포트폴리오_및_이력서_준비

기술 인터뷰 이전 과정들은 어떤 목적이 있는가?

- 모두가 아는 불편한 진실

- 누구나 "매력 있다"라고 생각하는 커리어는 -> 기본적으로 인지도가 높고, 도전하는 사람이 많다.
- 채용을 진행하려는 쪽은 -> 도전하는 모든 사람들을 만날 수 없다 (시간 리소스 문제)
- 그러면서 지원자 중 최고로 나은 사람을 뽑고 싶어한다.

- 많은 사람들이 모르는 불편한 진실

- 지원자를 가려내야 하는 입장에서는 모든 사람들의 이력서, 포트폴리오를 정성으로 읽어볼 시간이 없다.
- 그렇기 때문에 채용 낙방은 순수히 어떤 지원자가 타 지원자보다 역량이 *부족하다고* 겪는 일이 아니다.

ex) 사실 이력서는 면접 자리에 가서 훌쩍 넘겨다 보면서 질문거리를 만들기 위해 처음 보는 경우도 꽤나 된다는 것이지.

그러므로 중요한 것은:

엔지니어링 캐릭터 를 만들어가는 일

개발자 채용 순서:

> 서류 필터링->기술 인터뷰 필터링 대상자
> --> 최종 대상자 (임원/ 개발 디렉터 면접)

> 뒤로 갈수록 채용 리소스가 늘어난다 (주로 돈으로 환산되는 시간 리소스를 사용하므로)

- 기술 인터뷰를 할 때는 인터뷰어의 시간을 쓰게 된다. (억대 연봉 개발자의 한 시간!)
  - 그렇기 때문에, 채용시즌에는 인터뷰 때문에 본업에 문제가 생기기도 한다.
- 최종 인터뷰에서는 회사의 중요 직책이나 임원 이상의 인원으로 구성된 인터뷰를 진행. 이들이 인터뷰에 시간을 투자하는 것, 최종 오퍼가 나간 이후 연봉 협상 과정 자체가 시간 단위로 리소스가 나간다는 일.

그래서 서류 필터링에서는 채용 [대상자]가 확실히 ***아닌*** 사람들을 필터링해내는 것이 목적인 것이다. -> 절대 뽑으면 안 되는 사람을 떨구는 과정에서 같이 걸러지면 안 된다는 이야기!

인터뷰를 하는 측에서 중점으로 보는 것

- 타 지원자 대비, 채용을 진행하는 팀에 소속되었을 때 업무를 빠르게 익히고, 문제 없이 진행할 수 있는지를 본다.
    - 헤드헌터들을 통해 주고 받는 이력서는 잘 안 본다고 한다.
    - 뭔가를 빨리 배우고, 문제 해결을 빠르게 도달해 본 일이 있는가? *역량만 본다*는 말은 쌩 그짓말이다. Day 1 부터 문제를 해결할 수 있는 사람!
    - 언어 /프레임워크과 같은 기술적인 경험이 직무와 일치하는가? -> 스택
    - 어려운 문제가 닥치면, 그것을 해결할 기본기가 있는가? -> 자료구조, 알고리즘, DB, OS, 네트워크, 객체지향 프로그래밍 등의 CS적인 지식?
    - 기본기-> 코딩 테스트! -> 안 죽었다. (문제해결능력 ->)

> [기본기] : 이미 프로그래밍 언어를 아는 상황에서 하나의 언어를 새로 공부해야 하는데 2주 안에 라이브러리/프레임워크를 파악하고 과제를 풀어낼 수 있을 정도의 레벨

- 고용을 추진하는 우리팀의 약점을 지원자의 강점으로 채울 수 있는지 본다.
- 그럼, 개발자로서 중요하게 지키는 포인트가 무엇인지?
    - 1. 결과물의 안정성 -> 중요!
    - 2. 작업 속도 -> numerical하게 만들어지는 생산성
    - 3. 기술 트렌드 관심도 및 공유 마인드 -> 팀이 새로운 기술을 받아들이는 데 필요한 man/hour를 직접적으로 줄여줄 수 있음
    - 4. 기술 선택시의 정교한 트레이드-오프 능력 -> 제한된 자원 및 툴을 가지고 주어진 문제를 해결하는 것이 엔지니어의 핵심 역량!
    - 5. 설계 능력
        - 이는 문제 파악 및 리소스 할당 능력과도 맞닿아있음
        - ㄴ> 지원하는 팀에 대한 문제 /집중하고 있는 토픽이 무엇인지 인맥으로 알아내면 최고임

1~5 에 대해서 스스로 평가하면서 별 하나~다섯 개를 달아주면 그 profile이 당신의 엔지니어링 캐릭터!

