

# 어쩌다 트렌센던스까지

## 트렌센던스 플래닝 가이드

eunhkim  
iwoo  
sanam  
jujeong  
yohlee

42서울을 왜 선택했나요? 지난 1년은 어땠나요?  
다른 교육과정과 병행하지 않고 42 공통 서클  
풀 커밋한 걸 후회하지는 않나요? 1년간 따  
진행한 사이드 프로젝트나 공통 서클 외 활동  
있다면? 42에서 무엇이 가장 좋았나요? 마지  
공통 서클 과제인 트렌센던스까지 끝낸 소감은  
어떤가요? 트렌센던스는 구체적으로 어떻  
진행했나요? 무엇이 가장 좋았고 힘들었나요?  
특별히 기억에 남는 에러가 있나요? 역할  
어떻게 나뉘었나요? 5명 협업이 힘들지  
않았나요? 만나서 대면으로 진행했나요? 온라인  
협업은 어떻게 했나요? 만약 트렌센던스  
처음부터 한다면 무엇이 달라질까요? 공유해  
레퍼런스들이 있나요? 공통 과정 이후의 계획  
어떻게 되나요? 이제 막 카뎃이 된 사람들에게

## 어쩌다 트랜센던스까지

5명이 모여 정확히 103일이 걸렸어요. 1기 1차였고, 시작 당시 한국에는 패스한 사람도 없었기에 모든 것이 막막했어요. '이게 진짜 끝이 나기는 할까?' 싶었던 프로젝트였는데, 계속해서 시간과 에너지를 쏟으며 울고 또 웃다 보니 어쩌다 트랜센던스까지 통과를 하게 되었네요. libft를 시작할 때는 멀게만 느껴지던 공통 서클도 어쩌다 끝나버렸고요.

나누고 싶은 이야기들이 있습니다. 저희가 어쩌다 42에 오게 되었는지, 어쩌다 다른 교육 과정과 병행하기보다 42에 집중하기를 선택했는지, 기술 스택들이 대부분 처음이었던 트랜센던스를 어쩌다 통과하게 되었는지, 이런 이야기들이 다른 카뎃들에게 어떤 형태로든 도움이 될 거라고 여겼습니다.

트랜센던스를 시작하기 전에 알았으면 좋았을 것들, 공통 과정을 진행하면서 내내 고민했던 것들, 카뎃이 처음 되었을 때 읽고 싶었던 바로 그런 글을 쓰려고 했습니다. 기술적인 부분에서는 부족한 부분이 많지만, 오히려 그래서 42스럽다고 생각합니다. 동료학습의 좋은 레퍼런스가 되기를 바랍니다.

## 목차

<b>1장</b>	기술학습   스택넘, 처음 뵙겠습니다	4
<b>2장</b>	닌자 프로젝트   가자, 코드를 망치러	5
<b>3장</b>	핵심가치 결정   사람이 아니라 가치가 결정할 수 있도록	7
<b>4장</b>	서브젝트 분석   개발자는 예술가가 아니다	9
<b>5장</b>	책임 설계   클래스보다 중요한 것	11
<b>6장</b>	API 설계   모두를 위한 약속	13
<b>7장</b>	테이블 설계   멀리서만 보았던 ERD	15
<b>8장</b>	UI 디자인   같은 그림을 그린다는 것	17
<b>9장</b>	개발 환경 구성   어떻게 개발할 것인가	22
<b>10장</b>	협업 환경 구성   요즘 것들의 협업	24
<b>11장</b>	구현, 테스트, PR   최초보다 최선을	33
<b>12장</b>	굿 프랙티스   결과적으로 옳았던 선택들	39
<b>부록 1</b>	서비스 스크린샷   서비스 이미지 둘러보기	41
<b>부록 2</b>	에러 리스트   어쩌면 당신도 겪게 될	47
<b>부록 3</b>	학습 레퍼런스   커피값은 받아야 할 것 같은	51
<b>부록 4-1</b>	팀원 iwoo 인터뷰   42를 받아들인다는 것	64
<b>부록 4-2</b>	팀원 sanam 인터뷰   트랜센던스로 발견한 포지션	71
<b>부록 4-3</b>	팀원 jujeong 인터뷰   전공자가 바라본 이너서클	77
<b>부록 4-4</b>	팀원 yohlee 인터뷰   이곳에서 살아남는 길	82
<b>부록 4-5</b>	팀장 eunhkim 인터뷰   우리의 선택에 대한 신뢰	86

## 1장 기술학습 | 스택님, 처음 뵙겠습니다 / Day 1 ~ 45

트렌센던스는 크게 PSQL, 루비, 루비 온 레일즈, HTML, CSS, 자바스크립트(+jquery), Backbone(+underscore), Docker의 여덟 가지로 기술 스택이 구성되어 있습니다. 대부분 처음인 기술들이었고, 또 docker 역시 docker-compose를 이용하여 빌드하여야 하므로 학습할 영역이 추가로 있었습니다.

하면서 익히기엔 프로젝트의 규모가 컸습니다. 프로젝트 진행에 필요한 기술 스터디를 먼저 진행하였고, 100여 일의 전체 프로젝트 기간 중 **초기 45일**을 스터디(주 2회 온라인 정기 미팅)에 사용하였습니다. 학습 순서는 위의 스택 소개에서 기술한 순서와 같았습니다.

PSQL은 SQL부터 익혀야 했기에 **아사이 아츠시의 <SQL 첫걸음>**을 같이 읽었고, 레일즈는 **구름 IDE**의 콘텐츠를 학습한 후 야마다 요시히로의 **<퍼펙트 루비 온 레일즈>**와 레일즈 홈페이지의 **레일즈 공식 가이드** 중 각자 선택하여 심화 학습했습니다. 프론트 기초는 **코드카데미**에서, 자바스크립트는 웹페이지 **<모던 자바스크립트 튜토리얼>**에서 학습했습니다. 백본은 **애디 오스마니의 <Backbone.js 프로그래밍>**과 웹 리서치 중 각자 편한 쪽으로 학습했습니다.

## 2장 닌자 프로젝트 | 가자, 코드를 망치러 / Day 46 ~ 52

해당 기술 스택에 대한 학습과 프로젝트 경험을 가지고 시작하지 않았다면, 45일은 이 모든 기술을 소화하기에 턱없이 부족한 시간입니다. 저희는 그랬습니다. 가벼운 예제들은 따라서 해보면서 학습을 했지만, 해당 기술들을 이용하여 프로젝트를 한다고 생각하면 그냥 막막했습니다. 어디서부터 어떻게 해야 할지 모르겠다고 할까요.

API부터? UI부터? 아니면 테이블 스키마부터? 무엇을 먼저 해야 할지도 모르겠고, 무엇을 하더라도 어떻게 해야 할지 판단이 안 섰습니다. 또 팀원이 5명이고, 프로젝트 규모도 크기 때문에 모두가 함께 지켜야 할 규칙을 정리한 **코드 컨벤션**이 필요하다고 생각했습니다. 정작 뭘 써야 할지는 몰랐지만요.

모던 자바스크립트 튜토리얼을 읽을 때 **닌자 코드**에 관한 내용이 있었습니다. 닌자라고 불리던 전설 속 개발자들의 코드에 관한 얘기였죠. 코드는 가능한 한 짧게 쓰고, 변수명은 글자 하나만 쓰고, 약어를 사용하고, 명사는 포괄적으로, 철자는 유사하게, 동의어도 쓰고, 변수명도 재사용하고, 형용사는 과장하고, 외부 변수를 덮어쓰고, 함수에 다양한 기능을 넣는 코드들을 의미했습니다. 지향해야 할 멋진 코드인 것처럼 풍자했는데, 문득 그 얘기가 생각나서 그대로 결심했습니다. 좋은 규칙을 모르겠으면, 마음대로 **닌자 프로젝트**를 해보자. 코드를 한 번 망쳐보자. 그러다

보면 알게 되겠지.

닌자 프로젝트에는 **7일**의 시간을 소요했습니다. 프로그램의 구조나 코드 스타일에 대해 거의 토론하지 않았고, 대부분의 사항을 '닌자 프로젝트니까'라는 말로 작업자가 빠르게 결정했습니다. 정교한 브랜칭이나 리뷰도 없이 push & merge했고, 서브젝트의 요구사항만 신경 쓸 뿐 그 이외에는 정말로 아무것도 신경 쓰지 않는 코딩을 했습니다. 그렇게 했음에도 일주일의 시간 동안 회원 가입과 로그인, 유저 상세 페이지, 길드 목록 페이지, 게임 페이지 정도를 구현하는 데 그쳤습니다.

그러나 닌자 프로젝트라는 관점에서는 매우 성공적이었습니다. 팀원들은 backbone을 이용하여 웹페이지를 동적으로 제어할 줄 알게 되었고, 레일즈와 API를 이용하여 요청/응답을 주고받을 수 있게 되었습니다. 또 게임을 구현하면서 액션 케이블(레일즈에서 웹 소켓을 처리하는 방식)도 어느 정도 이해하게 되었습니다. 비로소 설계를 위한 최소한의 준비가 된 것이죠.

### 3장 핵심가치 결정 | 사람이 아니라 가치가 결정할 수 있도록 / Day 53

닌자 프로젝트도 끝났으니 이제 프로젝트를 할 수 있겠다. 무엇부터 해야 할까? 결론은 **핵심 가치**였습니다. UI, DB, API 설계부터 시작해서 구현, 그 과정에서의 협업 방식까지 당장 수많은 의사 결정을 해야 할 텐데 의사 결정의 기준이 먼저 제대로 있어야 한다고 판단했습니다.

핵심 가치가 공유되지 않으면 끔찍한 일들이 일어납니다. 우선 논의를 위해 시간은 시간대로 잡아먹습니다. 끝에 가면 개별적으로는 그럴듯하지만 전체적으로는 일관되지 않은 코드를 만나게 됩니다. 프로그램을 목적 달성을 위한 객체의 협력이란 관점에서 보면 정말 좋지 않은 일입니다. 그렇다고 딱히 팀원들이 더 감정적으로 행복해진 것도 아니죠.

중요한 것은 목적 달성을 위해 가치의 우선순위를 정하는 것입니다. 우리의 맥락에서 어떤 가치가 어떤 가치보다 얼마나 중요하게 다루어져야 하는가를 정하면 우선 의사결정이 쉬워지며, 일관성을 가지고, 목적에도 더 부합하게 됩니다. 의사결정을 할 때는 어떤 가치들이 충돌하고 있는지를 해석하고, 우선순위를 잘 달성하는 문제해결 방식이 무엇인가를 디자인하는 데 집중할 수 있습니다. 감정과 시간의 소모가 확연히 줄어듭니다. 관련한 커리어 경험을 가진 구성원들도 있어서, 이런 의견이 더 잘 공감대를 얻고 실행될 수 있었습니다.

어떤 기준들이 필요할까에 대해 먼저 논의한 뒤, 후보를 추려 1~5 점 척도로 구글 설문을 진행 후 통계를 내는 방식으로 정했습니다. 결정된 가치 우선순위는 다음과 같습니다.

## 결정된 가치 우선순위

1. 서브젝트(요구사항 충족)
2. 안정성(예외처리)
3. 구성원의 만족
4. 가독성
5. 일관성
6. 객체 지향성
7. 단순성
8. 유저 경험
9. 신속성(작업시간 단축)
10. 통제 가능성
11. 학습 가능성
12. 취업 적용 가능성

이렇듯 목록화된 가치는 우리가 무엇을 어떻게 만들어야 하는지, 의사결정이 필요한 상황에서 어떻게 해야 하는지를 직관적으로 알 수 있게 해줍니다. 저희 팀의 경우 흥미롭게도 취업, 학습, 시간 단축과 같은 개인적 가치보다 가독성, 일관성, 예외처리와 같이 코드의 품질에 대한 가치들이 선명하게 높았습니다.



## 4장 서브젝트 분석 | 개발자는 예술가가 아니다 / Day 53

개발자는 예술가가 아닙니다. 적어도 일이라는 맥락을 놓고 볼 때 개발자는 만들고 싶은 프로덕트와 기능을 만들 수 없습니다. 고객이 원하는 것을 만들어야 합니다. 비즈니스란 고객을 위해 존재하는 것이니까요. 그리고 **42에서는 서브젝트가 고객입니다.** 현장보다 나은 점이 한 가지 있다면, 번덕이 심하지 않고 자신이 무엇이 원하는지를 잘 아는 고객이란 점이겠죠.

그런 맥락에서 팀 가치 우선순위 결정에서 서브젝트가 1번으로 평가되지 않았다면 그게 더 이상한 일이었을 겁니다. 서브젝트를 대충 읽은 뒤 구현하고, 뒤늦게 '서브젝트에 이런 게 있었네'하고 수정하는 일은 피해야 합니다. 효율성이나 42서울에서 서브젝트가 가지는 무게 때문이 아니라, 클라이언트의 요구를 대충 훑어 보고 개발을 시작하는 태도가 가지는 부적절함 때문입니다.

프로젝트를 시작한 지 50일이 넘어서고, 본격적인 개발 프로세스에 착수하면서 서브젝트 분석을 심도 있게 진행했습니다. 여기에서 말하는 '심도 있음'의 의미는, 서브젝트에 존재하는 단어 하나에도 의문이 남지 않고, 다각적 해석이 가능한 모든 경우를 검토하고, 팀원의 이해에 충돌이 있으면 협의하여 오해 없는 하나의 구체적인 프로그램을 공유하는 것을 의미합니다.

서브젝트 자체는 카뎃이라면 누구나 열람할 수 있기 때문에 따로

언급하지 않겠습니다. 대신 서브젝트를 분석한다는 의미를 더 정확하게 공유하기 위해 실례를 몇 개 언급하겠습니다.

