핵심 질문, 답변 모음

- [ ] **왜 코딩을 하고 싶은지?** #지적호기심 #성장욕구
  - 도전하고 성장하는 삶을 살 수 있다는 것에 매력을 느꼈기 때문입니다.
  - (개발을 시작하기 이전에는 증권사를 준비했었는데요, 그 이유도 빠르게 변화하는 시장 환경을 분석하고, 경쟁하며 성장할 수 있는 삶이 기대가 됐기 때문입니다.)
  - 증권 취업 준비를 하면서 크롤링 기술이 필요해서 우연히 코딩을 접했는데요, 컴퓨터 원리를 알아가는 과정이 재밌게 느껴졌습니다. 이를 계기로, IT 분야에 관심을 가지면서 진로를 알아보던 중, IT 분야도 매번 성장을 요구한다는 점에서 저의 성향과 잘 맞는 분야라는 확신이 들어 개발자로서의 길을 선택했습니다.
  - 그 이후로 코딩을 시작했고, 매일매일 작게 나마 학습하고 성장하는 제 모습을 돌아볼때마다 뿌듯함을 느꼈고, 코딩을 계속함으로서 이 감정을 느끼면서 살고 싶습니다.
  - 지금처럼 더 계속 성장해서, 좋은 서비스를 만들고 세상의 불편함을 해소할 수 있고 싶습니다.
- [ ] **코딩 공부하기 전에 뭐했는지?**
  - 구직 활동을 하면서도, 매일 아침 글로벌 및 국내 주요 이슈를 분석하고 실제 시장에서는 어떻게 반응하는지를 매일 모니터링했습니다.
  - 또한, 금융학회에서 기업 분석 레포트를 작성한 경험을 바탕으로, 공백기 동안에도 꾸준히 글로벌 매크로 흐름을 파악하고 특정 기업에 대한 리포트를 작성하며 시장을 바라보는 시각을 넓혀갔습니다.
  - 이후에는, 전문성을 갖추고 취업 스펙트럼을 넓혀보고자, CPA 등의 자격증에도 잠깐 도전해보았습니다.
  - 하지만,학습을 진행하면서 변화가 적고 정형화된 작업을 한다는 느낌을 많이 받았습니다. 저는 그런 업무보다는, 유연한 사고로 새로운 문제를 접근하는 삶을 살고 싶었기 때문에 저랑 맞지 않는다고 생각해 CPA를 포기하게 되었습니다.
- [ ] **당신이 노력한 점은? (공백기 제외)**
  - 학부 시절, 금융 분야에 대한 관심을 바탕으로 다양한 활동과 학습을 통해 실무 역량을 키우고자 노력해왔습니다.
  - 대표적으로, 증권사 대학생 투자대회 입상도 했었고, CFA, 투운사 같은 금융 자격증을 취득하여 재무 및 투자 이론에 대한 기초를 다졌습니다.
  - 또한 교내 금융학회 활동도 적극적으로 참여했습니다.
  - 기업분석 프로젝트들을 진행하면서 팀원들과 함께 기업을 정말 세밀하게 분석하고 논리적 근거를 바탕으로 valuation을 도출하여 투자 판단을 내리는 경험을 했습니다.
  - 그 외에도 학회에서 주도하는 독서 및 채권 스터디에도 자발적으로 참여하며, 지식의 범위를 확장해나갔습니다.
- [ ] **협업하고 싶은 팀원은 어떤 사람인지**
  - [ ]
- [ ] **왜 크래프톤에 지원했는지?** #기초체력 #커넥션풀 #CPU #기본기중심훈련
  - 크래프톤의 CS, 알고리즘 등의 기본기 중심의 훈련 환경이 지금 저에게 필요하기 때문이었습니다.
  - 지금까지 다양한 기술과 개념들을 배우는 과정이 정말 재밌었지만, 한편으로는 기초가 탄탄하지 않아 어려운 기술을 배울 때 속도가 느리다는 느낌을 받았습니다.
  - 특히, 최근 팀 프로젝트에서 Spring batch를 활용해 백업, 복구 로직을 구현하면서 그 한계를 더욱 명확히 느꼈습니다. 예를 들어, CPU 코어수를 신경쓰지 않고 무작정 병렬 처리를 함으로써 오버헤드 문제를 일으켰던 점, 커넥션 풀 관리를 하지 않아 대용량 처리 시 커넥션 풀이 마르는 오류가 발생했던 점입니다. 만약 CS 지식이 풍부했다면 **설계와 구현 단계에서 효율적으로 시간을**쓰지 않았을까 싶습니다.

