목 차

(7) 프로그래밍 기술

- 가상함수
- 자바 가상 머신
- 네이티브앱, 웹앱, 하이브리드앱
- 디자인패턴

(8) 소프트웨어 개발 방법론

- 애자일
- 데브옵스

(9) 빅데이터

- 빅데이터 기술
- DBScan

(10) 인프라

- 도커
- 클라우드
- 마이크로서비스

중소기업, 대기업, 외국계, 기술사 시험까지 합격한 면접 노하우

1. 면접 보기 전에 알아둬야 할 점

2. 기술면접 면접과정 설명

3. 면접 TIP

(1) 자기 소개를 해보라는 것은?
(2) 압박 질문을 받았을 때는 태도와 대응이 중요하다.
(3) 면접은 만남이다.
(4) 면접은 이력서 작성부터 시작이다.
(5) 기술면접을 잘 보기 위해서는 기술적인 용어를 적절하게 사용하여야 한다.
(6) 면접자는 본인이 어떻게 면접관의 눈에 비칠까를 끊임없이 생각해야 한다.
(7) 자기소개에서 인용 문구는 겉치레로 들린다.
(8) 면접장에서 겸손은 자신감 있는 당당한 겸손이다.
(9) 자기 수준에 맞는 답변이 좋다.
(10) 답변은 너무 짧지 않게 하는 게 좋다.
(11) 전혀 모르는 질문이 나왔을 때
(12) 질문은 유도하는 것도 방법이다.
(13) 마지막으로 하고 싶은 말이 있는가라는 질문에 적당한 답은

4. 면접 전 준비할 것

기술면접에 자주 나오는

CS 지식

1. CS 기본지식

(1) 자료구조

프로그램을 개발할 때 자료를 저장하는 코드를 짤 때가 많다. 자료라는 것은 데이터를 의미하고, 데이터의 예를 들면 도서 관리 프로그램의 경우는 도서가 데이터가 될 수 있고, 해당 도서를 대출한 사람을 사용자를 데이터라고 할 수 있다. 프로그램에서는 데이터 저장하는 일이 많고, 이를 어떤 식으로 저장할지 중요하다. 데이터를 어떤 식으로 저장할지에 대해서 정리해 놓은 것이 자료 구조이다.

자료 구조는 데이터를 어떤 식으로 저장하고 관리할 것인지를 말한다. 자료 구조의 의미는 프로그램에서 사용하는 데이터를 어떤 식으로 저장하고 어떤 식으로 사용할 것인지 정해진다. 자료구조는 곧 알고리즘과 연결된다. 알고리즘의 흐름에 따라서 데이터를 어떻게 저장할지가 결정된다. 프로그램에서 데이터를 어떻게 다룰 것인가 등은 프로그램 개발에서 중요한 부분이다. 대표적인 자료구조에는 스택과 큐, 배열과 링크드 리스트, 트리 형태 등이 있다.

스택 vs 큐

1. 예상 질문

자료 저장 형태 중 스택과 큐에 대해서 설명하시오. 스택과 큐는 어떤 차이가 있는지 설명하시오. 스택과 큐를 적용할만한 예를 들어 설명하시오.

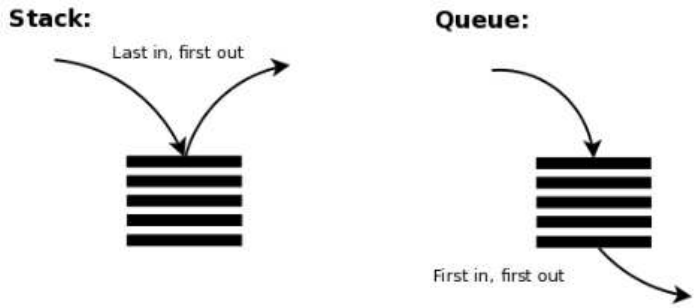
2. 이 질문을 하는 이유

스택과 큐는 자료 구조의 기본적인 개념이다. 각각 적합한 활용처가 있으며, 그 활용처가 무엇인지 숙지해야한다. 코딩의 기본이 되는 요소이기도해서 코드 레벨로 구현할 수도 있어야 한다. 기본 중에 기본이므로 반드시 숙달해야한다.

3. CS 지식

책이 책상 위에 쌓여 있다고 가정하자. 제일 아래에 있는 책을 꺼내는 것보다는 제일 위에 있는 책을 꺼내는 것이 더 쉽다. 이런 방식으로 자료를 저장하고 사용하는 형태의 자료구조를 스택이라고 한다. 쌓여 있다는 의미이기도 하다. 스택에 데이터를 입력했다가 가져오게 되면 가장 위의 데이터를 먼저 가져오게 된다. 그리고 다시 입력 후에 다시 가져올 때도 가장 마지막에 입력된 데이터를 가져오게 된다. 이렇게 먼저 쌓인 것은 나중에 꺼낼 수가 있고, 나중에 쌓은 것을 먼저 꺼낼 수가 있다고 해서 FILO (first in last out) 이라고 한다.

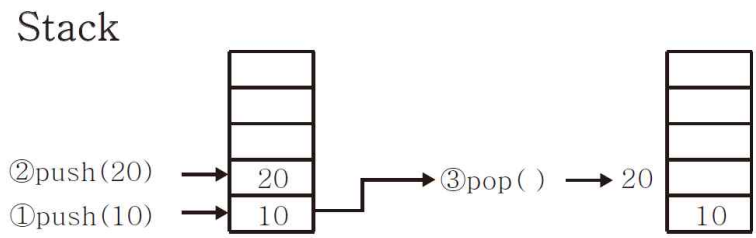
큐는 줄을 서는 개념이다. 먼저 줄을 서 있는 사람이 먼저 기회가 주어진다. FIFO(First in first out) 큐는 데이터가 들어가 있고, 먼저 들어갔던 데이터를 먼저 꺼내서 사용한다.

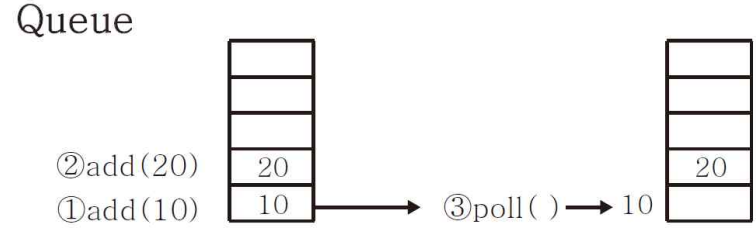


개념을 알았다면 구현을 어떻게 할 것인가. 사실 스택이나 큐는 많은 언어들에서 이미 라이브러리에 함수로 구현이 되어 있다. 자바의 경우도 다양한 큐들이 있고, 선택해서 사용하면 된다. 하지만 스택, 큐가 어떻게 구현되어야 하는지를 생각하고 구현할 줄 아는 것도 중요하다.

스택, 큐 둘 다 배열을 이용하여 구현 할 수 있다. 배열의 현재 위치를 가리키는 인덱스 변수를 하나 두고, 현재 위치를 변경하면서 값을 반환하거나 입력받으면 된다.

만일 둘 다 숫자를 입력받는다고 하면 숫자를 넣을 수 있는 push 함수와 값을 가져오는 함수로 구성할 수 있다. 내부적으로 add를 하게 되면 값을 저장하고, 인덱스의 위치를 바꿔주고, 값을 가져오는 함수는 현재 위치가 가리키는 곳의 값을 가져오면 된다.





Java 언어의 경우에도 Stack이나 Queue 객체가 있고, 그 객체를 이용해서 간단하게 사용할 수 있다. 실제 프로젝트를 할 때에도 그런 함수를 이용하면 된다. 하지만 그 내용이 어떻게 구성되어 있을지를 생각하는 것은 프로그램의 기본기가 있는지 확인하는 좋은 방법이기도 하다.

배열 vs 링크드 리스트

1. 예상 질문

배열과 링크드 리스트에 대한 차이점을 설명하세요. 배열과 링크드 리스트의 장단점을 생각하며 활용하면 더 좋은 예제를 말해보세요. 이진 트리 자료 구조를 배열로 구현하는 방법에 대해서 말해보세요.

2. 질문 의도

배열과 링크드 리스트는 자료가 저장되는 형태 중 가장 기본적인 형태이다. 둘의 차이점과 장단점에 대해서 정확하게 이해하고, 활용할 수 있어야하고 코드로 능숙하게 구현할 수도 있어야 한다.

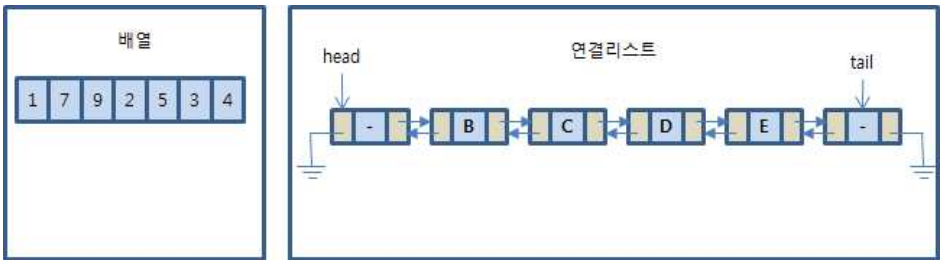
3. CS 지식

배열의 의미는 무언가가 나열되어 있다는 표현이다. 배열은 옆으로 나란하게 있는 모습을 말한다. 자료 구조는 '자료를 어떤 식으로 저장할까'의 방법이다. 그 대표적인 것이 배열이다. 옆으로 나란하게 저장하겠다는 의미이다. 숫자이든 문자이든 옆으로 나란하게 저장해야 하는 경우가 상당히 많다. 예를 들어 이러한 문장을 저장한다고 해도 옆으로 나란하게 저장하는 것이 맞고, 여러 가지 숫자를 저장한다고 해도 연속해서 저장하는 게 좋다. 그 이유는 연속된 숫자들을 가지고 하는 일이 많이 있기 때문이다. 예를 들면 학생들의 키를 저장해 놓은 경우 다음 학생의 키를 가지고 올 일도 있고, 그 키에 대한 수를 평균을 내기도 하고, 그러한 행위가 연속된 저장공간에 있는 것이 유리한 상황이 많다.

배열은 메모리 어느 위치에 한 개씩 저장되는 연속이라고 생각하면 된다. 예를 들면 1, 2, 3, 4, 5와 같은 숫자를 저장한다고 하면 이 값들이 메모리 내 연속된 주소에 저장이 된다. 연속되어 있다는 것은 바로 인근 저장소로 생각하면 된다.

배열은 a[] 와 같이 표현하고 위치를 나타낼 때는 a[1] 이런 식으로 위치를 가리킬 수 있다. 연속해서 값을 가져오고 싶을 때는 for, while과 같은 반복문을 사용하고, a[i] 와 같은 식으로 가리키는 위치 값을 바꿔주면 된다.

저장하는 기능 자체는 유사한데, 다른 방식으로 저장되는 것이 링크드 리스트이다. 링크드 리스트는 연결 리스트라고 하는데, 저장된 값과 다음 값을 연결하면서 값들을 저장한다.



[그림] 배열과 링크드 리스트

배열과 링크드 리스트는 값을 저장한다는 점에서는 유사하지만, 다른 점은 저장하는 방식이다. 그리고 여기서 더 중요한 것은 왜 다른 방식이 필요한가이고, 서로 간의 장단점을 아는 것이 중요하다. 꼭 배열이 더 간단하고 편리하고 좋은가 하면 배열의 단점이 있다. 배열은 미리 할당을 받아 사용을 한다. 100개 정도의 값을 저장할 것이라고 생각하고 미리 할당을 받아서 선언하다. 그런데, 두 가지 경우가 있을 수 있다. 100개를 사용한다고 해놓고선 사용하지 않는 경우이다. 이 경우는 사용하지 않은 만큼 자원이 낭비가 된다. 미리 할당을 받았기에 메모리를 사용한다고 보고 있고, 다른 변수들은 사용할 수 없게끔 확보가 된다. 자원 부족을 일으킬 수 있다. 그리고 또 하나의 경우는 100개를 사용한다고 예상했지만, 사용하다보니 더 필요한 경우가 생길 수 있다. 그런 경우 메모리를 조금씩 늘려서 사용한다는 것이 용이하지는 않다. 100개 이상의 데이터가 필요하면 배열을 필요한 만큼 다시 선언하고 그 곳으로 기존 내용들을 다 복사해야 하기 때문이다. 그러다 보니 꼭 필요한 만큼이 거의 완전하게 예측되는 경우에 사용하는 것이 좋다. 그리고 값의 입출력은 용이하나 중간에 값을 삽입한다든지 삭제 했을 경우 그 뒤에 있는 값들을 옮겨줘야 하는 경우라면 번거롭고, 오히려 프로그램 복잡도가 높아질 수 있다.
반면 링크드 리스트라면 배열이 가지고 있는 단점을 보완할 수가 있다. 사용량을 처음부터 설정해야 하는 것이 아니라 포인터로 가지고 있고, 필요한 만큼을 할당 받아서 연결시키는 방식이기 때문에 배열과 같은 메모리 관련 문제가 발생하지 않는다. 그리고 데이터 중간에 추가, 삭제가 용이하다. 기존 값을 가리키던 곳의 포인터를 추가된 값을 가리키도록 변경하고, 추가된 값을 가리키던 곳의 포인터를 삽입될 위치 뒤의 값을 가리키도록 변경하면 된다. 프로그램이 처음부터 조금 복잡해질 수는 있지만, 확장이 용이하다는 점에서는 어떤 프로

그램에서는 꼭 링크드리스트로 사용해야 하는 경우가 있다.
예를 들면 도서관리 프로그램을 개발해야 한다고 하면 도서가 몇 개일지 미리 정하는 것보다는 도서는 언제든지 신규 도서가 들어 올 수 있으므로 확장 가능하게 작성하는 것이 좋다. 물론 데이터베이스라는 프로그램들을 통해서 데이터가 유연하게 처리되지만, 프로그램에서도 특정 데이터를 읽어들일 때에도 유연하게 처리할 필요가 있다.
물론 링크드 리스트 구조가 장점만 있는 것은 아니다. 단점은 배열에 비해서는 느린 속도를 가진다. 다음 값을 가져오기 위해서는 링크드 리스트가 가르키는 다음 값을 찾아가야 하기 때문이다. 이는 시간 복잡도 상의 느리다가 아니라 메모리 상 값을 가져오는 물리적 속도가 느리다라는 관점이다.
배열과 링크드 리스트는 기본적인 프로그램 자료구조이고, 그 차이점에 대해서는 장점과 단점 그리고 어떨 때 사용하는 것이 좋다는 관점을 꼭 이해하는 것이 좋다.

트리 구조

1. 면접 예상 질문

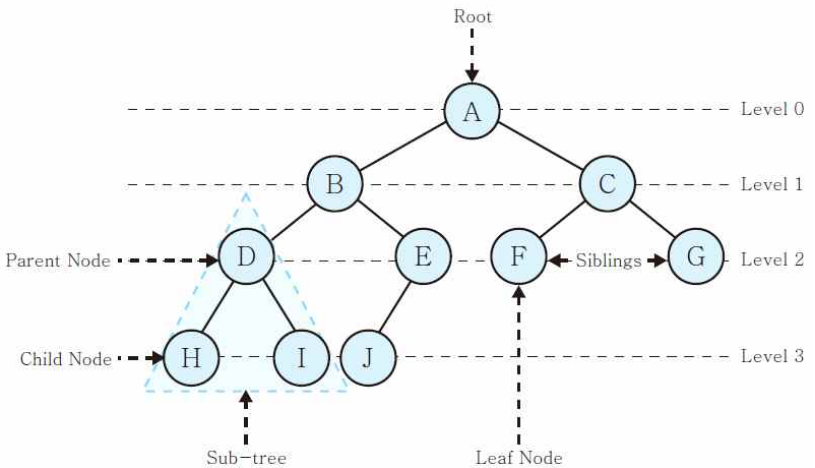
자료 구조 형태 중 트리 구조에 대해서 설명해보시오. 왜 트리 구조를 사용하는지 말해보시오. 트리 구조의 장점을 활용한 사례를 말해보시오.

2. 질문 의도

자료 구조의 기본 형태 중 하나로 많은 알고리즘에서 사용하는 구조이다. 왜 트리에 저장해야 하는지 이해하고, 실제 사용해보면서 어떻게 다룰 수 있는지도 파악하는 걸 권장한다. 트리 구조는 데이터를 단순 링크드 리스트나 배열에 저장하는 것보다 훨씬 효율적으로 저장하고 검색할 수 있어서 알고리즘에서 매우 중요하게 다뤄진다.

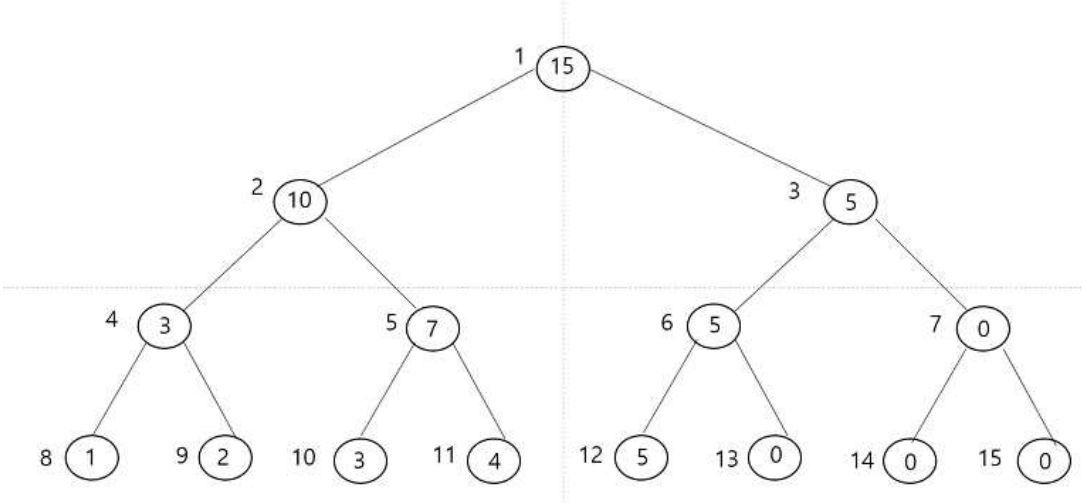
3. CS 지식

알고리즘 책을 보거나 강의를 듣다 보면 '트리'라는 용어를 자주 접하게 된다. 자료구조 책을 잠깐 봐도 트리 모양의 그림이 그려진 페이지들을 볼 수 있다. 트리는 나무를 나타내는데, 데이터를 이런 나무의 형태로 저장한다는 의미이다.

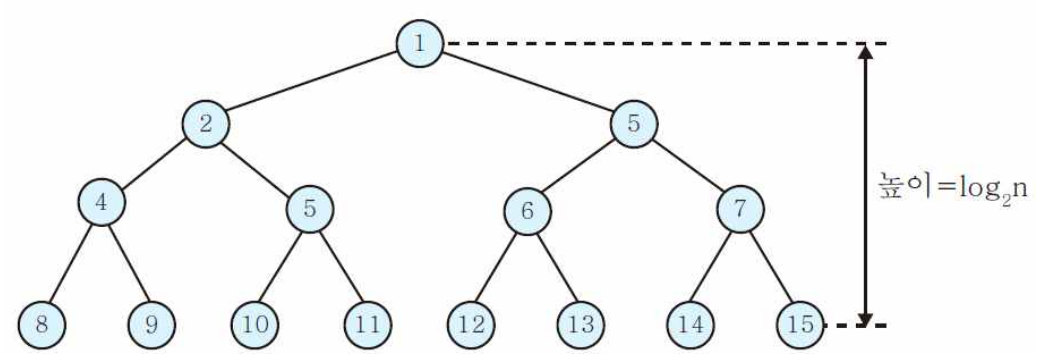


[그림] 트리 구조

나무에는 뿌리와 잎이 있듯이 자료구조를 저장하는 트리에도 root와 leaf node가 있다. 트리를 그냥 공부했겠지만, 사실은 왜 트리인가를 생각해보면서 이해하면 트리에 대해서 확실하게 이해할 수 있고, 활용도 잘 할 수 있게 된다. 사실 여러 가지 언어에서 트리를 직접 구현할 일은 많지 않다. 대부분 트리에 대해서 이미 구현되어 있고, 라이브러리로 제공이 된다. 그런 것을 이용하여 데이터를 저장하거나 활용하면 된다. 하지만 트리 구조에 대해서 이해하는 것은 프로그램 개발의 이해도가 높아지고, 알고리즘 시험 같은 곳에서는 나오는 문제를 해결할 수도 있다.
트리는 트리와 같은 형태로 데이터를 저장하는 자료 구조의 일종이다. 자료를 저장할 때 왜 트리 모양으로 저장해야 하는가. 무엇이 유리한가 생각해 볼 수 있다. 트리는 부모와 자식 노드들로 이뤄진다. 부모는 위쪽에 있고, 자식은 아래쪽에 위치 한다. 그리고 특정 기준에 따라서 부모의 아래 오른쪽 또는 왼쪽에 자식 노드들이 연결된다. 트리 구조가 꼭 자식 노드가 2개여야만 하진 않는다. 여러 가지 다양한 트리 형태가 있다. 자식 노드가 하나만 있는 경우, 또는 3개 이상을 가진 경우도 있다. 이들 모두가 트리라고 한다. 어떤 노드가 가지고 있는 자식의 수를 그 노드의 차수라고 한다. 그래서 3개의 노드를 가지는 경우는 차수를 3이라고 한다. 2개의 자식 노드로 구성된 트리는 특별히 이진 트리라고 한다.
트리는 배열이나 링크드 리스트의 자료구조들을 통해서 구현할 수가 있다. 트리에 저장하면 유리한 이유는 트리로 저장한 뒤에 순회라고 하는 트리를 탐색하는 과정에서 몇몇 가지 단순 배열이나 링크드 리스트가 갖지 못한 장점들이 생긴다.
대표적인 활용사례는 인덱스 트리 또는 정렬을 들 수가 있다.



인덱스 트리는 트리의 노드들에 인덱스가 있다고 가정하고 자식 노드들에 값이 저장될 때 해당하는 부모 노드에 자식 노드들의 값들의 합들을 넣어둔다. 그렇게 해서 상위 부모 노드들까지 해당 자식들의 값을 모두 더 해 놓는 경우 자식들 중 일부분에 대한 합을 구할 때 그 계산 속도가 빨라지게 된다. 자식 노드들의 구간 합이 특정 위치의 부모 노드에 저장된 값이기 때문이다. 연산이 빠른 속도로 된다.
그리고 정렬 상태를 유지할 수 있는 장점이 있다. 정렬 상태를 유지하기 위해서 내부적으로는 heap 형태의 트리 구조를 사용하는데, heap 구조의 트리라는 것은 저장된 값 중 최솟값 또는 최댓값을 리턴하고 제외한 값의 나머지를 다시 정렬로 저장하는 방식을 말한다. 이렇게 저장을 유지하게 됨으로써 이를 활용하여 priority Queue (우선순위 큐라고도 하고, 큐 형태의 자료구조 이면서 특징은 숫자를 넣게 되고, 그 숫자가 항상 정렬이 유지된다는 것이 특징이다) 와 같이 항상 정렬 상태를 유지하는 구조에 활용될 수 있다.
이러한 트리를 사용하는 이유 중 하나는 시간 복잡도라는 문제의 해결에 있다. 시간 복잡도는 알고리즘에서의 가장 중요하게 생각하는 부분이기도 하다. 시간 복잡도는 시간이 얼마나 복잡하다는 의미라기 보다는 얼마나 오래 걸리냐 그리고 그 오래 걸리는 방식이 왜 그러한가를 계산해보는 것이다. 1에서 N까지의 숫자를 더한다고 해서 for문을 사용해서 더하는 것은 시간복잡도 상 N까지 반복을 해야 하기에 시간 복잡도를 O(N) 이라 할 수 있다. 그리고 이중 반복이 되어진다면 N의 제곱 시간복잡도 O(N^2) 에 해당한다. 이러한 시간 복잡도를 낮추는 노력은 알고리즘 개발에 꼭 필요한 과정이다. 일련의 배열에 저장된 값을 찾는 것은 특별한 알고리즘을 사용하지 않으면 1~N 중 값을 찾아야 하니 O(N)의 시간복잡도를 갖는다. 트리에 저장되어 있는 경우는 logN 시간으로 줄어들 수 있다. 갑자기 log가 나와서 당황스러울 수 있지만, N=8인 경우를 생각해보면 숫자가 8개 있다는 의미이고, 이 경우 특정 숫자를 찾는데는 최악의 경우 8번이 걸린다. 시간복잡도는 항상 최악의 경우를 포함해야 한다. 그런데, 트리에 저장되어 있는 경우 3번 만에 특정 숫자를 찾을 수 있다. 2라는 숫자를 찾는 경우 4를 기준으로 아래 그리고 그 다음 아래 1, 2 중 위쪽 숫자를 선택하면 된다. 2라서 빨리 찾았다고 볼 수 있고 그러한 경우 배열도 2번 만에 찾는다고 생각할 수 있지만, 8을 찾는다고 하면 배열의 경우 8번만에 찾을 수 있고, 트리에 저장된 경우는 4보다 위쪽 그리고 6보다 위쪽 7, 8 중에 8을 찾을 수 있다.



이런 식의 여러 가지 장점을 가지고 있어서 알고리즘에서는 사용되는 곳이 많다. 그렇게 사용되어지는 것이 많은 만큼 내용에 대한 이해가 필요하다.

(2) 알고리즘

알고리즘은 프로그램 개발과 직접적인 연관이 있다. 프로그램을 개발한다는 것이 어쩌면 알고리즘을 만든다라고 할 수 도 있다. 그렇다고 프로그램이라는게 복잡한 알고리즘만 있는 것은 아니지만 그만큼 프로그램을 개발한다는 것은 여러 가지 알고리즘을 포함하고 있다.

그렇다고 프로그램 개발이라는게 알고리즘들이 복잡하게 개발을 해야 하는 것은 아닐 수도 있다. 그리고 매번 알고리즘을 새로 만들어서 개발을 하진 않는다. 알고리즘을 사용할 상황이 되어도 이미 기존의 라이브러리에서 훌륭한 함수를 제공하기도 한다. 그래도 역시 프로그램 개발에 있어서 몇 개의 알고리즘 지식 또는 그에 버금가는 알고리즘적 사고가 필요하다.

정렬

1. 면접 예상 질문

정렬 방식 중 알고 있는 방식을 말해보시오. 정렬 방식의 종류 중에서 가장 효율적인 정렬 방식을 말해보시오. 퀵 정렬에 대해서 말해보고, 어떤 장점이나 효율이 있는지 말해보시오.

2. 질문 의도

데이터는 자료 구조 내에 저장되고 검색된다. 데이터를 효율적으로 검색하기 위한 여러 가지 방법이 있는데 대표적인 것이 '정렬'이다. 자료를 정렬하면 검색 속도가 향상된다. 이 정렬에는 다양한 방법이 있고 각 방법은 장단점을 지니고 있어서 상황에 맞게 정렬 알고리즘을 사용해야한다. 그러므로 대표적인 정렬 알고리즘을 잘 이해하고 적용하는 능력이 필요하다.

3. CS 지식

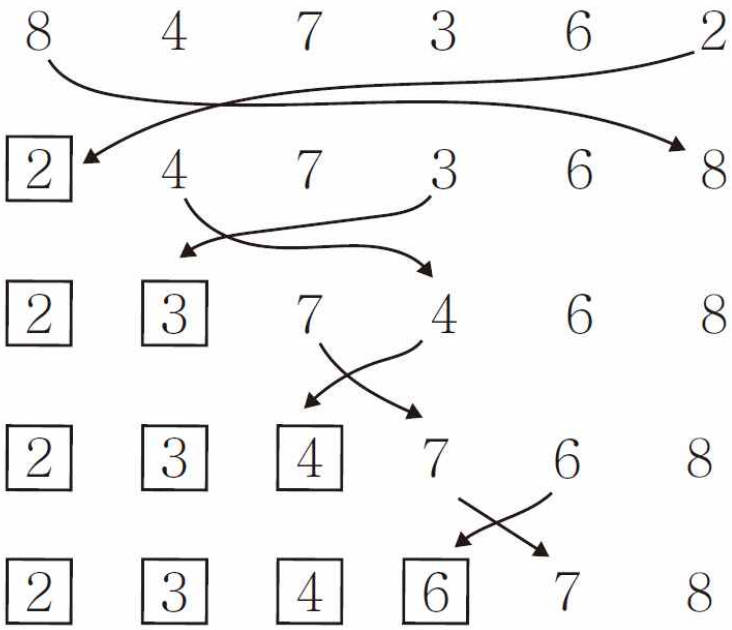
정렬은 알고리즘에서 가장 기본적인 것이라고 할 수 있다. 그리고 정렬은 많은 상황에서 필요로 한다. 쇼핑몰만 들어가봐도 가격으로 정렬하기도 하고, 평점으로 정렬하기도 한다. 오름순위로 내림순위로 정렬하는 것이 여기서의 정렬이다. 정렬을 어떻게 할 것인가가 알고리즘적인 숙제이다.

정렬을 하는 것을 어떻게 할지 생각해보면 대표적으로 과거에는 학교에서 키 순서대로 앞자리부터 앉게 하곤 했는데, 그러려면 학생들 모두 일어서게 하고, 한 줄로 선 다음에 두 사람씩 서로 키를 비교해보고 키가 작은 사람을 앞으로 보내면 된다. 그렇게 하다 보면 몇 번의 반복 끝에 키 순서대로 서게 될 것이다. 이것은 모든 학생이 사람이고 거의 동시에 서로 간의 키를 대조해서 앞으로 보내주면 되기 때문에 비효율적으로 보이진 않는다. 이러한 정렬을 공기 방울이 물 안에서 위로 올라오는 형상과 비슷하다고 하여 버블 정렬이라고 한다. 그런데, 알고리즘적으로 이 방식이 아주 효율적인 것은 아니다.