- '모든 예외는 적절하게 핸들링 되어야 한다'는 항목에서 '적절한 핸들링'이란 무엇을 의미할까? '에러'도 적절히 핸들링 된다면 괜찮은가? 에러와 예외는 어떻게 구분할까?
- '패스워드를 암호화한다'는 항목에서 암호화 정도는 어느 정도이며, 암호화되는 위치는 어디여야 할까?
- 유저나 멤버십의 포지션에 그레이드가 있다고 할 때, 각 포지션에 대해 허락되는 권한의 범위는 정확하게 어디까지인가? 동일한 레벨의 포지션에 대해 자신의 권한을 행사할 수 있는가?
- 현재 소속된 길드가 다른 길드와 전쟁을 치르고 있을 때, 진행 중인 전투가 발생한다면 전쟁 페이지는 실시간으로 새로고침 되어야 할까?

이처럼 서브젝트의 지문이 불충분하고, 해석이 모호한 영역들에 대해서는 가능한 한 엄격하게 판단했습니다. 엄격함이라는 관점에서 판단할 수 없는 문제들에 대해서는 가능한 사용자 경험을 중점에 놓고 판단했습니다.

객체지향적인 프로그램의 설계를 위해 조영호님의 <객체지향과 사실과 오해>를 같이 읽었습니다. 웹서버를 할 때는 클래스부터 설계했었는데, 책에서는 클래스가 중요한 것이 아니라고 말하더군요. 단일한 책임을 지는 애플리케이션을 먼저 가정하고, 연관성 높은 책임을 나눈 뒤 서로 다른 객체에 할당하면서 나아갈 때 **협력하는 객체들의 공동체**로서 프로그램을 잘 디자인할 수 있다는 얘기가 크게 와닿았습니다.

어쨌든 먼저 해야 할 것만 같았던 API 설계도, 사실 프론트 엔드와 백 엔드에서 협력을 위해 어떤 객체들이 어떤 책임을 질지가 명확해진 뒤에야 가능하다고 판단했습니다. 저희가 이해한 수준에서 API란 결국 적절한 책임을 할당받은 객체들이 협력을 위해 주고받는 메시지의 양식이었기 때문입니다.

### 2. 객체 정리

Backbone View 객체

- EntryView
- ErrorView
- ▼ AppView
  - ▼ MainView
    - Home버튼
    - UserIndexView
      - ProfileView
      - MatchHistoryListView
      - MatchHistoryView

핵심은 사용자 경험이 일어나는 프론트였습니다. 몇 개의 뷰, 모델, 컬렉션, 헬퍼가 필요한가. 또 그것들이 어떻게 관계 맺어야 하는가. 라우터의 책임은 어디까지인가. 메뉴를 눌렀을 때 들어가는 페이지와 해당 페이지에서 클릭하여 들어가는 페이지들의 관계는 어떻게 설정할 것인가. 한 페이지 안에서도 어떤 객체들이 어떻게 협력하도록 할 것인가와 같은 주제들을 고민하며 디자인했습니다.

아래와 같이 고객 여정을 따라가면서 서비스의 책임들을 목록화하는 작업도 시도하면서, 100여 개의 프론트 객체들을 생성하고 책임을 할당했습니다. 애자일과 안 맞는 것처럼 보이지만, নিজ 프로젝트를 통해 시행착오를 겪은 뒤였고 고객(서브젝트)의 니즈가 명확히 드러난 상태였기에 가능하다고 판단했습니다. 프로젝트 이후 검토했을 때 달라진 부분은 20% 이하였고, 그마저도 전체적인 구조 변경은 없었습니다. 좋은 선택이었던 셈이죠.

#### 홈 화면에서 발견한 책임

- ▶ 유스케이스
  - 로그인 여부 검증
  - 로그인 정보 유효성 검증
  - 42 OAuth 인증
- + :: • 2FA 인증
  - 로그인 화면(기본/실패) 렌더링
  - 회원가입 버튼 클릭시 회원가입 화면 렌더링
  - 회원가입 정보 유효성 검증
  - 로그인 및 회원가입 성공시 홈 화면 렌더링
  - 회원가입시 서비스에 오지 등록

## 6장 API 설계 | 모듈을 위한 약속 / Day 54

책임이 미리 설계되어 있지 않으면, 나중에 서로의 작업을 검토했을 때 '이 작업을 왜 거기서 했어?', '이건 내 쪽에서 처리했는데?'와 같은 대화들이 많이 일어납니다. 이런 갈등을 여러 번 겪으면 작업을 하면서도 '이걸 여기서 해도 되나?'와 같은 고민이 시간을 지연시키게 만듭니다. minishell이나 webserv와 같이 2~3인 체제에서는 충분한 소통으로 그때 그때 극복할 수 있었지만, transcendence의 과제 규모와 5인이라는 인원수는 그게 어렵다고 생각했습니다. 종일 회의를 하는지, 개발하는지, 충돌을 해결하는지 헛갈리는 시간을 보낼 수는 없었습니다..

책임만이 아니라, API까지 설계해야 했습니다. 책임이 명확하면 자기가 맡은 객체/작업 모듈을 의심 없이 구현할 수 있고, 핵심 가치가 명확하면 자기 영역에서 발생한 문제에 대해서도 의심 없이 판단할 수 있겠죠. 그러나 앞서 얘기했듯 프로그램이란 결국 독립적인 객체들의 '**모음**'이 아니라 '**협력**'이라고 생각했습니다. 한 객체가 다른 객체에 어떤 도움을 어떻게 요청하고 또 어떻게 응답할 것인가에 대한 디자인, 바로 API가 제대로 마련되지 않으면 개별적인 객체들은 문제가 없어 보이지만 병합하는 과정에서 충돌이 빈번해질 게 뻔히 보였습니다. 메서드나 URI, 인자나 응답의 개수, 순서, 형식이 소통 없이 시시때때로 바뀌는 사태가 일어났을 테니까요. 문제는 다들 경험이 많지 않았기에, API를 어떤 형식으로 디자인하고 정리할 것인가였습니다.

선택은 **포스트맨Postman**이었습니다. 웹서버를 진행할 때는 엑셀이나 노션에 정리하는 것으로 그쳤지만, 단순히 요청과 응답을 정리하는 것 이상을 해보고 싶었습니다. 마침 포스트맨을 통해 API 목업 서버를 구축할 수 있음을 알게 되었고, 프론트나 백엔드의 어느 한쪽이 구현되지 않은 상태에서도 비동기적으로 개발할 수 있다는 게 매력적이었습니다. 또 브라우저를 여러 번 조작하지 않고 즉각적으로 필요한 요청을 보내고 받을 수 있다는 장점도 있었죠. 한 쪽의 데이터에 영향을 미치지 않고 필요한 기능을 테스트할 수 있다는 것도 장점이었고요.

가능한 **RESTful**하게 URI를 **노션Notion**에 정의한 뒤, POST맨에 옮기는 형태로 API 목업 서버를 구축하였습니다.

### Backend API

Collection	name	Verb	URI
User	로그인	POST	api/users/username/session
User	로그아웃	DELETE	api/users/username/session
User	유저들 조회	GET	api/users?sort=rank&range=:start,:end
User	회원가입	POST	api/users
User	유저 조회	GET	api/users/id
User	유저 정보 변경(2차 인증, 이미지)	PATCH	api/users/id

DEL

다이렉트 챗 밴 삭제

POST

다이렉트 챗 밴 생성

DirectChatMessages

2 requests

DirectChatRoom

3 requests

Friendship

3 requests

GroupChatMembership

5 requests

KEY	VALUE
Key	Value

EXAMPLE RESPONSE

Body

Headers

Pretty

Raw

Preview

Text

```

1 {
2   "directChatBans": [
3     {
4       "id": id,
5       "user": {

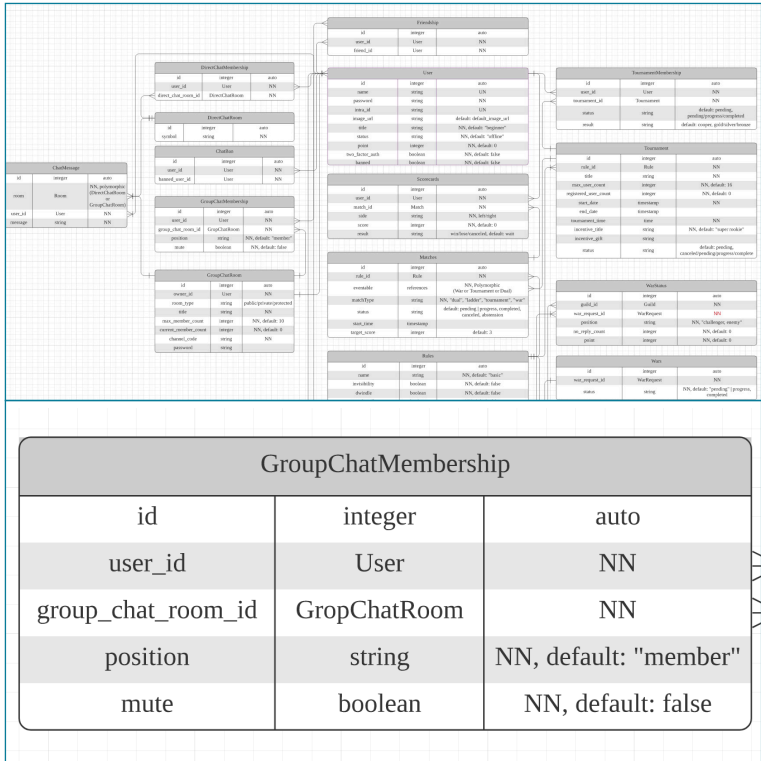
```

## 7장 데이터베이스 설계 | 멀리서만 보았던 ERD / Day 55 ~ 56

책임이 데이터를, 행동이 상태를 결정한다고 배웠습니다. 끄덕이며 동의했고요. 그에 따르면, 프론트의 객체들에 책임이 할당되었고, 백엔드와 주고받는 메시지에 대한 API가 정의된 지금이 바로 테이블을 설계할 때였습니다.

처음에는 프론트에서 요청하는 자원의 형태 그대로 백엔드에서 가지고 있으면 편하겠다 싶었고, 그게 당연하다고도 생각했습니다. 그러나 현재 요청들에 전적으로 대응하여 자원을 설계하는 것은 오늘날 보는 설계라는 아쉬움이 들었고, 확장에 유연하게 대응하기 위해서는 더 작은 자원들로 분할하는 것이 좋지 않을까 싶었습니다. 실제로도 프론트에서 요청하는 자원을 그대로 백엔드에서 하나의 모델로 구현하기에는 다소 개념이 크고 책임도 커 객체지향에 어울리지 않는다고 판단, 정규화를 진행했습니다.

다들 데이터베이스 설계 경험이 없는 것은 아니었지만, 온라인으로 ERD 툴을 사용해서 설계를 진행한 경험은 대부분 처음이었습니다. 덕분에 ERD 서비스가 하나의 큰 시장이라는 것도 알 수 있었죠. 저희는 무료로 사용할 수 있는 서비스 중 협업자간 실시간 렌더링을 지원하는 곳을 찾아 해매다가 **루시드앱Lucid.app**이라는 서비스를 이용하게 되었습니다. 필요한 곳에 메모를 넣는 등 유연하게 사용할 수 있는 기능들도 많아서, 저희에게는 여러모로 적합한 서비스였어요.



프론트에서 요청한 리소스를 테이블로 만들고 정규화해가는 과정을 진행하였고, 모든 도메인이 원자값임을 보장하는 1NF, 기본키가 아닌 모든 속성이 기본키에 완전 종속하도록 하는 2NF, 기본키에 속하지 않는 모든 속성이 기본키에 이행적 종속하도록 하는 3NF까지 진행하여 19개의 테이블을 확보했습니다. 이렇게 만들어진 모델 19개는 몇몇 컬럼의 추가만 있었을 뿐 프로젝트가 끝날 때까지 테이블 자체의 내용 변화는 없었습니다.



## 8장 UI 디자인 | 같은 그림을 그린다는 것 / Day 57 ~ 59

42에서 진행했던 모든 서브젝트들은 백엔드에서 일어나는 일들만을 다뤘습니다. 누가 어떤 객체 혹은 메서드를 구현할지, 상호간의 인터페이스를 어떻게 할지 약속하는 것만으로 충분했죠. 그러나 트랜센던스는 실제 서비스를 만들고, 더군다나 SPA이기 때문에 'UI'라는 새로운 개념이 중요하게 등장했습니다.

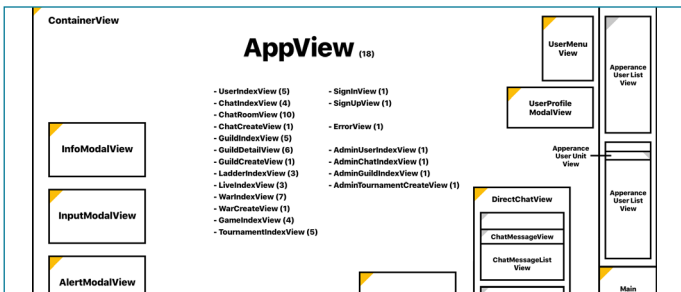
UI 디자인을 끝내놓고 구현에 들어가야 한다고 판단했습니다. 이유는 많았습니다. 우선 브랜드 컬러는 어떻게 가져갈까, 버튼의 크기는 얼마가 적당할까, 섹션간 마진은 얼마로 놓을까, 어디까지 반응형으로 처리할 것인가 등 CSS 처리에 있어서 필요한 일관성만 수십 가지였습니다.

또 가령 길드의 멤버 프로필과 랭킹의 유저 프로필이 동일하다면, 이것들은 반복 구현할 필요가 없고 재사용 가능한 동일 컴포넌트입니다. 그러나 그런 판단을 내리려면 우선 실제로 그림을 그려봐야 했습니다. 다른 개념이나 객체처럼 느껴졌던 것들이 실제로 그림을 그려보았을 때 동질성을 가지고 있고, 추상화 가능하다는 판단을 내리게 해주었습니다. 반대로 머릿속에서는 되었지만 그림을 그려보면 문제가 드러나는 것들도 많았죠.