- 팀에서 원하는 연차 및 직급에 맞는 성과나 경험치 나 있는지 살펴보기
    - 오랫동안 경험해 왔는데 물경력이 티가 나거나, 학습의 결과가 좋지 않다면-> 이것은 레드 플래그이다.

- 프로젝트 코드를 볼 수 있게 공개했다면, 그것이 트렌딩한 컨벤션에 맞는지와 좋은 아키텍쳐로 구성했는 지를 본다.
    - 공개하지 않는 게 더 나은 듯한 프로젝트를 올리는 사람들도 매우 많다. *진지하게 되돌아 보자.*

예시) 개발자 면접을 보러 와서 라이브러리 관련 질문을 받았을 때 "이 문제는 라이브러리 문제이기 때문에 라이브러리 만든 사람이 업데이트를 해줘야 문제를 해결할 수 있다" 따위의 대답을 하는 자가 과연 개발자라고 볼 수 있는지? 에 대한 질문이 블라인드에 올라왔다고 한다.

ㄴ> 특정 라이브러리를 활용하려면 fork를 떠와서 문제되는 부분을 직접 수정, 핸들링 할 각오는 되어 있어야 할 것이다.

기술 인터뷰에서 보지 않는 것 / 절대적 기준이 아닌 것

- 지원자의 성격, 성실함, 도덕성, 커뮤니케이션이 유연하다 라고 소개하는 내용
  - 서류로 믿을 수 있는 것이 없음.
  - 커뮤니케이션이 잘 되는 것은 더 이상 차별점이 아닌, 20대 중후반일 때 당연히 갖춰야 하는 사람의 *기본*

- 사진, 생김새, 나이
  - 외모가 개발 작업의 리소스를 줄여주지 *않는다*.
    - 노션 이력서로 사진 등을 먼저 넣는 사람들이 종종 있는데, 쓸모없는 페이지라고 생각될 뿐니

- 얼마나 많은 프로젝트 / 경험을 했는가?
  - 숫자가 중요한 게 아님. 어떻게 했는지가 중요. 이상한 방법으로 오랫동안 만든 것은 최악이다.
    - Github 하루하루 잔디 심기라는 말이 있지만, 하루하루 의미없는 커밋만 올리고 있다면 -> 그게 좋은 영향을 미칠까?

- 각종 자격증 및 토익 성적
    - 개발자는 자격증 문제를 푸는 게 일인 사람이 아니며, 영어가 필요할 시에는 인터뷰에서 영어 인터뷰를 진행하게 될 것임. (Google Korea, Coupang 등)
    - 그나마 정보처리기사는 괜찮음.

- "절대적인 기준의" 출신 대학, 출신 학과, 휴각 기간, 학점 등
    - 다만 다른 경쟁자보다 드러나기 위해선 위의 사항이 아닌 -> 다른 면모로 매력을 보여줘야 함.
    - 1년 동안 배출되는 인-서울 컴퓨터공학과 학생들이 매우 많다는 이야기다. 그들이 모두 기회를 얻지도 못한다.
    - 오픈소스 커미티 등등등

바이브 코딩으로 다 할 수 있는데 왜 굳이 프로그래밍을 해야 하나요?

-> 대기업들은 보안 등의 이슈로 죄다 GPT 못 쓴다는 것을 기억하기!

이력서와 포트폴리오의 차이

- 이력서
  - 짧고 간결한 개조식 표현 위주
  - 디테일보다는 워딩 / 성과 위주로 표현
    - ※ 잘 설명할 수 있는 키워드 위주
    - 이는 면접자가 무엇을 물어볼 지 도와줌
  - 경험한 팀 / 프로젝트에 대한 자세한 내용보다는 나의 경험 위주로 서술
    - "내가" 어떤 기술을 배우고 도입했는지
    - 어떤 성과를 냈는지

- 포트폴리오
  - 사진, 글, 영상등을 활용, 경험을 풍부하게 표현
  - 나의 성과와 다뤄본 기술적인 경험을 디테일하게 나타내고, 프로덕트에 대한 배경 등에 대한 내용도 서술 (학습한 것, 시행착오 사항들)
  - 코드를 넣긴 하던데... 과연 볼까?

ㄴ> 이는 PM이나 디자이너의 이력서에 가까움

포폴을 꾸민다 하더라도, 코드 자체를 올린다기보다는 내가 해결한 문제, 내가 한 설계 등등을 어필할 것!