이런 식의 알고리즘적 고민은 프로그램 개발에 있어서 기본적으로 생각해야 할 필요한 지식이고, 또 그런 식의 효율적인 사고를 할 수 있는가는 프로그래머의 역량이라고 생각할 수도 있어서 중요하게 생각한다. 귀찮으니깐 그냥 반

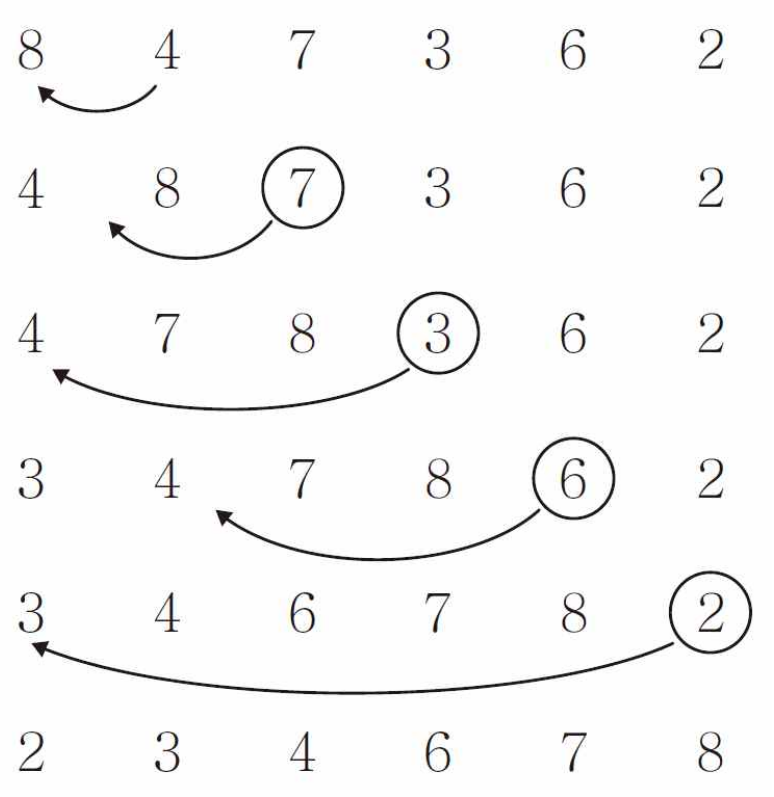
복문 3번 돌려서 프로그램 개발하면 된다는 식이라면 곤란한 상황으로 빠질 수도 있다.
버블정렬보다 효율을 높인 정렬은 몇 가지가 있다. 선택 정렬, 병합 정렬, 삽입 정렬, 퀵 정렬 등이 있다. 정렬 방식이 조금씩 차이가 있는 것인데, 효율적인 것은 퀵정렬, 병합 정렬이다. 알고리즘별로 어떻게 하는지 간단히 알아보고 대표적인 병합 정렬은 자세히 알아두면 좋겠다.

• 선택 정렬: 여러가지 숫자들이 있고, 오름 차순으로 정렬을 한다고 했을 때 데이터들 중에서 가장 작은 숫자를 찾아서 가장 앞으로 위치 시키고, 나머지 데이터 중에서 가장 작은 데이터를 그 다음 위치시키고 하는 식으로 순서를 찾는 것이다. N^2의 시간 복잡도를 가진다.



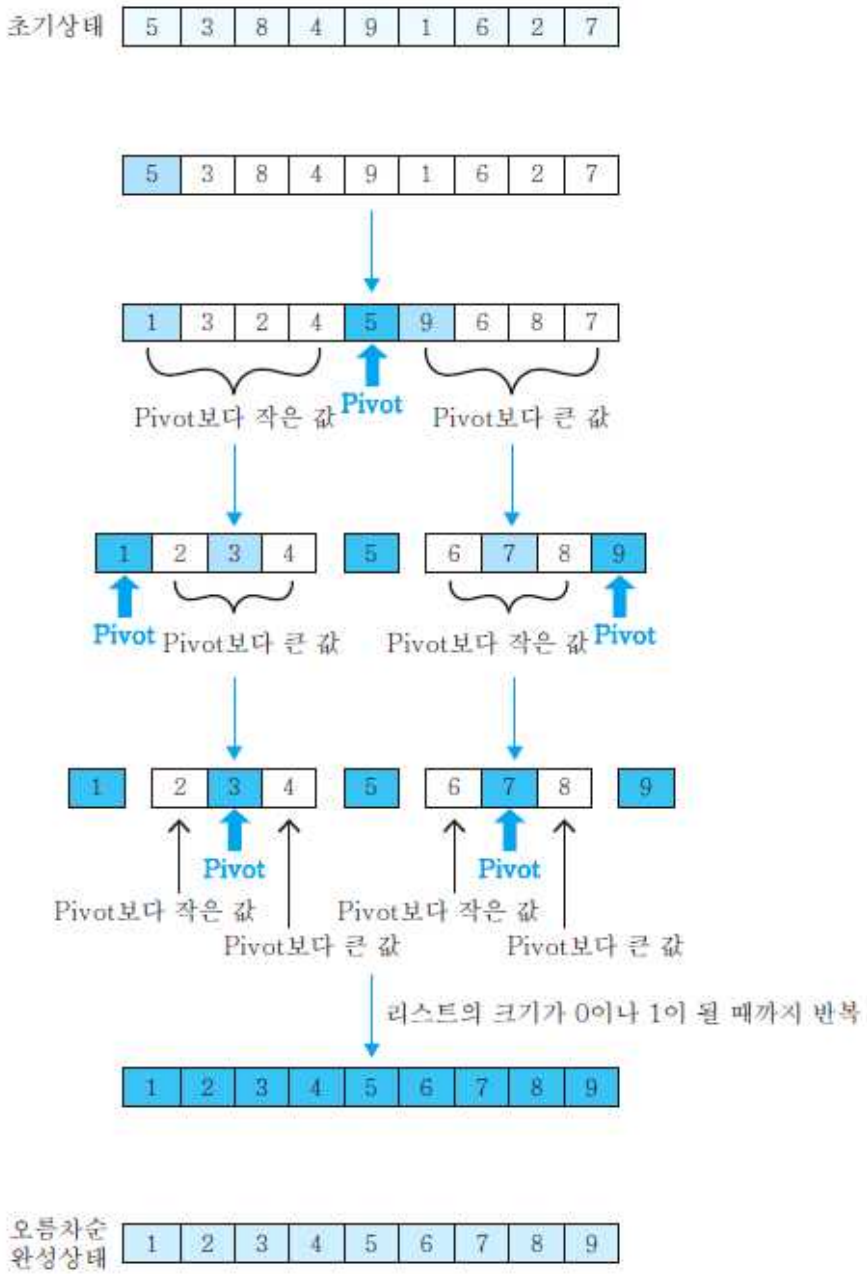
선택 정렬 예) 첫 번 째 줄의 숫자들 중 가장 작은 2를 맨 앞으로 위치 시킨 다음 그 자리에 있는 8과 교체 그 다음 작은 수를 찾아서 교체를 반복

• 삽입 정렬: 숫자들을 정렬할 때 앞에서부터 진행을 할 때 그 범위를 늘려가면서 다음 나오는 숫자를 맞는 위치에 삽입을 시켜주면서 진행을 한다. 만약 앞에 정렬된 숫자들 보다 크면 그대로 넘어가고 작다면 해당 위치를 찾아줘서 넣는다. 그러다 보면 선택 정렬보다는 빠른 성능을 가질 수 있다. 하지만 최악의 경우를 생각해본다면 O(N^2)의 시간 복잡도를 갖게 된다.

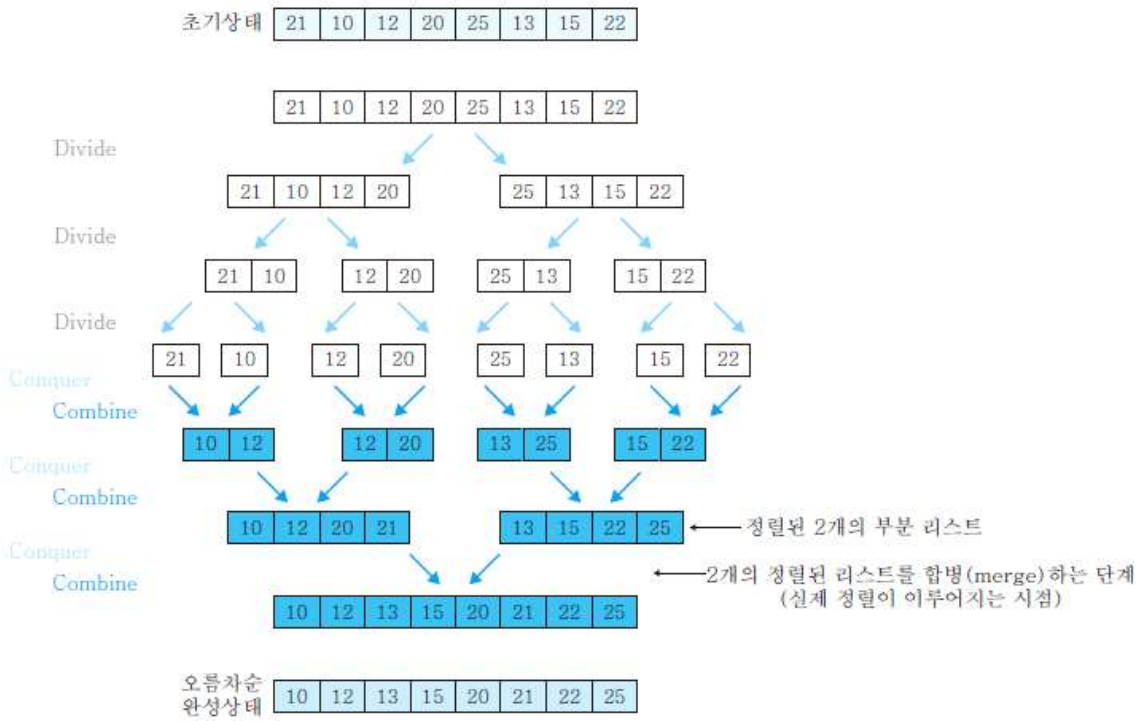


삽입 정렬 예) 4가 8보다 작으니 삽입, 7이 8보다 작으니 삽입
앞 숫자들 들어갈 위치에 삽입 이것을 반복

• 퀵 정렬 - 숫자들 중 한 숫자를 정한다. 다른 숫자들을 그 숫자보다 큰 부류와 작은 부류로 나눈다. 큰 부류는 오른쪽, 작은 부류는 왼쪽에 놓는다. 나눠진 부류에서 위를 반복한다. 더 이상 부류로 나눌 수 없을 때 까지 진행한다.



• 병합 정렬 - 병합 정렬은 전체 정렬하고자 하는 대상을 분리해서 정렬을 하고 다시 합치는 방식이다. 8개의 숫자를 정렬한다고 했을 때 4씩 2개로 앞뒤 리스트를 분리하고, 또 2개씩 4개로 또 분리한다. 그렇게 되면 2개의 숫자 중 크기를 비교할 수 있고, 2개씩 정렬된 그룹들 간에 요소를 비교하면서 또 정렬을 할 수가 있다. 그렇게 비교를 하면서 다시 채워 넣고 합치는 방식이다.



병합 정렬이 알고리즘에서 말하는 Divide and Conquer의 예라 할 수 있다. 문제를 작게 분할하고, 해결해 간다는 의미이다. 이러한 방식은 구현을 할 때 재귀호출이라는 것을 하게 되는데, 재귀 호출은 자기 자신을 다시 호출하는 것을 의미하는 것으로 문제를 작게 나눈 것을 계속적으로 호출하면서 풀어간다.

재귀호출의 대표적인 예가 factorial

int factorial(int n) (참조 코드로 컴파일 환경이나 개발 언어에 따라서 다를 수 있다.) { if (n == 1) // n이 1일 때 return 1; // 1을 반환하고 재귀호출을 끝냄

return n * factorial(n - 1); // n과 factorial 함수에 n - 1을 넣어서 반환된 값을 곱함
}

시간 복잡도가 좋은 것은 아니지만, 코드를 간단하게 문제를 풀 수가 있다. 병합 정렬도 이렇게 분할하는 것을 다시 재귀호출을 해줌으로써 구현이 가능하다.

// 합병 정렬 (참조 코드로 컴파일 환경이나 개발 언어에 따라서 다를 수 있다.) void merge_sort(int list[], int left, int right){ int mid;

if(left<right){ mid = (left+right)/2 // 중간 위치를 계산하여 리스트를 균등 분할 -분할(Divide) merge_sort(list, left, mid); // 앞쪽 부분 리스트 정렬 -정복(Conquer) merge_sort(list, mid+1, right); // 뒤쪽 부분 리스트 정렬 -정복(Conquer) merge(list, left, mid, right); // 정렬된 2개의 부분 배열을 합병하는 과정 -결합(Combine) } }

구현 관점으로 재귀호출을 통해서 이렇게 구현한다는 것을 아는 것이 개발 능력에도 중요하고, 병합 정렬이 정렬 중 가장 효율적인 방식이라는 점도 중요한 부분이다. 최악의 경우에도 NlogN의 시간 복잡도를 가진다.

동적 계획법

1. 면접 예상 질문

• 동적 계획법으로 문제를 해결한다는 것은 어떤 식으로 문제를 푸는 것인지 말해보세요. 점화식에 대해서 말해보고, 점화식으로 푸는 예제를 말해보세요. 메모이제이션에 대해서 말해보고, 왜 이런 방식으로 문제를 푸는 것이 효율적인지 말해보시오.

2. 질문 의도

알고리즘 중에 중요한 해결 방식인 동적 계획법에 대해서 이해하고 있는지 여부를 통해 알고리즘에 대한 이해 정도를 가늠할 수 있기 때문에 알아둬야 하는 내용이다. 동적 계획법에 필요한 점화식은 문제마다 다르기 때문에 상황에 맞게 생각해내야 하는데, 이러한 방식이 있다는 사실 자체를 아는 것은 사전 지식으로써 필요하다. 개발을 효율적으로 하기 위해서 필요한 지식이 될 수 있어서 알아둬야 할 필요가 있다.

3. CS 지식

동적 계획법은 문제를 풀어가는 방법이다. 특정 문제들을 풀 때 알고리즘을 사용하면 쉽게 풀리기도 한다. 이 방법은 어떤 큰 문제가 있다고 하면 그 큰 문제를 풀려고 할 때 작은 부분으로 나눈다. 그리고 작은 부분이 반복되는 패턴이 있는지 살핀다. 그 작은 패턴을 푸는 것이 모여서 큰 문제를 풀 수 있는지를 생각한다. 그리고 그것을 일종의 수식으로 만들어 낸다. 이를 점화식이라고 한다. 그 작은 불씨가 전체를 태워버린다고 기억해도 좋을거 같다.

그리고 중요한 부분은 이 반복되는 것의 결과 값을 저장해두고, 다음 동일한 계산 요청 시에는 저장된 값을 반환하여 계산 속도를 개선한다. 이렇게 계산 결과 값을 기억한다고 해서 메모이제이션이라고 한다. 한번 계산한 것을 배열과 같은 저장장소에 기억해 놓는다는 의미다. 여러번의 반복 계산이 있다고 할 때 다시 반복하지 않고, 저장된 값을 사용한다.

동적 계획법 문제를 조금 간단한 예제로 생각해보면 다음과 같다.

Factorial 문제를 재귀함수로 구현이 가능하다. 그런데, 이 재귀함수를 매번 사용한다고 했을 때 예를 들면 10!을 구하는 사용을 하고 또 15!을 구한다고 했을 때 재귀함수를 10!만큼 하고, 그 뒤에 동일한 함수를 호출하여 15!까지 함수를 호출하게 되면 계속 반복되어지는데, 예를 들면 10!을 계산하고 나서 그 값을 저장해둔다. 그리고 15!이 호출되면 10!까지 결과 값이 있으므로 그 값에 15부터 11까지만 곱해주면 된다. 물론 15! 이후에 그 이하의 숫자는 모두 저장되어 있고, 그 위의 숫자가 불렸을 때 더해진만큼 저장해두고 사용하면 최초 불릴 때는 시간이 걸리겠지만, 이후에는 빠른 속도로 값을 알 수가 있다. 기존 재귀로 구현했던 함수를 아래와 같이 개선할 수 있다.

int f[N]; int factorial(int n) { if (n == 1) // n이 1일 때 return 1; // 1을 반환하고 재귀호출을 끝냄

if(f[n] != 0) return f[n];

return f[n] = n * factorial(n - 1); // n과 factorial 함수에 n - 1을 넣어서 반환된 값을 곱함
}
(이 소스는 참고 코드로 언어나 컴파일러에 따라 다를 수 있음)

동적 계획법으로 가능한 알고리즘들이 많이 있고, 이것을 생각해 내는 것은 쉽지만은 않다. 이러한 복잡한 문제를 단순화 하여 문제를 풀어가는 것도 프로그램의 중요한 부분 중 하나이다.

(3) 보안

IT에서 보안은 중요하다. 보안이 중요한 이유는 해킹 시도라든가, 데이터 유출과 같은 문제가 발생할 수 있기 때문이다. 보안 기술에 대해서는 먼저 해킹의 방법에 대해서 알고 있어야 한다. 그래야 사전에 해당하는 해킹 방법의 시도를 막을 수가 있다. 보안을 안다고 해서 보안 솔루션을 만들거나 보안 관련 프로그램을 개발에서만 필요한 것이 아니다. 프로그램 특히 웹사이트나 데이터베이스 관련해서는 보안은 기본이다. 매번 프로그램 개발 시 발생되는 버그처럼 개발 때마다 생기는 문제는 아니지만, 한번의 문제가 대형 문제로 될 수 있어서 매우 중요한 부분이 된다. 개발자가 모두 염두해야 하는 분야이다.

SQL Injection

1. 면접 예상 질문

- SQL Injection에 대해서 말해보시오.
- SQL Injection은 어떤 상황에서 발생될 수 있는지 말해보세요.
- SQL Injection을 보완하기 위한 방법은 무엇인지 말해보세요.

2. 답변 핵심 KEYWORD 및 질문 의도

SQL, 데이터베이스

보안은 모든 개발자가 알아야 하는 기본적인 내용이다. 프로그램 개발에 있어서 보안을 염두에 둔 개발이 필요하다. 보안 지식을 통해 개발 시 유의할 점에 대한 사전 지식이 있는지 알아볼 수 있다. SQL Injection은 웹 개발 또는 DB에 접근 가능한 모든 상황에서 고려해야 하고, 관련 가이드도 미리 숙지하고 있어야 큰 규모의 시스템 사고를 막을 수가 있다. 분야를 막론하고, 보안 의식과 기술에 대한 이해, 방어 전략을 알 필요가 있다.

3. CS 지식

SQL는 데이터 베이스를 조회/수정/삭제 할 수 있는 명령어 집합이다. 대부분의 데이터 베이스는 이런 명령을 통해서 데이터를 가져오고나 수정하거나 삭제할 수 있다.

사이트나 서비스는 대부분 그 데이터를 데이터베이스에 저장된다. 사용자에 의해서 데이터가 조회되거나 수정/삭제가 된다. 그러한 기능을 위해 프로그램 또는 사이트 등에서 SQL 명령어를 데이터 베이스에 전달하게 된다. SQL Injection은 SQL 명령어 전달 과정에서 이 SQL 문장에 특별한 어떤 명령어들을 포함시켜서 원래 프로그램이 의도한 것이 아닌 명령들을 보내 시스템을 해킹할 수가 있게 된다.

그럼 어떻게 이 과정이 가능할까. 데이터 베이스 시스템 자체를 접속하거나 뚫고 들어가기는 어렵다. 하지만 SQL 명령이 전달되고 있는 연결이 되어 있는 곳에 데이터를 변조해서 넣는 것은 가능할 수가 있고, 이를 통해서 해킹 시도가 이뤄질 수가 있다. 프로그램과 데이터베이스는 연결이 되어 있고, 이 연결

된 것으로 명령어들이 주고 받기 때문이다.
그렇다면 어떻게 명령어를 변조할 수 있었을까 ? 웹사이트 내에는 여러가지 사용자에 의해서 입력 받을 수 있는 항목들이 있다. 이때 데이터를 입력하고 확인 버튼 등을 누르게 되면 그것이 데이터 베이스 시스템에 입력이 된다. 입력이 되어지는 것은 대부분 명령어가 조합되어서 데이터 베이스로 전달이 된다. 이 입력 시 에디트 박스에 스크립트형 문장을 입력하거나 네트워크 전송이 되어질 때 URL 값을 변조하여 전달하므로써 프로그램이 원래 동작과 다른 수행을 하게끔 한다.
이때 해당 SQL 문장을 주입한다고 해서 SQL Injection이라고 한다. 데이터 베이스에 비정상적인 동작을 일으킨다. 이는 취약점을 틈타는 공격으로 공격 자체는 어렵지 않다. 예를 들면 Select * from table where id='id' and password='password'; 라고 했을 때

ID: admin Password: ‘ ’OR ‘1’ = ‘1’

Select * from table where id='admin' and password=' ' OR '1' = '1'; 라고 했을 때 로그인이 성공 될 수 있다.

따옴표를 올바르게 닫으며 pasword=' ' 만듦과 동시에 SQL 구문에 OR '1' = '1'을 붙이게 되면 논리가 OR가 붙게 되면서 '1' = '1'은 항상 참이 되어 로그인을 성공하게 된다.

이를 막기 위해서는 유저에게 받은 값을 바로 조합하여 SQL문장을 만들면 안된다. 그리고 값 속에 정규 표현식이라든가 값을 다시 한번 체크할 필요가 있다. 이렇게 SQL Injection이 무엇인지 인지하고, 입력받는 곳에서 문제가 없는지 생각해야 한다.

대칭키, 공개키

1. 면접 예상 질문

대칭키와 공개키의 개념의 차이는 무엇인지 설명해보시오. 대칭키와 공개키에 사용되는 알고리즘은 무엇이 있는지 말해보시오. 대칭키와 공개키를 사용했을 때 발생할 수 있는 문제는 무엇인지 말해보시오.

2. 답변 핵심 KEYWORD

암호키, 개인키, 부호화, 복호화

대칭키와 공개키는 전자인증에 관련된 보안 기술로서 이 기술을 통해 인터넷에서 상호 간의 신뢰할 수 있는 정보전달이 가능하다. 이 키를 활용한 인증을 통해서 신뢰할 수 있는 사용자인지 확인한 후 통신을 시작한다. 키를 공개할 것인지, 비밀로 유지하면서 통신 주체끼리 소유한 키를 대조할 것인지에 따라서 방법이 달라진다.

실데이터를 공유할 때는 대칭키를 이용하고, 대칭키를 공유할 때는 공개키 방식을 이용한다.

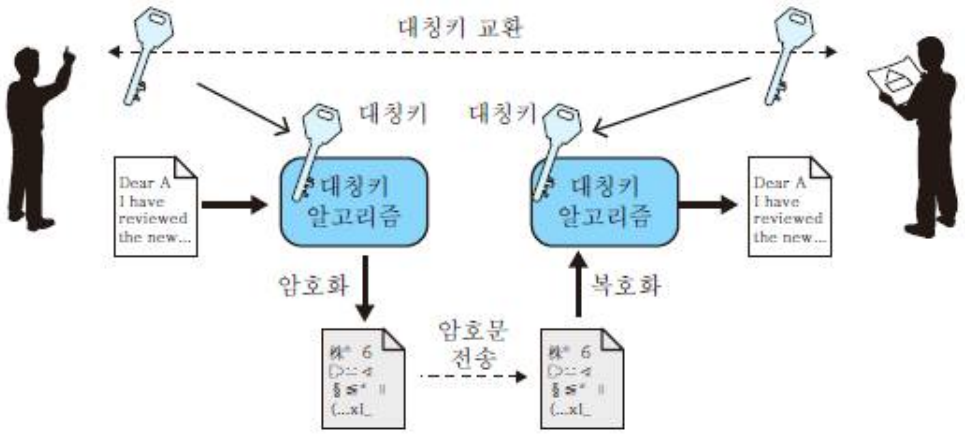
3. CS 지식

보안은 기밀한 데이터가 외부로 유출되지 않게 하는 것이다. 침입을 막는 보안도 있고, 보안은 외부에 유출이 되어도 암호화 같은 것이 되어서 다른 사람은 볼 수 없게 하는 것도 있다. 암호화는 이처럼 데이터를 아무나 볼 수는 없게 하는 기술이다. 그리고 암호화를 하여 데이터를 전송하여 볼 수 있는 권한이 있는 사람은 볼 수 있는 기능을 제공한다. 우리가 은행이나 증권사이트에서 로그인할 때 본인이라는 인증을 하는 것이 공인인증서인데, 이러한 인증서 개념도 이러한 암호화 기술을 근간으로 한다.

대칭키라는 말은 키가 대칭이 된다고 해서 대칭키이다. 대칭이 된다는 의미는 잠근 열쇠와 푸는 열쇠가 동일한 키이다. 이 키로 잠그고 풀고자 하는 사람도 동일한 키가 있어야 한다.

이러한 대칭키는 외부에 노출이 되어서는 안된다. 누구나 이 키만 있으면 해당 키로 암호화된 데이터를 풀 수 있기 때문이다. 이렇게 대칭키는 키로 암호화하고, 해당 키로 복호화를 하는 개념이다. 암호화한 키는 교환 과정을 거쳐서

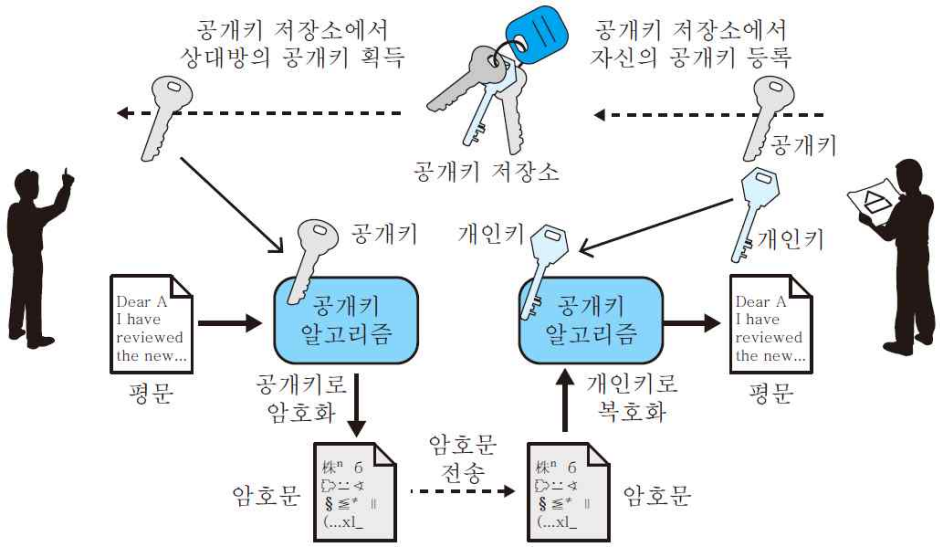
전달이 된다.



[그림] 대칭키 전달 과정

대칭키 암호화 방식은 AES 방식을 사용한다. AES(Advanced Encryption Standard)는 1997년 DES를 대체하기 위한 대칭키 블록 암호 기술이다. DES(Data Encryption Standard) 대칭키 블록 암호화 기술로 가장 널리 사용되어졌다. 하지만 56bit라는 짧은 키로 되어 있어 해킹 우려가 있어 현재는 사용되지 않고 있다.

공개키는 키가 공개되어지는 부분이 있어서 공개키라고 부른다. 공개키로 암호화 된 것을 복호화 하기 위해서는 반드시 개인키가 있어야 한다. 공개키로 암호화를 해두고, 풀기 위해서는 개인키가 필요한 것이다. 공개키 또는 키가 비대칭이여서 비대칭키라고도 한다.



[그림] 공개키 전달 과정

비대칭키는 공개키(Public Key)와 개인키(Private Key)가 하나의 쌍을 이루고 있다. 다른 사람들에게 공개된 키로서 정보를 암호화할 수 있는 공개키와 사용자 본인만 알고 있어서 암호를 풀 수 있는 키인 개인키를 하나의 쌍을 이루고 있다. 비대칭키를 사용한 암호화 방식에는 공개키로 암호화를 하는 경우와 개인키로 암호화를 하는 경우로 구분할 수 있다. 공개키로 암호화를 선택했다면 데이터 보안에 중점을 둔 것이고, 개인키로 암호화를 진행했다면 안전한 전자서명을 통한 인증 과정에 중점을 둔 것으로 해석할 수 있다. 암호화할 때의 키와 복호화할 때의 키 값이 다르면 비대칭키라고 한다. 비대칭키의 알고리즘은 RSA, 디피-헬만(Diffie-Hellman), 타원곡선암호(ECDSA) 등이 있다. 대표적인 RSA는 1977년 3명의 MIT 교수(Rivest, Shamir, Adleman)가 고안한 큰 소인수의 곱을 인수 분해하는 수학적 알고리즘을 이용하여 만든 공개키 알고리즘이다. 속도가 느린 단점이 있다.

(4) 데이터 베이스

데이터 베이스는 데이터가 저장되는 시스템이다. 데이터는 시스템에서 다루는 가장 중요한 내용이다. 데이터를 관리하기 위한 시스템이 많다. 웹사이트에서 보여주는 대부분은 데이터라고 말할 수 있다. 데이터 베이스는 DBMS라고 하고, SQL 명령으로 조회를 한다. 요즘은 IoT 디바이스가 늘고, 빅데이터 시대가 되면서 비정형 데이터도 많이 다뤄지게 됐다. 그러면서 NoSQL 데이터 베이스를 많이 필요로 했고, 이에 대한 이해도가 필요하다. NoSQL은 SQL과 같지 않다는 의미로 non SQL 또는 non relational의 의미이다.
데이터베이스는 관계형 데이터베이스 보다 제한이 덜 받는 빠른 속도로 저장, 검색하기 위한 방식으로 저장이 된다. NoSQL은 여러 대의 컴퓨터에 데이터를 분산하여 한다. 대용량 처리가 가능하고, 확장성이 좋고, 모바일 기기 데이터나 다양한 포맷의 데이터도 저장이 쉽게 가능하다.

직접적으로 NoSQL DBMS를 다루지 않더라도 프로그램 개발을 할 때에도 이에 대한 이해는 필요하다. 기존 SQL이 아닌 Script 형태와 같은 조회문을 개발해야 하기도 하다.

SQL vs NoSQL

1. 면접 예상 질문

SQL과 NoSQL의 차이를 말해보시오.

• NoSQL의 등장배경을 설명하고 SQL과 다르게 만들어진 사유를 말해보시오.

왜 다르게 만들어졌는지에 대한 사유가 CS 지식 란에 없는 것 같아서 아쉽습니다.
제가 생각하는 NoSQL의 등장 배경은 "RDMBS의 Write 속도 한계를 넘기 위해서"입니다. 이런 내용이 CS 지식에 들어가 있으면 좋을 것 같습니다.