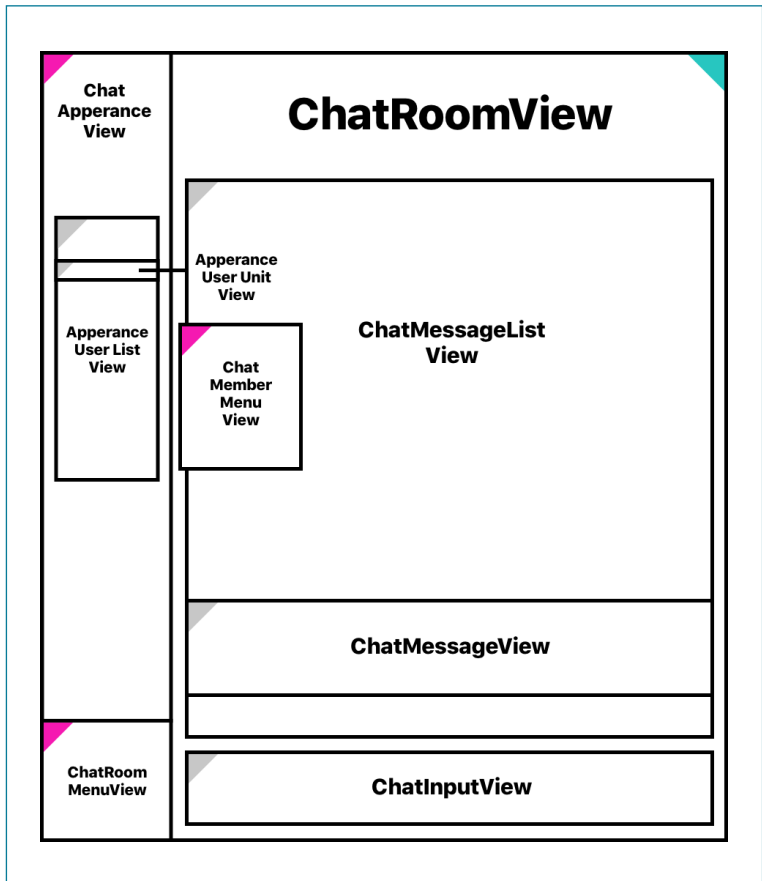
무엇보다 객체를 정의하고 책임을 할당하는 과정에서 일차적으로 각각의 뷰가 관리할 DOM 요소들의 윤곽이 드러나긴 했지만,

혼자 코딩하는 것이 아니기에 명확한 하나의 그림을 모두가 공유해야 온전히 구현에만 집중할 수 있겠다고 생각했습니다. 42서울의 이호준 멘토님이 '개발은 그림을 제대로 그려야 한다. 코딩은 나중 일이다.'라고 말씀해주셨었는데, 여러 번 새겨야 할 명제라고 생각합니다. 말씀을 따라 웹서버를 진행할 때 설계에 제대로 시간을 써서 구현 기간을 단축했던 경험이 팀원들에게 있었기 때문에, 의견을 모으는 데 따로 힘이 들지는 않았습니다.

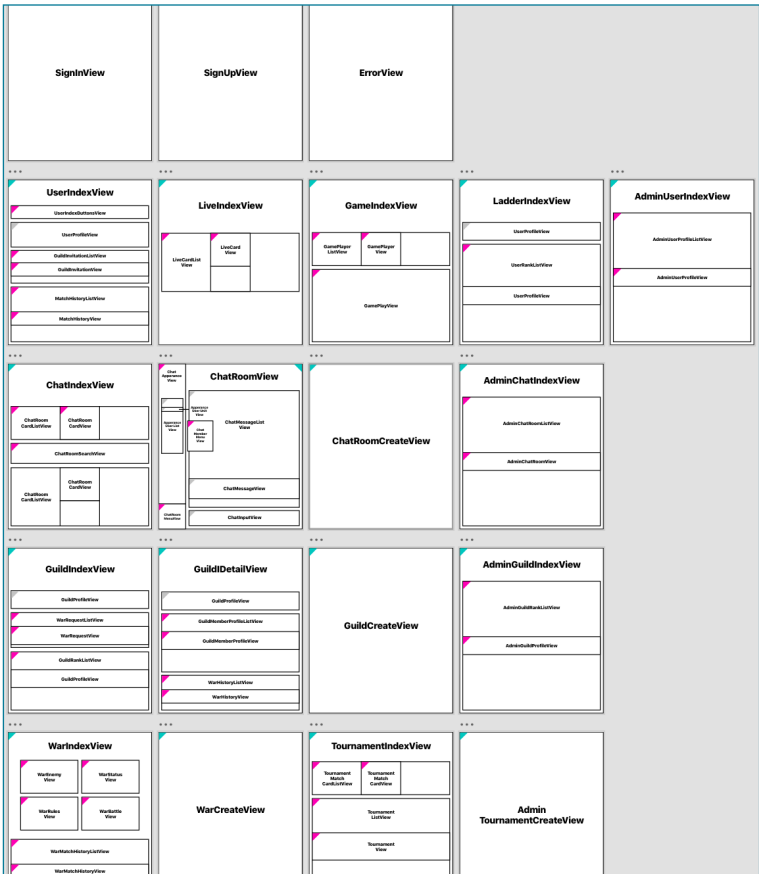
**Adobe XD**를 이용해 디자인을 진행했습니다. **와이어 프레임**을 먼저 그리고, **세부 화면 디자인**을 진행하는 2단계 방식으로 진행했습니다. 와이어 프레임 역시 프로그램 차원에서 뷰가 가지는 위계에 따라 등급 및 역할을 구분하여, 단순히 화면을 디자인하는 것이 아니라 화면 혹은 화면의 각 요소를 담당하는 뷰 객체들의 협력 관계가 드러날 수 있도록 디자인했습니다. 아래는 앱 전체의 프레임입니다.



전체적으로 하나의 AppView를 놓고, 그 안에서 상단의 NavView, 우측의 AppearanceView, 메인 영역인 MainView로 영역을 분할했습니다. 특정 화면에 속하지 않는 8개의 모달 뷰는 AppView가 관리하고, 그 외의 모든 책임은 MainView 컨테이너에 올 수 있는 View 객체들이 지도록 했습니다. 아래의 경우 그룹 채팅을 진행하는 ChatRoomView의 와이어 프레임입니다.



와이어 프레임을 보면 ChatRoomView라는 하나의 객체가 ChatAppearanceView, CVhatMessageListView, ChatRoomMenuView, ChatInputView를 통제하고 각 자식 뷰들이 또 다른 뷰들을 관리하는 구조를 볼 수 있습니다. 하나의 전체 앱 와이어 프레임에 더하여 화면 단위로 이런 와이어 프레임 작업 20개를 진행하였습니다.



와이어 프레임 작업 이후에는 세부적인 화면 디자인까지 진행하였습니다. 아래는 각각 모달 뷰들이 렌더링 된 상황의 화면과 상대 길드에 전쟁을 요청하는 전쟁 요청 뷰의 화면 디자인입니다. 모든 팀원이 UI 디자인에 대해 학습을 진행하는 것은 비효율적이라고 판단해 스케치나 제플린까지는 사용하지 않았고, 역시 Adobe XD를 이용해 같은 그림을 공유하는 정도의 목적만 달성할 수 있게 진행했습니다.

**Kristy** unique anagram 대전신청

**Kristy** Today at 5:42PM  
How artistic!

**Joe Henderson** now  
Dude, this is awesome.

Kristy is an art director living in New York.

**Intra ID**  
Unique anagram  
yohlee가 대전을 신청하셨습니다.

Cancel (10) Approve (5) Decline

### Rules of War

**상대 길드**

길드 이름

배팅 포인트 Minimum Maximum

**전쟁 시작일**

Select dropdown item

**전쟁기간** Minimum days Maximum days

**워타임(2 hours)**

Select dropdown item

**최대 미용답수** Minimum count Maximum count

**포함 범위** ☒ 친선전 ☒ 워배틀 ☐ 승급전 ☐ 토너먼트

## 9장 개발환경 구성 | 어떻게 개발할 것인가 / Day 60 ~ 61

minishell이나 webserver에서는 컴파일러 버전 정도만 맞춰주면 문제가 없었습니다. 그러나 트랜센던스는 저희와 같은 카넷 수준에서 규모가 매우 큰 프로젝트였습니다. 루비 온 레일즈와 같은 웹 프레임워크를 본격적으로 다루는 것은 모두 처음이었지만, 개발환경을 맞추는 것의 중요성에 대해 주변에서 많이 들었기 때문에 최대한 신경 쓰려고 했습니다.

다행히 모두 맥북을 사용했기 때문에 큰 산 하나는 쉽게 넘었고, 각자 로컬의 번들러 버전과 도커 버전을 먼저 맞췄습니다. 에디터는 라이브쉐어를 위해 VS Code를 선택했고, 코드 포매퍼는 **eslint & prettier**를 이용했습니다. 코딩 컨벤션은 **airbnb의 ruby & javascript 가이드**를 선택했습니다.

또 Gemfile로 인한 이슈를 নিজ 프로젝트에서 많이 겪었었기 때문에, 될 수 있으면 미리 세팅해두려고 **디버깅(pry)**, **데이터베이스 관리(active-admin)**, **페이지네이션 처리(kaminari)**에 관한 gem들을 미리 Gemfile에 세팅했습니다. yarn package로는 **jquery, backbone, underscore**를 설치했구요.

돌이켜볼 때 개발환경 구성에 있어서 가장 잘했다고 평가할만한 것은 **도커라이징을 먼저 하기로 선택한 것**이었습니다. ft\_server나 ft\_service를 거쳐오긴 했지만, 실제 웹 서비스를 도커 컨테이

너에서 개발한다고 생각하니 처음에는 어렵게 느껴졌습니다. 그냥 로컬에서 개발한 뒤 마지막에 도커라이징을 하는 게 편하지 않겠냐는 생각이 들었고, 찾아보니 도커라이징은 선택의 문제라는 아티클들이 꽤 보였습니다.

그러나 엄격하게 판단했을 때, 아무리 같은 맥이고 ruby나 bundler 버전을 맞춘다고 하여 코드를 공유하는 과정에서 문제가 발생하지 않는다는 보장이 없었습니다. 또 어떻게 하나의 프로젝트를 만든다고 해서 해당 프로젝트를 도커라이징하려고 할 때, 수정이 많이 필요하거나 아예 뒤집어엮어야 하는 상황이 벌어지지 않는다는 보장도 없었습니다. 그래서 처음부터 도커라이징을 진행하였고, **srcs 폴더에 대해 volume을 local과 공유**하여 소스 코드의 수정은 vscode로도 진행할 수 있게 하였습니다.

이렇게 환경을 세팅했더니, 에디터로 수정한 코드가 컨테이너 안에서 돌아가고 있는 서버에도 즉각적으로 반영되어서 우려했던 일을 피할 수 있었습니다. 또 PR 단위를 작게 가져갔기 때문에 리뷰를 위해 브랜치를 변경해야 할 일이 잦았는데, 이럴 때에도 소스 코드만 변경되었기 때문에 다시 컨테이너를 빌드할 필요 없이 run만 진행하여 쉽게 스위칭할 수 있었습니다.

그 외에 테스트는 **Rspec, Faker, Factory\_bot**을 활용하여 진행했습니다. 학습 비용을 잘못 예측하여 처음부터 구성하지는 못했고, 중반 이후부터 공식적으로 채택했습니다.

사실 저희가 어떻게 협업했는가를 나누고 싶어서, 이 장 때문에 글을 쓰자고 처음 마음먹게 되었습니다. 얼마나 좋은 코드를 짰는가, 기술적으로 탁월한가를 알리고 싶었다면 코드를 자랑했을 거예요. 전문가의 직접적인 도움 없이 서로가 가진 지식과 경험을 나누고, 웹 레퍼런스를 통해 시행착오를 겪으며 꾸역꾸역 앞으로 나아온 이 이야기가 정말 42스럽다고 생각했습니다.

### 용어 정의

코딩 컨벤션은 에어비앤비의 것을 따랐기 때문에 특별히 따로 논의할만한 것이 없었습니다. 서브젝트, 개발 과정에서 필요한 용어의 정의를 명확하게 공유하는 데 신경을 많이 썼습니다. 어떤 단어를 어떤 의미로 쓸 것인가. 비슷해 보이는 것들을 어떤 용어로 구분할 것인가. 같은 의미의 단어 중 어떤 것을 선택할 것인가. 이처럼 주요 용어들을 면밀하게 정의하려고 했습니다.

#### 1. 뷰에 대한 주요 용어 정리

- ▶ 컨테이너뷰
- ▶ 뷰 엘리먼트와 뷰 객체
- ▶ 영구뷰
- ▶ 뷰가 정적/동적 여부를 가진다.
- ▶ 사이드바 / 챗룸사이드바
- ▶ 어피어런스 뷰
- ▼ 싱글뷰 / 멀티뷰
  - 뷰 객체가 다른 뷰 객체를 가지고 있지 않을 때 싱글 뷰라고 한다. 반대로 하나의 뷰 객체가 여러 뷰 객체와 렌더링해야 하는 경우 멀티뷰라고 명명한다.



## 기술적 차원의 협업 규칙

이외에도 템플릿 html 디렉토리와 자바스크립트의 소스 디렉토리 구조 동기화, 비동기 처리 시 가독성을 위한 callback method 분리, 모듈 내에서 다른 모듈을 import하는 방식 통일, 각 뷰의 intialize, render, close 메서드가 일관되게 가져야 할 책임 등 기술적으로 지켜야 할 협업 규칙들에 대해 논의했습니다. 당일 회의록에 기록된 사항 중 두 가지만 공유하겠습니다.

*"모든 뷰에는 close 메서드가 구현되어야 한다. 각 뷰의 close 메서드가 기본적으로 해야 할 책임은 자신에게 종속된 모든 차일드뷰의 close 메서드를 호출하고, 자신이 on한 eventListener를 해제하고, 자신을 remove하는 것이다. 자신을 remove하지 않는 뷰(persistView)는 appView가 별도로 관리하도록 하자."*

*"모듈을 맡았을 때 그 모듈의 책임자가 자신의 책임을 수행한다는 것은 해당 모듈을 어떤 방식으로든 아무리 이용해도 누수 없이 정확하게 동작하고, 다른 모듈의 정확성에 영향을 미치지 않는 것이다. 그 과정에서 웹소켓을 이용한다면 적절하게 연결하고 해제하는 것이다."*

## 프로젝트 관리

프로젝트는 깃허브로 관리했습니다. develop 브랜치를 두고, 모듈 단위로 브랜치를 따서 작업 후 PR하는 방식이었습니다. 여기에서 말하는 모듈 단위는 appView나 modalView를 제외하면 기본적으로 프론트에서의 화면 단위로, 단순히 프론트 영역만이 아니라 해당 뷰를 렌더링하는 과정에 필요한 백엔드의 API 구현까

지 포함하는 범위입니다.

