있는 기회를 우리가 피신 형태로 만들어서 하고, 에티튜드에 관련된 부분도 피신형태로 만들어서 운영을 해 보고. 사실 그거는 쉬운 문제가 아니에요.

예전에 제가 넥스트나 이런 다른 데서 교육을 해봤더니, 그런 게 교육을 하기 위해서는 우리 리소스가 들어가잖아요.

돈이든 노력이든 이런 게 들어갈텐데, 효과가 되게 안나와. 사람의 행동 패턴을 바꾸는 거기 때문에, 지식을 쌓는게 아니라. 그래서 그거를 하기 위해서 학생들을 놓고 비포 애프터를 평가해야하잖아요. 그런 연구를 좀 하고 그 다음에 그걸 실제 피신으로 만들어서 뭐가 있는지를 측정을 해보고.

(그런 동영상이 있으면 각 층에 비치된 tv에 틀어놓아도 좋을 것 같은데요)

좋은 의견이야. 적어놔야지.

(이번엔 다른 주젠데, 아까 신입개발자 때 이야기를 많이 해주셨잖아요? 만약 학장님께서 신입 시절로 돌아간다면 바꾸고 싶은 한가지가 있을까요?

저한테 되게 어려운 질문인데 저는 컴퓨터를 거의 배우고 처음부터 그냥 회사들이랑 프로젝트를 그냥 했어요.

그래서 리터럴리 신입 개발자라고 회사에 들어간 적이 진짜 거의 없었어요. 처음부터 그냥 회사들이랑 프로젝트를 하면서 배웠어요. 처음에 예를 들어 동네 약국에 이제 되게 복잡하잖아요? 약국의 재고관리를 어떻게 해야하나. 데이터베이스를 써가지고 하는 것 그런 작업. 실험실에서 실험 장비에서 데이터를 꺼내는 걸 쌓아서 분석하고 이런거 하고, 자동차의 문을 버튼으로 열 때 만드는 정치를 어떻게 프로그램하고, 엘리베이터 제어를 어떻게하나. 그런걸 그냥 학부때 그냥 하면서 배웠거든요. 그냥 혼자서 프리랜서로 그런 일을 많이 해서 특정 회사의 신입사원이라고 들어간 적은 박사과정 때 처음이었어요. 개발자가 10 명인데. 사실 개발자 중에 제가 꽤 시니어 였었기 때문에 배우는 입장이 아니었고 팀을 그냥 리드하는 입장이어서 그렇게 그냥 신입사원이었거든요. 그니까 이제 제가 신입사원의 입장에서 뭔가를 보는 것은 거의 경험이 없는 것 같고요. 신입사원을 보는 관점에서는 이제 신입 개발자를 보면 느끼는 바는 많이 있죠. 일단 얘는 잘한다 못한다는 느낌이 이제 거의 바로 들어오고요. 거꾸로 신입사원도 그거를 느껴요. 회사에 딱 가면, 이제 오늘 나랑 같은 공간에 일하는 사람이 10명이야. 그러면 한 2~3일 지나면 이 10명의 실력을 나래비로 세울 수 있어요. 거의 틀리지 않아요. 얘는 진짜 잘 하는 애고, 걔는 왜 여기 있는지 모르겠어. 이 나름대로 세우는 게 누구나 다 가능하고 그게 거의 틀리지 않아요.

30분 45초