SQL과 NoSQL의 특성에 따른 활용 사례를 말해보세요.

2. 답변 핵심 KEYWORD

정형데이터, 비정형 데이터, 빅데이터, 관계형 DB

데이터베이스는 데이터의 양이 단순히 늘어나는 수준을 넘어서 빅데이터로 진화하면서 그 형태 자체가 바뀌고 있는 상황이다. 특히 SQL에서 NoSQL로의 변화는 기존 데이터베이스를 개선하는 수준이 아니라 패러다임 자체가 전환되는 것으로 볼 수 있다. IoT나 빅데이터에서 필요로 하는 구조로써의 데이터베이스의 진화에 대해, 왜 이렇게 바뀌어야 했는지 신기술에 적용된 데이터베이스 구조에 대한 이해가 필요하다.

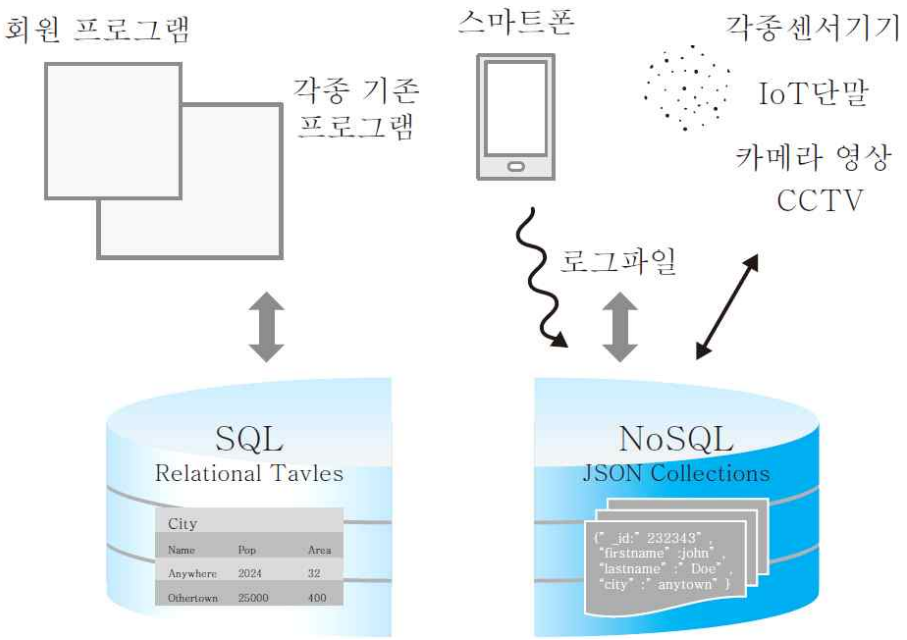
3. CS 지식

기존의 데이터 베이스는 대부분 SQL이라는 구조화된 명령어를 통해서 데이터를 삽입, 수정, 삭제를 하였다. 테이블이라는 것이 존재하고, 이곳에 데이터가 저장되고, 저장될 때에는 필드라는 것으로 구분이 되어 행이 채워지는 형태였다. 고정된 필드 값들이 존재하고, 어떤 형태의 값이 들어갈지 사전에 정의되어야 하고, 그 정의된 값이 아닌 경우는 형식이 잘못되었다는 에러를 리턴하였다.

NoSQL(Not only SQL)은 기존 방식의 관계형 DB가 아니라 대량 데이터를 위한 고속처리를 위한 DB이다. IoT, 빅데이터등의 활용이 증가함에 따라서 NoSQL은 많이 사용되어지고 있는 추세이다.

왜 NoSQL인가 기존 DB는 데이터 베이스 모델링할 때 관계를 맺고 그 관계 구조 상에서 데이터가 저장이 되고, SQL은 이를 조회하기 위한 문장인데, 이

문장은 이 관계를 기본으로 SQL문이 만들어진다. 이런식의 관계지향은 아니다. NoSQL은 기존 방식의 SQL을 사용하지 않는다. 매우 직관적이며 자료 그대로를 넣는다. 기존 DB가 관계를 나누고 사용할 때는 조합해서 사용했다면 NoSQL은 사용하고자 하는 값을 위주로 저장을 그대로 해둔다. 그리고 그대로 꺼내서 사용한다. 기존 관계형 DB의 예제로 학생 테이블, 과목 인덱스 테이블, 과목 점수 테이블 등이 있어서 저장을 할 때 정규화를 하다보면 테이블은 각각 분리가 된다. 사용할 때는 join을 해서 사용을 한다. NoSQL은 어플리케이션에서 사용할 데이터 관점으로 저장을 해둔다. 저장되는 데이터량은 많아진다. 대신에 고속처리에 적합하다. NoSQL DB는 자료형태를 그대로 저장할 수 있는 장점도 있다. 예를 들면 json과 같은 형태의 일종의 문서를 그대로 저장하는 것을 지원한다.



NoSQL형태의 DB는 여러가지가 있다. 대표적으로 MongoDB, HBase, Casandra 같은 것들이 있다. 이 종류 별도로 사용 목적이 약간씩 다르다. 이러한 NoSQL은 로그 수준의 내용들을 DB에 쌓고, 이용할 수 있다. MongoDB의 경우 특징은 기존 SQL이 아닌 JavaScript 문법과 같은 명령어를 통해서 다룰 수가 있다.

NoSQL이라는 것은 이러한 기존 방식의 SQL과는 달리 미리 구조를 정의하지 않는다는 의미이다. 예를 들면 NoSQL은 정의되지 않은 필드도 운영 중에 추가를 할 수가 있다. 이는 예상하지 못하는 데이터도 받아들일 수 있다는 의미이다.

기존에 회원관리와 같이 데이터를 미리 정의해서 관리하는 것은 SQL로 관리하면 되었다. 그런데, 데이터 형태를 알 수 없거나 비정형 데이터를 받아 들일 수 있어야 했는데, 이러한 데이터의 형태와 무관하게 저장을 할 수 있게 정의하여 사용하는 데이터관리 방식이다.
대표적으로 카산드라, 몽고DB, HBase와 같은 데이터 베이스가 있다. 빅데이터에서 수집되는 데이터들은 약속되어진 데이터보다는 비정형데이터가 많이 있을 수 있다. IoT디바이스 등에서 데이터가 나올 때, 그 형태가 다양하다.
몽고DB의 경우는 Json 포맷과 같은 데이터를 저장하고, 업데이트하기에 빠른 성능을 가진다. HBase의 경우 데이터를 수집하고 적재 그리고 빠른 속도의 통계를 내는데 유리하다. 카산드라는 분산 데이터를 사용하여 대용량의 데이터를 처리할 수 있다.
모든 데이터 베이스가 NoSQL 형태로 가야 하는 것은 아니고, 기존 데이터 베이스 구조라면 SQL형태로 유지하면 된다. 데이터 수집이라든가 비정형 데이터를 다룬다고 하면 기존 SQL과 하이브리드 형태로 데이터 베이스를 구성할 수 있다.

(5) 네트워크

웹사이트로 이뤄진 서비스나 시스템등은 모두 네트워크를 통해서 데이터를 주고 받는다. 단순하게 컴퓨터의 데이터만으로 되는 서비스는 많지 않다. 웹사이트 기반으로 된 시스템도 많고, 어플리케이션이더라도 서버와 통신을 한다. 이때 네트워크는 기본적인 연결이 되고, 대부분 네트워크 기반의 프로그램이다. 그리고 서버를 구성할 때 어떤 식으로 연결이 되는지 그리고 어떻게 분산 환경을 만드는지 등도 중요하다.

TCP vs UDP

1. 면접 예상 질문

• TCP와 UDP 통신의 차이를 말해보시오. TCP방식의 연결 방식에 대해서 말해보시오. 기존 IPv4의 한계를 벗어나기 위해 IPv6 채택 배경과 활용에 대해 설명 하시오.

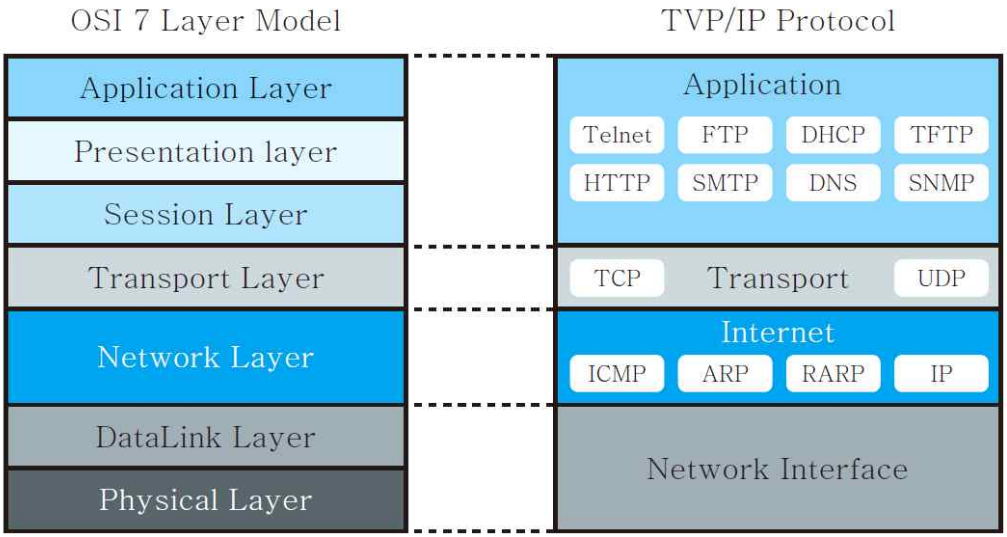
2. 답변 핵심 KEYWORD

SYN, ACK, 전송계층(Transport Layer)

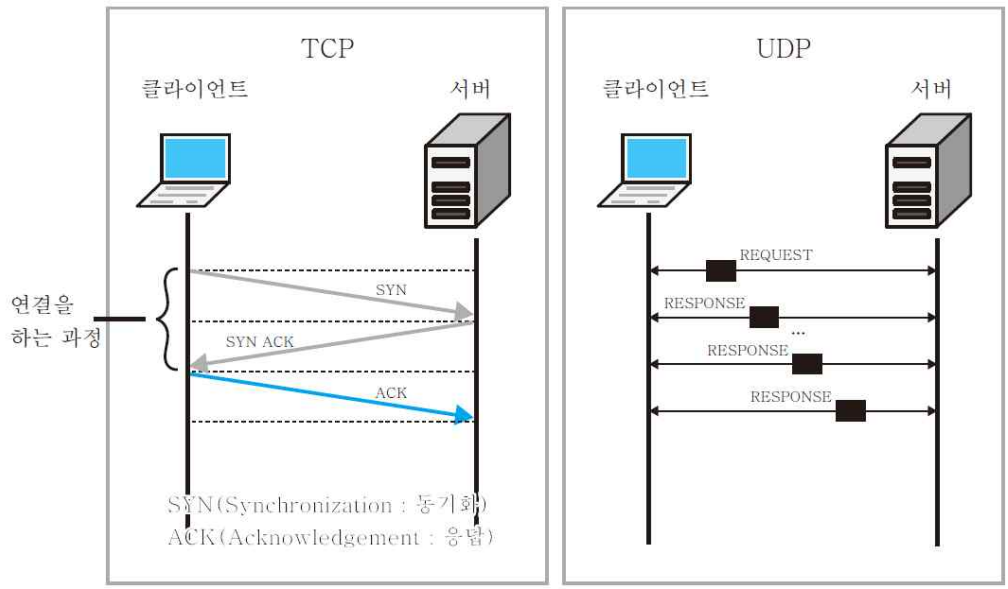
TCP/IP 또는 UDP는 통신의 가장 기본적인 요소이자 핵심이다. 대부분의 프로그램이 웹기반이거나, 웹 기반이 아니더라도 통신은 기본이기 때문에 네트워크 통신에 대한 기본적 이해가 필요하다. 데이터 손실이 없고 신뢰할 수 있는 통신을 위해서는 TCP 통신을 사용하고, 데이터 손실을 감수하면서 빠른 통신을 위해서는 UDP를 사용한다. 이러한 차이가 있는 것은 TCP 통신을 진행할 때는 신뢰할 수 있는 통신을 진행하기 위한 단계를 거치고, 새롭게 통신할 때마다 데이터를 잘 수신했는지까지 확인하고 수신 못했을 경우 재전송하는 과정이 있기 때문이다.

3. CS 지식

데이터를 인터넷으로 주고 받는 것을 데이터 통신이라고 하는데, 이러한 데이터를 주고 받게 되는 기본 요소가 네트워크 이다. 네트워크에는 Layer라고 하는 계층이 있는데, 이 계층 간 프로토콜에 의해서 주고 받게 된다. 이러한 계층에는 네트워크는 OSI 7계층이 있고, TCP/IP 4계층으로 구분될 수 있다.



응용 계층은 우리가 화면에서 보는 통신 프로그램들이다. 응용 프로그램도 세션이나 표현 계층을 포함해서 통신을 하게 된다. FTP라든가 우리가 항상 사용하는 인터넷 브라우저등이 포함된다. 그리고 데이터를 통신하기 위한 아래 Layer로 가는데, 이 중 TCP와 IP가 있다. IP는 어디에서 출발해서 어디로 전달할지 주소와 같은 개념이고, TCP는 데이터를 실어 나르는 역할을 한다.
대표적인 트랜스포트 계층이 TCP인데, TCP의 경우 데이터를 전달해주는 레이어에 해당이 된다. 이때 전달을 해주는데 있어서 비손실이 되게 전달을 하는 방식이 있고,(TCP) 손실이 되는 것을 감안해서 전달하는 방식도 있다.(UDP)
채팅의 경우는 반드시 TCP로 전달을 해야 한다. 왜냐하면 문자는 한글자도 틀리면 안되서 그러하다. 그런데, 영상을 주고 받는다고 했을 때는 약간의 손실이 있어도 영상을 볼 수가 있다. 여러 가지 방법으로 전달을 하겠지만, 손실이라 해서 데이터가 많이 훼손되는 것은 아니다. 데이터 정상 수신에 대한 체크를 다시 하지 않고, 유실되어도 다시 재전송을 요구하지 않는 수준이다.



기술적인 차이로 보면 TCP의 경우는 연결을 하고 데이터를 주고 받을 때 데이터 무결성 체크를 하고, 무결성에서 어긋나는 경우 그 경우는 다시 데이터를 재요청을 해서 받는 과정을 거친다. UDP의 경우는 그러한 무결성 체크를 하지 않고, 데이터를 계속 받는 차이가 있다.

TCP 데이터 전송의 경우 주고 받는 과정에서 연결을 하는 과정이 있고, 데이터의 재전송등이 이뤄지는 경우가 있어서 UDP 통신에 비해서는 느릴 수 있다. UDP는 연결 설정이나 데이터 재전송이 없기 때문에 속도가 빠른 대신에 일부 데이터 손실이 발생될 수 있다.

그래서 영상이나 음성의 경우는 UDP통신으로 하여 많은 데이터를 빠른 속도로 처리하되 일부 데이터 손실에 대해서 감안하고 통신을 하게 된다.

HTTP vs HTTPS

1. 면접 예상 질문

HTTP 프로토콜에 대해서 말해보시오. HTTP와 HTTPS 프로토콜의 차이에 대해서 말해보시오.

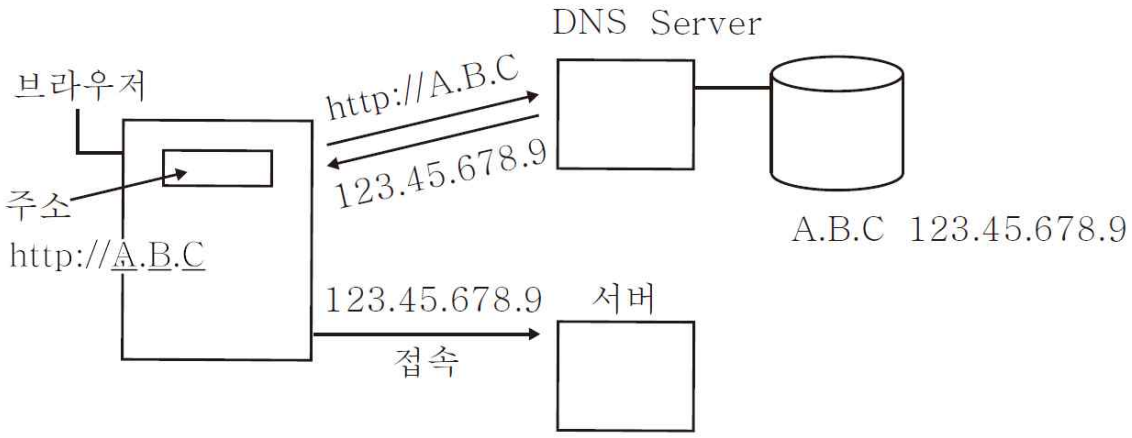
2. 답변 핵심 KEYWORD

프로토콜, 암호화, 클라이언트/서버 통신, URL, HyperText
HTTP는 웹 페이지를 송수신하는 기술을 시작으로 대부분의 데이터를 송수신하는 프로토콜이 되었다. OpenAPI를 사용하면서 이 API를 호출하는 것도 HTTP통신 방식을 이용한다. 그리고 HTTP와 더불어 보안 측면이 강화된 HTTPS까지 이해하는 것도 중요하다. HTTP가 단순 프로토콜로 HTML데이터를 송수신하는 프로토콜이라면 HTTPS는 세션 레이어의 암호화가 포함된다. 이 암호화를 통해서 전송 데이터를 제 3자가 식별할 수 없게 된다.

3. CS 지식

클라이언트/서버 통신 방법 클라이언트는 우리가 사용하는 컴퓨터 또는 스마트폰과 같은 것이 될 수 있다. 기본적으로 클라이언트 측에서 먼저 요청을 한다. 요청을 한다는 것은 어떤 기능을 요구한다는 것이다. 카카오톡도 사용자 중 누군가가 메시지를 작성하고 전송 버튼을 누르게 된다. 카카오톡의 서버가 어떤 구조인지 정확히는 알 수 없지만, 기본적으로 서버는 기다리고 있다. 어떤 요청을 받을 준비를 하고 있는 것이다. 그래서 어떤 클라이언트가 어떤 요청을 하는지 또는 연결을 하고 싶어 하는지를 알 수 있다.
웹사이트 접속 시 서버와 클라이언트 동작을 생각해본다면 웹브라우저에서 특정 URL로 접속을 하게 되면 그 URL의 주소값을 DNS라는 서버를 통해서 가져온다. 시스템은 숫자 주소인 IP라는 것이 있어야 해당 서버에 연결을 할 수가 있다. 하지만 사람이 숫자를 외우는 것보다는 이름으로 아는 것이 쉽기때문에 도메인이라는 것을 사용한다. 클라이언트 측에서는 웹브라우저를 이용하고 도메인을 입력하고 접속을 하면 해당 주소에 있는 서버가 응답을 해준다. 이때 웹서버의 경우 URL이라는 주소로 접속을 해서 파일이나 이미지를 요구하게

된다. 그 요구되는 컨텐츠를 클라이언트로 전송한다.

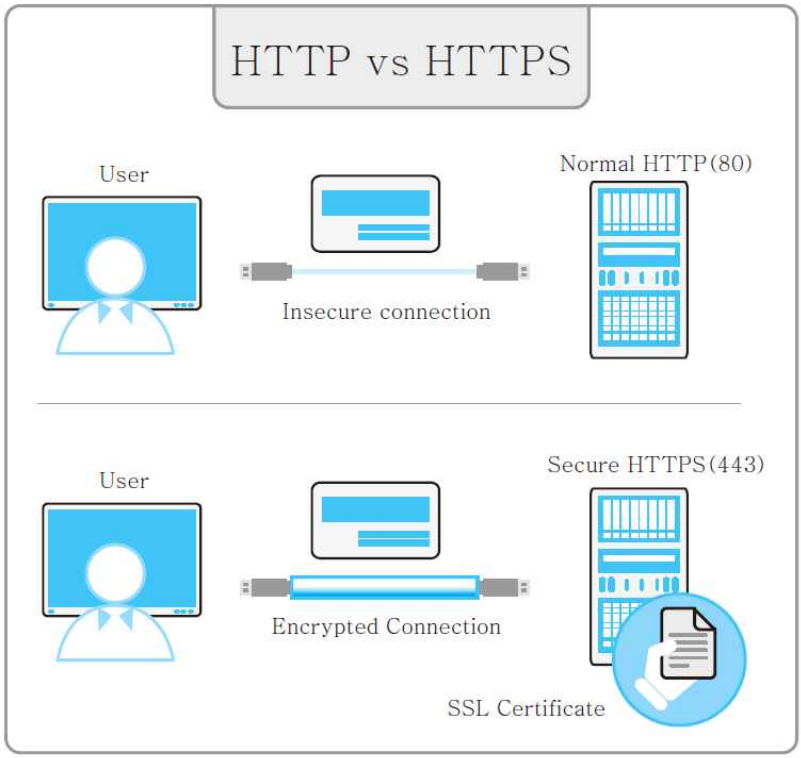


이렇게 클러이언트 접속을 받아 응답을 해주는 것이 서버의 기능이다. 다음에 사용자가 클라이언트에서 다른 메뉴를 누르거나 선택하면 다시 그 URL을 요청하고 서버는 응답을 해주게 된다.

HTTP라는 말은 너무 흔하게 듣지만 기술적 측면을 보면 이러하다. HTTP(HyperText Transfer Protocol) 프로코톨은 하이퍼텍스트를 전송하기 위한 통신 규약이다. 우리가 사용하는 브라우저에서 URL을 입력하면 해당 URL에 위치한 서버와 통신을 하게 되고, 웹페이지들을 받아서 브라우저에 보여준다. 대표적으로 GET, POST와 같은 메소드가 있어서 우리가 브라우저 상에 데이터를 입력하기도 하고, 데이터를 받아보기도 한다. HyperText 라는 개념은 웹페이지에 보면 링크로 다른 경로나 이미지, 영상들을 볼 수 있는데, 이를 담아내기 위한 텍스트 형식이다.

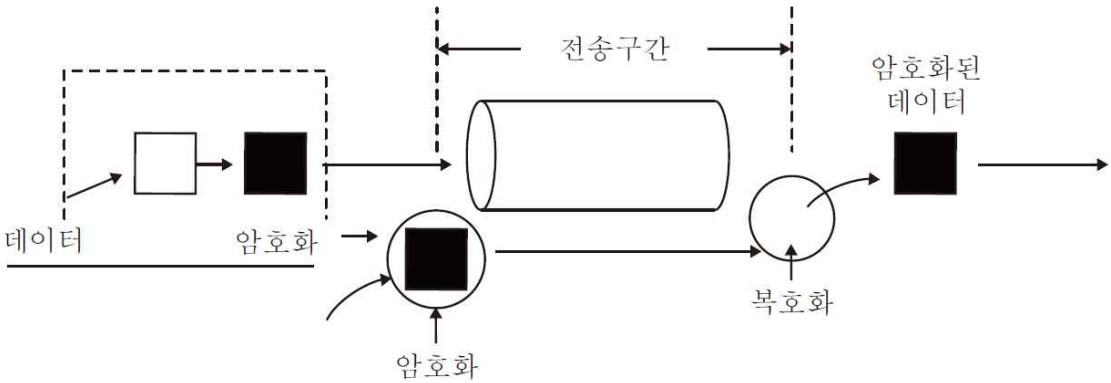
HTTPS는 HTTP over TLS Session로 HTTP에 보안 기능을 추가한 것이다. 통신 단에서의 암호화 기반 인터넷 보안 프로토콜이다. 인터넷 통신의 개인정보 보호, 인증, 데이터 등이 외부로 유출되지 않도록 통신단을 암호화 하는 것이다.

에 암호화를 하는 것이다. Secure를 해주면서 Layer 암호를 해주는 것이다.



[그림] HTTP vs HTTPS

데이터 자체를 외부로 유출이 되어도 보지 못하게 암호화 하는 것이 있고, 처음부터 외부로 못나가게 암호화 하는 것이 있는데, HTTPS는 통로를 암호화 하는 것이다. 네트워크에 데이터가 오가면서 해킹이라든가 데이터를 빼서 볼 수가 있는 상황에서 데이터 암호화가 되어 있다면 복호화가 어려울 것이고, 통신 단에서 암호화가 되어 있으면 데이터를 쉽게 가져갈 수 없다.



보안상의 이유로 웹사이트는 모두 https 통신만으로 접속되게끔 하였다. 이러한 적용 이후로 브라우저들에서는 https 접속을 기본으로 하고, http로 접속하려고 하면 경고 문구를 띄워준다.

로드 밸런싱

1. 면접 예상 질문

• 로드 밸런싱이 어떤 개념인지 설명하고, 어떤 상황에서 로드 밸런싱을 해야 하는지 말해보시오.
• 로드 밸런싱 구성시 네트워크 밸런싱과 어플리케이션 밸런싱의 개념에 대해 설명하시오.
• 로드 밸런싱 중에 후단에 있는 서버 중 하나가 응답이 없을 때 처리되는 방법에 대해서 말해보시오.

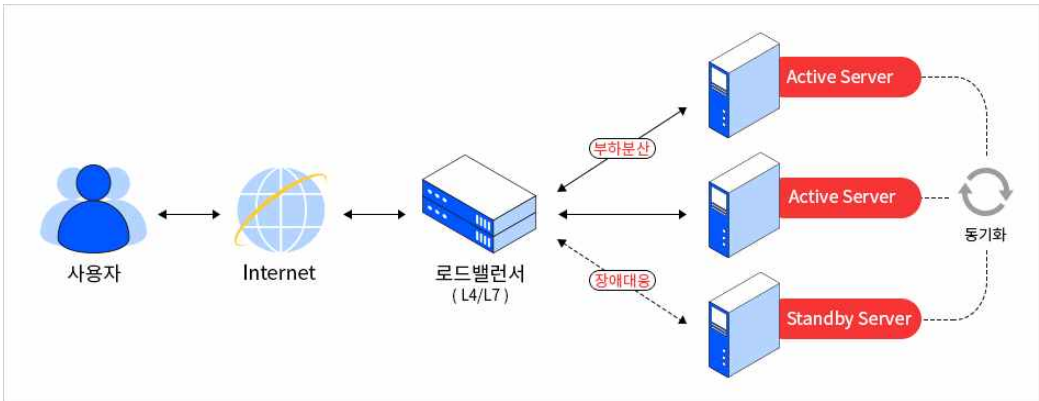
2. 답변 핵심 KEYWORD

트래픽 분산, 부하 분산, L4 로드밸런싱, L7 로드밸런싱
서버 관리 중 중요한 것은 안정성이다. 이를 위해서 로드 밸런스 기술을 이용한다. 로드 밸런서는 서버의 부하를 분산하는 것이 목적이다. 이 작업을 로드 밸런싱이라고 한다. 로드 밸런싱은 서버 이중화의 중요한 기술이다. 서버를 여러 대 연결해서 트래픽을 분산함으로써 서버 한 대가 고장나더라도 다른 서버가 동작 중이기 때문에 서비스 안정성을 높여준다. 즉, 한 서버가 어떠한 이유로든 응답이 없다면 다른 서버들로 트래픽이 넘어가게 되어서 전체적인 서비스는 유지될 수 있다.

3. CS 지식

로드 밸런서는 서버에 부하를 분산해주는 장치이다. 부하가 한 쪽으로 몰리는 것을 분산하여 수행하므로 전체적인 시스템 효율을 높혀준다. 공항 같은 곳에 가보면 줄을 길게 서 있는 사람들이 있고, 여러 군데로 나눠서 사람들의 요청을 처리해준다. 이렇게 하면 전체 사람들의 요구를 처리하는 것이 빠르게 진행될 수 있을 것이다. 여러 대의 서버로 트래픽을 나눠서 수행하려고 할 때 어떤 곳으로 가면 되는지 연결 해주는 것을 로드 밸런싱이라고 한다.
균등하게 또는 특정 규칙에 맞게 분산을 해준다. 분산을 해주는 장치가 없다면 한군데로 트래픽이 몰려 버릴 것이다.

시스템에서 처음으로 트래픽을 받는 것이 이 로드 밸런스 장치이다. 로드 밸런스가 앞에서 전체 트래픽을 받게 된다. 그리고 그 뒤에 서버들로 전달을 해준다. 로드 밸런서 뒤에는 웹서버나 WAS(Web Application Server) 서버가 있기도 한다. WAS는 DB 조회나 다양한 로직 처리를 요구하나 장치에 애플리케이션을 수행해주는 서버이다.
로드밸런서 뒤에는 한 대부터 여러 대까지의 서버가 있을 수 있다. 여러 대의 서버에 균등하게 보내주기도 하고, 트래픽의 분산 비율을 정해줄 수도 있다.



[그림] 로드 배런서와 서버 연결

로드 밸런서가 중요한 이유는 로드 밸런서 뒤에 특정 서버가 장애가 났을 때 다른 정상적인 서버로 트래픽을 보내줌으로써 전체적인 서비스는 중단되지 않게 할 수 있다. 최근 서버 운영 트랜드는 물리적 서버가 문제가 발생될 수 있다는 가정 하에 전체 서비스가 이상이 없도록 설계하고 운영을 한다.
서비스를 운영하는 운영자들을 이러한 장애에 굉장히 민감하고 이를 위해서 로드밸런서와 연결된 서버들 중 일부를 강제로 중단되게 해서 전체 서비스에 이상이 없는지 확인하는 테스트를 주기적으로 하기도 한다.
그리고 지역자체가 DNS라든가 망 문제로 인해서 접속이 안될때는 지역간 Global 로드밸런서를 두고, 이중화를 하여 다른 정상적인 지역에서 서비스를 하게도 한다. 그렇다 보니 이 로드밸러서 자체에 문제가 생기면 더 큰 장애가 되기도 한다.
로드 밸런서는 과거에는 L4 스위치 Layer 중 4계층(Transport) 부하 분산, Layer 7계층(Application)에서 부하 분산을 하는 장비가 있었다. 4계층 부하분산이란 포트와 헤더를 구분하여 분산을 해준다. 그러면 포트별로 구분이 가능하다. 특정 서비스는 특정 포트를 사용하기에 해당 포트별로 분산을 해줄 수가 있다. L7의 경우는 어플리케이션 레벨로 패킷을 들여다 보고 분기 해줄 수

가 있다. 특정 프로그램까지 구분하여 스위칭 해줄 수가 있다.