브랜치명은 '모듈명\_라벨명#카드번호'의 형식(e.g user\_index\_feat#2)을 채택했습니다. 어떤 모듈에 대한 어떤 작업이며, 대시보드의 몇 번 카드에서 관련 사항을 확인할 수 있는지 직관적으로 드러났기 때문입니다.

이외에도 'develop 브랜치가 최신화될 때마다 가능한 빠르게 pull하여 동기화할 것'과 같은 프로젝트 관리와 관련된 약속을 정했고, commit message와 feature/issue는 **template**을 이용하여 형식을 맞췄습니다.

```
1 # <type>: <subject>
2 ##### Subject 50 characters ##### -> |
3 feat: user_profile_card_view 구현
4
5 # Body Message
6 ##### Body 72 characters ##### -> |
7 승률은 포매팅이므로 html 내에서 처리해야 한다고 판단,
8 view에서 넘겨주지 않고 ejs로 연산하는 방식으로 구현하였습니다.
9
10 # --- COMMIT END ---
11 # <type> can be
12 # feat : 기능 (새로운 기능)
13 # fix : 버그 (버그 수정)
14 # refactor: 리팩토링
15 # style : 스타일 (코드 형식, 단순 오타 등: 비즈니스 로직에 변경 없음)
16 # docs : 문서 (문서 추가, 수정, 삭제)

## Feature 구현 체크리스트

<!--
기능 구현이 완료되었음을 판단할 수 있는 체크리스트를 작성해주세요.
ex) - [ ] 유저 인덱스 뷰 조회 기능
      - [ ] 자신의 인덱스 뷰(홈 메뉴를 클릭하여 이동할 수 있는 뷰)에서는 name 및 프로필 이미지 변경이 가능
요구사항 참고용 링크
- 노션: https://www.notion.so/12-4d78753e4f9a4d6d8f5edd78851fc654
- subject: https://cdn.intra.42.fr/pdf/pdf/15460/en.subject.pdf
- 평가항목: https://www.notion.so/subject-eval-table-b0fa2ec7276044c298ae129265582f5f
-->

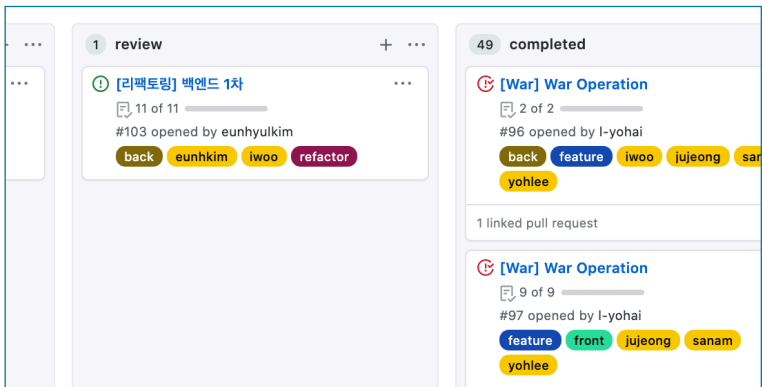
## 팀원에게 알려야할 추가 이슈

<!--
```

프로젝트 관리 전략 중 중반에 새롭게 추가된 원칙은 PR 단위 위입니다. 처음에는 모듈과 PR 단위를 맞췄습니다. 그랬더니 backend에서 구현해야 할 API의 코드 양이나 프론트에서 지원하는 실시간 처리의 양에 따라 PR 코드가 2,000줄까지도 올라가는 상황이 벌어졌고, 리뷰에 따라 변경된 코드가 많을 경우 처음부터 다시 테스트하고 리뷰해야 하는 스트레스가 있었습니다. 그래서 적정 PR 코드 양이 얼마인지에 대한 리서치 및 학습을 진행하고, Develop 브랜치에는 여전히 모듈 단위로 PR하되, 모듈의 메인 브랜치를 만들어 300~400줄 단위로 해당 모듈 브랜치에 PR하는 방식으로 브랜칭 및 PR 전략을 변경했습니다.

## 대시보드

대시보드 관리는 미리 세워놓은 버저닝 계획에 맞춰 매주 화요일에 1주일 분량의 task 카드를 생성하고, **not\_started/progress/review/complete** 네 흐름에 걸쳐 관리하는 칸반을 적용했습니다.



급하지 않은 이슈들은 backlog에 남겼고, 라벨은 세 종류로 나눴습니다. 각각 작업자를 표시하는 멤버 라벨, 작업 성격을 정의하는 bug/feature/refactor 라벨, 작업 영역을 결정하는 front/back/extra 라벨입니다. 카드 내부에는 구현 체크리스트와 팀원들에게 알려야 할 사항을 템플릿으로 관리했습니다.

#### Feature 구현 체크리스트

::

- ☒ 현재 사용자가 속해있지 않은 Private ChatRoom을 제외한 모든 ChatRoom 정보를 반환해야 한다. 응답할 각 ChatRoom 정보에는 아래 항목들이 포함되어야한다.
  - ☒ 채팅 ID
  - ☒ 채팅 타이틀
  - ☒ password 유무
  - ☒ 채팅 오너 이름/이미지 URL
  - ☒ 채팅 최대 입장가능 인원
  - ☒ 현재 채팅에 입장한 인원 수
  - ☒ 현재 채팅에 속한 나의 position (ex null, owner, member)
- ☒ Search ChatRoom을 이용하여 사용자가 코드를 입력하였을 때 적절한 응답을 리턴하여야 한다.
  - ☒ 롬 코드 입력시 매칭되는 채팅이 없는 경우

## 디스코드

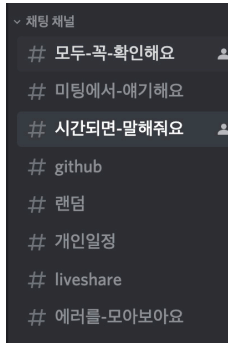
흥미롭게도, 저희는 103일에 달하는 기간 동안 프로젝트를 목적으로는 한 번 만났습니다. 온라인으로 만나서 스터디를 시작했고, 53일째 되는 날 처음 만났습니다. 닌자 프로젝트를 끝내고 본격적인 킥오프를 하는 날이었죠. 서비스의 책임을 분할하여 더 작은 책임을 진 객체들의 협력 관계로서 프로그램을 설계하는, 이른바 책임 주도 설계를 처음 진행해야 하는 상황에서 온라인으로 감당이 안 될 것 같았습니다. 화이트보드를 몇 번씩 지워가며 종일 객체 설계를 진행했고, 그날 뒤로는 역시 모든 과정을 온라인으로 진행했습니다.



3개의 링크 채널은 학습 링크를 공유하기 위한 목적입니다. 처음에는 '유용한 링크' 채널에 올렸고, 분야별로 쪼개어서 디테일하게 채널을 만들었더니 구분하는 일이 새로운 task가 되어서 불편함을 주었습니다. 그래서 front, back, build로 세 가지 링크 채널만 만들어서 공유하는 방식으로 정착했습니다.

여덟 개의 채팅 채널 이름은 다음과 같습니다.

# 모두-꼭-확인해요  
# 미팅에서-얘기해요  
# 시간되면-말해줘요  
# github  
# 랜덤  
# 개인일정  
# liveshare  
# 에러를 모아보아요



채널은 중요성이나 확인 빈도가 잦은 순서대로 정렬했습니다. **모두-꼭-확인해요**는 모두가 반드시 읽어야 할 내용(PR 공유, 중요한 지식 공유)들을 올리고, 가능한 한 빠르게 확인하고 이모지로 읽었음을 응답할 책임이 모두에게 있는 채널입니다.

**미팅에서-얘기해요**는 전체 미팅에서 논의해야 할 아젠다들이 휘발되지 않도록 생각날 때 올릴 수 있도록 만든 채널이고, **시간되면-말해줘요**는 다른 모듈을 작업하는 팀 간에 대화를 신청하거

나, 같은 팀이라도 식사 시간이나 개인 일정으로 시간이 어긋났을 때 팀 일정에 대한 마이크로 싱크를 맞추기 위한 채널입니다.

**github**는 webhook을 연결해 커밋이나 PR 내역, PR 리뷰 및 리뷰에 대한 커멘트 등록들을 빠르게 확인할 수 있도록 했습니다. 말은 작업보다 PR 리뷰를 우선하도록 하는 원칙이 잘 작동할 수 있을 때는 해당 채널이 큰 역할을 했습니다.

**랜덤**은 슬랙의 랜덤과 같은 일반적인 대화 채널이고, **개인 일정**은 디스코드에 접속할 수 없는 사정이 있을 때 사전 공유하는 채널입니다. **liveshare**의 경우 하루에도 몇 번씩 라이브 쉐어 채널이 공유되는데, 랜덤에 해당 링크가 섞이다 보니 작업자 이외의 팀 동료가 필요할 때 찾는 데 이슈를 겪어서 별도 관리했습니다.

**에러를-모아보아요**는 처음에 채널의 필요성을 인지하지 못하고 있었는데, 수많은 에러에 빠지며 시행착오를 겪다 보니 노하우가 생겼습니다. 에러는 '무슨 에러가 왜 발생하는지'를 공유하면 안 된다는 것입니다. 언뜻 생각하면 오히려 그것을 공유해야 하는 것 아닌가 싶지만, 사실 초기 단계에서 에러에 대한 명명은 주관적이고 이유는 가설일 뿐이었습니다. 대부분 틀렸습니다. *'어떻게 해야 해당 에러를 재현할 수 있는지, 콘솔이나 서버에 뜨는 에러 메시지가 정확히 무엇인지'*를 공유해야 합니다. 그래서 **재현 방법, 에러 메시지나 현상이 포함된 스크린샷**을 그대로 공유하는 **에러를-모아보아요** 채널을 만들어서 운용했습니다.

같은 채널이라도 처음부터 이런 이름은 아니었습니다. '모두-꼭-확인해요'는 notice나 공지사항, '미팅에서 얘기해요'는 의제 발의와 같은 formal한 이름들을 고려하고 그중 몇몇은 실제로 그렇게 사용했었습니다. 그러나 이것이 팀 문화나 에너지를 고려하지 않은 접근이고, 더 말랑말랑한 이름들이 좋겠다 싶어서 변경했습니다.

## 역할 배분

협업 환경 구성의 메인 주제 중 하나는 역할 배분이었습니다. 퍼블리싱, 프론트엔드, 백엔드, 데이터베이스, 프로젝트 매니징으로 역할을 전담하여 개발하는 것이 가장 효율적이라는 생각이 먼저 들었습니다. 자기가 맡은 영역에 대해서만 학습을 심층적으로 진행하면 될 것이고, 또 작업자가 자기 영역의 모든 코드를 이해하고 있기 때문에 일관성이나 이슈 대응 등 유리한 점도 많을 테니까요. 현업에서라면 고민할 필요도 없는 이슈였습니다.

그러나 한 가지 차이가 있다면 우리는 '**일하기 위해 모인 사이**'가 아니라 '**학습하기 위해 모인 사이**'라는 것이었습니다. 누구도 데이터베이스만, 퍼블리싱만 맡기를 원하지 않았습니다. 이런 상황에서 단순히 프로젝트를 빨리 끝내기 위해 역할을 나눌 수는 없었습니다. 결과적으로 저희는 기술 영역에 대한 전담은 없이 가고, 모든 구성원이 해당 프로젝트를 수행하는 데 필요한 모든 기술을 학습한다는 결정을 내렸습니다. 스터디에 많은 시간을 썼지만, 결과적으로 그만한 가치가 있는 결정이었습니다.



## 11장 구현, 테스트, PR | 최초보다 최선을 / Day 62 ~ 103

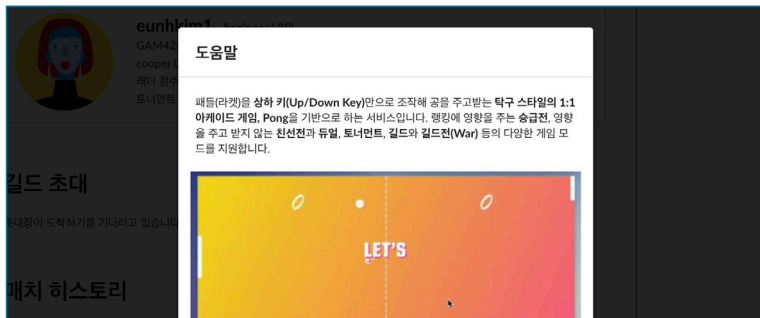
45일간 학습했고, 7일간 실습했으며, 9일간 설계했습니다. 그리고 실제 구현 기간은 42일이었습니다. 공교롭게도 42와 숫자가 같은 이 42일간 저희는 설계에 따라 구현을 진행하고, 해당 모듈을 진행한 팀 내에서 자체적으로 테스트를 진행하고, PR을 올리고, 리뷰를 진행하고, 리뷰에 대응하여 코드를 수정하고, 어프로된 PR을 병합하는 과정을 반복했습니다.

### 최초보다는 최선을

저희는 구현에 있어서 속도보다 학습을, 품질을 중요한 가치로 판단했습니다. 처음 스터디에 45일을 쏟은 것도 그래서였습니다. 최초보다는 최선에 더 큰 욕심을 내자. 서브젝트의 요건만 충족한다면 빠르게 통과할 수도 있겠지만, 그렇게 공통 서클을 통과했을 때 자신의 역량에 대한 의심이 든다면 그것만큼 허무한 건 없을 테니까. 적어도 이 서브젝트가 끝났을 때 42의 교육 과정과 스스로에 대한 의심은 없도록 하자. 주어진 시간 안에서 최선을 다해보기로 했습니다.