- 이제는, 기술을 배우는 것에 집중하기 보다는 개발자에게 필요한 기초 체력을 다지고 싶습니다. #기초체력
- 이를 통해 앞으로 마주할 다양한 문제를 유연하고 쉽게 해결하고, 새로운 기술들을 빠르게 습득하는 개발자가 되고 싶습니다. #다양한문제-유연하게해결 #새로운기술-빠르게습득

- [ ] **어떤 개발자가 되고 싶은지?**
  - 탄탄한 기본기를 바탕으로 기술 트렌드를 빠르게 흡수해 문제를 다각도로 해결할 수 있는 개발자가 되고 싶습니다. #탄탄한_기본기 #기술트렌드_흡수 #다각도_해결
  - 빠르게 변화하는 개발 환경 속에서 새로운 기술을 배우고 적용하는 것은 필수적입니다.
  - 하지만 단순히 트렌드만 좇는 기본기가 부족한 개발자가 되기를 원하지는 않습니다.
  - 기술은 끊임없이 변화하지만, **기초 역량과 개념에 대한 깊은 이해는 변하지 않는 자산**이라고 믿기 때문입니다.
  - 저는 이러한 **기본기**를 바탕으로 문제를 다양한 방법으로 접근하며, 상황에 적합한 솔루션을 도출할 수 있는 개발자가 되고자 합니다.
    #더좋은기술 #혁신 #더좋은서비스
- [ ] **앞서 못다한 이야기가 있다면? ⭐**
  - 아직 별거 아닌 실력이지만, 개발 공부를 시작한 이후로부터는 정말 치열하게 살아온 것 같습니다.
  - 늦게 시작한 만큼, 다른 사람들을 따라가기 위해 더 노력해야 한다는 생각만으로 코딩공부에 몰두하고 살았습니다.
  - 이러한 삶을 살고 있는 저에게 크래프톤의 커리큘럼은 정말 최적의 환경이라 생각합니다.
  - 지금도 저는 정글에서 배우게 될 낯설지만 근본적인 개념들을 배울 수 있다는 큰 기대감을 갖고 있습니다. 쉽지 않은 과정이겠지만, 개발자가 되기 위한 필수 퀘스트를 깬다는 마음으로 도전하고, 마주하게 될 높은 벽들을 하나씩 넘어가며 성장해나가고 싶습니다.
- [ ] **만약 크래프톤 정글에서 떨어지면 이후 행보는?**
  - 우선은, 현재 부트캠프가 끝난 뒤 지난 내용들을 되돌아보는 시간을 가질 것 같습니다. 짧은 시간 안에 다양한 기술을 배우느라 쫓아가는데만 집중했던 것 같습니다. 따라서, 7개월 동안 배웠던 내용들과 그 과정에서 있었던 CS개념들을 정리해볼 것입니다.
  - 동시에, 하루에 3~4시간씩 알고리즘 학습에 집중하여 코딩 테스트를 준비할 것 같습니다.
  - 또한, 개인 프로젝트가 부족해서, **개인 프로젝트를 기획하고, 하루에 조금씩이라도 개발을 진행**하며 실전 감각을 유지할 것입니다.
  - 마지막으로, **부트캠프에서 지원해주는 인턴십 프로그램이나 채용형 교육 과정에 지원**하여 실무 경험을 쌓을 수 있는 기회를 적극적으로 모색할 것입니다.
  - 그리고 이번에 저저번주에 정보처리기사 필기를 합격했는데요, 실기 시험을 준비할 것 같습니다.
- [ ] **크래프톤에서 맘에 드는 커리큘럼은?**
  - Pintos 과정 (간단한 OS Framework)
    - OS의 핵심 요소를 직접 구현하는 과정 ➡ OS 이해도 높이고 싶다.
    - 커널 스레드, 파일 시스템 등을
    - 프로그램의 원리 이해
- [ ] **스스로 문제를 해결했던 경험은???**
  - **Spring batch에서 직렬화 문제 얘기**
    - Spring batch에서 Step끼리 자원을 공유하는 최적의 방법을 발견해, 성능을 높였던 적이 있습니다.
    - (최근 팀 프로젝트에서 대용량의 기사 데이터를 주기적으로 S3백업하고 DB에 저장하고 복구하는 배치 작업을 담당했었습니다.)
    - Spring batch를 활용하다보면 Step끼리 자원을 공유하는 일들이 많은데요,