그러한 시스템을 전체 시스템의 트래픽 흐름 최상단에 위치해두었다. 예를 들어 아마존 클라우드 시스템으로 전환이 되면 로드 밸러서 장비를 메뉴 상에서 Elastic Load Balancer라는 기능을 추가를 해줌으로써 바로 위치할 수가 있고, 이런 로드밸러서는 용량이 유연하게 늘어나게 되어서 트래픽을 부하 분산할 수가 있다.

(6) 운영체제

운영체제는 컴퓨터를 구성하고 있는 CPU, 메모리, 하드디스크 등에서 프로그램이 실행되도록 해준다. 어떻게 동작되는지 기본적인 원리에 대해서 이해할 수 있다. 운영체제를 잘 알고 있으면 프로그램의 동작에 대해서 더 잘 이해할 수 있다. 프로그램 개발을 하다 보면 멀티 프로세스, 멀티 쓰레드와 같은 프로그램 개발을 할 때가 있다. 운영체제의 원리에 대해서 이해 하면 이러한 특성에 맞는 개발을 할 수가 있다. 고급 엔지니어가 되어갈수록 이런 기본기를 필요로 한다.

프로세스 vs 쓰레드

1. 면접 예상 질문

• 프로세스와 쓰레드의 차이를 설명하시오.
• 쓰레드가 필요한 경우에 대해서 예를 들어 설명하세요.
• 프로세스 여러 개를 띄우는 경우과 쓰레드를 여러 개 띄우는 경우에 대한 차이를 설명하시오.

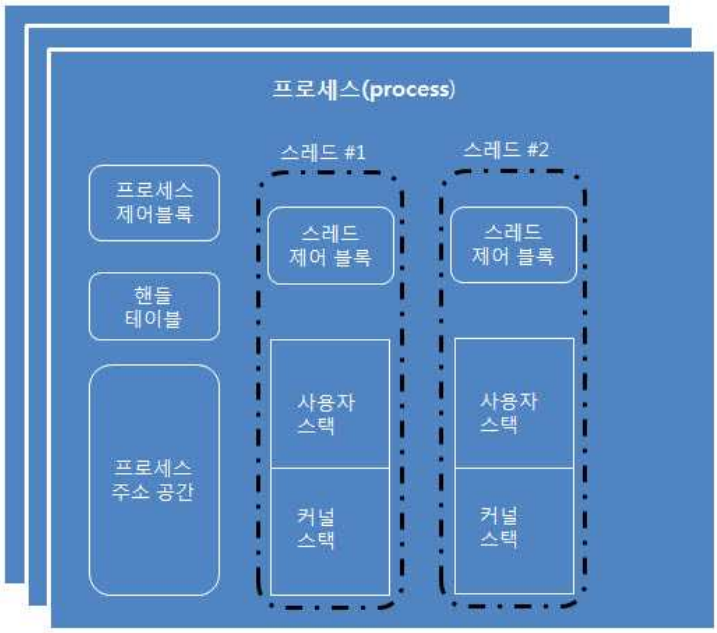
2. 답변 핵심 KEYWORD

문맥 전환(Context Switching), 멀티 쓰레드, 싱글 쓰레드

프로세스와 쓰레드는 컴퓨터 구조에 대한 이해가 필요하지만 개발할 때에 알아야 하는 핵심 기술 중 하나이다. 프로그램이 실행되어 동작되고 있는 상태가 프로세스이다. 프로그램이 여러 개 실행되고 있다는 것은 프로세스가 여러 개 띄워져 있다는 의미이다. 쓰레드는 각 프로세스의 실행 단위이고, 하나의 프로세스에 여러 쓰레드가 실행될 수 있다.

3. CS 지식

컴퓨터를 구동하고 있는 것은 CPU, 메모리, 하드디스크등으로 구성된 하드웨어와 이 하드웨어들을 구동 시켜주는 운영체제 그리고 운영체제 위에서 실행되고 있는 프로그램 즉 소프트웨어로 구성이 된다.
소프트웨어가 실행되고 있지 않을 때는 프로그램은 파일 형태로 하드디스크에 저장되어 있다. 이 프로그램이 실행되는 순간 프로그램이 컴퓨터 메모리에 일부가 적재되고 실행되는데 이를 프로세스라고 한다. 프로세스는 실행되고 있는 프로그램으로 생각하면 된다.
각각의 프로세스는 자신의 실행 이미지 일부와 실행에 필요한 추가적인 메모리 공간을 가지고 있다. 이것은 각 프로세스마다 구별되며 해당 프로세스가 접근하고자 하는 파일, I/O장치들에 대해서 또한 프로세스 단위로 할당 받아 관리된다.



[그림] 프로세스와 쓰레드

쓰레드는 운영 체제에서 프로세서 별로 시간을 할당하는 기본 단위로 하나 이상의 쓰레드가 해당 프로세스 내에서 코드를 실행한다. 한 프로세스는 기본적으로 하나의 쓰레드가 실행된다.

동시에 여러 부분을 실행시키고 싶은 경우는 여러 개의 쓰레드를 띄워서 사용할 수가 있다. 이렇게 실행되는 단위가 쓰레드이다. 이렇게 여러 개의 쓰레드가 한 프로세스의 여러 부분을 실행하는 것을 멀티 쓰레드 프로그램이라고 한다.

이러한 쓰레드는 같은 프로세스에 있는 자원과 상태를 공유한다. 같은 프로세스 내에 있는 쓰레드는 같은 주소 공간에 존재하게 되며 동일한 데이터에 접근할 수 있고 하나의 쓰레드가 수정한 메모리는 같은 메모리를 참조하는 쓰레드에 영향을 미치게 된다.

예를 들어 하나의 쓰레드에서 오픈한 파일을 다른 쓰레드가 사용할 수 있다. 프로세스가 종료되면 거기에 속해있던 쓰레드도 함께 종료된다.

쓰레드의 필요성 하나의 프로세스 안에서 여러 개의 루틴을 동시에 수행하여서 수행 능력을 향상시키려고 할 때 쓰레드를 사용하게 된다. 또는 어떤 OS들에서는 네트워크 통신이라든가 화면 처리를 별도 쓰레드로 처리하여 통신의 응답을 기다리는 동안 화면이 멈추지 않게 해준다. 서로 같은 자원을 사용하는 경우라면 동기화 문제가 생길 수 있다. 어떤 쓰레드가 수정을 하고 있다든지 특정 자원을 사용하고 있으면 동시에 사용할 수 없거나 잘못된 값을 참조할

수도 있다.
쓰레드는 응용 프로그램을 개발할 때 어떤 식으로 사용할지 충분히 고려되어야 한다. 쓰레드는 동시에 프로그램 여러 기능을 수행하게 함으로 성능을 좋게 해주지만, 너무 많이 사용하는 경우는 오히려 성능 저하를 일으킬 수 있다. 쓰레드는 운영체제로부터 짧은 시간동안 CPU 자원을 사용하는데, 너무 많은 쓰레드가 있는 경우는 쓰레드 전환이 빈번하게 일어 나면서 성능 저하를 일으킬 수 있다. 이렇게 쓰레드 전환을 문맥 전환(Context Switching)이라고 한다. 문맥 전환은 CPU가 하나의 프로그램에서 필요로 하는 명령어들을 처리하면서 해당 명령어가 참고하는 메모리 영역이라든가 참조 변수들로 세팅했다가 이제 다른 프로그램에서 사용하려고 기존 변수들을 다른 프로그램의 상태로 바꿔주는 것을 의미한다.

데드락

1. 면접 예상 질문

- 데드락 현상에 대해서 말해보시오.
- 데드락이 발생했을 때 해결방법에 대해서 말해보세요.
- 데드락이 발생되지 않게 하는 사전 방지 방안에 대해서 말해보세요.

2. 답변 핵심 KEYWORD

교착 상태, 자원 점유, 예방, 회피, 해결

데드락은 운영 체제에서 핵심적으로 알아야 하는 개념이다. 데드락이 어떤 상황에서 발생할 수 있는지 이를 회피하거나 예방, 해결할 수 있는 방법은 무엇이 있는지 알아야 한다. 멀티 쓰레드 프로그램이 많이 개발되기 때문에 이러한 기본적인 원리에 대해서 파악하고 숙지해야한다.

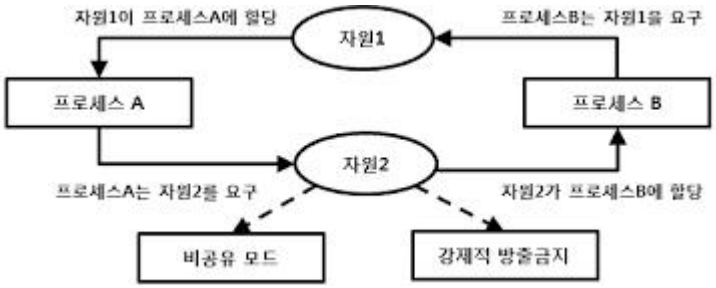
3. CS 지식

교착 상태라고도 하는데, 차가 다니는 도로에서 사거리에 차들이 꽉 막혀 있는 상황을 연상하면 된다. 차들이 사거리에서 각자 원하는 방향으로 가고자 계속 진입할 때 오히려 완전히 막히게 되고, 멈추고 더 이상 전진을 할 수가 없다. 이 상태를 데드락이라고 한다. 이때 데드락을 피하기 위해서는 어떻게 해야 할까? 사거리에 보면 교통 신호등이 있고, 어떤 때는 빨간불을 통해서 차량의 진입을 막고, 다른 차량이 어느 정도 지나가고 나서 신호등이 바뀌면서 다른 차량을 보낸다. 이런 과정은 사거리의 통행을 위해서는 반드시 필요하다.

그럼 데드락은 왜 발생하는 걸까? 서로 자원을 사용하려고 하기 때문이다. 컴퓨터 내에서는 여러 가지 자원들이 있다. 메모리나 CPU, I/O 장치들이 있는데, 하나의 프로세스가 사용하는 것은 문제 되지 않는데, 다른 자원이 사용하려고 할때 기존 프로세스가 놔주지 않으면 사용을 못하게 된다.

그런데, 단순히 사용을 못하면 기다리면 되는데, 왜 교착이 발생될까? 이런 상황을 보면 된다. 프로세스A가 자원1을 사용하고 있다가 자원2가 필요해서 그 자원을 사용하려고 보니 프로세스B가 이미 자원2를 사용하고 있다. 그러면 기다리면 되는데, 프로세스B도 자원1을 요구를 하고 있는 상황이라면 프로세스A

가 자원1을 해제 해주거나 프로세스B가 자원2를 해제 해주면 되지만, 지금 진행되고 있는 상황이기에 해제를 해줄 수가 없는 상황이 된다.



[그림] 교착 상태

운영체제에는 수많은 프로세스, 쓰레드가 수행되면서 여러가지 일을 동시에 하고 있기에 충분히 이런 일이 발생될 수 있다. 그럼 어떻게 이런 상황을 방지 또는 해제 할 수 있을까? 사실 해제 하기보다는 방지를 하는 것이 더 좋은 방법이다. 방지를 위해서는 자원 사용을 Shared로 하는 간단한 방법이 있고, 프로세스가 시작하려고 할때 필요한 자원을 모두 점유해 버리는 방법도 있다. 이는 그렇게 되었을 때 최소한 프로세스A가 필요한 자원1, 자원2를 점유하고 시작하면 B프로세스 수행 대기가 길어질 수는 있지만, 최소한 프로세스A 실행을 마칠 수가 있다. 비선점하는 방법도 있다. 다른 프로세스가 요구할 때 일단 자원을 받납하고, 다른 프로세스가 사용할 때까지 기다리는 방식이다. 아니면 자원에 고유 번호를 할당하고, 순서표를 뽑아서 기다리게 하는 방식도 있다.
교착상태가 이미 되어 버렸을 때 해제하는 방법도 있다. 간단한 방법은 교착상태엔 프로세스를 모두 중지시킨다. Kill 하는 것이다. 과거 윈도우가 멈춰서면 Ctrl-alt-del을 눌러서 프로세스를 죽이곤 했는데, 이런 방법으로 생각하면 된다. 교착 상태 프로세스 모두를 죽이지 않고, 순차적으로 Kill을 하여 교착상태가 풀어 보는 것도 방법이다.
그리고 다른 방법으로 교착 상태의 프로세스가 점유하고 있는 자원을 다른 요구하는 프로세스에게 할당하고, 교착 상태 프로스세를 일시 정지시키는 방법이다. 그리고 우선 순위가 낮은 프로세스, 수행된 횟수가 적은 프로세스 위주로 자원을 선점하게 해주는 방식도 있다.
데드락에 대한 이해를 위해서 현실 세계에서 경험한 데드락 상태가 있다.
직장 내에서 특정 업무를 하고 싶다고 예를 들면 DBA(데이터 베이스 관리자)를 원한다고 하면 회사 책임자가 DB 업무 경험이 있는지 묻게 된다. 왜냐면 회사가 운영하는 시스템의 DB를 맡긴다는 것이 경험 없는 사람에게 맡기기는

어렵기 때문이다. 그래서 묻는다. 경험이 있는지? 그런데, DBA를 경험을 해보려면 맡겨야 할 수 있다. 맡기려면 경험이 있어야 한다. 이 상태로 계속 반복 질문이 되면서 결론이 나지 않는 상황도 데드락 상황이라고 말할 수 있다. 아 그럼 실제 이런 상황에서는 어떻게 해결이 될까? 데이터 베이스 담당과 같이 중요한 업무는 대개 주담당자를 두고 부담당자로 업무를 시작하게 된다. 그러면서 기존 경험자의 기술과 경험을 들으면서 간단한 일부터 하게 된다.

캐시 메모리, 버추얼 메모리

1. 면접 예상 질문

• 캐시 메모리에 대해서 설명하시오.
• 캐시 메모리가 없다면 생길 수 있는 현상에 대해서 설명하시오.
• 버추얼 메모리에 대해서 설명하고, 버추얼 메모리가 필요한 이유에 대해서 설명하시오.

캐시 메모리에 대해서 설명하시오.

캐시 메모리는 CPU가 자주 참조하는 데이터를 저장해두는 작은 용량의 메모리입니다. 메모리와 CPU 사이에 위치하여, CPU가 메모리에 접근할 때 캐시 메모리에서 먼저 데이터를 찾아보고, 존재하지 않으면 메모리에서 데이터를 가져와서 CPU에 전달합니다. 이를 통해 CPU의 성능을 향상시킬 수 있습니다.

캐시 메모리가 없다면 생길 수 있는 현상에 대해서 설명하시오.

캐시 메모리가 없다면 CPU가 메모리에서 데이터를 가져오는 데 걸리는 시간이 지연될 수 있습니다. 이는 CPU가 메모리에서 데이터를 가져오는 데 필요한 시간이 상대적으로 오래 걸리기 때문입니다. 따라서, CPU의 성능이 떨어지게 되어, 프로그램의 실행 속도가 느려질 수 있습니다.

버추얼 메모리에 대해서 설명하고, 버추얼 메모리가 필요한 이유에 대해서 설명하시오.

버추얼 메모리는 하드 디스크의 일부를 메모리처럼 사용하는 기술입니다. 이를 통해 물리적인 메모리가 부족한 경우에도 프로그램을 실행할 수 있습니다. 버추얼 메모리는 각 프로세스의 가상 주소 공간을 물리적인 메모리와 결합하여 프로세스가 필요로 하는 메모리 공간을 늘리는 것처럼 보이게끔 해줍니다. 이를 위해 하드 디스크의 일부를 메모리처럼 사용할 수 있도록 하고, 프로세스가 필요로 하는 메모리를 일시적으로 하드 디스크에 보관해두었다가 필요할 때 메모리로 다시 올려 사용합니다. 버추얼 메모리는 프로그램의 실행에 필요한 메모리를 보다 효율적으로 관리할 수 있도록 해주며, 물리적인 메모리의 한계를 극복할 수 있습니다.

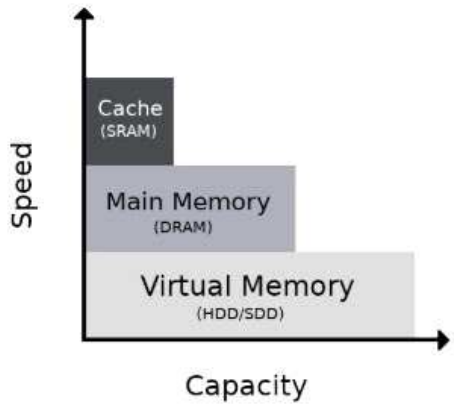
2. 답변 핵심 KEYWORD

캐시 힛(Cache Hit), 지역성(Locality), 페이징(Paging), 세그먼트(Segment), 외부 단편화 현상

메모리는 저장 장치 중의 일부인데, 캐시 메모리와 버추얼 메모리는 특징도 다르고, 역할도 다르다. 캐시 메모리의 경우 CPU와 메인 메모리 사이에서 CPU가 실행해야 할 데이터들을 미리 캐시 해둠으로써 전체 시스템 구동을 더 빠르게 하기 위한 것이라면 가상 메모리는 디스크를 활용해서 메인 메모리의 한계를 넓혀줌으로써 성능을 높여주는 역할을 한다.

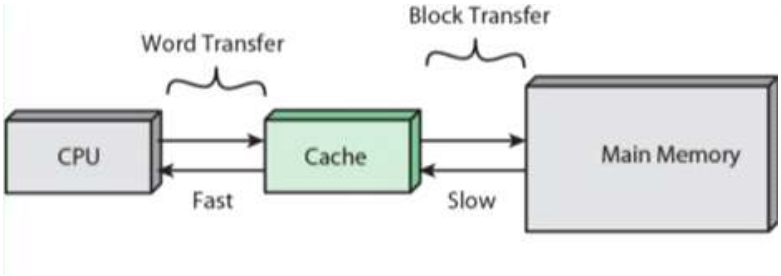
3. CS 지식

캐시 메모리는 메인 메모리와 CPU간의 데이터 속도 향상을 위한 중간에 위치한 메모리이다. 그럼 왜 이런 캐시 메모리가 생긴 걸까? 그것은 전체 시스템 향상을 원하긴 하지만, 비용을 무한정 늘릴 수 없기 때문이다. CPU와 가까운 메모리일수록 값이 비싸다. CPU내에도 메모리가 있는데, 이 메모리가 가장 비싸고, 가장 성능이 좋다. 하지만 무한대로 이 메모리를 늘릴 수 없기에 큰 사이즈의 메인 메모리가 있다. 속도 측면의 보완을 위해서 자주 사용하는 데이터를 캐시 메모리로 올리게 되면 빠른 속도로 CPU가 데이터를 참조할 수 있게 된다.



[그림] 메모리의 속도와 비용과 관계

캐시라는 의미가 메모리에서뿐 아니라 인터넷 사이트에서도 캐시를 사용하여 로컬 컴퓨터에 파일을 저장해두고 동일한 사이트를 재접속했을 때 이미 받았던 이미지들은 다시 받지 않고, 로딩해줌으로 빠르게 표시해주기도 한다.

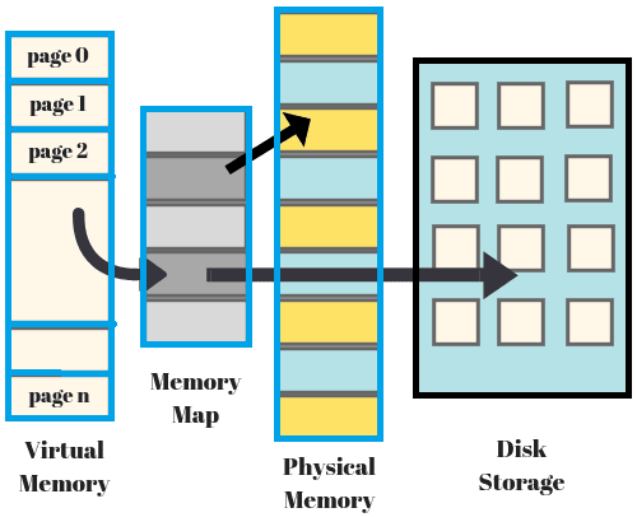


[그림] 캐시 메모리

캐시 메모리에서의 중요한 성능은 CPU에서 참조하려는 데이터가 캐시 메모리에 올라와 있는가 이다. 그래서 캐시메모리에는 되도록 빈도수가 높은 데이터 위주로 배치하게 된다. 이를 Hit율이라고 한다. CPU가 찾는 데이터가 캐시 메모리에 있는 경우가 hit를 한 것이다. 캐시 메모리에 어떤 데이터를 올려 놓을지 여러 가지 방식의 데이터 교체 방식이 있다.
먼저 들어온 데이터 위주로 적재하는 FIFO, 최근에 자주 사용되었던 데이터를 유지하는 방법 MRU(Most Recently Used)이나 최근 가장 사용하지 않은 데이터를 내리는 방식 LRU(Least Recently Used), 이후 참조되지 않을 가능성이 높은 데이터를 내리는 것도 방법중에 하나이다.
이런 방식은 모두 근거가 있다. 공간적인 지역성 또는 시간적 지역성이라고도 말하는데, 예를 들면 현재 수행되고 있는 프로세스가 계속 수행될 가능성이 높기에 현재 참조하고 있는 데이터가 더 많이 자주 참조되어질 가능성이 높다. 컴퓨터에서 특정 작업을 하고 있다고 하면 한동안은 그 작업이 계속되어지기 때문이다. 그리고 시간적 지역성은 for, while등에 의해서 다시 접근이 되어지는 데이터들이 있다. 시간적으로 집중되어서 사용되어질 수 있다는 의미이다.

버추얼 메모리는 과거 메모리가 부족하던 시절에 프로그램이 실행되기 위해서는 어느 정도의 메모리 공간이 필요한데, 부족할 때 이를 위해서 고안된 방식이다. 메모리가 아니지만 하드디스크를 마치 메모리처럼 확장해서 사용할 수 있게 해준 방식이다. 컴퓨터 시스템의 메모리는 한계가 있다. 무한정 늘릴 수가 없다. 메모리는 고가이기도 하고 어플리케이션에서 요구하는 메모리는 늘어났다. 이러한 상황에서 프로그램이 원활하게 실행될 수 있게끔 보조역할을 한 것이다. 그렇다고 하드디스크가 메모리로 바뀌는 것은 아니다. 메모리에 올라가는 데이터 중 일부를 하드디스크에 그대로 보관했다가 다시 메모리로 올리는(SWAP) 형태로 메모리의 확장으로 보이게끔 해준다.
가상 메모리라는 것은 물리적 메모리보다 보다 많은 메모리를 프로그램에서 사용할 수 있도록 디스크의 일부를 메모리처럼 사용하는 논리적 메모리이다.

아래의 그림과 같이 표현 할 수 있다.



[그림] 페이징 기법

하드 디스크의 특정 공간을 페이징이라는 개념을 통해서 메모리처럼 사용하는 것이다. 가상 기억 장치를 위해서는 물리적 메모리에 데이터를 체계적으로 적재하기 위해서 일정한 크기로 나누어 페이징 처리를 한다.

페이징 기법이라는 것은 프로그램을 메모리에 적재할 때 균일하게 나눠서 페이지 단위로 올리는 방식이다. 메모리의 히트율에서 보면 지역성이라는 특성을 갖기 때문에 프로그램의 일부를 올리면 메모리에서 효과적으로 사용을 하다가 내려오게 된다. 또 다른 프로그램의 일부도 동시에 메모리를 사용할 수 있는 것이 장점이 있다.
가상 주소와 페이징 시스템 페이지로 구성되고 여러 개의 프로그램이 동시에 메모리를 사용하고 있다면 실제 주소에 담겨진 데이터를 어떻게 각각의 프로그램이 인식할 수 있을까? 실제 메모리와 프로그램의 일부인 데이터를 연결시켜주는 정보목록이 있어야 한다. 이것이 페이지 테이블(Page Table)이다.
페이지 단위가 크다 보면 메모리 사이 사이에 비 효율적인 사용이 발생 될 수 있다. 그 사이를 분할해서 사용하는 방식이 세그먼트 방식이다. 세그먼트로 분할하게 되면 메모리의 빈공간을 줄 일 수가 있다. 하지만 이 방식의 단점은 데이터를 자르다 보면 잘려진 데이터로 인해서 의미를 알 수 없게 되기도 한다.
단편화(Fragmentation) 세그먼트 기법의 서로 비슷한 속성을 갖는 데이터를 물리적으로도 근접 시킬 수 있는 장점이 있지만, 서로 다른 크기를 갖는 세그

먼트를 메모리에 적재하고 내리다 보면 메모리의 사용에 있어서 더 이상 사용할 수 있는 좁은 메모리 공간이 여러 개가 생기는데 이를 '외부 단편화 현상'이라고 한다. 이러한 단편화 현상에 대해서는 결국 메모리 재배치를 통해서 해결이 된다. 페이징에 비해서 주소 변환이 없다는 장점도 있었지만, 이렇게 메모리 재배치라는 오버헤드를 갖게 된다.

(7) 프로그래밍 기술

프로그래밍은 개발에 핵심이기도 하고 가장 중요한 요소이다. 이러한 프로그램을 위한 기초 지식을 쌓는 것은 필요로 하고, 개발에 있어서도 배경 지식이 될 수 있다. 프로그램을 무작정 만들기 시작하는 것보다 기초적인 지식을 쌓고, 만들어보는 것이 더 견고한 프로그램 능력을 갖게 되는 길이다.

가상함수

1. 면접 예상 질문

• 가상함수의 개념을 말하고, 어떤 때 사용하는지 말해보시오.
• 가상함수에서 사용되는 개념은 객체 지향의 어떤 특성을 이용하는 것인지 설명해보시오.
• 오버라이딩과 오버로드 함수에 대해서 차이점을 비교하여 설명하시오.

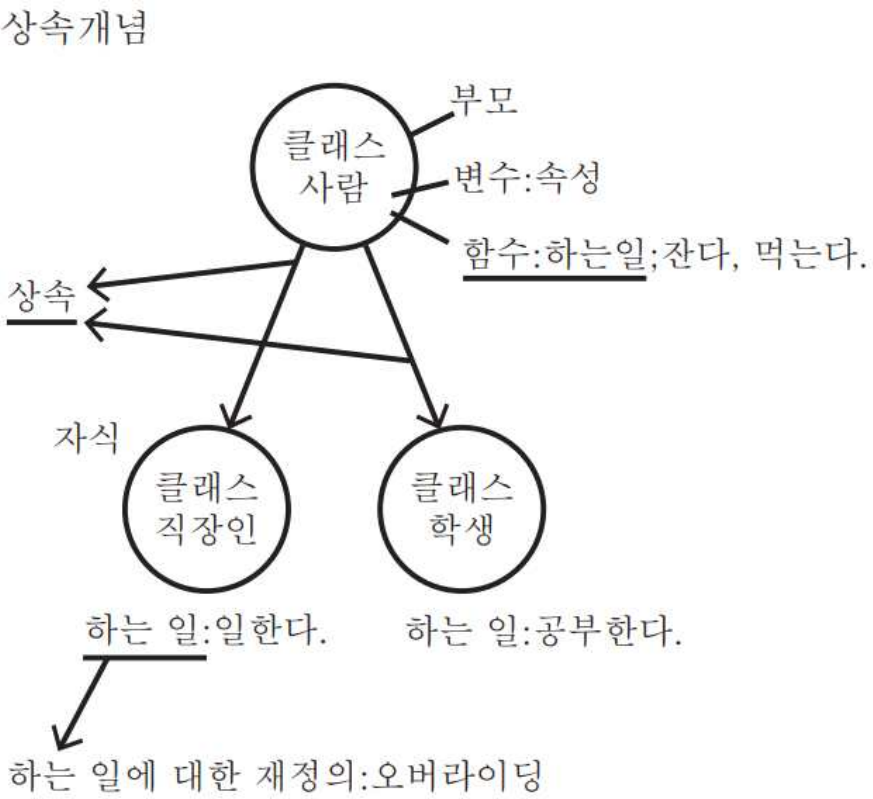
2. 답변 핵심 KEYWORD

함수, 상속, 오버라이딩, 인터페이스
프로그램을 개발할 때 가상함수의 개념을 알고 사용할 수 있어야 하고 객체 지향의 몇 가지 개념 또한 알고 있어야 한다. 객체 지향 프로그래밍에서, 가상함수는 상속을 받아서 사용하는 곳에서 재정의하여 사용할 수 있게끔 함수에 대한 정의만 해둔 상태의 함수이다. 이렇게 재정의하는 것을 오버라이딩이라고 하며 가상 함수의 경우 재정의 해도 되고, 안해도 무관하다.

3. CS 지식

함수라는 것은 특정 기능을 수행하는 역할을 한다. 우리는 함수라는 말을 학교에서 수학 시간에서 배운 적이 있다. y = ax + b 이런 식이 함수이다. x의 값을 넣으면 특정 값을 더하거나 빼거나 하는 등의 값을 변화시키고, 결과를 돌려준다. y = f(x) 라고 표현하고, x를 입력으로 받고 특별한 기능을 해줄 수도 있다.
물론 이런 형태만 있는 것은 아니지만, 이처럼 특정 기능을 하는 것이 함수이다. 그러면 가상함수는 무엇일까? 가상함수는 함수가 존재하지 않은 채 가상으로 있는 함수이다. 이게 인터페이스라고 해서 함수만 정의를 하고 상속 받은 함수 내에서 어떻게 사용할지는 정한다.
이때 두 가지 개념의 이해가 필요하다. 상속의 개념을 알아야 하고, 함수 재정의를 알아야 한다. 상속은 객체 지향에서 나오는 부모와 자식의 개념이다. 학생을 정의할 때 사람으로부터 상속을 받을 수가 있다. 모든 학생은 사람이다. 학생에는 사람의 속성이 모두 포함되어 있다. 사람을 먼저 정의하고, 상속을

받아서 학생을 정의할 수 있다. 직장인도 사람으로부터 상속받을 수가 있다. 그런데, 학생과 직장인의 일과는 다를 수가 있다. 하루에 하는 일을 함수로 정의하고, 하루에 하는 일은 모두가 있고, 그 내용은 다를 수가 있다. 그래서 그런 개념 중 부모에 정의된 아직 내용이 정의되지 않은 빈 이름을 가진 함수가 가상 함수이다.
그러면 함수 재정의를 보면 함수 재정의는 오버라이딩이라는 용어로 표현하는데, 다시 어떻게 사용할지 정의 하는 것이다. 부모에서 정의 했지만, 자식은 그 함수를 동일하게 만들고, 다시 기능을 넣을 수가 있다.