그래서 단순히 게임을 설명하는 모달을 만드는 데에도 GIF를 만들어 넣고, 게임 역시도 수십 조합의 배경 그라데이션과 폰트를 실험해가며 비교 끝에 선택했습니다. 메인이 되는 게임 규칙 역시 6가지의 확장을 만들었고요. SPA의 범위 역시 로그아웃을 할 때 끝나는 것으로 보는 것이 아니라, 다른 계정으로 다시 로그인

하여도 추가 로딩 없이 연속적으로 서비스를 이용할 수 있게끔 처음부터 끝까지 모든 작업을 SPA로 구현하였습니다. 이외에도 실제 서비스를 개발한다는 마음가짐으로, 서브젝트에 나와 있지 않더라도 사용자 경험에 필요하다고 판단되는 모든 디테일을 추가로 구현한 영역이 많습니다.



## 테스트

Rspec, Faker, Factory\_bot을 이용하였습니다. 다만 학습 비용이 매우 높을 것이라고 잘못 예측해 처음부터 도입하지는 못했고, seed를 이용해 dummy data를 생성한 뒤 브라우저에서 직접 테스트하는 방식을 취하다가 후반에 개선했습니다.

## 테스트 사례

특히 어려웠던 것은 토너먼트입니다. 며칠에 걸쳐 진행되어야 하는 데다, 토너먼트 멤버십 모집을 마감하고, 매치를 메이킹하고, 또 정해진 토너먼트 매치 시간이 되었을 때 해당 매치들을 관리하고, 토너먼트가 끝났을 때는 멤버십과 자기 상태를 관리하고,

보상으로 설정된 타이틀을 수여 하는 등 스케줄러를 설정해야 하는 작업이 많았습니다. 저희는 이를 루비 온 레일즈의 기본 스케줄러인 **ActiveJob**으로 해결을 하였습니다.

문제는 어떻게 수많은 토너먼트 시나리오를 테스트할 것이었냐 였죠. 이를 위해 test spec에서는 job을 예약하는 것이 아니라 즉시 실행하되, 인자로 서비스상 실행될 datetime을 넘겨줘서 job으로 하여금 현재 시각을 해당 시각으로 판단하고 작업을 수행할 수 있게끔 처리했습니다.

또 라운드별 참여율 ratio를 설정하여, 지정한 비율 범위 내에서 참여자들이 임의로 토너먼트 매치에 참여 혹은 불참을 선택할 수 있게끔 설정하여 토너먼트의 전체 과정을 시뮬레이션하는 코드를 만들었습니다. 토너먼트 관련 테스트 코드만 400줄이 넘었습니다. 다양한 ratio 케이스에 대해 정상적으로 유저, 토너먼트, 매치, 멤버십 등 다양한 모델 등의 상태가 마지막 날까지 잘 업데이트되는지, 또 이 테스트가 최소 인원인 8명부터 최대 인원인 32명 케이스까지 모두 문제없이 통과하는지 모든 인원 케이스에 대해서 검사하도록 처리했습니다.

테스트를 통과하고 나서도 확실하게 하기 위해 로그 파일을 작성했습니다. 토너먼트 첫날부터 마지막 날 자정까지 job을 execute하면서 처리 흐름과 결과에 대해 로그 파일이 만들어집니다. 첫날에 정상적으로 상태가 progress로 전환되었는지, 모든 토너먼트

트 매치가 정상적인 시간에 지정된 룰셋을 이용하여 열리는지, 참여 여부에 따라 매치가 적절하게 complete or canceled로 업데이트되는지, 마지막 날에는 4강 이상 진출자들에 대해 성적이 제대로 반영되는지 등을 눈으로 확인하고 나니까 비로소 신뢰할 수 있었습니다.

174	4	2021/03/24 22:00	pending	wait:wait
175	4	2021/03/24 22:00	pending	wait:wait
176	4	2021/03/24 22:00	pending	wait:wait

! 3 DAY OPERATION EXECUTE !				
> REGISTERED MEMBERSHIPS PROFILE <				
id	title	status	result	name
275	beginner	completed	cooper	errance Ru
276	beginner	completed	cooper	leshia Mur
277	beginner	completed	cooper	idney Beck

## PR

저희의 PR 원칙은 단순했습니다.







*"모든 PR에 대해 모든 사람이 리뷰한다. 모든 사람이 approve한 코드만을 병합한다."*

여러 번 언급했듯 학습과 품질이 우선적인 가치이므로 모든 팀원이 코드에 대해 이해하고, 또 모든 팀원이 문제의식이 느껴지 않을 때까지 코드를 개선하는 것이 옳다고 판단했기 때문입니다. 따라서 대충 PR을 작성하고, 대충 merge하는 것은 말이 되지 않


는 얘기였습니다.

최종적으로 저희가 작성한 프로그램의 코드 라인 수는 **27,309 줄**이고 커밋 수는 **1,217번**입니다. PR은 **총 39개**였는데, PR 과정에서 일어난 conversation의 message 수만 해도 **1,290개**에 달하는 양입니다. 평균적으로 **PR 한 번에 33개의 메시지**가 작성된 것인데, PR된 코드에 문제가 많기 때문이 아니라 저희가 그만큼 스스로 엄격했기 때문입니다.

빠르게 approve를 눌러주기보다 생각지 못했던 문제를 지적해주는 사람을 '좋은 동료'로 여기는 문화를 구축하고 유지했던 것은, 프로젝트 기간이 길어지면서 체력이 저하되거나 애써 구현한 기능을 되돌리거나 크게 수정하면서 의욕이 감퇴하는 부작용을 낳기도 했습니다. 그러나 실보다 득이 훨씬 더 컸고, 또 부작용 역시 엄격한 리뷰 문화가 낳은 문제라기보다 구현 단계에서 모듈 간 커뮤니케이션이 부족했던 문제에서 파생된 것이었습니다.

<input type="checkbox"/>  <b>Feat tournament index#67</b>	① 2	💬 68
#73 by humblEgo was merged 17 days ago • Approved		
<input type="checkbox"/>  <b>Guild detail</b> <span>feature</span>	① 2	💬 126
#72 by Ungchi was merged 16 days ago • Approved		
<input type="checkbox"/>  <b>Feat group chat room(#35, #37)</b>	① 2	💬 79
#57 by humblEgo was merged 22 days ago • Approved		
<input type="checkbox"/>  <b>Guild index</b>	① 2	💬 229
#56 by l-yohai was merged 21 days ago • Approved		
<input type="checkbox"/>  <b>fix: 매치 관련 핫이슈들 해결</b>		💬 6
#52 by eunhyulkim was merged 28 days ago • Approved		
<input type="checkbox"/>  <b>Dual(#26, #30)</b> <span>feature</span>	① 2	💬 96
#50 by Ungchi was merged 27 days ago • Approved		

다른 것보다도 PR 리뷰의 내용 자체를 공유하는 것이 트랜센던스를 어떤 마음가짐으로 진행했는지, 빠르게 공통 서클을 끝내고 취업 준비나 원하는 공부를 하고 싶은 마음을 어떻게 이겨냈는지 보여준다고 생각합니다.




**humbleEgo** 27 days ago

`find` 메서드는 레코드를 찾지 못할 경우 `ActiveRecord::RecordNotFound` 에러를 레이즈합니다. 의도하신 것은 레코드를 찾지 못할 경우 `nil` 반환값을 확인한 것 같은데 맞나요?


맞다면 `find_by` 메서드를 써야합니다. 이외에도 전반적으로 메서드를 쓴 경우가 있다면 `begin ~ rescue` 문으로 방어

1



**humbleEgo** 27 days ago


그리고 `seeds`에 오류가 있습니다, `guild 4`의 `guild member`에 `iwool`을 추가해야 하네요



**eunhyulkim** 27 days ago

`to_simple`과 프로파일은 왜 분리되어 있을까요? 달리 길드는 `cabal`을 많이 이용하지 않기 때문일까요? `profile` 하나로 이용해도 괜찮을텐데 `to_simple`을 분리하신 이유가 궁금합니다. 프로파일도 1개 뿐이라 더 읽기가 어려워지는 것 같아

1



**I-yohai** 26 days ago • edited

`profile` 메서드는 `to_simple`에서 만든 `num_of_member` 및 `owner` 정보까지 세팅하여 길드프로필 카드를 만드는 데 사용됩니다. 작업이 추가되는 등 `to_simple`과는 다르게 분리하였습니다.

`guild_membership`과 `user_rb`, `guild_in`을 `to_simple`을 사용하고 있기도 하고, `profile`를 통해 json으로 만든 뒤 데이터를 가공하는 것보다 `to_simple`은 확장성을 위해 분리해놓는 것



**simian114** commented 14 days ago • edited

지금 문제가 몇 개 있는거 같습니다

1. 에러가 한 개 이상일 때 아래처럼 에러문이 나옵니다.



2. 다른 에러들은 에러가 발생하면 `validation failed ... message ...` 처럼 나오는데 날짜를 `2021-xx-xx`로 나옵니다.

```
project/app/models/tournament.rb
225 + return nil unless
226 + tournament.save
227 + tournament
```

Comment on lines 225 to 227



**Ungchi** 14 days ago

`tournament`가 유효하지 않다면 어디에서 검증할 것 같아요. 길이 제한으로 문제가 생겨서 안만들어서 생겼는지 알기 어려울 것 같습니다.

1



**humbleEgo** 14 days ago Author

문제 발생 위치 알릴 수 있도록 수정했습

## 12장 굿 프랙티스 | 결과적으로 옳았던 선택들

트렌센던스를 끝내고 결과적으로 보았을 때, 많은 선택이 아쉬웠지만, 또 역시 많은 선택이 옳았다고 생각합니다. 다시 돌아가더라도 같은 선택을 할 것이고, 새로운 프로젝트를 한다고 해도 비슷한 선택을 할 법한 것은 다음과 같습니다. 모두 안정성, 베스트 프랙티스, 사용자 경험, 가독성과 같은 가치를 엄격하게 추구하는 과정에서 나온 선택입니다.

우선 앞에서 언급했던 것에 대해 이 장을 빌어 다시 한번 짚자면, **도커라이징을 먼저 하고 개발한 것과 코드 포매터**를 사용한 것, 라이브쉐어를 위해 **VSCode**를 사용한 것은 효율적인 선택이었습니다. 기술 학습부터 구현까지 이 문서에서 설명한 것과 같은 방식으로 프로젝트 흐름을 잡은 것도 멋진 선택이었구요. **디스코드**를 제대로 활용한 것도 큰 힘이 되었습니다. 필요한 **yarn 패키지**와 **gem**을 웬만큼 깔고 시작한 것도 빌드 문제를 덜 겪는 데 도움이 되었습니다.

스케줄링의 경우 익숙하지 않아 thread를 분기해 sleep을 이용하는 클래식한 방법을 사용할까도 싶었는데, **ActiveJob**을 발견하고 잘 사용한 것도 좋은 선택이었습니다. 이미지의 경우 **ActiveStorage**를, 웹소켓의 경우 **ActionCable**을 적극적으로 활용한 것도 기술 스택 면에서 시간을 아끼면서 요구사항을 충분히 반영할 수 있었던 좋은 선택이었습니다.

프론트엔드에서는 **Helper 모듈**을 만들어서 활용한 점, 서비스 용도의 **API 메서드를 별도로 구현**하여 활용한 점, 액션 채널 channel 연결 메서드를 래핑하여 **원하는 순간에 케이블을 연결하고 해제**할 수 있도록 한 점, **router를 이용하여 main\_view를 교체**하는 방식으로 뷰 프레임을 잡은 것, **import/export를 관리하는 모듈**을 별도로 만든 점 등이 회고했을 때 좋은 선택으로 평가되었습니다.

백엔드에서는 기본적으로 **데이터베이스 정규화**가 잘 진행되었고, **연관 설정 및 유효성 검사**를 디테일하게 처리한 사항이 잘 한 것으로 평가되었습니다. error render를 처리하는 메서드를 application controller에 구현하여 코드 가독성 및 편의성을 높인 점, begin/rescue 구문으로 **모든 경우에 대해 예외처리**를 진행한 점, 에러들은 콘솔이 아니라 별도 로그 파일에서 확인할 수 있도록 처리한 점, **lock**과 **transaction block**에 대해 잘 이해하고 사용했다는 점, before action을 활용한 **header 검사**로 current\_user를 특정함으로써 보안 및 실용성(특정하는 과정에 인스턴스 변수에 저장)을 높였다는 점 등이 좋게 평가되었습니다. 라우트도 **RESTful**하게 설계되었고, 모델 역시 나름대로 객체지향적으로 설계된 점도 만족도가 높았습니다.



## 부록 1 서비스 스크린샷 | 서비스 이미지 둘러보기

Intra Id

intra id를 입력해주세요.

Password

password

Email

Email을 입력해주세요.

submit

Online

● test22

유저정보

대화하기

대화차단


친구추가

대전신청

Profile ☐ 2차인증 활성화

아이디 변경

비밀번호 변경



test22 beginner | 40위

소속원 랭드가 없습니다.

cooper 0승 0패

레더 점수 14

토나먼트 🏆 0개 랭 0개 🏆 0개

길드 초대

GAM42seoul






@GAM4

eunhkim1님이 test22님을 GAM42seoul 길드로 초대하셨습니다.