- 기존에는 Step끼리 자원을 공유할 때 JobContext를 활용하는 방식을 채택했었는데요. 완성하고 나니, DB 작업 및 직렬화 비용으로 성능이 좋게 나오기 힘들거라 생각했습니다.
- 대안을 알아보던 중, 기술블로그에서 자원을 공유하는 3가지 패턴에 관한 글을 읽고 제 코드를 상황에 맞게 리팩토링 했었습니다.
- 배치 재시작 시 정합성을 해치지 않고, 동시성을 제어할 수 있는 범위내의 데이터는 싱글톤 패턴의 컨테이너를 만들어 적재함으로써 직렬/역직렬화에 드는 CPU 사용량을 줄이고 DB I/O 작업을 최소화 했습니다. 또한, 그렇지 않은 데이터는 promotionLister라는 것을 활용해 승격화의 원리를 이용해서, Job과의 결합도를 낮추고 안정성을 높일 수 있었습니다.

☰ Spring batch 질문

1. Job
   - **여러 Step을 묶어서 실행**하는 배치 작업의 최상위 단위
   - **전체 배치의 흐름을 관리**
2. Step
   - 배치 작업에서 **하나의 독립된 실행 단위**
   - 하나의 Step이 실제 처리 로직을 실행한다.

> *쉽게 말하자면 Job은 '무엇을 처리할지'를 정의, Step은 '어떻게 처리할지' 를 담당*

3. Context
   - Spring batch는 실행 중 발생하는 데이터나 상태값을 공유하기 위해 ExecutionContext를 제공합니다.
   - 2가지 종류가 있으며, 자동으로 생성되는 메타 데이터뿐만 아니라 개발자가 직접 데이터를 적재할 수도 있습니다.

☐ *코드 품질과 개발 속도가 충돌했을 때, 어떻게 우선순위를 결정해야할까요?*

- 프로젝트의 성격과 상황에 따라 다르다고 생각합니다.
- 처음에는 높은 코드 품질로 완성도 높은 코드를 작성하는 것이 좋은 개발자인 줄 알았습니다. 하지만, 실제 개발 환경에서는 **마감 기한 준수**와 **피드백 속도** 또한 매우 중요한 요소라는 것을 알게 되었습니다.
- 따라서 현재는 **문제를 최대한 회피**하는 범위 내에서 **우선 기능을 구현**하고, 이후 **점진적으로 품질을 개선**해나가는 방식이 더 현실적이고 효과적이라고 판단하고 있습니다.
- 물론 **서비스의 특성에 따라 접근 방식은 달라져야** 합니다. 예를 들어, 금융 서비스처럼 안정성이 절대적인 경우에는 속도보다 품질과 안정성을 우선시하는 것이 맞습니다.
- 결국 중요한 것은, 서비스에 맞게 상황에 맞게 우선순위를 유연하게 정하는 것이 중요하다고 생각합니다.

☐ **왜 본인이 적합할 거라 생각하는지?** #자기주도성 #몰입 #강한학습의지

- 험난한 정글에서 살아남을 수 있을거라고 생각하는 2가지 이유가 있습니다.
- 저는 첫번째 장점은 스스로 학습하는데 익숙하다는 것입니다. 부트캠프 이전에도 스스로 학습 계획을 세워 자바, 스프링, JPA 를 독학하였습니다. 단순히 누군가 먼저 가르쳐주기를 기다리기보다는 지금 나에게 어떤 개념과 기술이 필요한지 스스로 판단하고, 이에 맞는 자료를 선별해 학습하는 방식이 몸에 익어 있습니다. 이런 자기주도적 학습 태도는 문제 상황을 스스로 정의하고, 탐구하고, 해결 방법을 찾아내는 힘이 필요한 크래프톤 정글의 운영 방식과 적합할 것입니다.
- 두번째로, 강한 학습의지(를 바탕으로 구준히 학습을 지속해왔습니다). 개발 공부를 시작한 이후부터는 6개월 넘게 하루도 빠짐없이 학습을 이어오고 있습니다. 이는, 단순히 남들보다 늦게 시작해서 시간을 많이