자바 가상 머신

관련성 높은 직무: FE / BE / AI(ML) / Android / iOS / Data / Graphics / Hardware

1. 면접 예상 질문

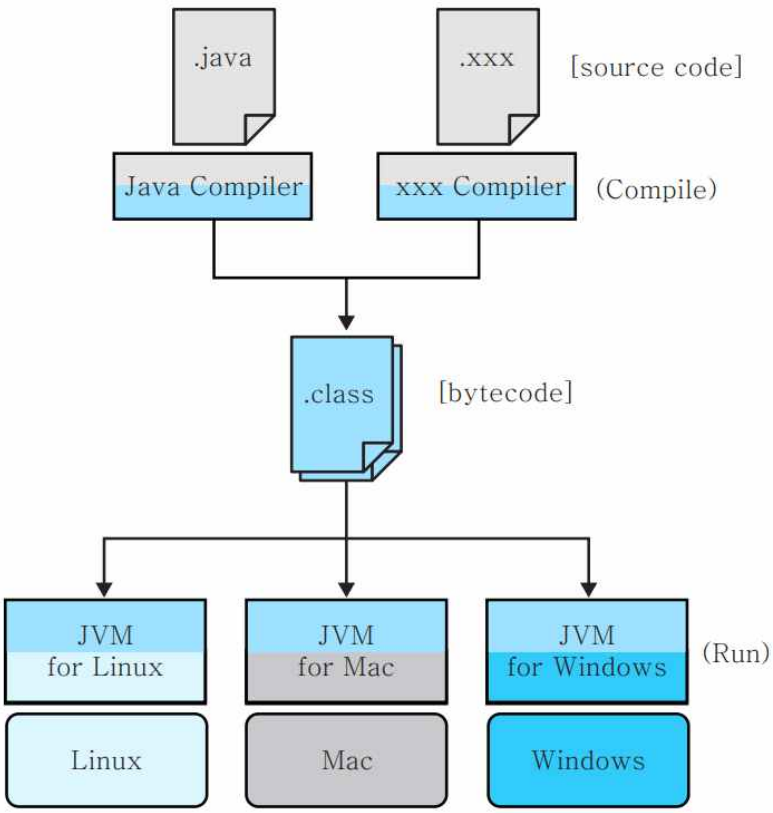
1. 자바 가상 머신에 대해서 말해보세요.
2. 자바 가상 머신의 구성 요소에 대해서 말해보세요.
3. 자바 가상 머신의 장점은 무엇이고, 왜 널리 사용하게 되는지 말해보세요.

2. 답변 핵심 KEYWORD

JVM(Java Virtual Machine), ByteCode, Garbage Collection
자바 가상 머신은 자바가 실행되기 위한 환경이다. 자바의 확산에는 이러한 가상 머신의 역할이 크다.

3. CS 지식

자바는 가장 많이 사용되는 언어 중 하나이다. 그 이유 중 하나가 플랫폼 독립적이기 때문이다. 플랫폼 독립적이라는 의미는 어떤 플랫폼이든 수행이 가능하다는 의미이다. 안드로이드에서 사용되어지는 앱도 자바로 개발되어지고, 웹서버에서 구동되는 서버 프로그램도 자바로 개발되고 수행될 수 있다. 어떻게 이것이 가능할까? 이는 자바가 동작될 수 있는 가상 머신이 해당 플랫폼들에 존재하기 때문이다. 이렇게 자바가 수행될 수 있게 만들어주는 미들웨어를 자바 가상 머신이라고 한다.



[그림] 자바 가상 머신의 요소

이런 식의 가상 머신을 통해서 수행되다 보니 약간 수행속도가 느린 점이 단점일 수 있지만, 하드웨어의 발달로 인해서 단점에 대한 부각보다는 한번 개발되어지면 어떤 플랫폼이든 수행이 될 수 있는 장점이 더 크게 작용을 하게 된 것이다. 자바를 개발할 때에는 자바 환경을 꼭 설치를 해준다.

그리고 자바 바이트 코드는 JVM이 이해할 수 있는 변환된 코드이다. 코드의 명령어 크기가 1바이트라서 자바 바이트 코드라고 한다. 이러한 자바 바이트 코드의 확장자는 .class이다. 자바 바이트 코드는 JVM만 설치되어 있으면, 어떤 운영체제에서라도 실행될 수 있다.

그리고 자바에서는 메모리 사용 후 별도 해제가 없이 자동 관리가 된다. C언어나 C++언어에서는 개발을 할 때 메모리를 사용하면 반드시 메모리 해제 작업을 해줘야한다. 자바는 JVM에서 메모리 관리를 해주는데, 주기적으로 Garbage를 모아주는 작업을 한다. 이를 Garbage Collection이라고 한다. 이때 Garbage는 객체가 만들어지고, 메모리를 사용하다가 코드가 종료되고 만들어진 객체가 더 이상 쓰이지 않을 때 이것을 가비지(쓰레기)라고 부른다.

네이티브앱, 웹앱, 하이브리드앱

1. 면접 예상 질문

네이티브 앱, 웹앱, 하이브리드 앱의 차이점을 말해보시오. 네이티브 앱과 하이브리드 앱의 장단점을 비교해보세요. 어떤 때 네이티브 앱, 하이브리드 앱으로 개발 시 좋을지 예를 들어서 설명해 보세요.

2. 답변 핵심 KEYWORD

모바일 앱, 모바일 웹, 풀브라우징

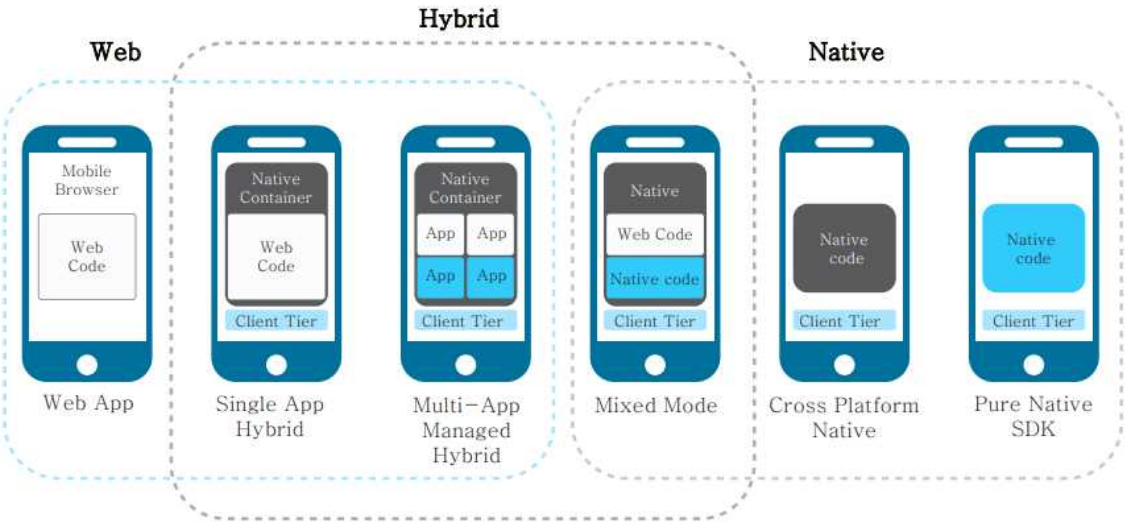
스마트폰의 발전에 따라서 실생활에 필요한 다양한 앱들이 나오게 되었고, 이를 통해서 IT가 발전하고, 앱이 진화되었다. 이 가운데 네이티브 앱, 웹앱, 하이브리드 앱에 대해서 어떤 기술로 만들어졌는지 구분할 수 있어야 한다.

3. CS 지식

모바일앱이 있고, 모바일웹이 있다. 하이브리드앱이라는 말도 존재한다. 모바일앱은 모바일에 설치되는 어플리케이션으로 우리가 사용하는 카카오톡이나 은행 프로그램 같은 것을 말한다. 모바일웹은 모바일에서 보지만, 내부가 웹 기반으로 된 앱을 말한다. 네이버 앱 같은 경우는 네이버가 설치가 되지만 내부를 보면 웹으로 되어 있고, 물론 PC에서 보는 화면과 달리 모바일에 맞춰서 웹이 개발되어진 형태이다. 하이브리드앱은 메뉴와 같이 고정되어 있는 것은 안드로이드나 아이폰에서 제공하는 API를 가지고 네이티브 앱으로 개발을 하고 자주 바뀌는 화면의 내용은 웹으로 하는 형태이다. 앱이라는 말도 사실 어플리케이션의 준말이다. 윈도우도 설치해서 사용하는 PC에서 실행되는 어플리케이션이 있고, 웹 사이트를 접속하여 사용할 수 있는 웹 어플리케이션이 있다. 모바일 내부에 브라우저가 있고, 그 브라우저를 통해서 내용을 보여주는데 PC처럼 보여주기 보다는 모바일에 맞게 사이즈가 조정되거나 재배치 되어 보여준다.

풀브라우징이라는 말과 함께 그런 기술이 가능하다는 얘기가 나온 적이 있었는데, 풀브라우징은 전체 사이트가 모두 보여진다는 의미이다. 과거 단말에서

는 그러한 풀브라우징도 쉽진 않았다. 그래서 별도의 메뉴가 있고, 모바일에서 가능한 수준의 UI로 기능이 제공이 되었었다. 그런데, 풀브라우징이 되면서 모바일에서도 웹사이트를 접속할 수 있게 된 것이다.



[그림] 네이티브 앱, 웹앱, 하이브리드앱

웹앱이라는 말도 있다. 웹 기반으로 구동되는 어플리케이션이다. 요즘 웹사이트에서 왠만한 기능을 모두 수행할 수 있다. 구글은 웹사이트에서 문서 편집이나 엑셀등 다양한 기능들을 할 수 있게 해준다. 이는 나의 컴퓨터에서 설치 하지 않아도 편리하게 사이트를 통해서 기능을 사용할 수 있다는 장점이 있다. 이미지 변환이나 문서 변환도 웹사이트에서 Drag&Drop으로 되는 경우다 많다. 이렇게 복잡한 기능이 가능하고, 웬만한 기능들이 수행이 가능한 이유는 웹사이트가 정교해지고, 네트워이 발달하고, 고사양의 컴퓨터가 되면서 기존에 가지고 있던 병목이 되었던 부분이 거의 없어졌기에 설치된 어플리케이션만큼이나 빠른 속도로 기능 수행을 할 수 있게 된 것이다.
또한 웹앱이 설치되는 어플리케이션보다 유리한 점은 앱을 설치하고 나서 앱이 수정이 발생하거나 기능이 추가가 되었을 때는 앱을 업데이트를 하여야 한다. 앱을 업데이트하려고 하면 사용자가 그 과정을 해줘야 하는데, 웹앱의 경우는 그렇게 하지 않고, 서버에서 수정을 하면 사용자가 다시 접속을 하게 되었을 때 자연스럽게 바꿀 수가 있다. 이러한 장점이 있다.
단점이 없는 것은 아니다. 설치하는 앱을 네이티브 앱이라고 하는데, 네이티브 앱이 웹앱보다 속도가 빠르다. 아무래도 웹브라우저 기반으로 서버에 접속을 해서 메뉴를 가져오고 그 메뉴에 해당하는 기능을 수행하려고 다시 서버를 접속해서 응답을 받으면 아무래도 속도가 느리게 된다. 네이티브 앱은 그에 비해

서 상당히 빠른 속도로 기능 제공을 할 수가 있다.
그래서 하이브리드 앱이라는 것도 나오게 된다. 하이브리드 네이티브앱과 웹앱을 적절하게 하여 일부 자주 사용하는 메뉴는 네이티브 앱으로 만들고, 화면만 웹으로 제공을 한다. 그렇게 해서 장점을 두가지 모두 살리려고도 한다. 이것이 최고 일거 같지만, 하이브리드 앱도 전체 성능이 조금 떨어지고, 일부 수정이 필요할 시에 재배포를 해야 하기에 네이티브앱과 웹앱을 두가지로 제공하는 경우가 더 많다.

디자인패턴

1. 면접 예상 질문

디자인 패턴은 무엇인지 설명해보세요. 디자인 패턴 중 아는 패턴에 대해서 예를 들어서 설명해보세요. 디자인 패턴 중 싱글톤 패턴의 내용과 특징 그리고 어떤 원리로 가능한지 말해보세요.

2. 답변 핵심 KEYWORD

싱글톤 패턴, 공통 모듈, 레퍼런스 아키텍처

개발을 하다 보면 몇 가지 것들을 다른 곳에서도 사용할만한 공통적인 모듈들이 있다. 디자인 패턴은 이러한 모듈을 만드는 베스트 안을 모아둔 것이다. 이를 이해할 때 수준 높은 개발을 할 수 있으며 기업체가 기존에 이런 방법론을 채택해서 개발했다면 이를 이해하는 데 필요한 지식이 된다. 소프트웨어 개발에 있어서 가장 비효율적인 것은 이미 잘 만들어진 것을 다시 만드는 일이다. 그런 의미에서 디자인 패턴을 알고 사용하는 것은 개발 수준을 높일 수 있는 방법이다.

3. CS 지식

개발을 하다 보면 어떤 기능들에 대해서는 유사하게 계속적으로 개발되어지는 경우가 있다. 예를 들면 로그인 화면이라든 파일 전송 같은 공통 기능들은 기존에 만들었던 것을 사용할 수도 있다. 이렇게 보면 개발 소스의 재사용 정도로 할 수 있을텐데, 디자인 패턴은 그 상황에서 어떻게 개발하면 좋을지에 대해서 패턴을 레퍼런스로 만들고, 그것을 참고한다.

그렇게 되면 새롭게 개발을 해야 하는 부담도 덜고, 상당히 수준 높은 아키텍쳐를 바로 개발에 이용할 수가 있다. 그렇다고 디자인 패턴이라는 것이 바로 소스로 제공되는 라이브러리와는 성격이 다르다. 어떤 상황에서 어떤 식의 패턴을 사용하면 좋다는 템플릿 같은 것으로 생각하면 된다. 이는 그 유사한 기능 개발을 할 때 처음부터 고민해야 하는 것을 줄이면서도 있을 법한 시행착오까지 감안하여 설계된 패턴들인 것이다. 객체를 생성할 때, 구조에 대한 패턴 그리고 특정 행위들을 할 때 패턴들이 있다.

대표적인 것이 싱글톤(Singleton) 패턴이다. 싱글톤 패턴은 객체를 단 한개만 생성해서 활용할 때 유용하다. 객체를 생성하다 보면 생성하는 곳에서 사용하고 마치기도 하고, 또 여러 곳에서 여러 개를 만들기도 한다. 하지만 하나의 객체로 계속 사용해야 하는 경우가 있는데 이때 싱글턴 패턴을 사용하게 된다. 객체의 생성과 관련된 패턴으로서 특정 클래스의 객체가 오직 한 개만 존재하도록 보장한다. 즉 클래스의 객체를 하나로 제한한다.
객체가 선언되어지고, 인스턴스가 생성되는데, 이미 생성되어 있다면 getInstance()라는 함수를 통해서 해당하는 객체를 리턴해준다. 이렇게 되면 추가적인 생성이 아니라 기존 생성되어진 객체를 리턴하면서 계속 활용할 수 있게 해준다.

(8) 소프트웨어 개발 방법론

소프트웨어 개발이라는 것은 프로그램 개발이라는 단어와 약간의 차이가 있다. 유사한 단어라고 생각될 수도 있지만, 소프트웨어 개발은 프로그램 개발이 포함하고 있다. 소프트웨어 개발은 프로그램 개발보다 더 범위가 큰 표현이다. 하나의 프로그램을 여러 사람이 기능 또는 화면으로 나눠서 개발을 하는 경우가 있는데, 이 때 적용되는 개발 방법론을 소프트웨어 개발 방법론이라고 한다. 혼자 하는 개발에 대한 것이 아닌, 개발팀 단위로 체계적으로 일하는 것에 대해 정해진 규칙같은 것이라고 보면 된다. 그렇다 보니 다양한 개발 방법론이 존재하고, 시대와 상황에 맞게 적절한 개발 방법론을 적용하여 진행하게 된다. 민첩하게 대응하는 것이 필요한 경우 빠른 개발 방법론을 사용하기도 한다.

애자일

1. 면접 예상 질문

애자일 방법론의 개념에 대해서 말하고, 나오게 된 배경에 대해서 말해보세요. 애자일에서 스크럼, 스프린트, 칸반에 대해 설명해 보세요. 애자일 방법론을 적용하면 좋을만한 개발의 예를 말해보세요.

2. 답변 핵심 KEYWORD

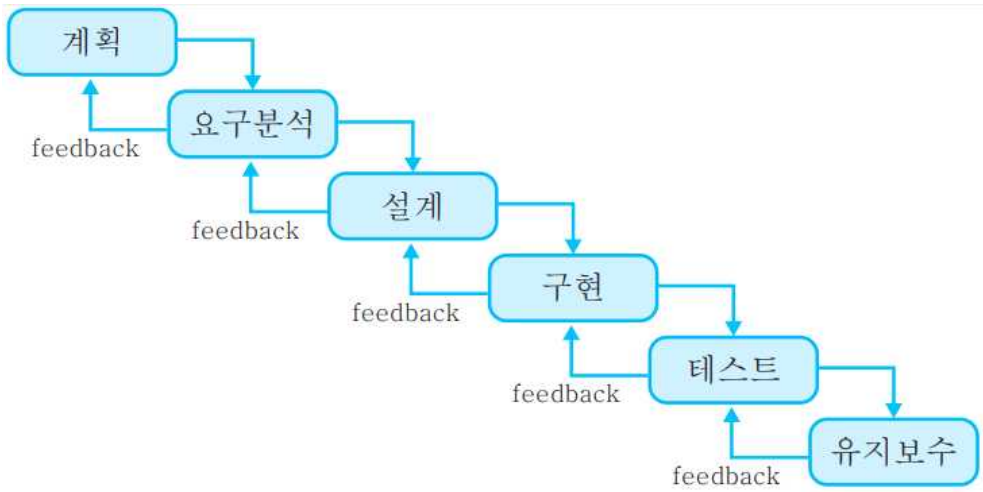
폭포수 개발방법론, 스크럼, 스프린트, 칸반

개발 방법론도 어떤 개발을 하고, 어떤 조직인지에 따라 다르다. 애자일 방법론은 프로토타입과 같이 가벼운 시제품 형태를 만들어서 빠르게 피드백을 받아서 개선하는 방식에서 필요한 방법이다. 기존의 개발 방법은 설계가 꼼꼼하게 이뤄지고 한단계씩 진척을 시키는 폭포수 모델이 많았다. 긴시간이 주어지는 대규모 SI 과제에서 많이 쓰는 방식이다. 그런데 최근의 서비스, 앱 개발등을 보면 초기 개발한 프로토타입을 통해 요구사항을 받아서 개선 방향을 모색한다.

3. CS 지식

프로그래밍은 프로그램을 만드는 것을 의미한다. 알고리즘 등을 사용해서 특정 기능을 개발하는 것이다. 대규모 시스템을 개발 한다면 여러 사람이 함께 만들기 때문에 체계가 있어야 한다. 같은 소스를 여러 사람들이 개발할 때 규칙을 정하고, 체계를 갖추는 것이 개발 방법론이다. 시스템 내에 상호 간의 통신을 하는 규약을 만들고자 할 때 그것을 특정 사람이 임의로 만든다면 문제가 생길 수 있다.

그렇기 때문에 시스템을 구축할 때는 표준화를 하고 개발방법론을 정하고 이후에 개발에 들어가게 된다. 과거에 가장 많이 사용되었던 개발 방법론은 폭포수 개발 방법론인데, 폭포수를 연상시키는 이미지로 구성되어 있다.



[그림] 폭포수 개발방법론

이 폭포수 개발방법론은 어떻게 시스템을 만들지에 대한 기획 과정을 거치고, 이후 시스템 내부를 설계하도록 되어 있다. 전체 프로그램을 집이나 건축물로 비유한다면 설계도를 만드는 과정으로 생각하면 된다. 설계도에 따라서 건축을 시작하게 되고, 이후 테스트 과정을 거쳐서 수정 또는 보완을 하게 된다. 그렇기 때문에 당연히 어떤 모양을 어떻게 만들지 정하고 그에 따라 설계도를 제작하는 모든 과정은 한 치의 오차도 없도록 치밀하고 정교하게 진행되어야 한다. 건물을 지을 때도 각 층마다의 도면이 나와야하는 것은 물론 수도, 전기, 엘리베이터 등의 모든 요소가 세밀하게 고려될 필요가 있다.

소프트웨어 공학에서의 개발 방법론도 그런 차원에서 설계도를 매우 중시했고 그렇게 만들어졌다. 그런데 소프트웨어는 애초부터 건축물과는 조금 차이가 있다. 건축물은 한번 건축을 시작하면 되돌릴 수 있는 부분이 매우 한정적이며, 혹여 되돌리는 과정을 거칠 수 있다고 한들 그 과정에서 심각한 문제가 발생할 수 있다. 하지만, 소프트웨어의 경우는, 건축물과 달리 되돌리는 것이 불가능한 수준도 아니거니와 만들어가면서 설계를 조금씩 발전시켜 나가는 것이 오히려 더 좋을 수도 있다.

개발 시에 소프트웨어 특성에 맞게끔 빠른 대응을 하는 방식이 바로 애자일 프로그래밍이다. 애자일은 우리 말로 '민첩한'으로 번역되는데, 단어 뜻에서부터 알 수 있듯이 순간순간 민첩하게 대응하고 반응하는 것을 의미한다. 그렇기 때문에 애자일은 설계 시간을 짧게 가지고, 개발 진행 과정 중 수정될 수 있음을 감안한다. 그리고 기존과 달리 상호 간의 피드백을 중시하는 방식으로 진행

된다. 매일 아침에 가벼운 형태의 짧은 만남을 가지고, 오늘 내가 진행할 업무와 해당 업무를 진행하기 위해 필요한 상호 간의 지원 사항을 이야기한다. 이를 스크럼이라고 한다.
애자일은 개발 방법론이자 조직 문화이다. 유연하게 대처하자는 목표를 가지고 있고, 공유된 현재 상황에 따라 계획도 얼마든지 바뀔 수 있다. 심지어 일을 쪼개고 나눠서 다른 사람이 자신의 업무를 빨리 끝냈다면 내 업무를 그 사람이 진행할 수도 있다. 자신의 업무를 직접 배정할 수도 있다.
또한 애자일은 수평문화를 지향한다. 수평으로 의사소통이 이뤄질 때 일이 더 잘 진행될 수 있다고 생각한다. 모두가 의사 표현을 자연스럽게 할 수 있고, 그렇기 때문에 의사 결정도 보다 합리적으로 내릴 수가 있다.
애자일 방법론은 어쩌면 개발 방법론 그 자체에 집중한 것이 아니라 실제 개발 과정에 집중하였기에 탄생하게 된 방법론 계의 혁신적인 변화로 볼 수도 있을 것이다. 애자일은 민첩함으로 직역될 수 있지만, 그렇다고 애자일 방법론을 단순히 민첩하게 개발하는 것 정도로 생각하고 해석하기에는 애자일 방법론 자체가 가지는 의미가 매우 넓다.
기존의 대부분의 방법론은 제조에서 나온 것으로 보인다. 제조나 건축, 결과적으로 무엇을 만든다는 것에 있어서는 소프트웨어나 하드웨어 역시 비슷한 측면을 가지고 있기 때문에 방법론적으로 일맥상통한 면이 있었기 때문일 것이다.
그런데 애자일은 어쩌다가 등장하게 되었을까? 필자는 두 가지 시대적 상황의 만남으로 보았다. 말 그대로 민첩한 대응이 필요했기 때문. 변경사항을 바로바로 반영할 수 있다는 것은 소프트웨어가 가지는 가장 큰 장점일 수도 있다. 자동차를 생산할 때는 2년 정도의 단위로 모델을 기획하고 설계한다고 한다. 물론 이것도 조금 빨라질 수는 있지만, 기본적으로 2년이라는 시간이 소요된다는 것이다. 건축 역시 마찬가지로 몇 년의 시간을 들여 진행하기 때문에 사실상 실제로 건축하는 시점은 기획 및 설계 시점에서부터 한참이 지난 후다. 그러나 소프트웨어는 오늘 아침 회의로 결정된 내용을 오후에 개발하여 반영할 수도 있다. 실제로는 이렇게까지 빠른 속도로 반영하기는 어렵지만, 아예 불가능한 일은 아니다. 왜 민첩함이 필요했을까 생각해보면, 그 민첩함이 가능한 분야였고 그러한 민첩함을 사용자가 필요로 했기 때문이라고 얘기할 수 있다. 사용자들은 끊임없이 좋은 기능을 요구하고, 그 요구사항을 충족시키지 못했을 경우 금세 좋은 기능을 가진 앱이나 서비스로 이동한다. 그러므로 사용자의 요구사항을 빠르게 대응할 수 있는 민첩함이 소프트웨어 개발의 필수요건이 되었다는 것이다. 또한 주체가 생산자에서 소비자로 옮겨졌다는 것도 하나의 이

유가 될 수 있다. 물론 당연히 개발의 주체는 생산하는 사람이기는 하지만, 이제 생산자가 만들고 싶은 대로만 개발할 수 있는 시대는 아니라는 것이다. 불편해도 아쉬운 대로 쓰는 시대가 아니다. 도구는 너무 많고, 유사한 앱들도 너무 많다. 골라 쓰기도 바쁘기 때문에 불편하면 굳이 사용하지 않는다. 선택지가 많기 때문에 선택은 소비자가 한다. 그렇기에 소비자가 원하는 것을 민감하게 캐치하고 반영하는 것이 필요해졌다.
그래서 애자일 방법론이 필요해진 것이다. 기민하게 요구사항을 찾고 분석해서 설계하고, 그 설계를 반영하고 개발하고 반영하고, 빠르게 요구사항을 반영하는 것과 동시에 시장 반응을 관찰한다. 애자일 방법론만이 시대 요구를 따라잡을 수 있는 것이다.
전통적인 개발 방법론부터 지금의 개발 방법론까지, 세상에는 다양한 개발 방법론이 존재한다. 그러나 개발 방법론은 어디까지나 방법론일 뿐이다. 개발 방법론을 완벽하게 따르기 위해서 개발이 존재하는 것이 아니라 개발을 보다 더 잘하기 위한 방법의 측면에서 개발 방법론이 존재한다. 개발만 잘하면 되는 문제이다. 그래서 만일 개발 방법론을 적용하는데 문제가 많이 생기거나, 개발 방법론의 모든 것을 전부 맞추기가 어렵다면 과감해질 필요가 있다. 형식을 따르는 것보다는 내용을 채우는 것이 중요하고, 실질적으로 필요한 것에 집중하는 것이 낫다.
개발 방법론을 도입하면, 그 방법론이 무엇이냐만 지나치게 의식하는 경우가 있다. 개발 방법론이란 것은 개발을 진행하며 굉장히 자연스럽게 생겨난 형태인데, 우리는 단순히 그 겉모습만 따라하기 바쁘다.
스크럼을 간단하게 하라고 해서 간단하게 하고, 서서 하라고 해서 서서 한다. 그 근본 취지를 좀 더 생각해 볼 필요가 있다. 예를 들면 외국 사람들은 많은 대화를 하면서 의견을 발전시켜 나간다. 아침에 출근을 해서 차를 한잔 마시면서 오늘 해야 할 일들과 현재의 문제들에 대해 자연스럽게 얘기를 하며 일을 시작하고자 하는 것이다. 우리가 그것을 흉내 낸답시고 회의실에 다 모여서 어색하기 짝이 없는 분위기로, 회의를 짧게 하라고 하니까 형식적인 몇 마디만 나누고 회의를 파한다면 그것은 의미 없는 시간, 형식적인 시간이 되어버린다. 문제가 생기면 민첩하게 대응하며 별도의 회의시간을 가지는 것이 아니라 바로바로 옆에 사람들과 대화하면서 해결해나간다. 그것은 그저 실리콘밸리 어디선가 일했던 팀들의 자연스러운 문화이자 모습이었을 텐데 우리는 그저 애자일을 방법론으로 규명하고 형식을 따라간다. 그럴 것이 아니라 하나의 문화라고 생각하고 최대한 장점을 흡수하고, 어려운 점은 제외해서 우리가 잘할 수 있는 새로운 방법론을 만드는 것도 좋다고 생각한다.

쉽게 말해 이번 우리 프로젝트의 개발방법론은 애자일이야. 이런 식의 정의는 우스운 발상이다. 우리 팀의 개발자들 성향, 성숙도, 일의 내용 모두를 고려해서 우리만의 룰을 정해보는 것도 좋은 방법이 될 것이다.

데브옵스

1. 면접 예상 질문

개발 방법론 중 데브 옵스의 특징에 대해서 말해보세요. 데브 옵스가 나온 배경과 장점에 대해서 말해보세요. 데브옵스에서 기존 전통적 형상관리 방안과 달라지는 형상관리에 대해 설명해 보세요.

2. 답변 핵심 KEYWORD

애자일, 개발/운영

개발 방법 중에 데브 옵스는 개발을 하면서 운영도 하는 것이다. 왜 이런 방법이 나오게 되었을지에 대해서 아는 것도 중요하며 실제 기업들에서 많이 도입하는 방법론이여서 사전에 알고 있을 필요가 있다. 이전에는 개발은 개발조직에서 하고, 운영은 운영조직에서 하다보니 서로의 업무 내용을 파악하지 못하거나 운영에 관한 배려가 없는 시스템이 만들어지는 경우도 있고, 또 유기적으로 유연하게 수정하면서 대응해야할 사항이 많은데, 운영 조직에서 안정성을 최우선으로 생각하기에 데브옵스가 요즘 같이 민첩하게 대응해야 하는 상황에서 필요한 조직 구성이 된 것이다.