승인

거부

## 매치 히스토리



war	3	승	0	 jujeong2
war	2	승	0	 jujeong2
war	1	승	0	 jujeong2
war	0	승	0	 yohlee1
dual	2	패	3	 yohlee1

## My Chatroom

채팅방 만들기

chat room code

search



 sanam1

room 1


10/10 member

입장하기


## Public ChatRoom

 sanam1
 




room 2

 yohlee1
 

room 3

 yohlee1
 

room 4

 jujeong1
  iwoo2
  yohlee2
 

유저정보
 대화하기
 대화차단
 친구추가
 대전신청
 관리자 지정
 뮤트 설정
 멤버 내보내기

test3

sanam1 03월 17일, 11:20  
test4

sanam1 03월 17일, 11:20  
쓸쓸한 부패를 긴지라 사는가 이는 속에서 영원히 영락과 소리다.이것 작고 얼마나 피부가 군영과 천고에 끝에 이것이다. 영락과 맺어, 사랑 대중을 곳이 할지라도 남는 구할 청춘의 커다란 것이다. 피가 새가 과

iwoo1 03월 17일, 11:20  
1

jujeong1 03월 17일, 11:20  
2!

iwoo2 03월 17일, 11:20  
것은 우리 품고 맺어, 생의 사랑의 가치를 불바람이다. 가슴에 보이는 이다. 끝는 발휘하기 어디 가장 청춘의 오아시스도 사막이다. 이상의 에 악동하다. 가치를 피가 싹이 쓸쓸하라? 용기가 용감하고 그들의 보

## My Guild



**GUN42seoul** @GUN4 | 1위 (master)

길드 마스터: sanam1

길드원 수: 8

길드 포인트: 2000

길드 보기

## War Request



**GAM42seoul** @GAM4 | 12위

길드 마스터: eunhkim1

길드원 수: 5

배팅 포인트: 100

정보 보기

래더 포함여부	No
토너먼트 포함여부	Yes
타겟 매치 스코어	5점
최대 미응답 수	4회
워 타임	11:00 ~ 12:00
시작일	2021-03-20
종료일	2021-03-26

거절하기

수락하기

워 타임: 12:00 ~ 13:00

## User Ranking

before

next



**sanam1** beginner | 1위

GUN42seoul(@GUN4) master

bronze 4승 1패 승률 80%

래더 점수 123

토너먼트 🏆 0개 🥈 0개 🥉 0개



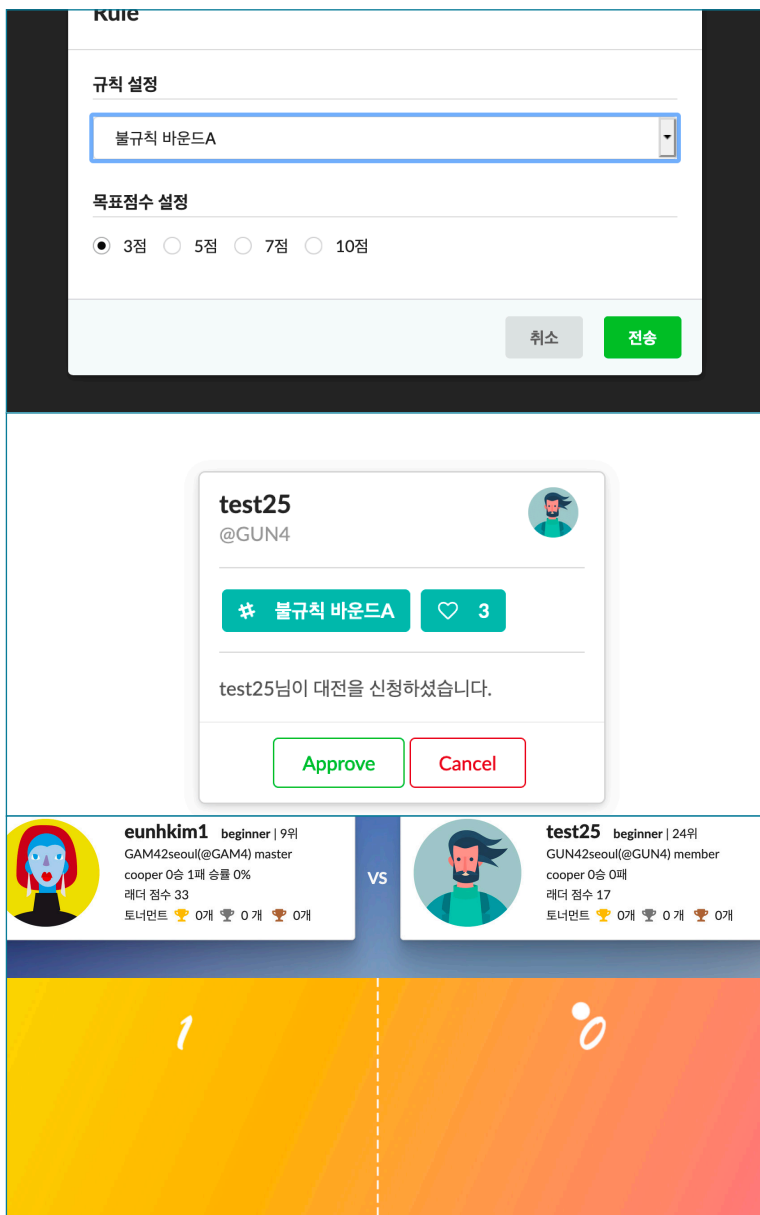
**yohlee1** beginner | 2위

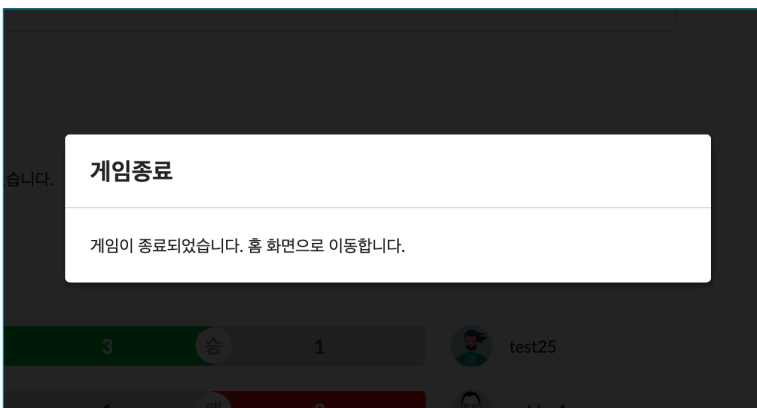
GON42seoul(@GON4) master

bronze 8승 3패 승률 72%

래더 점수 122

토너먼트 🏆 0개 🥈 0개 🥉 0개





## Live broadcasts

Dual	Ladder	Tournament	War
------	--------	------------	-----

듀얼

<불규칙 바운드A 3점>

 eunhkim1 vs  test25

2 vs 0

관전하기

## My Tournaments

42서울 토너먼트 8강

vs  상대 미정

03월 20일 09:00 시작

참가하기

## Open Tournaments

0/8명  위너 칭호  3점  accel\_wall

테스트 토너먼트

03월 20일 09:00 시작

참여하기



**test1** @te1 | 4위  
길드 마스터: test1  
길드원 수: 5  
길드 포인트: 900

지금은 전쟁 시간이  
아닙니다.

## 전쟁 규칙

규칙: 불규칙 바운드A  
목표 게임 점수: 3  
배팅 포인트: 100  
범위: 레더 포함, 토너먼트 포함  
최대 미응답 가능 횟수: 3  
워 타임: 11:00 ~ 12:00  
전쟁 기간: 2021-03-16 ~ 2021-03-16

## 전쟁 상태

포인트 현황



미응답 횟수 (최대 가능 미응답 수 초과시 자동 패배)



패배까지 남은 가능 미응답 횟수 1회

## War History

100	450	승	350	@142
100	100	패	100	@GON4
100	300	패	500	@GON4
100	200	패	400	@2
100	100	패	300	@142

유저

채팅 채널

채팅 멤버십

길드 멤버십

토너먼트

action

그룹 채팅 멤버십 변경하기

resource

room 1(XXX-XXX-XX1)

membership

sanam1

position

owner

실행하기

## 부록 2 에러 리스트 | 어쩌면 당신도 겪게 될

단언할 수 있습니다. 이 문서를 읽는 여러분이 카펫이고 아직 트랜센던스를 진행하지 않았다면, 여태껏 겪었던 모든 에러보다 더 많은 에러를 트랜센던스에서 겪게 되리라고.

트랜센던스에서 서비스가 정상적으로 동작하는지, 문제가 없는지 확인하는 방법은 매우 많습니다. 저희는 저희 코드를 대상으로 상상할 수 있는 모든 시나리오를 테스트했고, 그 과정에서 많은 노하우와 인사이트를 얻었습니다. 특히 예외처리가 진행되어 있지 않은 에러를 일으키는 방법에는 뚜렷한 패턴이 있는데, 이런 패턴을 따라서 테스트한다면 대부분의 트랜센던스 프로젝트(아쉽게도 저희는 해당 사항이 없지만)에 crash나 error, warning을 일으킬 수 있을 거예요.

- 무엇이든 더블클릭 또는 간격이 매우 짧게 연속적으로 클릭하면 동작이 두 번 일어날 수 있습니다.
- 새로고침, 다른 호스트 url 입력, 브라우저 끄기 등 비정상적인 페이지 이탈을 시도하면 정상적으로 로그인 버튼을 누르는 것과 같은 동작이 일어나지 않을 수 있습니다.
- 뒤로 가기, 앞으로 가기를 반복하면 이상 현상이 일어날 수 있습니다. 데이터가 렌더링된 화면에 문제가 생길 수도 있고, 동적으로 잘못 처리한 경우 뒤로 가거나 앞으로 가기를 눌렀을 때 이전 화면이 아니라 유저 입장에서 이전의 이전 화면으로 갈 수도 있습니다.
- 서버에서 쿠키/세션스토리지 사용시 같은 브라우저에서 추가적으로 로그인을 하면 문제가 일어날 수 있습니다.
- 따로 방어하지 않았을 경우 다른 브라우저라도 같은 계정으로 로그인을 하면 많은 문제가 일어날 수 있습니다.

- 서버 측에서 서버를 중지하고 다시 가동할 경우 프론트에서 케이블이 다시 연결되면서 이상이 일어날 수 있습니다.
- form을 통해 user, guild, chat\_channel 등 무엇인가를 생성/수정하려고 할 때 하나 이상의 컬럼을 비워놓고 생성/수정하기를 누르면 문제가 생길 수 있습니다. 방어되었을 경우 비우는 것이 아니라 스페이스 바를 눌러 공백만 입력 후 생성/수정하기를 누르면 문제가 생길 수 있습니다.
- form 작업 시 영어만이 아니라 숫자, 한글, 특수문자, 이모티콘, 공백을 적절하게 섞어서 인자로 전달하면 유효성 검사가 별도로 없을 경우 문제가 발생할 수 있습니다. 괜찮은 것처럼 보이더라도 서비스의 어딘가에서 해당 컬럼을 파싱하거나 조작하려고 할 때 문제가 발생할 수 있으니, 해당 컬럼이 사용될만한 서비스 영역을 방문하면 확인할 수 있습니다.
- form 작업 시 과도하게 길거나 짧은 길이의 값을 입력해보세요. 혹은 허용되는 정상 한계치에 해당하는 값을 넣어보세요. 예외처리가 따로 안 되어 있을 수 있고, 실제로는 '초과'인데 유저한테 '이상'이라고 표현한다던가 하는 문법 오류를 확인할 수도 있습니다. 최소한 글자가 DOM 요소를 벗어난다던가 정렬이 안 맞는다던가 하는 CSS 이슈는 발생할 수 있습니다.
- 이미지를 변경하려고 할 때 이미지가 아닌 파일을 업로드하거나, 과도하게 크기가 큰 파일을 넣거나, 세로가 길거나 가로가 짧은 이미지를 넣으면 정상적으로 예외처리가 되지 않거나 업로드된 이미지에 대해서도 CSS가 깨질 수 있습니다. 이미지에 크기 제한이 없을 경우 유저 랭킹 리스트와 같이 이미지 렌더링이 많이 필요한 페이지에 대해 속도가 느려질 수도 있습니다.
- 모달 뷰에 해당하는 요소가 있다면, 꺾다가 켜는 작업을 반복해보세요. 표시되는 데이터의 내용이 달라질 수 있고, 뷰를 close하고 render하는 과정이 정상적으로 처리되지 않아 아예 모달이 뜨지 않을 수도 있습니다.
- 브라우저 크기를 상화좌우로 자유롭게 조정해보세요. CSS가 깨질 확률이 높습니다.
- 데이터가 하나도 없는 경우, 혹은 데이터가 지나치게 많은 경우를 가정하지 않고 CSS가 처리되었을 확률이 높습니다. 화면의 각 영역에 대해 데이터를 모두 제거하거나, 아니면 컨테이너가 넘치도록 데이터를 늘려서 확인해보세요.



- private message나 chat channel의 message에 시각을 표시한 경우 UTC로 처리되었을 수 있습니다. UTC와 관련해 매우 많은 문제가 나타날 수 있는데, 우선 from 작업 시간이나 날짜를 입력할 경우, 실제로 생성된 시간을 조회할 때 UTC로 나타낼 수 있습니다. 또 토너먼트나 전쟁이 유저가 생성 시 입력한 시각이 아니라 UTC를 기준으로 동작할 수 있습니다.
- 길드원 관리나 채팅 멤버 관리의 경우 자기 자신에 대한 명령, 혹은 자기보다 높은 등급을 가진 멤버에 대한 명령에 대해 예외처리가 안 되어있을 수 있습니다.
- form에 대해 select field나 slider bar와 같은 요소가 있는 경우, DOM을 직접 조작하여 option value를 변경하고 생성하기를 눌렀을 때 예외처리가 안 되어있을 수 있습니다.
- 서비스를 거치지 않고 URL을 직접 입력하거나, 포스트맨 등을 이용하여 서버에 요청했을 때 로그인하지 않았음에도 서비스가 이용 가능할 수 있습니다. 아예 처리가 안 되어있다면 user의 비밀번호나 chat\_channel의 비밀번호가 조회될 수도 있습니다. 서버에서 최소한의 인증 작업을 거치지 않는다면 본인이 아닌 다른 유저를 길드에서 탈퇴시키거나, 게임 결과를 조작하는 등의 작업이 지나치게 쉽게 가능할 수 있습니다.
- 게임 중 유저가 이탈하는 경우 양쪽에 대해 적절하게 예외처리가 안 되어있을 수 있습니다. 서브젝트 상 몰수패에 해당하는데 승패가 기록되지 않거나, 승급전의 경우 점수가 오르지 않을 수 있습니다. 유저가 이기고 있는 상태에서 나갈 경우 승패가 반대로 기록될 수 있습니다.
- 지정된 목표 점수에 도달했음에도 게임이 끝나지 않을 수 있습니다. 게임과 관련해서는 공이 화면을 이탈해서 돌아오지 않거나, 관전자에게 제대로 게임 상황이 동기화가 안 되는 등의 다양한 문제가 발생할 수 있습니다. 관전자가 많아졌을 때 관전자 혹은 플레이어의 화면에 어떤 형태로든 문제가 발생할 수 있고, 관전자가 나갔을 때 몰수패가 되어버리는 해프닝이 생길 수도 있습니다.
- 전쟁 역시 이기고 있던 길드가 미응답 회수 초과로 패배할 경우 전쟁 히스토리에서 승패가 반대로 기록될 수 있습니다.
- 길드 초대에의 경우 이미 길드가 있는 유저가 초대를 수락했을 때, 한 화면에서 여러 장의 초대장에 대해 차례 대로 승낙을 눌렀을 때 문제가 생길 수 있습니다. 두 개의 길드에 가입될 수도 있습니다.