투자하려 했던 것만은 아닙니다. 새로운 지식을 배우는 과정 자체에서 흥미와 성취감을 느꼈기에, 자연스럽게 학습이 일상으로 자리 잡았습니다.
  - 정글에서 마주하게 될 과제들은 결코 쉽지 않겠지만, 앞서 말씀드린 스스로 학습하는 습관과 강한 학습의지가 있다면, 난관들도 헤쳐나갈 수 있으리라 믿습니다.
- [ ] **공부는 어떻게 하시나요?**
  - 단순 개념인지 프레임워크 등의 넓은 개념인지 등 학습 범위에 따라 다릅니다.
  - 프레임워크의 넓은 범위를 공부할 때는 모두 공부할 수 없기 때문에, 핵심을 모아 놓은 강의나 영상을 보면서 전체적인 개념을 파악하려고 합니다. 또한 학습 도중 중간중간 내부 구현체를 파헤쳐 분석하거나, 레퍼런스, 유명 회사의 기술블로그 등을 참고하면서 공부합니다.
- [ ] **Mock 과잉 사용이 리팩터링에 취약하다는 문제를 발견했다고 했습니다. 실제로 어떻게 책임을 분리해 테스트 구조를 개선했나요?**
  - 초기에는 서비스 레이어에 대부분의 비즈니스 로직을 구현하면서, 해당 레이어가 많은 책임을 지고 있다는 사실을 인지하지 못했습니다. 하지만 테스트 코드를 작성하면서 Mock 객체에 과도하게 의존하고 있는 현상을 발견하였고, 그 결과 프로덕션 코드 리팩토링 시 테스트 코드가 깨지는 구조적인 문제를 겪게 되었습니다.
  - 이러한 문제를 해결하기 위해 저는 서비스 레이어의 책임을 분리하는 방향으로 리팩토링을 진행했습니다. 단순히 도메인 클래스에 비즈니스 로직을 넣는 것뿐만 아니라 POJO와 일급 컬렉션 등을 활용하여 개선하였습니다. 이렇게 분리된 객체들은 자체적으로 로직을 처리하게 되어, 테스트 시에도 외부 의존 없이 순수 단위 테스트가 가능해졌습니다.
  - 서비스 레이어는 흐름 제어와 외부 인터페이스와의 연동에만 집중할 수 있게 되었고, 전체 시스템의 유연성과 테스트 안정성이 크게 향상되었습니다(으며). 이를 통해 책임 분리의 중요성을 체감할 수 있었습니다.
- [ ] PATCH vs PUT
  - PATCH : 자원의 일부를 수정하고, 멱등성이 없을수도 있다.
  - PUT : 전체 자원을 교체. 멱등성 O
- [ ] RESTA API 설명
- [ ] 이벤트 버블링 사용 이유 `$(documnet).on("click ~~~ ")`
  - 이벤트 버블링 사용 시
    - 요소가 동적으로 생성됐더라도 이벤트가 버블링되어 document 객체까지 올라오면서 감지가된다.
    - event는 하위 -> 상위 요소로 전파된다
  - Direct Binding 방식
    - 현재 DOM에 있는 요소만 해당된다.
    - 즉, 동적으로 추가된 요소에는 작동하지 않는 단점
- [ ] e.preventDefault() 쓴 이유
  - 브라우저의 기본 동작을 차단하기 위해 (링크 이동, 폼 제출 등)
  - 필요없는 경우도 있을 수 있지만, 후에 제가 아직 잘 모르는 프론트 코드에서 의도치 않은 동작이 발생할 것을 대비해 전부 적용시켜놨습니다.
- [ ] Bootstrap쓰지 않고 Bulma를 쓴 이유
  - 학습자료에서는 2가지 css 프레임워크를 제공했었는데요, 헷갈리지 않게 하나만 사용하는게 좋다고 생각했다.
  - 아이콘도 그렇고, 미니멀한 디자인이 좋아보여서 Bulma를 쓰게 되었습니다.
- [ ] .replace(/\s+/g, " "); 의미는?
  - \s+ : 하나 이상의 공백

- /g : 전체 문자열에 적용

- [ ] 버튼 디자인 방식

```HTML
<div class="buttons is-mobile is-vcentered is-justify-content-center px-2 is-flex-wrap-nowrap">
```

```CSS
.buttons.is-flex-wrap-nowrap .button {
  flex: 1;
  min-width: 0;
}
```

- [ ] 어려웠던 점
  - 익숙하지 않은 언어들로 하다보니 배우는게 두려웠다.
  - 다른 프로젝트와 시험과 일정이 겹쳐서 공부할 여유 시간이 얼마 안됐다.
  - 헷갈렸던 점은
    - CSS
    - 필드값 or Key에 접근할 때 접근법이 헷갈렸다. 메서드로 접근해야할지 속성접근법으로 해야할지
  - 프론트의 경우 하나의 파일에 저장하다보니 코드가 길어져서 헷갈렸다.
- [ ] IF문만 쓰고 else 안 쓴 이유
  - 가독성 향상 : 중첩이 늘어나면 가독성이 떨어진다. 비정상 흐름을 빠르게 분기 처리하고, 정상흐름을 쭉 읽히도록