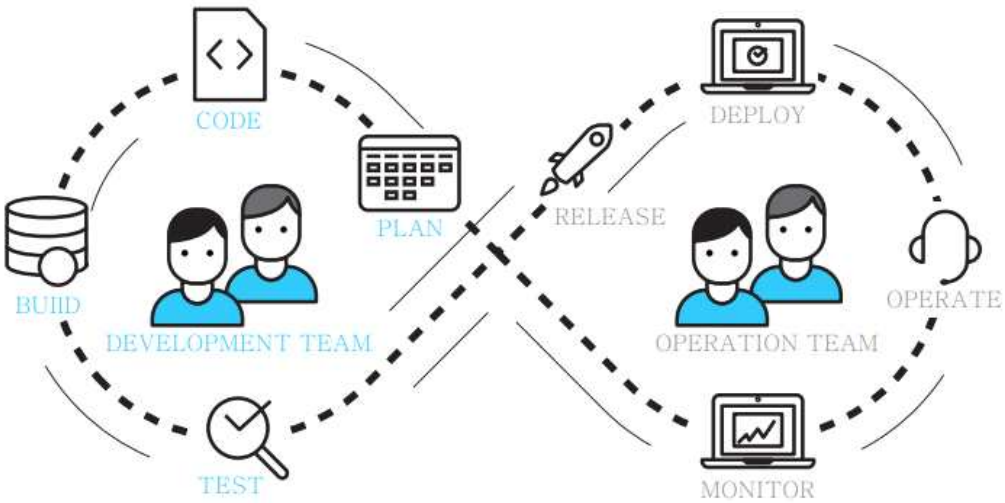
3. CS 지식

DevOps는 Development와 Operation의 합성어이다. 개발과 운영 함께 잘해보겠다는 것이다.

왜 이런 개념이 나왔을까? 원래 구축을 하고 운영으로 인수인계를 한다. 구축하는 사람들을 개발팀이라고 부르고, 그 구축된 시스템을 운영해주고 유지보수해주는 팀을 운영팀이라고 부른다. 소프트웨어 개발은 사람이 하기에 사람에 대한 이해가 필요하다.

각각 소속된 사람은 어떤 심리를 가지고 있을까? 구축팀은 어떻게든 납기 내에 고객이 원하는 시스템을 구축을 해야 한다. 어떤 상황에서도 일정 준수가 목표이다. 기능은 요구사항대로 수행되면 되고, 일정을 준수하려고 한다. 운영팀의 입장은 다르다. 실제 운영을 해야 하는데, 요구된 기능을 일정에 겨우 맞

춘 시스템은 문제 투성이 인 경우가 많다.
운영을 해야 하는데, 뭐가 뭔지 모르겠고, 시스템도 엉망인데다가 소스도 엉망이다. 어딜 손대야 할지도 모르겠는데, 메뉴얼 마저 부실하다. 그리고 개발팀 사람들은 개발이 끝났다고 어디론가 하나둘 도망가기에 바쁘다. 이게 현실이다. 그래서 운영팀은 욕을 하면서 어떻게든 운영은 해야 하니 소스를 손봐가면서 어떻게든 운영을 한다. 기능들은 어떻게든 동작은 되는데, 뭐 하나 하려면 왜 이렇게 느린지. 그리고 무슨 기능을 하려고 하는데, 낯선 것들 투성이다.
실제 사용할 사람들에 대한 배려는 전혀 없다. 왜 이렇게 되었을까? 개발팀은 고객의 요구사항에 촛점을 맞추고, 그 요구사항을 구현하기에 바빴다. 운영자들이 시스템을 어떻게 편리하게 잘 사용할 수 있는지까지 배려해줄 여유가 없다. 운영자들이 시스템을 하다 보면 너무 번거롭거나, 버튼의 위치가 불편하거나 너무 느리거나 불편하기 짝이 없다.



[그림] 데브옵스

그래서 개발팀과 운영팀이 하나가 되어 움직이는 것이다. 운영팀이 실제 운영시 필요한 것들을 개발에 넣고, 개발팀은 운영관점에서 개발을 하고 구축이 끝나고 운영 시작이 아니라 구축을 하면서 운영을 하는 것이다. 애자일하게 개발을 하고, 그것을 운영에 반영하는 것이다. (CI) 애자일과 CI가 같이 돌아가면서 운영 중심의 개발이 되는 것이다. DevOps의 취지는 그러하다. 방법론 자체가 중요한 것이 아니라 개발을 하되 운영 관점에서 개발을 해보자는 취지이다. 많은 곳에서 시도를 하고 있다. 자칫 운영도 하고 개발도 하는 애매한 모습의 팀이 만들어지지기는 하지만, 필요한 방향이다.

(9) 빅데이터

몇 년 전부터 아마도 빅데이터라는 용어가 IT분야에서 가장 핫한 용어가 되었다. 초기 빅데이터 기술은 기술을 연구하는 단계라고 하면 이제 많은 기업에 활용하는 단계로 변화했다. 초기에 수많은 돈을 들여서 빅데이터 시스템을 구축한 회사들에서 ROI 관점으로 과연 투자 대비 효과가 무엇인가라는 생각하여 빛 좋은 개살구 같은 존재로써 몇년을 지내다가 IoT 기기의 발전, 사용자의 행동 패턴을 분석하려는 마켓 변화 그리고 시스템들이 발전하면서 데이터 분석이 결국 승부처라는 생각에 너도 나도 없이 빅데이터를 적용해보고자 하고 있는 상황이다. 심지어 공공기관에서 공무원들의 민원들을 분석을 하여 이후 발생 될 문제에 대해서 빅데이터 분석을 통해서 대비하는 등의 준비를 하고 있다. 예를 들면 불법 쓰레기가 버려졌다는 민원을 접수 받으면 며칠 뒤에 해당 지역에 쥐가 출몰한다는 민원이 들어오는 상관 관계 분석을 통해서 미리 방역 조치까지 겸하여 한다던지 하는 빅데이터 기술을 이용한 예측을 실무에서 적용하고 있는 상황이 된 것이다.

빅데이터 기업에 취업을 하려는 상황에서는 빅데이터 기술에 전반을 이해하고 그리고 내가 역할을 하고자 하는 부분에 대해서 깊이 있게 지식을 연구하여서 준비할 필요가 있다. 그리고 빅데이터 시스템 도입은 꼭 빅데이터 개발을 하고 있는 회사가 아니더라도 IT 시스템을 구축하는 것에 기본이 되었다. 대개 시스템들이 데이터 분석을 하고자 하여 최소한의 비용으로 빅데이터 분석을 고려하고 있다.

빅데이터 기술

1. 면접 예상 질문

빅데이터 기술에 대해서 말해보세요. 빅데이터 기술을 위한 절차에 대해서 말해보세요. 빅데이터 분석 알고리즘 및 모델링에 대해서 설명해보세요.

2. 답변 핵심 KEYWORD

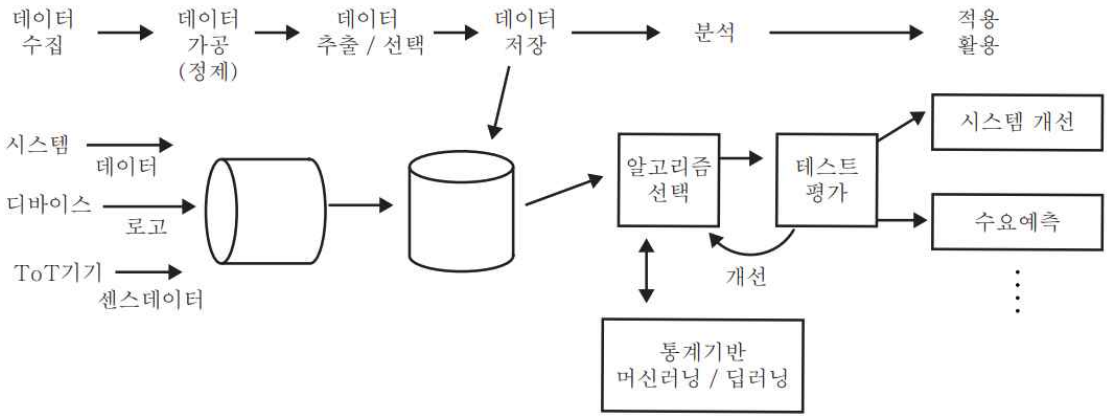
수집/정제/분석/가시화, 로그, 비정형 데이터, NoSQL

빅데이터는 몇 년 전부터 아주 자연스럽게 프로젝트나 서비스에서 도입하여 사용하고 있는 기술이고, 이제는 분석을 기반으로 하는 서비스 예측이나 경영 정보를 제공하는 현실이다. 이에 대해서 개발자들도 알아야 할 필요가 있다. 필수 기술이 되어진 만큼 빅데이터 인프라를 어떻게 구축 할 수 있는지, 그리고 빅데이터 절차 수집, 정제, 분석, 가시화와 같은 절차 내 기술요소에 대해서 알아둘 필요가 있다. 이제 빅데이터 시스템 구축이나 분석 방법을 아는데 그치지 않고, 기업들은 이것으로부터 수많은 가치있는 결과를 기대하는 상황이다.

3. CS 지식

IT기술은 기존 업무를 편리하게 해준다거나 빠르게 처리해 주는 역할을 해왔다. 기존 IT가 기능으로써의 가치를 제공했다고 하면 빅데이터 기술은 의사 결정이나 의미 분석을 제공하는 더 뛰어난 가치를 제공한 것이다.

빅데이터의 구성은 데이터를 수집하고, 정제하고 분석하고 그 과정에서 의미있는 상관 관계등을 분석하고 이를 도식화하여 보여주는 것이다. 기존 데이터와는 달리 이때의 데이터는 정보 성격의 데이터 뿐만 아니라 기존에 버려졌던 또는 참고하지 않았던 데이터도 포함한다. 과거에는 시스템의 로그 파일은 문제가 발생했을 때 찾아보는 수준으로 적극적으로 활용하지는 않았다. 빅데이터에서는 이러한 로그 파일의 내용도 적극적으로 참고 할 수가 있다.



빅데이터는 IoT와도 연결이 될 수 있다. IoT 기기들에서는 수많은 정보들이 실시간으로 모일 수 있다. 이러한 데이터를 단순 기록물이나 정보 저장형태가 아니라 분석용으로 데이터를 수집하게 된다. 매장에서 고객의 이동 경로에 따른 상품의 배치등도 상당히 중요하게 상호관계가 있을 수 있다. 그래서 이러한 데이터 분석을 통해서 좀더 경영에 도움이 될만한 의사 결정을 할 수가 있다.
데이터를 수집할 때 이 데이터는 정의가 잘되어진 데이터라기 보다는 비정형 데이터가 있을 수 있다. 이를 위해서 기존 방식의 데이터베이스에 SQL이 아닌 비정형 데이터로 NoSQL 데이터 베이스를 사용해야 할 수 있다.
이는 고속의 데이터 처리나 대용량의 데이터 베이스 처리에 적합하다. 그리고 모든 수집된 데이터로 분석을 하기는 데이터의 양도 많고, 유의미할지에 대한 의문이 있을 수 있다. 이를 정제 또는 전처리라는 과정을 통해서 분석하려는 데이터 중심으로 정제를 하게 된다. 그리고 각종 분석의 알고리즘 등을 통해서 분석하고자 하는 대상들을 분석한다.
데이터 분석에 관련해서는 철저한 사전 검증을 통해서 이뤄진다. 빅데이터 플랫폼을 도입 시 최소한의 목표치와 기대하는 바는 준비가 되어야 한다. 어떤 데이터를 모으고, 어떤 상관 관계를 기대한다가 있어야 효율적인 분석 및 가치를 도출할 수 있다. 무작정 수집 하다보면 가치 정보가 나올 수 있다는 식은 불필요한 데이터만 쌓일 수가 있다.
데이터 분석을 통해서 의미 있는지 가시적으로 보여주는 것이 가시화(Visualization)이다. 가시화를 필요로 하는 이유는 빅데이터의 최종 결과물을 보는 사람이 회사 내에서 경영을 수행하는 사람 또는 기획하는 사람들로 데이터 베이스를 바로 접근해서 데이터를 본다기 보다는 화면에 표시되는 그래프나 차트를 통해서 보는 것이 내용 파악에 쉽기에 그렇게 제공을 한다. 빅데이터의 중요한 영역이기도 하다.

빅데이터가 초기에 데이터 수집 및 플랫폼 구축에 비중이 커서 하둡이라든가 스파크과 같은 툴을 다룰 수 있고, 데이터 연동등에 목표가 있었는데, 클라우드 시스템등에서 빅데이터를 지원하는 PaaS 서비스[1] 등이 있어서 시스템 구축 보다는 의미 있는 데이터 분석에 가치 제공에 집중하려고 하는 상황이다. 원래의 목적에 맞게 집중을 하면 된다. 시스템은 제공되어지는 것을 사용했을 때 구축 비용 감소라든가 시간 절감을 기대할 수 있기때문이다. 분석가들도 데이터 분석을 어떻게 할 것인가 보다 어떤 데이터를 분석하여 어떤 점에서의 의미 있는 분석을 할 것인가라는 주제 좀 더 포커스를 해야 하는 상황이다.

1) 데이터를 수집, 적재 할 수 있는 기술과 데이터 초고속으로 처리 할 수 있는 플랫폼 데이터 연동을 위해서 빅데이터를 지원하는 클라우드 기능이 있음.

DBScan

1. 면접 예상 질문

데이터 분석 기법의 종류에 대해서 아는 대로 말해보세요. 데이터 분석 방법 중 클러스터링 방법이 필요한 상황과 방법에 대해서 말해보세요. 클러스터링 방법 중 DBScan 방법에 대해서 말해보세요.

2. 답변 핵심 KEYWORD

클러스터링, K-means

데이터 분석 방법에는 여러 가지 방법이 있다. 그 중 클러스터링이라는 기술이 있다. 이는 데이터의 유사도에 따라서 묶는 방법인데, 데이터 분석에 대한 기본적인 지식으로 볼 수 있다.

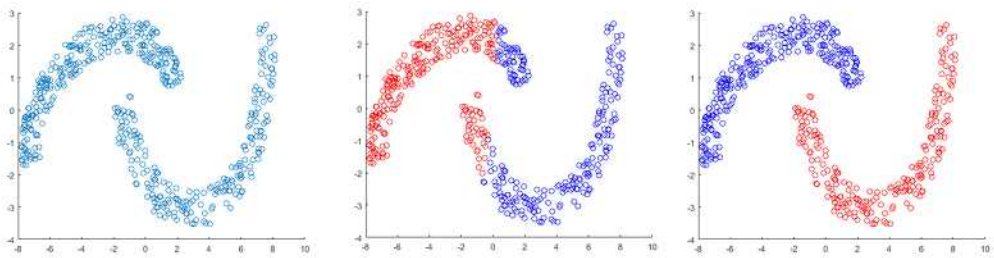
3. CS 지식

빅데이터 분석 시에 알고리즘을 보면 상관관계 분석이나 예측 분석등 다양한 분석 기법들이 있다. 그 중에서 DBScan은 클러스터링 기법에 해당한다. 클러스터링은 군집이라고 하는데, 유사한 데이터를 묶는 것이다. 클러스터링이 필요한 이유는 방대한 데이터에서 유의미한 데이터들을 묶음으로써 그 전체 데이터를 분류 할 수 있다.

쉬운 예를 들면 제품이 좋은 평을 받고 있는지 SNS서비스등에서 댓글을 분석하고 싶다고 했을 때 수많은 댓글이 달리고 있는 상황에서 이 댓글을 사람이 일일이 보지 않으면 그 내용이 어떤 것인지 파악하기 힘들다. 중요한 점은 댓글들에 좋은 평이 달리고 있는지 알고 싶다는 것이다. 그러한 때에 댓글을 수집하고 하나 하나 워드로 집계를 하고 그 집계된 워드에 감정에 대한 평점을 메길 수가 있다. 이때 그 워드가 동일한 단어가 아니라 유사한 단어들일 수 있다. 그러면 이러한 단어에서 비슷한 단어들을 클러스터링을 할 수가 있다. 이 클러스터링을 통해서 군집을 형성하고 어떤 군집이 더 많은지를 가시적으로 볼 수가 있다.

클러스터링 기업 중에는 대표적으로 DBScan과 K-means 알고리즘이 있다.

K-means는 단순 가까운 거리인지를 측정하는 것이다. K-mean에서 K는 몇 개의 점을 기준으로 할 것인가 이다. 전체 집합을 몇 개로 나눌 것이다. K가 2이면 2개로 분류하는 것이다. 2개로 분류할 때 기준점에서 가까운 2개의 집합을 나눈다. K가 5이면 5개의 군집으로 분류하는 것이다.



[그림] K-means

DBScan은 방식이 약간 다른데, 좀 더 유의미하게 나눌 수가 있다. 이것이 가능한 이유는 DBScan에서 DB(Density-based)는 밀도를 의미한다. 밀도로 가까운 것을 찾고 그 것을 그룹으로 형성한다. 그리고 몇 개인지 사전에 정해주지 않고, 데이터를 보고 근접한 것들을 찾는 것이다. 조금 더 유의미한 결과를 찾을 수도 있고, 몇 개로 나눌지도 알고리즘적으로 계산을 해준다.

(10) 인프라

기존 IDC(데이터 센터)에서 수많은 기업들 클라우드로 이전 했거나 이전을 고려하고 있다. 이미 클라우드 기술은 검증이 되었고 비용 효율적인 측면, 안정적인 측면에서 검증이 되었다. 초기에는 클라우드에 데이터를 올리는 것이 안전한가라는 생각으로 반신반의했던 기업들도 오히려 자체적으로 관리하는 것보다 안전할 수 있고, 확장이 용이한 점에서 많은 기업들의 호응을 받았다. 클라우드의 특징 중 하나는 이용이 간편하다는 점이다. 기존 IDC에서 장비를 신규로 하나 도입하기 위해서 수많은 의사 결정과 서버에 대한 전문성이 필요했고, 적절한 용량이라는 산정이 어려움이 있었다. 서버 구매에서 셋팅하기까지 적게는 한달 길게는 수개월이 걸릴 수도 있었는데, 클라우드의 경우는 편리하게 신규 가상 서버를 띄우기까지 몇 분으로 가능하다. 이러한 점에서 개발자도 서버를 쉽게 다룰 수 있다는 점에서 많은 기업들이 서버 관리자를 별도로 두지 않고, 개발자로써 대응하려는 모습도 보인다.

도커

1. 면접 예상 질문

• 도커 기술에 대해서 말해보세요.
• 도커과 기존 컨테이너 기술의 차이와 활용 방법에 대한 차이를 말해보세요.
• 기존 서버를 도커 기반으로 전환하려고 할때 고려할 점과 전환 방법에 대해서 말해보세요.

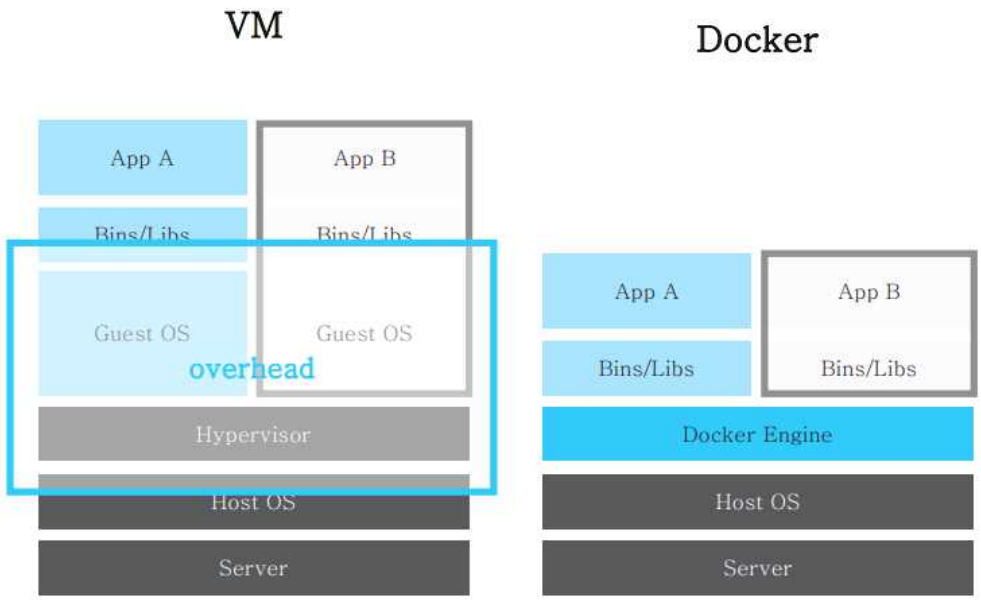
2. 답변 핵심 KEYWORD

컨테이너, 쿠버네티스, VM, 클라우드
도커는 구글이 다년간 서비스 운영 체제로 사용한 만큼 서비스 운영 시 분명한 장점이 있고, 전 세계적인 추세이기 때문에 알아둘 필요가 있다. 도커 기반과 단순 클라우드의 차이점에 대해서 명확하게 알아야 하고, 도커로 전환했을 때 얻을 수 있는 장단점에 대해서 알아야 한다. 그리고 도커로 전환이 된다고 했을 때 어떤 방식으로 전환할 수 있을지에 대해서 전략적인 접근이 필요하다.

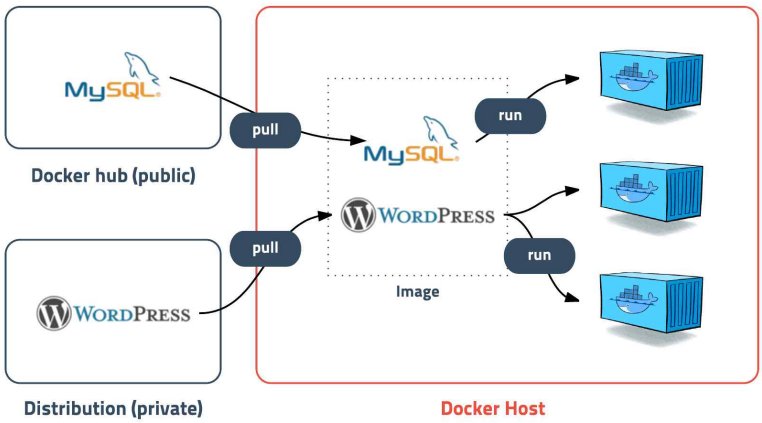
3. CS 지식

도커라고 하면 항구에서 선박이 오가는 것이 연상이 된다. 컨테이너라고 하면 일정 크기의 화물을 넣는 규격화된 대형 화물 상자를 의미한다. 컨테이너에 어떤 물건이 들어 있는지는 모르지만, 배에 싣고, 이동하거나 물건을 나르기에는 규격화 되어 있어서 좋다. 컨테이너는 어떤 내용을 담고 있다.
컨테이너라는 것이 일반적인 기술이고, 도커는 컨테이너 기술을 이용해서 만든 대표적인 제품이자 기업명이다. 워낙 유명하다보니 그것이 일반 명사처럼 쓰여지고 있다.
도커는 컨테이너로 구현된 기술이고, 구글이 개발한 컨테이너 운영 기술이 쿠버네티스이다.
도커의 철학은 기존의 VM(Virtual Machine)의 발전에 있다. VM은 가상화 기술로써 하나의 컴퓨터 하드웨어에 운영 체제를 여러 개 수행될 수 있게 하거

나 여러 대의 컴퓨터를 하나의 운영체계로 보이게 하는 등의 가상화 기술이 발전하였다.
이로 인해서 클라우드 기술이 본격화되고, 클라우드 내에서는 인스턴스라는 이름으로 가상 머신을 사용자들에게 제공을 하였다. 가상 머신은 어떤 컴퓨터에서 동작되는지 알 수 없지만 어찌됐든 운영체제로써의 역할을 수행한다.
그런데, 이러한 클라우드 서비스 내에서 기능으로 제공받을 때는 운영체제 영역이 겹치더라도 크게 상관은 없지만, 하드웨어를 준비하여 운영체계를 여러 개 구동 시킬 때는 이 운영체계의 부분이 오버헤드로 되어질 수가 있다. 윈도우에도 Virtual Box라든가 VMware와 같은 툴들이 있는데, 이러한 툴들을 통해서 여러 개를 운영체계를 띄운다고 하면 하나의 컴퓨터에서 여러 VM을 띄울 때 각각 Guest OS가 중복되어서 가지고 있어야 했다. 이때 Guest OS를 중복으로 가지지 않고 어플리케이션과 라이브러리만 가지고도 독립된 실행 환경을 만들 수 있어서 이 컨테이너 기술이 매우 유용하다.

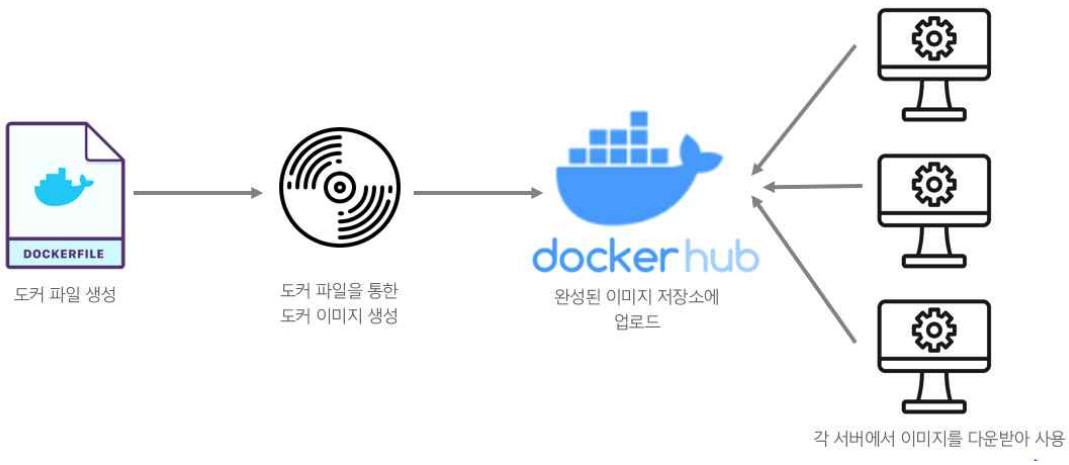


그리고 도커/컨테이너가 각광을 받았던 부분은 필요한 설정에 관해서 별도로 가져갈 수 있는 환경 독립성에 있다. 예를 들면 JBoss나 Tomcat이 웹 어플리케이션 서버로 존재한다. 이러한 어플리케이션을 매번 설치할 수도 있고, 하나의 VM에 설치하고 이것을 복제해서 사용할 수도 있다. 하지만 이 또한 번거로운 일이도 또 서비스 형태에 맞게 특정 설정같은 것을 한 경우라면 더욱 그렇다. 이러한 설정까지를 포함해서 도커 이미지로 작업을 하면 그 이미지를 공유 받아서 올리기만 하면 필요로 하는 설정까지 된 WAS서버가 구동이 된다.



[그림] 도커 작업

그리고 결정적으로 기존 모노릭스(하나의 시스템 내 모든 기능을 통합해서 넣었던 개발 형태)의 단점은 한번의 수정이 있어도 다시 배포를 해야 하는 문제가 있었고, 이를 인스턴스로 나눈다고 하지만, 이때 불필요한 운영체제가 늘어나고, 라이브러리가 서로 간 충돌이 될 수 있는 문제도 있었다.
이러한 것을 컨테이너의 기술을 통해서 필요한 라이브러리와 어플리케이션만 포함되는 이미지를 만들어서 해결을 했다.



[그림]

그러한 것을 도커 이미지라고 하고, 이 이미지에 업데이트 하거나 다운받아서 사용할 수 있고, 내가 만든 이미지를 업로드 할 수도 있다. 이것을 레포지토리라고 한다. 여러 개의 도커를 띄우는 것도 몇 줄의 명령어로 가능하고, 자동으로 재기동 되거나 운영의 가용성 측면에서 놀라운 대응을 해준다.

클라우드

1. 면접 예상 질문

클라우드의 개념이 기존 데이터 센터와 비교했을 때 차이는 무엇인지 말해보세요. 클라우드에서 IaaS/PaaS/SaaS에 대해 설명해보세요. 기존 서버들을 클라우드로 이전하고자 할 때 고려해야 할 점과 효율적인 방법에 대해서 말해보세요.

2. 답변 핵심 KEYWORD

가상화 기술, 데이터 센터, Iaas, Paas, Saas

클라우드는 이미 많은 서비스 또는 기업에서 도입하였고, 도입하고 있는 추세이다. 개발자들도 클라우드 개념을 이해해야하고, 사용할 수 있기를 원하고 있다. 또한 기존 서비스들을 클라우드로 이전하는 전략에 대해서도 생각해볼 수 있어야 한다. 비용관점의 ROI 분석, 서비스의 트래픽 분석, 시간대별 날짜별 추이 등에 의해서 아키텍쳐를 설계해 볼 수 있다.

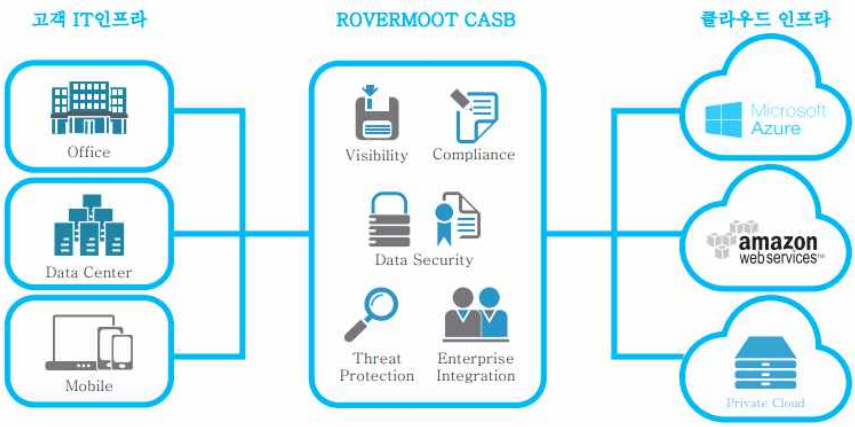
3. CS 지식

클라우드 컴퓨팅은 클라우드라는 말은 구름이라는 의미에서 나왔다. 어딘가 존재하는 컴퓨터들이다. 어딘가는 존재한다. 하지만 굳이 우리가 알아야 하는 것은 아니다. 물리적으로 어떻게 구성되어 있는지 몰라도 된다. 모른다는 것은 큰 리스크이기도 하지만, 편리함을 주는 측면도 있다.

과거 서버 하나를 구입해서 운영을 하는 것까지는 큰 비용과 시간이 들었다. 클라우드 컴퓨터는 이미 서버가 준비되어 있고, 바로 이용할 수 있다. 과거 서버를 구입하는 것은 매우 큰 돈이 든다. 수천만원에서 수억까지 투자를 해야 서버를 구입할 수 있다. 하지만 클라우드 컴퓨터는 매우 적은 금액으로 이용할 수 있다. 편리함과 비용측면은 매우 좋은데, 보안 관점에서 우려를 하는 사람들이 있다. 물론 우려되는 부분이 있다. 예를 들면 데이터가 타 회사의 서버에 저장된다는 점 이점 때문에 시스템의 전체를 클라우드로 옮기는 것에 대해서 보수적인 입장이였다.