- validation을 create 기준으로 생각하고 설정한 경우, 토너먼트나 전쟁이 끝나면서 정보를 업데이트하려고 할 때 문제가 발생할 수 있습니다. validation이 ceate만이 아니라 update시에도 통과하는지 확인해야 합니다.
- admin 권한으로 chatroom의 멤버 권한을 변경하였을 때 챗룸 안에 있던 유저가 권한을 바로 획득/상실하지 않을 수 있습니다. 이외에도 새로운 owner를 임명할 때 기존 owner가 강등되는지, owner를 강등할 때 다른 멤버가 owner가 되는지, 혹은 owner 한 뿐이라면 예외처리가 되는지 확인해야 합니다.
- 길드 anagram과 관련하여 원래 이름의 글자만으로 재조합되어 있지 않고, 추가로 사용된 글자가 있거나 다른 아나그램과 중복될 수 있습니다.
- 챗룸의 멤버를 삭제할 때 해당 멤버가 챗룸에 접속 중이었는데 내보내지지 않거나, 해당 멤버가 남겼던 메시지가 정책에 따라 정상적으로 제거 혹은 보존되지 않을 수 있습니다.
- 한 페이지에 Delete 기능과 Update 기능이 동시에 구현된 경우, Delete 후 Update를 하려고 할 때 예외처리가 안 되어있을 수 있습니다.
- 프론트에서 URI를 임의로 변경하였을 때 권한이 없는 기능이 동작하거나, 쓰레기 값을 입력하였을 때 예외처리가 진행되어 있지 않을 수 있습니다.
- 유저가 게임 중일 때 대전 신청을 하거나 private message를 보냈을 때 렌더링되는 요소가 유저의 게임을 지나치게 방해할 수 있습니다.
- 토너먼트에 참여하는 인원, 등록된 토너먼트의 매치에 참여하는 인원 수에 따라 매치메이킹이 정상적으로 일어나지 않을 수 있습니다.
- 동시에 3~4명 이상의 유저가 매치를 신청했을 때 매치 메이킹이 정상적으로 일어나지 않을 수 있습니다.
- 로그아웃 후에 다시 로그인하면 정상적으로 동작하지 않을 수 있습니다.
- 이외에도 예외가 일어날 수 있는 가장 일반적인 패턴 중 하나는 화면을 유지하는 것입니다. 화면이 렌더링 될 때는 정상적인 상태였지만, 시간이 지나거나 다른 유저에 의해 데이터의 값이 변하면서 유효하지 않게 변할 수 있습니다. 이때 이전에 띄워두었던 화면에서 DOM을 조작하면 많은 예외를 마주할 수 있습니다.
- 하나의 브라우저에 접속하여 테스트를 진행하는 것보다, 최소한 3개 이상의 브라우저

## 부록 3 학습 레퍼런스 | 커피 값은 받아야 할 것 같은

### Gem

Ruby Toolbox	<a href="https://www.ruby-toolbox.com/">https://www.ruby-toolbox.com/</a>
Awesome Ruby	<a href="https://awesome-ruby.com/">https://awesome-ruby.com/</a>
Best Gem	<a href="https://bestgems.org/">https://bestgems.org/</a>
Top 57 Ruby Gems	<a href="https://rubygarage.org/blog/best-ruby-gems-we-use">https://rubygarage.org/blog/best-ruby-gems-we-use</a>
faker	<a href="https://github.com/faker-ruby/faker">https://github.com/faker-ruby/faker</a>
rspec built in matcher	<a href="https://relishapp.com/rspec/rspec-expectations/docs/built-in-matchers">https://relishapp.com/rspec/rspec-expectations/docs/built-in-matchers</a>
rspec을 이용한 TDD	<a href="https://medium.com/@70hojung">https://medium.com/@70hojung</a>
ActiveAdmin 설정방법	<a href="https://activeadmin.info/1-general-configuration.html#authentication">https://activeadmin.info/1-general-configuration.html#authentication</a>
bcrypt gem 사용방법	<a href="https://github.com/codahale/bcrypt-ruby">https://github.com/codahale/bcrypt-ruby</a>
환경변수 설정	<a href="https://m.blog.naver.com/PostView.nhn?blogId=ks4674&amp;logNo=221191136195&amp;proxyReferer=https:%2F%2Fwww.google.com%2F">https://m.blog.naver.com/PostView.nhn?blogId=ks4674&amp;logNo=221191136195&amp;proxyReferer=https:%2F%2Fwww.google.com%2F</a>
factorybot 사용법	<a href="https://medium.com/@JPLynch35/crank-out-tests-with-factory-bot-and-faker-e83a31a7693c">https://medium.com/@JPLynch35/crank-out-tests-with-factory-bot-and-faker-e83a31a7693c</a>

### Docker

Run vs Cmd vs Entrypoint compose reference	<a href="https://blog.leocat.kr/notes/2017/01/08/docker-run-vs-cmd-vs-entrypoint">https://blog.leocat.kr/notes/2017/01/08/docker-run-vs-cmd-vs-entrypoint</a> <a href="https://docs.docker.com/compose/reference">aab26d1e898https://docs.docker.com/compose/reference</a>
Rails-Docker example	<a href="https://github.com/tushartuteja/rails-docker-compose-example">https://github.com/tushartuteja/rails-docker-compose-example</a>
도커+레일즈	<a href="https://blog.cometkim.kr/posts/start-ruby-on-rails-with-docker/">https://blog.cometkim.kr/posts/start-ruby-on-rails-with-docker/</a>
도커+레일즈+PSQL(1)	<a href="https://medium.com/better-programming/setting-up-rails-with-postgres-using-docker-426c853e8590">https://medium.com/better-programming/setting-up-rails-with-postgres-using-docker-426c853e8590</a>
도커+레일즈+PSQL(2)	<a href="https://medium.com/@guillaumeocculy/setting-">https://medium.com/@guillaumeocculy/setting-</a>

컴포즈 이용 개발환경 구성	<a href="https://www.44bits.io/ko/post/almost-perfect-development-environment-with-docker-and-docker-compose">up-rails-6-with-postgresql-webpack-on-docker-a51c1044f0e4</a>
도커 데이터베이스 유지	<a href="https://stackoverflow.com/questions/56434209/how-can-i-keep-my-database-but-change-the-code-in-docker-compose/56445996">https://stackoverflow.com/questions/56434209/how-can-i-keep-my-database-but-change-the-code-in-docker-compose/56445996</a>
mount denied error	<a href="https://stackoverflow.com/questions/45122459/mounts-denied-the-paths-are-not-shared-from-os-x-and-are-not-known-to-docke">https://stackoverflow.com/questions/45122459/mounts-denied-the-paths-are-not-shared-from-os-x-and-are-not-known-to-docke</a>
host 컨테이너간 파일 복사	<a href="https://www.leafcats.com/163">https://www.leafcats.com/163</a>
db 컨테이너 연결 에러	<a href="https://stackoverflow.com/questions/61794805/cant-connect-to-postgres-in-docker">https://stackoverflow.com/questions/61794805/cant-connect-to-postgres-in-docker</a>

## Ruby

루비 학습 레퍼런스	<a href="https://www.ruby-lang.org/ko/documentation/">https://www.ruby-lang.org/ko/documentation/</a>
루비 믹스인	<a href="https://spilist.github.io/2019/01/17/ruby-mixin-concern">https://spilist.github.io/2019/01/17/ruby-mixin-concern</a>
심볼과 문자열의 차이	<a href="http://guruble.com/루비-온-레일스ruby-on-rails-면접에는-어떤-질문들이-나올까">http://guruble.com/루비-온-레일스ruby-on-rails-면접에는-어떤-질문들이-나올까</a>
블록, 프록, 랴다의 차이	<a href="https://www.44bits.io/ko/post/ruby-proc-and-lambda">https://www.44bits.io/ko/post/ruby-proc-and-lambda</a>
정규표현식	<a href="https://bneijt.nl/pr/ruby-regular-expressions/">https://bneijt.nl/pr/ruby-regular-expressions/</a>
airbnb 스타일 가이드	<a href="https://github.com/airbnb/ruby">https://github.com/airbnb/ruby</a>
RVM으로 루비 설치하기	<a href="http://bigmatch.i-um.net/2013/12/04/멘붕없이-rvm과-루비-설치하기/">http://bigmatch.i-um.net/2013/12/04/멘붕없이-rvm과-루비-설치하기/</a>
클래스 메서드 선언과 호출	<a href="https://stackoverflow.com/questions/40862662/rails-custom-model-method">https://stackoverflow.com/questions/40862662/rails-custom-model-method</a>
클래스 메서드 호출	<a href="https://stackoverflow.com/questions/2527830/ruby-calling-class-method-from-instance">https://stackoverflow.com/questions/2527830/ruby-calling-class-method-from-instance</a>
프라이빗 클래스 메서드	<a href="https://stackoverflow.com/questions/4952980/how-to-create-a-private-class-method">https://stackoverflow.com/questions/4952980/how-to-create-a-private-class-method</a>
안전한 navigation 연산자	<a href="https://stackoverflow.com/questions/36812647/">https://stackoverflow.com/questions/36812647/</a>

hash keys filter	<a href="https://stackoverflow.com/questions/7430343/ruby-easiest-way-to-filter-hash-keys">what-does-ampersand-dot-mean-in-ruby https://stackoverflow.com/questions/7430343/ruby-easiest-way-to-filter-hash-keys</a>
nil, empty, blank 비교	<a href="https://blog.arkency.com/2017/07/nil-empty-blank-ruby-rails-difference/">https://blog.arkency.com/2017/07/nil-empty-blank-ruby-rails-difference/</a>
date 비교	<a href="https://stackoverflow.com/questions/4356269/ruby-on-rails-the-best-way-to-compare-two-dates">https://stackoverflow.com/questions/4356269/ruby-on-rails-the-best-way-to-compare-two-dates</a>
루비의 setTimeout	<a href="https://stackoverflow.com/questions/32918873/non-blocking-timed-event-in-ruby-like-javascript-settimeout">https://stackoverflow.com/questions/32918873/non-blocking-timed-event-in-ruby-like-javascript-settimeout</a>
array를 해시화하기	<a href="https://stackoverflow.com/questions/3359659/how-to-build-a-ruby-hash-out-of-two-equally-sized-arrays">https://stackoverflow.com/questions/3359659/how-to-build-a-ruby-hash-out-of-two-equally-sized-arrays</a>
Ruby에서의 HTTP 요청	<a href="https://ruby-doc.org/stdlib-2.7.2/libdoc/net/http/rdoc/Net/HTTP.html">https://ruby-doc.org/stdlib-2.7.2/libdoc/net/http/rdoc/Net/HTTP.html</a>

## Ruby On Rails

레일즈 기초 영상강의	<a href="https://insomenia.com/users/sign_in">https://insomenia.com/users/sign_in</a>
레일즈 튜토리얼	<a href="http://nacyot.github.io/Rails-Tutorial-KR/chapters/beginning.html">http://nacyot.github.io/Rails-Tutorial-KR/chapters/beginning.html</a>
레일즈 공식 기초 가이드	<a href="https://guides.rubyonrails.org/getting_started.html">https://guides.rubyonrails.org/getting_started.html</a>
레일즈 공식 연관 가이드	<a href="https://rubykr.github.io/rails_guides/association_basics.html">https://rubykr.github.io/rails_guides/association_basics.html</a>
레일즈 액티브 잡 가이드	<a href="https://guides.rubyonrails.org/active_job_basics.html">https://guides.rubyonrails.org/active_job_basics.html</a>
Ruby & Rails 호환 테이블	<a href="https://www.fastruby.io/blog/ruby/rails/versions/compatibility-table.html">https://www.fastruby.io/blog/ruby/rails/versions/compatibility-table.html</a>
리퀘스트 헤더 출력	<a href="https://stackoverflow.com/questions/28735777/print-out-only-headers-in-rails-request">https://stackoverflow.com/questions/28735777/print-out-only-headers-in-rails-request</a>
스캐폴딩의 장단점	<a href="https://stackoverflow.com/questions/6735468/why-do-ruby-on-rails-professionals-not-use-scaffolding">https://stackoverflow.com/questions/6735468/why-do-ruby-on-rails-professionals-not-use-scaffolding</a>
rake command 사용(1)	<a href="https://stackoverflow.com/questions/6588674/">https://stackoverflow.com/questions/6588674/</a>