이력서와 포트폴리오 둘 다 꼭 들어가야 하는 것

- 각 이력 / 프로젝트에서 다뤄본 기술적 키워드, 혹은 업무 커뮤니케이션 능력
    - 언어, 프레임워크, 아키텍쳐 패턴, 라이브러리, 사용 가능한 외국어
    - 외국어 (특히 영어) 어필은 아직 유효함
- 어떤 성과를 냈는지?
    - 내가 구현한 기능, 어떤 개선 사항을 이뤄냈는지
    - 여전히 이걸 쓰지 못하고 abc 했음! 정도로 썼을 가능성도 있음
- 내가 어떤 것을 얼마의 기간동안 배웠고, 배움의 결과는 어떤 것인지
    - 프로젝트의 문제 해결 과정에서 학습한 것과 기간 및 결과 -> 스터디/커리큘럼 -> "ㅁㅁㅁ 주제를 한달동안 ㅁㅁ 교재로 공부해 보았으며 이를 ㅇㅇ 개발할 때 ㅂㅂ하는 방식으로 적용해 봄"
        - 빨리 배우는 것을 어필! -> 짧은 시간에 다량의 학습을 한 것을 어필, 적용력 강조
    - 각 과목 및 학점. 하지만 자신이 없다면 안 드러내도 괜찮음
        - 학점이 안 좋다면 이력서에 학점을 드러내지 마시오.

이력서 작성 방법

- 열람자가 이력서를 모두 보고 나서 흥미를 갖게 하는 게 아니고, 이력서 ***상단의 콘텐츠***를 보고 매력을 느끼게 하기
    - 나의 최대 장점 및 성과 요약을 이름/ 인적사항 다음으로 양식 최상단에 작성

- 지원하는 공고에서 요구하는 기술 경험과 매칭되는 기술 키워드를 작성하기
    - React, AWS, MVVM, Clean Architecture, MSA etc.
- 나의 성과를 수치로 나타낼 수 있다면 최대한 나타내기
    - 로직 및 걸계를 개선하여 클라우드 인스턴스 비용을 ~% 감축
    - 전체 100명의 참가자 중, 3위로 입상

- 작업 결과물에 대한 근거를 나타내고 싶다면 압축 웹 링크로 표시
    - 예시: 깃헙에 호스팅된 오픈 프로젝트 / 오픈소스 기여 결과 등
- 이름, 이메일, 각 프로젝트 및 경험마다의 ***기간***, 역할, 활용가능한 외국어와 그 레벨을 나타내는 것은 기본
    - 영어 커뮤니케이션 레벨 등등

예시 이력서 살펴보기

(개인정보가 있다 보니, 이 부분은 느낀 것 위주로 정리)

- 졸업한 학교가 나오기보다는, ***어디에서 어떤 문제를 풀고 있는 사람인지***가 가장 먼저 나오는 게 좋다.
- gitHub 등 활용, 링크를 풀어서 적어놓을 것.
- 철저하게 기술 위주의 내용 적기.
- 음원 스트리밍 서비스 등등
- 포인트 / 개념을 알기 좋게 -> 그리고 알고 있는지 물어보기 좋게 키워드를 잘 드러내서 쓸 것.
- 담백하게 이야기하기
- 특정 환경에서 특정 기술을 활용한 도전 등등을 활용
- 기술 인터뷰에서 떨어지는 경우 -> 이력서에서 기술한 부분을 물어봤을 때 제대로 설명하지 못하는 경우가 대다수이다.

> - 그러므로 이력서에서 내가 잘 설명할 수 있는 개념들 위주로 후킹을 하는 것도 좋은 전략

앞으로 어떤 경험치를 쌓아야 할까?

- 뭔가를 적당히 만들어 본 것 만으로는 부족하다.
  - 프로젝트 / 경험마다 기술적 선택을 할 때마다 충분한 근거를 가지고 진행해야 할 필요 있음.
- 구체적이고 명확한 목표를 가진 상태에서 경험을 쌓는 것에 집중할 필요 있음
  - ex 1_)이 프로젝트에서는 기존에 구현하던 것 대비 x% 높은 퍼포먼스를 달성해 보자.
  - ex2) 기존 프로젝트에서 해보지 않은 로깅 시스템을 설계해서 마케팅 퍼포먼스 성과를 측정할 수 있는 제품을 만들어 보자.

- 현직에서 해당 기술 분야에 대해 집중하고 있는 부분이 어떤것인지 **파악**하고, 그것을 ***빠르게*** 학습해 보자. 기존에 잘 하던 것보다는 모르는 것을 채우는 데 집중할 것
  - 예시) Rust, 함수형 프로그래밍 등

- 개발자는 백지상태에서 새로운 것만을 만들기보다는, 이미 있는 것을 더 좋게 개선하는 것이 대부분의 업무이다. 과거에 만들었던 것을 **더 좋게** 만들어보는 것에도 집중하고, 그 성과를 기록하자.
  - 예시: 42 서울 결과를 re-factoring 하기, 재설계