클라우드의 안정성을 설명할 때 이를 은행으로 비유한다. 집에 있는 개인 사설

금고에 돈을 보관하는 것보다 은행에 맡기는 것이 더 안전하다고 할 수 있다. 물론 그 클라우드 서비스 업체가 망하면 어떻게 하냐고 물을 수 있다. 물론 그 리스크도 있다. 그 업체가 망하는 확률과 직접 서버를 운영하다가 서버가 문제가 생겨서 데이터가 날라가는 확률 사실 어떤 것이 더 높은지는 모르겠지만, 사실 이러한 부분의 의사 결정을 하는데 첫번째 고민할 사항은 아닐 것이다. 이미 클라우드 서버는 시장에서 신뢰는 충분히 얻은 상황이다.
시스템의 구성의 발전은 메인 프레임 큰 서버를 한대 두고, 단말기에서 입력, 출력 위주로 했던 시대 클라이언트 서버, 단말기의 성능이 향상되고, 분산되었던 시대 분산 컴퓨팅이라고도 볼 수 있다.



[그림] 클라우드 인프라

기업에서 IT의 초기 투자 비용 절감, 시스템의 신속한 구축 등의 이유로 클라우드 도입을 고려한다. 시스템 한 대를 추가하는데, 몇 분이면 가능하다. 과거 시스템 구축 시 시스템을 구매하려면 견적부터 발주, 시스템 설치, 셋업 등 수많은 절차로 상당히 긴 시간이 필요로 했다. 그것이 하루 내에만 된다고 하면 혁신 수준이다. 그리고 시스템을 구축했는데, 추가로 몇 대 더 필요하다고 했을때 시간은 또 하염없이 기다려야 했다. 그리고 갑자기 사용량이 늘었을 때나 갑자기는 아니더라도 수일 내에 2배로 증가할 것을 예측하는 상황에서 시스템 용량을 늘려야 하는 경우라면 상황은 더 복잡해진다. 하지만 클라우드 컴퓨팅을 이용한다면 이런 상황에 대해서 쉽게 대응이 가능하다.
초기 비용은 굉장히 적게 든다는 것과 또한 사용하는 만큼 과금을 한다는 장점이 있다. 시스템은 초기에 구입 비용이 굉장히 크다. 과거 한대다 1억을 호가 했다. 왜냐면 개인용 컴퓨터가 아니라 서버로써 많은 접속자를 처리해야 하고 1년 정도의 사용자 증가를 예상해서 서버를 미리 구매를 해야 했다. 그리

고 초기에는 사용자가 많지 않은 동안은 고가의 서버를 구매하고도 그 서버는 계속 쉬고 있어야 했다. 그러가다 용량이 다 찰 때가 되면 급하게 서버를 또 증설해야 했다. 증설하는 시스템도 비효율적으로 충분히 늘려야 했다. 클라우드 컴퓨팅은 초기 비용이 적을 뿐만 아니라 증설시 용이하고 그리고 증설에 따라서 사용한만큼만 지불을 할 수 있다. 물론 용량이 매우 큰 인스턴스를 셋팅하고, 자원을 별로 활용하지 않는다면 설정한 사람이 몫이지만, 알뜰하게 적절한 것으로 선택해서 설정한다면 충분히 효율적으로 활용할 수가 있다. 그래서 늘어난 인스턴스 만큼만 지불하면 된다.
클라우드 컴퓨팅이 가능한 기술적인 근거는 가상화이다. 여기서의 가상화는 우리가 일상적으로 말하는 가상과는 약간 거리가 있다. 가상이라고 하면 없는 것을 있는 것처럼 보이는 정도 아닐까 가상현실, 현실에 없는 것을 보여주는 것이다. 클라우드 컴퓨팅의 가상화가 가지는 의미는 이런 개념이다. 한 대의 컴퓨터를 여러 대의 컴퓨터처럼 나눠 쓸 수 있는 것 이 과정에서 실제 물리적으로 나눈다기 보다는 가상적으로 나눈다고 해서 가상화이다. 이러한 것이 가능한 것이 내부 기술 두 가지가 있다. VM(Virtual Machine)과 분산처리 기술이다. VM은 하나의 OS환경에서 어플리케이션이 실행할 수 있는 독립된 영역을 만들어주고 실제로 OS내부에 또 다른 OS가 있는 것처럼 동작되는 것이다. 분산처리는 여러 대의 컴퓨터에 나뉘어진 자원이나 데이터를 하나처럼 처리할 수 있는 기술이다. 이것은 대용량의 데이터 처리를 요하는 곳에서 필요로 한다.

마이크로서비스

1. 면접 예상 질문

마이크로서비스의 개념 대해서 장점과 단점에 대해서 말해보세요. 기존 서비스를 마이크로서비스로 전환하고자 할 때 고려할 점과 방법에 대해서 제시해보세요. Monoltic한 소프트웨어 구성과 마이크로서비스와의 차이점에 대해 설명해 보세요

2. 답변 핵심 KEYWORD

모노리틱(Monolithic), MSA(Micro Service Architecture, 토폴로지(Topology), 망형, REST API

마이크로서비스는 서버를 여러 모듈로 나누고 설계함으로써 안정성을 더 높이는 서버 설계 방식이다. 이를 도입하는 다국적기업과 국내외 기업이 늘고 있다. 마이크로 서비스 아키텍쳐를 이해하려면 이 아키텍처가 왜 나왔는지에 대해 알아야한다. 이는 기존 모노릭(Monolithic)아키텍처로 운영할 때 장애 관리 등이 힘들었기 때문이다. 마이크로 서비스 아키텍처가 이러한 단점을 극복하고, 전체 서비스 안정화를 위해서 나온 것임을 생각하면 왜 구조가 분산되어 있는지 대해서도 이해할 수 있다.

3. CS 지식

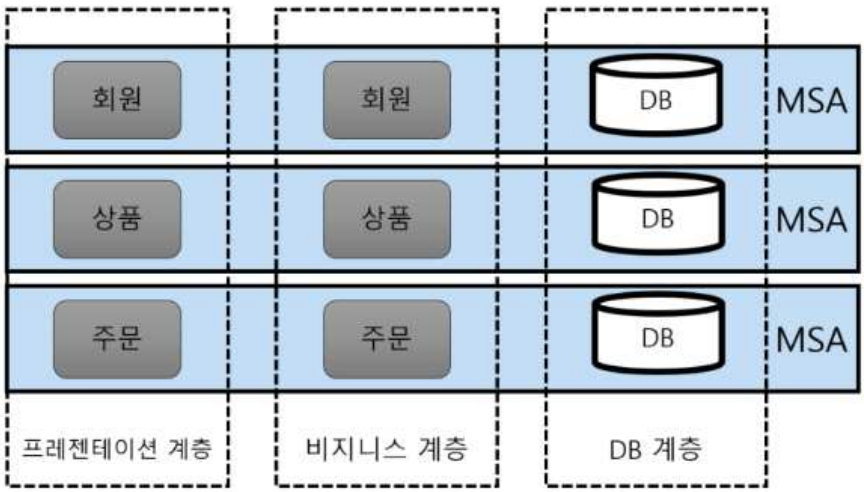
기술을 이해하기 가장 좋은 것은 그 기술이 왜 나왔는지를 아는 것이다. 마이크로 서비스라는 기술을 잘 적용한 사례가 넷플릭스이다. 그렇다면 왜 넷플릭스는 마이크로 서비스를 도입했는가를 보면 된다.

마이크로 서비스는 기존 모노로닉 구조의 통합 시스템을 기능 단위로 쪼갠 것이다. 그리고 해당 기능이 안되는 순간에도 전체 시스템에는 영향을 적게 한 것이다.

시스템이 대규모가 되면서 전체 시스템 장애와 같은 대규모 장애가 몇 번 나면서 이를 어떻게 하면 견고하게 유지할까 를 계속 고민한 것이다.

대표적인 클라우드 업체 아마존도 시스템은 장애가 발생할 수 있다고 말한다. 대신 그러한 때 어떻게 유연하게 대체 하는지 여러 가지 대안을 기술적으로

제공한다. 대표적으로 데이터 베이스 fail-over 기능이 있다. 메인 데이터 베이스 서버가 문제가 생기면 다른 쪽의 데이터 베이스가 바로 연결되어 서비스에 문제가 없도록 해준다.
결국 시스템(하드웨어 or 인프라)을 믿지 않고, 아키텍쳐 적으로 풀어 본 것이다. 마이크로 서비스는 말 그대로 아키텍쳐를 쪼갠 것이다. 거대하고 복잡해지는 서비스의 일부분으로 나누고, 각각 별도로 수행될 수 있게 만든다.
기존 시스템이 하나로 되어 있었던 것을 Monolith이라 한다. 여기에 사용자 관리부터 업무 기능까지 모두 통합되어 있었다. 이로 인해서 시스템도 복잡해졌지만, 또 문제가 생겼을 시에 그 문제도 한꺼번에 영향을 주었다. 그리고 배포 시에도 일부가 변하더라도 다 배포를 다시 해야 하는 단점이 있었다. 이를 Micro service 형태로 바꿔서 시스템 영향을 줄이게 한다. 바꾸게 되어 있을 때 모듈간의 복잡도는 있지만, 각각을 모듈별로 분리함으로써 기존에 Monolith 구조가 가지고 있던 문제를 해결할 수가 있었다.



[그림] 마이크로서비스

그렇게 구성이 되면 일부 모듈이 안되더라도 다른 기능은 정상적으로 될 수 있다. 확장성도 있다. 그렇게 만들다 보면 서비스를 추가하는 것이 매우 쉽고 유연하다. 큰 시스템은 작은 모듈로 나눴으니 서로 간의 통신을 해야 한다. 여기서 중요한 것이 생겼는데, 작은 단위의 시스템 간에 어떻게 연결할지이다. 어떤 식으로 네트워크를 연결할지가 고민인데, 이를 토폴로지라고 한다. 토폴로지 방식은 버스형, 트리형, 링형, 성형, 망형이 있다. 이 중 망형은 각 노드를 1:1로 연결하는 구조로 안정적으로 많이 사용된다.

위치와 접속 가능 상태를 알아야 함. 상태 정보를 어떻게 알 수 있을까? 어딘가에 저장해두고 그 곳을 바라보면 된다. (비슷한 예로는 DNS 서버가 있다.)
그러면 이러한 것을 Distributor로 정의하고 서버 자신의 상태 저장해둔다. Distributor의 통신은 TCP가 유리하다. 그리고 Distributor가 종료되는 상황도 고려해야 한다.
그리고 시스템 간 인터페이스 어떤 규약으로 연결할 것인가? 대표적인 REST API 이다.
시스템 간의 연결이 매우 중요하다. 그런데, 복잡하게 하지 않았다. 가장 단순하고 보편적이고 쉬운 REST 방식의 통신이다.
결국 중요한 것은 시스템을 사용하는 흐름과 그 흐름 중에 안 될 수 있는 시스템에 대한 방어 로직을 준비하여 Seamless하게 전체 시스템이 잘 동작되게 하는 것이다.

중소기업, 대기업, 외국계, 기술사 시험까지
합격한

면접 노하우

1. 면접 보기 전에 알아둬야 할 점

면접은 회사에 입사하기 위해 필수적으로 거치는 채용 과정이다. 즉, 회사가 사원을 채용하고자 할 때 꼭 필요하다고 생각하는 과정이라는 것이다. 채용 과정 중 가장 중요한 과정이라고 할 수 있다. 물론 면접자는 본인의 관점에서 최선을 다해 준비하겠지만, 면접을 통과하기 위해서는 면접관과 회사의 입장에서 객관적으로 자신을 바라볼 필요가 있다. 이에, 필자의 면접관으로서의 경험을 바탕으로 면접자들이 면접 시 어떤 점들을 준비하면 좋은지 등에 대해 이야기 해보고자 한다.

이력서만 봐도 면접자의 정보를 충분히 알 수 있지만, 굳이 면접을 보는 이유는 직접 보고 들으면서 면접자에 대해 더 구체적으로 파악하고 싶어서이다. 그러다 보니 인상이 매우 중요하게 작용한다. 사실 첫인상만으로도 면접의 당락이 좌우될 수 있고, 그 정도까지는 아니라고 하더라도 면접자의 첫인상을 보고 내린 자신의 판단을 검증하는 방향으로 면접 질문을 하게 될 수도 있다.

첫인상이라는 것은 사람의 모든 감각을 통해서 얻을 수 있는 직관이다. 이 직관은 어디서 오는 것일까? 사람은 사람을 보면 오감을 넘어서는 육감을 통해서 느껴지는 종합적인 판단이 생긴다. 실제 필자도 면접관으로 수많은 면접자들을 봤는데, 많은 경우 면접자가 면접장에 입장하는 순간 마음속에서 당락이 결정되기도 했다. 그 결정이 대부분 첫인상이었다.

그렇다면 누군가 이렇게 물어볼 수 있다. '원래 고약하게 생겨 먹은 것을 어떻게 합니까?' 생김새와 인상은 다르다. 생김새는 윤곽을 말하고, 인상은 마음을 반영해서 얼굴에 나타나는 것을 말한다. 즉 인상은 마음의 자세에 따라서 달라질 수 있는 것이며, 훈련을 통해서도 충분히 바꿀 수 있는 것이라는 이야기이다.

좋은 인상을 갖기 위해서는 어떤 마음을 품어야 하는가?

첫째는 '자신감'이다. 자신감은 실력이 근거가 되었을 때 가장 멋지게 뿜어져 나온다. '실력이 충분하다면 자신감은 당연히 가지고 있는 것 아닌가' 라고 생각할 수 있지만, 의외로 실력이 있음에도 자신감이 없는 경우도 많다. 그리고 실력이라는 것 자체가 상대적이기 때문에 생각하기에 따라서는 면접관들 앞에서 자신을 한없이 작게 여길 수도 있다. 근거 없는 자신감은 곤란하겠지만, 작은 일이라도 자신의 경험이나 성공 스토리에 충분히 자신감을 가지고 있는 것이 좋다.

둘째로 '웃는 인상'이다. 사실 면접장에서 박장대소 할 수도 없고, 히죽 히죽

웃을 수도 없는 일이다. 그런 경직된 분위기에서 웃는 인상은 어떻게 만들어질 수 있을까? 웃음을 머금고 있을 수 있다. 웃음을 머금었을 때 사람의 인상은 굉장히 밝고 좋아 보일 수가 있다. 그리고 웃음을 머금게 되면 본인에게도 여유가 생기게 된다. 웃는 얼굴에 모질게 질문하기는 어렵지 않겠는가.

셋째로 '말의 억양'이다. 부드럽고, 신뢰성이 들게끔 설득력 있는 답변들을 해나가야 한다. 면접관은 질문을 던지고 면접자의 답변을 들으며 지식을 충분히 갖췄는지, 태도는 어떠한지, 열정은 있는지 등을 보고 싶을 것이다. 이러한 태도를 어떻게 볼 수 있을까? 어렵거나 곤란한 질문을 하여 답변하는 것을 들어볼 수 있다. 이런 의도로 질문을 하는 경우 답변이 얼마나 정확한지에 대한 것도 중요하지만, 어떤 식으로 말을 하느냐도 매우 중요하다. 면접관은 답변을 들으며 이 면접자랑 같이 일하는 상황을 상상하게 되는데, 함께 일하면서 잘 맞을지, 골치 아플 일은 없을지, 어떤 일을 맡기면 잘할 수 있을지 등을 고민한다. 면접관의 이런 고민을 생각해본다면, 답변할 때 말의 억양과 태도도 매우 중요하다고 할 수 있다.

2. 기술면접 면접과정 설명

기술면접은 면접 중에서도 가장 중요도를 높게 두고 진행하는 과정이다. 인성이나 임원 면접이 당락을 크게 좌우하지 않고, 결격자를 구별하는 정도로 진행하는 것과 다르게, 기술 면접은 IT회사 입사 여부를 결정하는데 비중이 높고, 중요한 과정으로 여겨진다. 기술면접은 크게 신입과 경력으로 나눠서 생각해볼 수 있다. 경력 입사라고 하면 뽑고자 하는 포지션에 맞는 기술력이 있는지가 중점이다. 신입의 경우는 기본 지식, 가능성, 역량을 주로 본다.

외국계 IT 회사의 경우 1:1 면접을 3-4차례 하는 경우가 많고, 1시간 내외로 기술에 대한 깊이 있는 질문들을 주고 받는다. 외국계의 경우 지금 당장 필요한 역할을 해줄 수 있는 사람인지에 포커스를 두고 본다면, 국내 기업은 이 면접자가 회사와 적합한지를 주로 본다.

국내 기업의 기술 면접은 대개 3~4명 정도의 면접관들이 들어오는 경우가 많다. 면접관으로는 CTO(Chief Technical Officer: 최고 기술 리더) 나 개발 팀장급이 들어올 때도 있고 기술 실무자가 들어올 때도 있다. 면접자들의 기술 소양 및 기술 깊이를 측정하기 위해서는 잘 아는 사람이 배석해야 하기 때문에, 기술 면접에서는 실제로 기술을 가지고 현장에서 일하는 사람들이 참석하

는 경우가 많다.
가끔 면접 중반부터 면접자들이 압박 면접이라고 표현할 정도로 약간의 스트레스가 될 수 있을 정도의 심도 있는 질문을 할 때도 있다. 면접자의 지식 또는 기술의 깊이를 알고자 함이다. 마지막 질문까지 모두 정확하게 맞춰야 면접에 통과하는 것은 아니다. 면접관들 스스로가 생각하기에도 무리하다 싶을 정도로 물어볼 때도 있으니 너무 긴장하지 말고, 태도에 주의하며 차분하게 아는 선에서 답변하면 된다.
면접은 대부분 짧은 시간에 파악이 된다. 실리콘 밸리에서는 4초 만에 면접자의 당락이 좌우된다는 말이 있을 정도이다. 그런데 기술면접의 경우는 좀 다를 수 있다. 외모나 태도가 결정을 해버리는 것이 아니라 질문에 얼마나 내공 있는 답변을 하느냐가 중요하기 때문이다.

국내 SW서비스기업은 크게 IT서비스기업(네이버, 카카오, 라인 등)과 SI대기업(삼성SDS, LG CNS 등)로 나눠 생각해볼 수 있다.
IT서비스기업의 경우는 얼마나 개발을 잘하는지를 주로 본다. 지원자의 소규모 프로젝트나 포트폴리오에 조금 더 관심을 가지는 회사들이다. 개발에 얼마나 관심이 많은지, 재능이 있는지, 탄탄한 기본기가 있는지를 보는데, 개 중에서도 기본기를 상당히 깊이 있게 본다. 이미 코딩 테스트를 봤음에도 면접 시 손코딩을 시키거나 추가적으로 알고리즘 질문을 하는 식으로 지원자의 능력을 재차 확인한다. 함께 일할 때 전반적인 코드의 품질이 망가지지 않아야 하기 때문이기도 하고, 대규모 서비스를 운영할 때 잘못된 코드가 반영되어 성능이 떨어지거나 장애가 나기를 원치 않기 때문이기도 하다. 또한 개발자의 창의성과 개인의 역량을 중시한다. 기존 사용하던 개발 툴을 개선하는 등의, 좀 더 좋은 아이디어를 내고 개선점을 받아들이고 유연하게 대처하는 능력을 기대한다. 그렇기에 당연히 지원자가 스스로의 의지로 무언가를 만들어봤거나, 개선을 해본 경험이 있는지에 대해 관심을 가질 수밖에 없다.
또한 IT서비스기업은 개발환경들을 조금 더 빠르게 적용하고 있음을 이해할 필요가 있다. git은 물론이고, 자신이 지원한 분야의 최신 트렌드를 공부해두는 것이 좋다. 백엔드 직군을 예로 들면 DevOps 형태의 개발 조직, 쿠버네티스 인프라 환경 SPA 어플리케이션 구도, MSA 서비스 아키텍쳐 등에 대해서도 공부해두는 것이 도움이 될 것이다.

SI대기업은 지원자가 표준을 준수할 수 있는지, 얼마나 성실한지를 주로 본다. IT서비스기업들과 달리, 지원자의 소규모 프로젝트 진행 경험을 볼 때 얼

마나 좋은 아이디어인지, 기술에 대한 이해가 있는지와 같은 것보다는 함께 참여한 사람들과 커뮤니케이션에 문제는 없었는지, 문제가 발생했을 때 어떻게 풀어갔는지, 본인의 맡은 책임을 얼마나 끈기 있게 잘 했는지 등에 중점을 두는 경우가 많다. SI대기업에서는 대규모 프로젝트, 중소 프로젝트 등 프로젝트가 기준이 되는 경우가 많기 때문에 지원자가 해당 프로젝트 내에서 얼마나 잘 적응할지, 규칙에 맞게 행동할지, 개발 가이드에 맞게 개발을 할지 등에 더 관심을 가질 수밖에 없다.

모든 기업에서 성실함을 요구하겠지만, 그 요구하는 강도나 결이 다를 수 있다. SI대기업은 성실함을 요구하는 대신 창의성과 같은 능력에 큰 비중을 두지는 않는다. 튀는 행동보다는 준수한 행동을, 개성보다는 팀웍에 도움이 되는 사람을 선호한다. 특히 신입 사원의 경우는 성실하고 커뮤니케이션만 잘되면 괜찮다고 생각한다. 가능성이 있는 본인들이 잘 가르치고 키우면 된다고 생각하기 때문이다. 각 회사의 업의 내용을 생각해보면 왜 그러한 기준을 가지고 있는지, 어떤 인재상이 필요한지에 대해 이해할 수 있다. 면접 시에는 회사 측에서 원하는 인재상의 모습으로 재포장할 필요가 있다. 운동할 때는 운동복을 입고, 격식을 갖춰야 할 때는 옷을 말끔하게 차려입듯이, 동일한 사람이더라도 상황에 맞는 포장과 표현은 필요하다.

개발자로 면접을 준비하는 가장 좋은 방법은 직접 주제를 선정하고 프로젝트를 진행해보면서, 그 내용을 포트폴리오로 만들며 실제 개발 역량을 쌓는 것이다. 면접 시 그렇게 쌓은 역량을 내비친다면 면접관들은 지원자의 역량을 보다 넓은 방향으로 가늠해볼 수 있기도 하고, 직접 경험해서 얻은 지식이라는 그 자체로 높은 점수를 줄 수도 있다.

입사를 한 번에 성공하면 좋겠지만, 그렇지 않은 경우도 생길 수 있기 때문에 면접을 보고 나면 반드시 면접 내용을 복기해서 본인이 받았던 질문을 생각해보고, 자신이 어떻게 대답했엇는지 정리해보는 것이 좋다. 그리고 다음에 동일하거나 유사한 질문을 받았을 때 어떻게 답하는 것이 더 좋을지 생각해보는 것이 필요하다. 또한 우대 조건에 대해 물어보는 취업준비생들이 많은데, 우대 조건은 말 그대로 우대 조건이지 필수는 아니라는 것을 기억해라. 필수 사항을 잘하는 것이 훨씬 낫다.

만일 프로젝트를 해본 적이 있다면, 그리고 그 프로젝트 경험을 이력서에 기록할 것이라면 그 프로젝트의 내용만큼은 반드시 완벽하게 알고 있어야 한다. 특히 다른 사람과 함께 진행했던 프로젝트라면, 자기가 맡은 부분뿐만 아니라 다른 사람이 맡았던 부분까지 다 기억해두거나, 필요 시 함께 했던 사람에게

물어서라도 전체 내용을 정확하게 알고 있는 것이 좋다. 질문이 나왔을 때 “그 부분은 제가 맡았던 것이 아니어서 모르겠습니다” 라고 대답하는 것보다는 대략적으로라도 해당 부분을 먼저 설명하고 적당한 타이밍에 “그런데 제가 주로 맡아서 진행했던 부분은 이 부분이 아니라...” 등의 멘트를 통해 자신이 자신 있게 설명할 수 있는 영역으로 자연스럽게 면접관의 관심을 돌리는 것이 훨씬 좋다.

또한 프로젝트의 내용을 자세히 말하는 것도 좋지만, 진행 과정을 말하는 것도 도움이 된다. 프로젝트를 왜 했는지, 그것을 하면서 어떤 것을 배웠는지, 하면서 무엇을 느꼈는지, 동료들과 어떤 일이 있었고 그 일을 어떻게 해결했는지, 어려움을 어떻게 극복했는지, 동료들과 일하면서 부딪히고 배운 것은 무엇인지 등의 과정을 이야기해보자. 면접관들은 문제가 발생했을 때 면접자가 문제를 어떻게 극복해나가는지, 그 과정을 통해 어느 만큼 성장할 수 있을지, 위기 상황에 어떻게 대처할지와 같은 부분에 대해서도 궁금해하기 때문이다. 위의 이야기를 풀면서 자신의 문제해결능력을 강조하는 것도 좋다. 사실 회사에서는 면접자가 문제를 만나면 피해가거나 도망가는 사람인지, 지혜롭게 문제를 풀어나가는 사람인지를 판단하는 것이 매우 중요하다. 왜냐하면 회사에서는 여기저기서 문제가 발생하는 것은 사실 일상이고, 이러한 문제들을 직면했을 때 지혜롭게 문제를 해결해나갈 사람이 필요하기 때문이다.
면접자의 답변을 듣다 보면 사람의 수준이 간접적으로 드러난다. 사용하는 단어나 문장만 봐도 그 사람의 배경 지식이 어느 정도인지 가늠할 수 있다. 그렇기 때문에 답변을 조리있게 하는 것도 중요하다. 또한, 면접 시 답변은 두괄식으로 하는 게 좋다. 결론을 먼저 이야기하라는 말이다. 친구나 연인 간에는 자초지종을 전부 이야기하는 것이 더 흥미진진하고 서로의 교감에 도움이 될지 몰라도, 회사는 업무 중심의 조직이기 때문에 늘 결론이 가장 중요하다. 우선 결론을 말하고, 부연 설명 정도만 한 다음 면접관이 더 듣기를 원하면 더 자세히 말하는 것이 좋다.

3. 면접 TIP

면접관으로서 실제로 면접을 진행하면서 느꼈던 점을 모았다. 면접자들이 알고 있으면 좋을 만한 유용한 팁들이다.

(1) 자기 소개를 해보라는 것은?

면접관도 면접을 시작할 때 딱딱한 분위기를 조금은 부드럽게 하고 싶은 마음이 든다. 그리고 어떤 사람인지 정확하게 파악하지 못한 상태에서 첫 질문을 선택하는 것도 상당히 어렵다. 그럴 때 가장 무난한 질문이 자기소개이다. 자기소개를 들으면서 서둘러 이력서의 내용을 좀 더 확인할 수도 있다.

당연히, 면접관은 자기소개를 통해 면접자를 조금 더 알고 싶어한다. 하지만 의외로 자기소개를 통해서 얻을 수 있는 정보는 많지 않다. 면접자의 답변내용이 천편일률적이거나 추상적이거나 좋은 말로만 이루어져 있을 때가 훨씬 많기 때문이다. 그래서 자기소개를 질문해놓고서는 이력서만 들여다보는 경우도 많다.

그렇다면 면접자는 어떻게 준비를 해야 할까. 짧은 시간에 자기를 소개한다는 것은 쉽지 않은 일이다. 그렇기 때문에 자기에 대해서 돌아보고 생각하는 시간을 미리 가지는 것이 좋다.겨우 1-2분의 자기소개지만, 나는 누구인지, 무엇이 나를 가장 잘 표현하는 것인지를 충분히 고민해볼 필요가 있다. 단순하게 그럴싸한 자기소개 템플릿을 준비하는 게 아니라, 진짜로 자신에 대해서 돌아보고 자신의 정체성을 찾는다면 면접의 자기소개에서 정말 자신이 어떤 사람인지 녹여낼 수 있을 뿐만 아니라 본질적으로 인생이나 커리어의 방향을 잡는데도 도움이 될 것이다.

자기소개에 많은 내용을 넣기는 과한데, 그렇다고 짧게 하면 썰렁해질 수가 있기 때문에 어떻게 준비할지에 대해 고민하는 친구들이 많다. 언제 태어났고, 어디가 고향이고, 몇 남 몇 녀 중 몇 째이고.. 등등의 가족관계를 이야기하지 말자. 요즘에는 그런 식으로 대답하는 면접자가 많이 줄기는 했지만, 어쨌든 면접관은 그런 부분에 관심 자체가 없다. 자신에 대해 가장 알리고 싶은 내용, 자신을 부각시킬 수 있는 내용만 말해도 부족하다.

그렇다면 무엇을 말하면 좋을까? 자기소개부터 너무 격양돼서 말하는 것은 듣는 사람도 부담스럽다. 자신을 잘 표현하면서 자신에 대해 더 알고 싶게 만드는, 뭔가 자기만의 스토리의 예고편 같은 느낌이 가장 좋다.

(2) 압박 질문을 받았을 때는 태도와 대응이 중요하다.

보통 면접 때 처음에는 편하고 쉬운 질문을 할 가능성이 높은데, 어느 정도 시간이 흐르면 기술 지식이나 경험에 대한 깊이를 측정해보기 위해서 점점 어

려운 질문을 하게 된다. 면접자가 자신의 지식과 경험을 토대로 아무리 잘 답변하더라도, 깊이가 더하여지면 결국 모르는 질문이 나오는 순간이 생긴다. 물론 면접자가 정말 기술적으로 부족할 때도 있겠지만, 사실 가끔 면접관은 면접자의 태도를 보고자 일부러 면접자가 모르는 것이 나올 때까지 깊게 파고들어 물어보기도 한다. 압박처럼 여겨질 정도로 과도한 질문은 지원자의 기술 지식의 깊이 측정과 더불어 성향 파악을 하고자 하는 의도를 가지고 있다. 스트레스 상황에서 어떻게 반응하는지, 향후 일을 하게 될 때 팀웍에 좋지 않은 영향을 미칠 부분은 없는지, 고객을 상대로 하여 곤란한 상황 속에서도 평정심을 유지할 수 있는지 등도 가늠할 수 있기 때문이다. 그렇기 때문에 이때 면접자의 반응이나 태도, 모르는 것이 나왔을 때 대응하는 자세까지 전부 보게 된다. 지나치게 격양된 표현도 좋지 않지만, 쉽게 모른다고 인정하고 포기하는 태도도 좋지 않다. 최대한 내가 아는 선에서 이야기하려고 노력하는 것이 좋되 정말 모르겠을 때는 그런 부분에 대해서 내가 향후 어떻게 보완할 수 있는지 등에 대해서 이야기하는 것이 좋다.

이때 유의해야 할 점은, 감정적인 대응은 좋지 않다는 것이다. 자신의 의견을 강조하는 것은 좋지만, 질문자의 말을 부정하거나 자신의 말이 맞다는 식으로 이야기 하는 것은 좋은 느낌을 주지 못한다. 또한 간혹 IT 시스템을 구축하거나 개발해서 제품화하는 과정을 물어보면서 끊임없이 부정적인 질문을 할 때가 있는데, 이때는 최대한 끝까지 기술적인 답변을 제시하고자 하는 것이 좋다. 면접관은 곤란한 문제를 기술적으로 풀어나가는 과정에 대해서 보고 싶어 그런 질문을 했을 수 있다.

(3) 면접은 만남이다.

면접의 자리는 서로 간의 파악을 위한 만남의 자리이다. 결코 편안한 자리가 될 수는 없겠지만, 그렇다고 지나치게 긴장해서 경직되는 것은 좋지 않다. 긴장을 하게 되면 머리가 굳으면서 머리 속이 하얘지기 때문에 아는 것도 말하지 못하는 상황이 발생할 수 있기 때문이다.

면접은 정신없이 질문과 답변만 오가다가 끝나게 된다. 이때 질문에 대해 정확도 100%의 답변을 내는 것에만 급급하지 말고, 본인이 해당 직무의 적임자이고 일을 잘 할 수 있는 사람이라는 이미지를 심고자 노력하는 것이 중요하다. 면접관은 결국 이 대화 과정을 통해 면접자가 함께 일하기 괜찮은 사람인지에 대해 알아내고자 하는 것이기 때문에, 상대방과의 교감과 커뮤니케이션이

매우 중요하다.

그렇다면 인터뷰 하는 내내 면접관은 무슨 생각을 하고 있을까? 대개 면접을 준비하는 사람은 본인의 입장만 생각하기에도 급급해 면접관 입장은 생각해보지 않는다. 사실 면접관도 직장인일 뿐이다. 어쩌면 오늘 아침에도 지각하지 않기 위해 분주하게 준비해 회사에 도착한, 평범한 직장인 중 한 명일 수 있다. 다만 지금 회사의 지시에 의해, 혹은 함께 일할 좋고 괜찮을 사람을 뽑기 위해 면접관으로 앉아있는 것일 뿐일 수도 있다.

결국 면접은 만남이다. 남녀가 처음 만났을 때 서로 대화를 하며 상대방을 파악하고 좋은 점을 찾아보고 함께 시간을 보내도 좋은 사람일지 판단하듯, 면접도 마찬가지이다. 이 면접자가 매력적인 사람인지 성실한 사람인지 회사에 적합한 사람인지를 보는 것이다. 끊임없이 의심도 해보고 확인도 해보면서 혹시 이력서의 화려한 수식어구 속에 사실이 아닌 것이 있는지, 준비된 것처럼 보이는 꾸며진 답변들 속에서 이 사람이 진짜 어떤 사람인지를 알아내려고 할 것이다. 그래서 필자는 사실, 진정성 있게 사실을 근거로 해서 어떻게 살아왔고 어떻게 일할 것인지만 제대로 보여주면 된다고 생각한다. 정답을 맞춰야 한다는 강박적인 마음을 가지지 말고, 편안하게 나를 표현하는 것이 가장 좋다.

(4) 면접은 이력서 작성부터 시작이다.

면접 질문은 주로 면접자가 작성한 이력서의 내용으로부터 나온다. 이력서에 작성한 모든 내용은 질문의 대상이 될 수 있다는 것이다. 그렇기에 사실 이력서부터 잘 써야 기술면접을 잘 볼 수 있다. 여기서 이력서를 잘 써야 한다는 건 이력서를 자신이 잘 이야기할 수 있는 내용들로 구성해야 함을 의미한다. 어쩌다 한번 해본 기술이 요즘 떴다고 해서 그 키워드를 넣었다가 그것에 대한 질문이 집중해서 나온다면 낭패를 볼 수 있다.

이력서를 화려하게 작성하는 것도 좋지만, 자신을 잘 어필하는 방향을 유지하는 선에서 면접을 위해 전략적으로 접근하는 것이 더 좋다. 무조건 많은 것을 해봤다고 좋은 것은 아니다. 개발 관련 기술은 뭐 하나를 하더라도 제대로 하는 것을 더 중시한다. 왜냐면 면접자가 회사에 취업을 하고 나면 특정 업무를 하기 때문이다. 무엇이든 하나만 걸려라 식으로 쓸 수도 있겠지만, 그런 식으로 하게 되면 오히려 다 해봤는지 의심을 사게 될 수도 있다. 기간 대비 다 해봤다는 식의 이력서는 오히려 곤란한 질문을 받기 좋다.

면접관들은 끊임없이 의심한다. 이력에 작성된 내용이 사실인지 그리고 실제

해봤는지. 신입채용 시에도 마찬가지다. 학창 시절 팀 단위 프로젝트를 했다고 해도, 그것을 진행하면서 어떤 역할을 했는지, 무엇을 배웠는지 궁금해 한다. 이 면접자가 적극적이였는지, 팀웍을 해치지는 않았는지, 배움의 자세가 되어 있는지, 계속 성장할 수 있는지는 작은 일에서도 충분히 나타날 수 있기 때문이다.
과제의 주제가 화려하고 멋졌다고 해도 그 과제 제목의 화려함에 속지 않는다. 면접을 잘 보기 위해서 이력서를 잘 작성해야 하고, 이력서를 잘 작성하기 위해서는 그 과제를 수행할 때부터 성실하고 적극적으로 임해야 한다는 것이다.
이미 지난 일이고, 과제를 정리해야 하는 상황이라면 과제 주제를 적되 모든 것을 적지 말고, 본인이 한 업적 위주로 적어 질문의 폭을 줄이는 것도 방법이다.

(5) 기술면접을 잘 보기 위해서는 기술적인 용어를 적절하게 사용하여야 한다.

기술 면접을 잘 보기 위해서는 평소 내공이 많이 쌓여 있어야 한다. 그 사람이 얼마만큼 많은 개발을 해봤고, 기술 공부를 했고, 기본기가 있는지를 보고자 하는데, 이 역량은 결코 하루아침에 쌓을 수 있는 것들이 아니다.
그럼에도 불구하고 면접이라는 제한된 시간 동안 좋게 보일 수 있는 방법은 있는데, 사소한 말을 하더라도 기술용어를 적절하게 섞어서 표현하는 것이다. 이를 위해 면접 준비 기간에는 IT 관련 도서라든가 전자 신문 기존 컴퓨터 관련 서적을 많이 보면서 되도록 머리를 컴퓨터 관련 내용으로 채워두는 것이 좋다.

(6) 면접자는 본인이 어떻게 면접관의 눈에 비칠까를 끊임없이 생각해야 한다.

면접자들은 면접을 준비하면서부터, 심지어 면접장에서까지, 질문에 어떻게 답변할지에 대한 것들에만 집중한다. 시야가 굉장히 좁아져 있다. 이 면접이라는 상황을 객관적으로 봤을 때 가장 중요한 것은 면접자가 면접관에게 어떻게 비춰지느냐 이다. 질문을 답변하고 있는 자신의 모습이 중요한 것이지, 질문을 맞추느냐 못 맞추느냐는 두 번째 문제이다. 어떤 질문은 신입 수준 이상의 너

무 어려운 지식에 관련된 것이었을 수도 있고, 경험해보지 않은 영역과 관련된 것이었을 수도 있다. 그것을 면접관이 모르고 있을 것이라고 생각하지 말자. 면접은 필기 시험과는 성격이 다르다. 필기시험이라면 몇 개를 틀렸는지가 매우 중요겠지만, 면접에서는 아니다. 10개의 질문 중 몇 개를 틀렸느냐보다 더 중요한 것은 지금 자신이 면접관의 눈에 어떤 사람으로 보이느냐이다. 가능성 있는 사람인지, 자신감 있는 사람인지, 준비되어 있는 사람인지, 노력하는 사람인지 등, 자신이 어떤 사람인지 보여주는 것이 더 중요하다.

(7) 자기소개에서 인용 문구는 겉치레로 들린다.

자기소개를 할 때 간혹 멋지다고 생각하는 인용 문구를 사용하는 경우가 있는데, 아주 적절한 인용 문구를 골라 자신을 잘 표현하는 친구들도 있다. 하지만 이는 드문 경우에 속하고, 대부분의 경우 그 인용구가 매우 멋지다면 인용구만 기억에 남고, 멋지지 않다면 불필요한 수식 어구가 되어버린다. 자칫하다가 어색하게 연출되면 분위기 자체가 어색해지는 상황이 되어버리고, 암기식으로 보이면 연기하는 것처럼 보일 수도 있다. 면접관에게 진정성 없어 보이는 모습으로 비출 수 있기 때문에, 차라리 본인이 생각하는 가치관이나 생각을 표현한 문구를 만들어서 말하는 게 낫다고 생각한다. 인터넷에서 복사한 글 같은 것을 읽는 것으로 자기소개시간의 일부를 사용하기는 주어진 시간이 너무 아깝다.

(8) 면접장에서 겸손은 자신감 있는 당당한 겸손이다.

인생에서 겸손한 태도는 중요하다. 신입이든 경력이든 겸손한 사람을 싫어할 사람은 없을 것이다. 그렇다면 지식이나 경험적인 측면에서 겸손한 모습을 보이려면 어떻게 해야 할까? 간혹 '아직 부족한 점이 많다' 등과 같이 '부족하다' 라는 표현을 쓰면서 겸손을 부리는 사람들이 있다. 물론 대인 관계에서는, 자기가 무조건 잘났다고 하는 것보다 훨씬 나은 느낌을 주는 표현이기는 하다. 하지만 면접장에서는 굳이 사용할 필요 없는 단어이기도 하다.

신입이 부족한 것은 당연하다. 그 점은 면접관들도 이미 알고 있기 때문에 본인이 굳이 강조할 필요가 없다. 예를 들면, "어떤 기술을 잘하나요?" 라는 질문에 지나치게 겸손을 떨 필요는 없는 것이다. 실력이나 경험을 말해보라고 했을 때는 자신있고 당당한 태도로 내가 제일 잘하는 사람처럼 말하는 게 차라

리 더 낫다. 이렇게 자신감을 어필할 때 당당하게 보이고 싶고, 자신감 있게 보이고 싶다면 그 때만큼은 잠시 면접관의 눈을 응시하며 이야기해도 좋다. 물론 너무 덤비듯이 노려보면 곤란하겠지만 말이다.

(9) 자기 수준에 맞는 답변이 좋다.

자신이 아무리 자신감이 넘치는 사람이더라도 자신의 수준에 맞게 대답하는 것이 좋다. 자기가 신입인데, 마치 모든 것을 다 아는 양 말하다가는 그 자리의 면접관들에게 더 어려운 질문을 받을 수 있다. 자신을 객관적으로 판단하고 그 수준에 맞춰서 말하는 것이 좋다. 그리고 학생 신분에서 진행한 프로젝트라는 것을 면접관들도 어느 정도 감안하고 질문하는데, 그것이 다른 사람들과의 경험과는 다른 엄청난 일인 것처럼 얘기한다면 궁금증을 더 유발시키게 되기 때문에 사실에 입각한 객관적인 답변을 하는 것이 좋다. 자신감을 가지고 말하는 것과 사실을 부풀려 말하는 것은 다르다는 것을 기억해야 한다.

(10) 답변은 너무 짧지 않게 하는 게 좋다.

답변을 너무 짧게 한다면 해당 질문에 대해서 정확하게 모르거나 충분히 모른다는 이미지를 줄 수 있다. 따라서 기본적으로 답을 말하되 부연 설명도 해주는 것이 좋다. 그것이 이해도 잘되고, 듣는 사람도 충분히 납득할 수가 있기 때문이다. 이 때 단답형보다는 구체적인 기술적 경험을 포함하는 답변을 하는 것이 좋다.

대규모 회사의 경우는 질문의 범위나 질문지가 있을 수 있는데, 없는 경우도 많다. 그러한 경우, 이력서의 내용을 근거로 질문을 하게 된다. 이 때 면접관들은 다음 질문을 해야 하는 일종의 압박에 시달리게 된다. 또는 문제를 생각했는데 옆에 있는 면접관들의 눈치를 볼 수도 있다. '내 질문의 수준이 적절한가', '너무 낮은 수준은 아닌가' 등 말이다. 이런 것은 면접을 진행해본 사람만이 아는 고충이고, 느낌일 것이다. 이러한 때에 충분한 길이로 답변을 하면서 여유를 주는 것이 면접관들을 궁지에 몰지 않는 답변이다. 실제 필자도 수차례 면접에 들어갔는데, 어떤 경우는 답변자가 너무 빠르게 단답형으로 답변하여 다음 질문을 생각하기도 전이라 당황했던 적이 적지 않게 있다.

(11) 전혀 모르는 질문이 나왔을 때

면접에서 전혀 모르는 질문이 나오더라도 바로 '모르겠습니다.'라고 답변하기 보다는 최대한 생각하고 신중하게 답변을 해보려고 하는 게 중요하다. 만약 해당 질문에 대한 답은 모르겠지만 유사한 기술이 생각난다면 'OO 기술로 답변을 해도 될까요'라고 정중하게 묻고, 답변을 해보려는 시도도 괜찮다.
질문에 대해서 완전히 모를 때 한번 정도 모르겠다고 답변하는 것은 면접 전체의 당락을 크게 좌우하지 않는다. 질문마다 모른다고 했다면 그냥 부족한 사람이라 생각할 수 있지만, 전반적인 질문에 대해 답변을 성실하게 잘하려고 하다가 한두 개의 질문에 대해서 모른다면 그럴 수도 있다고 넘길 수가 있다. 면접관도, '경험이라는 것은 한계가 있으니 다른 경험한 것들을 잘 알면되지 않을까' 라고 생각해서 다음 질문으로 넘어간다.
그러므로 답변하지 못했다고해서 면접자의 마음이 요동칠 필요가 없다. 면접 질문에 답변하지 못한 것은 필기시험에서 정답을 틀리는 것과 다르다. 완전하게 모르는데 억지스럽게 답을 하거나, 답변하지 못했다고 당황하지 말고, 조금 다른 기술을 말하거나 면접관이 다음 질문을 할 수 있게 예의 있게 넘어가는 것도 괜찮은 방법이다.
물론 스스로의 답변 내용에 자신이 없다고 계속 넘어가서는 안되겠지만, 한정된 시간 속에서 다른 질문을 받아서 잘 대답할 수 있도록 시간 배분을 해보자는 것이다. 다시금 강조하지만 답변을 하려고 노력하는 자세는 유지해야 한다. 전체 면접 질문 중 한번 정도만 '죄송합니다. 질문 하신 내용에 대해서는 경험하진 못했습니다.' 정도의 답변을 하는 게 적절하다.

(12) 질문은 유도하는 것도 방법이다.

이력서는 면접관들이 질문을 뽑아내는 질문지이기도 하다. 이력서에서 내가 자신있고, 잘하고, 뽐내고 싶은 것을 노출하는 것이 방법이다. 그리고 답변을 하는 중에도 내가 좀 더 자신 있는 쪽의 얘기들을 끝부분에 붙여서 다음에 그것과 관련된 질문을 나오도록 하는 것이 좋다.
물론 이러한 답변 방법은 난이도가 높다고 느껴질 수 있다. 일단 면접관이 묻는 질문에 성실하게 답변을 하고, 끝부분에 자신의 경험이라든가 포토폴리오에 기재된 것과 같은 내용을 말하면서 궁금증을 유발할 수 있고 '제가 OO 수행 경험을 말씀 드려도 될까요' 라는 식으로 역질문을 해서 내 경험에 대해 말할 수 있는 기회를 얻을 수도 있다. 이렇게 해서 자신이 꼭 어필하고 싶은게 있다면 그 얘기를 할 시간을 만드는게 좋다.

(13) 마지막으로 하고 싶은 말이 있는가라는 질문에 적당한 답은

마지막 하고 싶은 말이 있는가라는 질문은 매우 형식적인 질문이다. 이러한 형식적인 질문은 사실 면접관이 면접을 마무리 하고 싶을 때 가장 많이 쓰는 질문 형태의 멘트이다. 일방적으로 '면접 끝입니다.' 라고 말하면 기회를 안준 것처럼 비춰질 수도 있고, 또 혹시 못 들은 얘기가 있을 수도 있다고 생각하기 때문이다. 꼭 하고 싶은 말이 있다면 이때 하는게 좋은데, 사실은 이때는 면접관도 면접을 마치고자 하는 마음이다. 그래서 너무 장황한 얘기를 시작하는 것은 좋지는 않다. 면접자 본인의 이미지 메이킹을 할 수 있는 찬스로 봐야 한다. 하고 싶은 말 없냐고 했을 때 사실 면접자들도 대부분 질문도 없고, 별 생각이 없을 수가 있다. 이대로 조용히 끝냈으면 하는 마음도 있다.
하지만 이때 조금 여유를 가지고 질문을 시도하는게 좋다. 너무 어려운 질문보다는 가벼우면서도 내가 이 회사에 관심이 많다 라는 느낌을 줄 수 있게 말이다. 입사 하게 되면 어떤 일을 하게 되는지 또는 최근 회사가 주력하고 있는 사업이 무엇인지 면접관들이 깊이 생각하지 않아도 편하게 답변할 수 있는 질문을 한다. 너무 민감한 질문은 피하는게 좋다. 그리고 2개 이상 하는 것은 좋지 않다. 면접을 마치려고 했던 상황이니 말이다. 또는 본인을 마지막으로 어필하는 것도 좋다. '꼭 일하고 싶다.' 와 같은 형식적인 멘트보다는 조금 더 현실적이고 와닿는 말이 좋다. 와닿으려면 문장에 논리가 맞아야 한다. '꼭 일하고 싶다.' 라는 말은 호소 밖에 되지 않는다. '열심히 하겠습니다.' 어떻게라는 부분도 없고, 설득력도 없다.
마지막 멘트는 미리 준비를 하는게 좋다. 면접의 마지막이라는 상황은 무조건 오기 때문이다. 한 문장정도를 준비한다. 그런데, 그것을 미리 준비해서 외운 것처럼 하지 말고, 그 자리에서 생각난 것처럼 평소 마음에 담고 살아온 것처럼 해야 한다. 본인들의 강점이 있을 것이다. 강점과 회사에서 필요로 하는 포지션의 니즈가 있을 것이다. 나의 꿈을 실현하겠다가 아니라 회사에 구체적으로 어떻게 기여하겠다라는 느낌이 좋다.

(14) 꿈이나 비전에 대해서 묻는다면

꿈이나 비전에 대해서 묻는 경우가 종종 있다. 면접에서 꿈을 묻는 이유가 무엇일지 생각해볼 필요가 있다. 면접관이 꿈을 묻는 이유는, 그 사람의 꿈 자체가 궁금하거나 미래가 궁금해서라기보다는 이 사람이 추구하는 방향성이 무엇

인지 궁금하기 때문이다. 이를 통해 지원자가 회사와 맞는 사람인지 엿볼 수 있다. 꿈이 아예 없는 것보다는 있는 것이 좋지만 꿈이 건물주라고 하면 설령 그게 사실이더라도 면접관에게 달가운 답변은 아니다. 회사 입사하는 사람이라면 어떤 분야의 전문가가 되고 싶다 정도가 무난하지 않을까 싶다.

그런데, 꿈을 종종 목표와 혼동해서 표현하는 사람들이 있다. 이를 듣는 사람도 혼란스럽다. 필자의 실제 사례인데, 회사에 3번이나 떨어진 면접자에게 꿈이 뭐냐고 물으니 지금 이 회사에 입사하는게 본인의 꿈이라고 말을 했다. 그래서 조금은 혹독하지만 '그렇다면 면접자는 회사에 입사하면 꿈은 없어지는지' 물었다. 면접자는 매우 당황해했고, 말을 잇지 못했다.

입사는 좋은 목표이지만, 꿈이라고 하기에는 여러 가지로 적절하지는 않은 것 같다. 물론 그 면접자 입장은 이해가 된다. 매우 되고 싶고, 달성하고 싶기에 꿈이라고 표현했겠지만, 오히려 달성이 가능해 버리면 '꿈' 보다는 '목표'라고 말하는 게 적절해보인다. 그래서 '꿈이 선생님이 되는 것' 이라고 하기 보다는 '어떠한 선생님으로 사는 것'이라고 하는게 더 바람직하다는 생각이 든다. 한편 거창하게 바로 사장이 되겠다거나 5년 뒤에 이 회사의 팀장이 되고싶다고 말한다면 도전적이고 진취적으로 들릴 수는 있지만, 도발적이고 허무 맹랑하게 들릴 수도 있다.

'꿈'에 대해 답변을 할 때는 노력하는 바가 점점 이뤄지는 모습을 근거를 통해 자연스럽게 표현하는게 좋다. 우선 '꿈'에 해당하는 것을 얘기하고 마지막에 이를 위해서 내가 오늘 무엇을 하고 있고, 무엇을 준비하고 있는지를 구체적으로 말하는 것이다. 나의 꿈이 단순한 백일몽이 아니라 내가 실제로 이뤄나가고 있는 과정임을 강조해줄 수 있다.

실제로 면접관에게 어필되는 쪽의 방향과 답변들이 있다. 그 지점을 잘 공략한다면 불리한 질문이 무조건 불리하지 않을 수도 있다. 고등학교까지만 나왔더라도 요즘은 블라이드 채용이라서 얼마든지 학점은행제등을 통해서 학사를 취득하고, 동일한 조건으로 입사 지원을 할 수가 있다. 비록 그 상황이 불리하더라도 그 속에서 '한시도 노력을 게을리 한 적이 없고, 끊임없이 배워야 한다고 생각했다.' 등으로 어필할 수가 있다.

면접 전 준비할 것

기술 면접 전에 본인의 이력서 내용 중 면접 질문이 될 만한 것들을 도출해보면 좋다. 앞에서도 언급했지만, 본인이 작성하여 제출한 이력서가 면접 질문의 범위가 된다. 따라서 이력서에 기재된 기술적인 키워드는 모두 면접 질문이 될 가능성이 있다. 각 용어 별로 그 내용을 정리할 필요가 있다.

이 과정이 첫 번째이고, 가장 중요하고 사실상 전부이다. 이를 어느 정도 준비했다면 이력서에 적은 기술적인 용어들로 무장할 필요가 있다. 예를 들면 '조금 느리게 동작 될 수 있다.'와 같은 표현도 면접에서 '성능 감소 현상이 발생될 수 있다.' 라고 표현할 수 있도록 트레이닝해야한다. 별 차이가 없어 보일 수 있지만, 상대방이 듣기에 더욱 기술적인 사람으로 보이는 게 유리하다. 이런 식으로 자신의 이력서를 정리하면서 대부분의 표현을 좀 더 기술적인 표현으로 바꿔볼 수가 있다.

그리고 지식, 특히 업무적으로 사용해야 할 지식이라면 한두 개는 자신이 확실하게 아는 게 중요하다. 예를 들어서 해당 전형이 안드로이드 개발자를 채용하는 전형이라면 안드로이드, 자바 개발자를 채용하는 전형이라면 자바를 공부하는 식으로 본인이 지원한 분야의 기술에 대해서 공부를 깊게 할 필요가 있다. 결국 해당 직무로 배치될 가능성이 높고, 그 일을 하게 될 가능성이 많기 때문에 면접에서도 '이 지원자가 그 일을 잘할 수 있는지' 보기 때문이다.

(1) 면접 2주일 전부터

자기가 했던 프로젝트나 포트폴리오 또는 이력에 대해서 전체적으로 자료를 취합해볼 필요가 있다. 여기저기 흩어진 것을 컴퓨터의 한 폴더에 모으고, 데이터를 정리해본다. 이렇게 하면서 자기가 과거 어떤 일들을 했는지 살펴볼 필요가 있다. 전체 자료를 다 모으면 자신이 사용할 데이터와 사용하지 않을 데이터를 구별하는 작업을 한다. 이력서는 이미 제출된 상태일 수 있지만, 자신이 좀 더 집중해서 다시 볼 데이터와 버릴 데이터를 구별하고 되도록 하나의 파일에 또는 하나의 자료로 만들어본다. 그렇게 여러 차례 모으는 과정과 취합 그리고 필터 하는 과정을 거치게 되면 나의 과거에 했던 프로젝트나 포트폴리오가 조금씩 떠오를 것이다. 그 과정에서 잘 기억이 나지 않는 것도 있을텐데, 그런 것들을 잘 메모해둔다. 그 내용들을 다시 찾아보거나 기록해두고 집중적으로 공부한다. 질문은 자신이 했던 일에서 가장 많이 나올 가능성이 높기 때

문이다.

(2) 면접 1주일 전부터

면접 2주일 전부터 취합 및 필터링을 통해 정리해왔다면 한눈에 보기 좋게 정리된 자료가 있을 것이다. 이것을 여러 번 보면서 자신의 머릿 속에 이미지로 넣고, 말하기 연습을 한다. 이때부터는 말로 자료의 내용을 표현해야 한다. 모르는 키워드를 만나더라도 그것을 설명하려고 노력해야 한다. 그 후에 정답을 찾아보는 면접 모드로 생활해야 한다. 막상 면접장에서는 아는 내용도 잊을 수가 있기 때문에 면접장에서의 답변을 몸에 배이게 하는게 좋다. 계속 말하는 연습이다. 모르는 내용도 찾아서 내 것으로 만드는 노력을 계속한다. 최대한 이해하려고 하고, 그것을 내 것으로 만들려고 하고, 컴퓨터 용어이나 인터넷에서 찾은 설명이 아니라 내가 머리로 이해하고 다시 정리한 말로 표현할 수 있어야 한다.

(3) 면접 하루 전

이제부터는 더 이상 모르는 용어를 찾지 말고, 잘 정리되어진 자료를 여러 번 반복하여 본다. 그 후에 자료를 보지 말고 자문자답을 해본다. 끊임없이 질문하고 답변을 하는 것을 반복하여 단순하게 암기 한 것이 아니라 몸이 반응할 정도로 자연스럽게 답을 할 수 있게 연습한다. 집 근처에서 돌아다니면서 또는 외부에서 연습을 해도 좋다. 그렇게 해서 최대한 집중력을 높이도록 해본다. 마지막 저녁 시간은 최대한 차분하게 자기 시간을 갖도록 하고, 모르는 게 갑자기 생각났다고 뭔가에 쫓겨서 더 알아야 한다는 마음보다는 편안한 시간을 보내는 게 좋다. 자극적인 드라마나 영화 시청보다는 명상정도가 좋다. 최대한 편안하게 잠을 잘 수 있도록 커피는 되도록 삼가는 게 좋다. 적게 마셔도 긴장이 되면 잠이 더 오지 않기 때문이다. 그렇게 최대한 숙면을 취하는 게 좋고, 영상 시청은 자신이 만든 이미지를 없애기 때문에 삼가는 게 좋다.

(4) 면접 당일

면접 당일 시간을 정확하게 확인하고, 장소까지 정확하게 확인한다. 이왕이면

조금 이르게 출발해서 늦지 않도록 하고, 늦지 않는 수준이 아니라 조금 여유 있게 도착하여 휴게 공간을 찾아보고 없다고 하면 면접장 근처에 가까운 커피숍이나 앉을 만한 곳을 찾아서 쉬면서 여유있게 기다리는게 좋다. 스마트폰을 보기 보다는 명상에 가까운 시간을 가지고, 그렇다고 자신의 자료를 다시 보지 말고, 머리 속으로만 생각하면서 준비하는게 좋다.

5) 면접 직전

심호흡을 하며 머리 속을 비우고 마음도 최대한 편하게 먹는다. 이제는 오히려 내려놓고, 잡고 있던 머리 속의 지식들도 내려놓는다. 왜냐면 자기가 지금 외우고 있고, 말하려고 준비한 것이 아니라 전혀 엉뚱한 질문을 받을 수도 있기 때문이다. 내가 뭔가를 계속 외우고 있으면 예상과 다른 질문이 나왔을 때 당황하면서 말의 스텝이 꼬일 수 있다. 편안하게 '면접은 만남이다.'라고 생각하고, '이 분들과 함께 일하게 될 가능성이 높다.' 등 조금은 편안한 생각을 하며 내가 이 분들 눈에 어떻게 비칠까를 생각해본다. 면접 질문에 대한 답변은 최대한 정확하게 하려고 하지만 혹시 못 맞춘다고 해도 자신이 한 일에 대해서는 당당하고 자신감있게 답변하는 것이 좋다. 직접적으로 '앞으로의 일도 잘 할 수 있다고 생각하고, 꼭 일을 해보고 싶으니 기회를 달라'라는 말을 할 수는 없더라도 그러한 마음을 품고 면접에 임하는 것이 좋다. 면접관도 지원자의 생각을 느낄 수 있기 때문이다.

(6) 면접 후 꼭 해야 할 일

면접 후에는 최대한 예의를 갖추어 퇴장하되 너무 주눅들거나 수축되어있지 말고, 밝은 미소와 꼭 다시 뵙고 싶다는 마음을 전달하며 인사를 하고 나가는 것이 좋다. 그리고 여기서부터 중요한 것이 있는데, 면접장을 나오는 순간 면접이 끝난 게 아니라 대개 면접장은 회사의 내부에 있을 가능성이 높다. 면접관들도 휴식을 취하기 때문에 화장실이나 휴게 공간 또는 복도를 지나갈 수가 있다. 이 때 지원자가 다른 사람과 통화를 하면서 면접을 마쳤다는 얘기를 하거나 본인 혼자서 고개를 떨구고 가거나 혼잣말 등을 할 때 들을 수도 있기 때문에 그 건물을 나올 때까지 최대한 예의 있는 모습과 자신감이 있는 모습을 갖추는 것이 좋다. 최대한 그 건물의 근처에서까지 말이다.

마지막으로 가장 중요한 것은 귀가 하였을 때 오늘의 면접 질문과 나의 답

변을 생각나는대로 메모해두고, 아차 싶었던 말이나 답변을 제대로 못했던 질문들에 대해서는 정확한 내용을 찾아서 정리하는 식으로 복기하는 것이 좋다. 오늘의 면접이 마지막이면 좋지만, 시작일 수도 있기 때문이다. 면접 후 복기는 본인을 업그레이드할 수 있는 방법이다. 계속 이런 식으로 나만의 면접 히스토리 관리를 하는 것이 좋다.