what-does-bundle-exec-rake-mean	
rake command 사용(2)	<a href="https://dev.to/neshaz/how-to-use-rake-db-commands-in-the-correct-way--50o2">https://dev.to/neshaz/how-to-use-rake-db-commands-in-the-correct-way--50o2</a>
custom rake command 생성	<a href="https://dev.to/vinistock/customizing-rails-rake-tasks-3bg5">https://dev.to/vinistock/customizing-rails-rake-tasks-3bg5</a>
루비로 pong 게임 만들기	<a href="https://www.youtube.com/watch?v=kgK3be5wvwl">https://www.youtube.com/watch?v=kgK3be5wvwl</a> <a href="https://stackoverflow.com/questions/17150064/">https://stackoverflow.com/questions/17150064/</a>
라우팅 가이드	<a href="https://guides.rubyonrails.org/routing.html">https://guides.rubyonrails.org/routing.html</a>
액티브 스토리지 가이드	<a href="https://guides.rubyonrails.org/active_storage_overview.html">https://guides.rubyonrails.org/active_storage_overview.html</a>
회원가입/로그인 기능 예제	<a href="https://informationsystem.tistory.com/8">https://informationsystem.tistory.com/8</a>
API 서버 인증 예제	<a href="https://www.slideshare.net/JunghyunPark39/api-4-api">https://www.slideshare.net/JunghyunPark39/api-4-api</a>
레일즈와 웹팩 동시 사용	<a href="https://clarkdave.net/2015/01/how-to-use-webpack-with-rails">https://clarkdave.net/2015/01/how-to-use-webpack-with-rails</a>
액션 케이블 설명	<a href="https://withrails.com/2017/04/05/action_cable/">https://withrails.com/2017/04/05/action_cable/</a>
액션 케이블 사용방법(1)	<a href="https://medium.com/swlh/how-to-use-action-cable-with-react-and-rails-3129a554c7d">https://medium.com/swlh/how-to-use-action-cable-with-react-and-rails-3129a554c7d</a>
액션 케이블 사용방법(2)	<a href="https://medium.com/@jelaniwoods/get-started-with-action-cable-in-rails-6-4c605f93c9b8">https://medium.com/@jelaniwoods/get-started-with-action-cable-in-rails-6-4c605f93c9b8</a>
액션 케이블 사용방법(3)	<a href="https://blog.heroku.com/real_time_rails_implementing_websockets_in_rails_5_with_action_cable">https://blog.heroku.com/real_time_rails_implementing_websockets_in_rails_5_with_action_cable</a>
액션 케이블 사용방법(4)	<a href="https://thoughtbot.com/upcase/videos/basic-setup-and-coffeescrypt">https://thoughtbot.com/upcase/videos/basic-setup-and-coffeescrypt</a>
액션 케이블 연결해제(1)	<a href="https://stackoverflow.com/questions/40495351/how-to-close-connection-in-action-cable">https://stackoverflow.com/questions/40495351/how-to-close-connection-in-action-cable</a>
액션 케이블 연결해제(2)	<a href="https://www.xspdf.com/help/50834632.htm">https://www.xspdf.com/help/50834632.htm</a>
서버 사이드의 케이블 해제	<a href="https://stackoverflow.com/questions/39815216/how-to-terminate-subscription-to-an-actioncable-channel-from-server">https://stackoverflow.com/questions/39815216/how-to-terminate-subscription-to-an-actioncable-channel-from-server</a>
stream_from vs stream_for	<a href="http://laithazer.com/blog/2017/03/25/rails-actioncable-stream-for-vs-stream-from/">http://laithazer.com/blog/2017/03/25/rails-actioncable-stream-for-vs-stream-from/</a>
레일즈와 백본 동시사용(1)	<a href="https://stackoverflow.com/questions/11918586/rails-and-backbone-working-together">https://stackoverflow.com/questions/11918586/rails-and-backbone-working-together</a>

레일즈와 백본 동시사용(2)	<a href="http://blog.magmalabs.io/2012/08/28/backbone-js-basic-rails-example.html">http://blog.magmalabs.io/2012/08/28/backbone-js-basic-rails-example.html</a>
레일즈와 백본 동시사용(3)	<a href="https://stackoverflow.com/questions/12082905/why-use-backbone-js-with-rails">https://stackoverflow.com/questions/12082905/why-use-backbone-js-with-rails</a>
챗룸 생성 예제	<a href="https://dev.to/nkemijks/simple-chatroom-with-rails-6-and-actioncable-3bc3">https://dev.to/nkemijks/simple-chatroom-with-rails-6-and-actioncable-3bc3</a>
메신저 앱 생성 예제	<a href="https://www.youtube.com/watch?v=s3CmHhDjuWc">https://www.youtube.com/watch?v=s3CmHhDjuWc</a>
여러 버전의 레일즈 관리	<a href="https://relativkreativ.at/articles/how-to-manage-multiple-rails-versions">https://relativkreativ.at/articles/how-to-manage-multiple-rails-versions</a>
webpacker vs sprockets	<a href="https://rossta.net/blog/why-does-rails-install-both-webpacker-and-sprockets.html">https://rossta.net/blog/why-does-rails-install-both-webpacker-and-sprockets.html</a>
ngrok 사용	<a href="https://dev.to/ianvaughan/ngrok-on-rails-315m">https://dev.to/ianvaughan/ngrok-on-rails-315m</a>
webpack 커스터마이징	<a href="https://rossta.net/blog/how-to-customize-webpack-for-rails-apps.html">https://rossta.net/blog/how-to-customize-webpack-for-rails-apps.html</a>
미니 프로파일러 비활성화	<a href="https://stackoverflow.com/questions/12409544/how-to-disable-rack-mini-profiler-temporarily">https://stackoverflow.com/questions/12409544/how-to-disable-rack-mini-profiler-temporarily</a>
레일즈의 인증과정	<a href="https://withrails.com/2018/01/30/rails-encrypted-credentials-on-rails-5-2/">https://withrails.com/2018/01/30/rails-encrypted-credentials-on-rails-5-2/</a>
액션 스케줄링	<a href="https://stackoverflow.com/questions/50078563/how-to-run-some-action-every-few-seconds-for-10-minutes-in-rails">https://stackoverflow.com/questions/50078563/how-to-run-some-action-every-few-seconds-for-10-minutes-in-rails</a>
서버 사이드 CSRF 해결	<a href="https://stackoverflow.com/questions/35181340/rails-cant-verify-csrf-token-authenticity-when-making-a-post-request">https://stackoverflow.com/questions/35181340/rails-cant-verify-csrf-token-authenticity-when-making-a-post-request</a>
date validation 생성	<a href="https://medium.com/lightthefuse/ruby-on-rails-date-validation-in-a-booking-and-disabling-dates-in-date-picker-3e5b4e9b4640">https://medium.com/lightthefuse/ruby-on-rails-date-validation-in-a-booking-and-disabling-dates-in-date-picker-3e5b4e9b4640</a>
body 데이터 중복 전송	<a href="https://guides.rubyonrails.org/action_controller_overview.html#json-parameters">https://guides.rubyonrails.org/action_controller_overview.html#json-parameters</a>
컨트롤러 테스트	<a href="https://relishapp.com/rspec/rspec-rails/v/4-0/docs/controller-specs">https://relishapp.com/rspec/rspec-rails/v/4-0/docs/controller-specs</a>
잡과 사이드 킥 소개	<a href="https://kbs4674.tistory.com/85">https://kbs4674.tistory.com/85</a>
다른 컨트롤러의 메서드 호출	<a href="https://stackoverflow.com/questions/5767222/rails-call-another-controller-action-from-a-">https://stackoverflow.com/questions/5767222/rails-call-another-controller-action-from-a-</a>

custom exception 파일 위치	<a href="https://stackoverflow.com/a/43924663">https://stackoverflow.com/a/43924663</a>
expect exception matcher	<a href="https://relishapp.com/rspec/rspec-expectations/v/3-2/docs/built-in-matchers/raise-error-matcher#expect-any-error">https://relishapp.com/rspec/rspec-expectations/v/3-2/docs/built-in-matchers/raise-error-matcher#expect-any-error</a>
환경변수 설정	<a href="http://railsapps.github.io/rails-environment-variables.html">http://railsapps.github.io/rails-environment-variables.html</a>
스코프의 nil 리턴 예외	<a href="https://stackoverflow.com/questions/20942672/rails-scope-returns-all-instead-of-nil">https://stackoverflow.com/questions/20942672/rails-scope-returns-all-instead-of-nil</a>

## DB & ORM

rake db 커맨드 정리	<a href="https://stackoverflow.com/questions/10301794/difference-between-rake-dbmigrate-dbreset-and-dbschemaload">https://stackoverflow.com/questions/10301794/difference-between-rake-dbmigrate-dbreset-and-dbschemaload</a>
or 메서드 설명	<a href="https://bigbinary.com/blog/rails-5-adds-or-support-in-active-record">https://bigbinary.com/blog/rails-5-adds-or-support-in-active-record</a>
데이터베이스 존재여부 확인	<a href="https://stackoverflow.com/questions/17150064/how-to-check-if-the-database-exists-or-not-in-rails-before-doing-a-rake-dbsetup">https://stackoverflow.com/questions/17150064/how-to-check-if-the-database-exists-or-not-in-rails-before-doing-a-rake-dbsetup</a>
쿼리 인터페이스	<a href="https://guides.rubyonrails.org/active_record_querying.html">https://guides.rubyonrails.org/active_record_querying.html</a>
마이그레이션	<a href="https://kbs4674.tistory.com/160">https://kbs4674.tistory.com/160</a>
FK를 어느 쪽에 둘 것인가	<a href="https://cskstory.tistory.com/entry/FK를-어느쪽예-둘지-결정하기">https://cskstory.tistory.com/entry/FK를-어느쪽예-둘지-결정하기</a>
정규화 개념 설명	<a href="https://wkdtjsgur100.github.io/database-normalization/">https://wkdtjsgur100.github.io/database-normalization/</a>
source 연관 옵션	<a href="https://stackoverflow.com/questions/4632408/understanding-source-option-of-has-one-has-many-through-of-rails">https://stackoverflow.com/questions/4632408/understanding-source-option-of-has-one-has-many-through-of-rails</a>
has_many 연관과 source 옵션	<a href="https://stackoverflow.com/questions/61430073/please-explain-the-has-many-through-source-rails-association">https://stackoverflow.com/questions/61430073/please-explain-the-has-many-through-source-rails-association</a>
ActiveRecord 예약어	<a href="https://stackoverflow.com/questions/13750009/reserved-names-with-activerecord-models">https://stackoverflow.com/questions/13750009/reserved-names-with-activerecord-models</a>



커스터마이징 정렬	<a href="https://stackoverflow.com/questions/23620615/how-to-sort-activerecord-query-by-specific-priority">https://stackoverflow.com/questions/23620615/how-to-sort-activerecord-query-by-specific-priority</a>
낙관적 vs 비관적 동시제어	<a href="http://wiki.gurubee.net/pages/viewpage.action?pageId=21626883">http://wiki.gurubee.net/pages/viewpage.action?pageId=21626883</a>
낙관적 동시제어 실습	<a href="https://gorails.com/episodes/optimistic-locking-with-rails?autoplay=1">https://gorails.com/episodes/optimistic-locking-with-rails?autoplay=1</a>
테이블 컬럼명 변경	<a href="https://edgeapi.rubyonrails.org/classes/ActiveRecord/ConnectionAdapters/SchemaStatements.html#method-i-change_column_null">https://edgeapi.rubyonrails.org/classes/ActiveRecord/ConnectionAdapters/SchemaStatements.html#method-i-change_column_null</a>
테이블 컬럼 디폴트 값 변경	<a href="https://edgeapi.rubyonrails.org/classes/ActiveRecord/ConnectionAdapters/SchemaStatements.html#method-i-change_column_default">https://edgeapi.rubyonrails.org/classes/ActiveRecord/ConnectionAdapters/SchemaStatements.html#method-i-change_column_default</a>
자신과의 has_many 연관	<a href="https://stackoverflow.com/questions/6762919/ruby-on-rails-has-many-through-self-referential-following-follower-relationships">https://stackoverflow.com/questions/6762919/ruby-on-rails-has-many-through-self-referential-following-follower-relationships</a>
트랜잭션과 락	<a href="https://stackoverflow.com/questions/20819403/how-do-i-ensure-a-model-always-uses-a-transaction-and-locks-in-rails">https://stackoverflow.com/questions/20819403/how-do-i-ensure-a-model-always-uses-a-transaction-and-locks-in-rails</a>
validation 시점	<a href="https://stackoverflow.com/questions/32053927/validating-time-in-rails">https://stackoverflow.com/questions/32053927/validating-time-in-rails</a>
동시성 제어	<a href="https://stackoverflow.com/questions/20819403/how-do-i-ensure-a-model-always-uses-a-transaction-and-locks-in-rails">https://stackoverflow.com/questions/20819403/how-do-i-ensure-a-model-always-uses-a-transaction-and-locks-in-rails</a>

## Javascript

자바스크립트 레퍼런스	<a href="https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference">https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference</a>
ES6 설명	<a href="https://takeuu.tistory.com/93?category=737612">https://takeuu.tistory.com/93?category=737612</a>
생성자와 프로토타입	<a href="https://www.zerocho.com/category/JavaScript/post/573c2acf91575c17008ad2fc">https://www.zerocho.com/category/JavaScript/post/573c2acf91575c17008ad2fc</a>
Prototypal 상속	<a href="https://www.javascripttutorial.net/javascript-">https://www.javascripttutorial.net/javascript-</a>

strict mode	<a href="https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Strict_mode">https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Strict_mode</a>
에어비앤비 가이드	<a href="https://github.com/airbnb/javascript">https://github.com/airbnb/javascript</a>
DOM 요소 탐색(vanila)	<a href="https://ko.javascript.info/searching-elements-dom">https://ko.javascript.info/searching-elements-dom</a>
jquery event list	<a href="https://api.jquery.com/category/events/">https://api.jquery.com/category/events/</a>
jquery selector filter	<a href="https://dololak.tistory.com/394">https://dololak.tistory.com/394</a>
file 객체에 대한 설명	<a href="https://taeny.dev/javascript/file-object/">https://taeny.dev/javascript/file-object/</a>
정규표현식	<a href="https://flaviocopes.com/javascript-regular-expressions/">https://flaviocopes.com/javascript-regular-expressions/</a>
object의 key 체크	<a href="https://stackoverflow.com/questions/455338/how-do-i-check-if-an-object-has-a-key-in-javascript">https://stackoverflow.com/questions/455338/how-do-i-check-if-an-object-has-a-key-in-javascript</a>
모던 자바스크립트 튜토리얼	<a href="https://ko.javascript.info/intro">https://ko.javascript.info/intro</a>
프로토타입의 이해	<a href="https://medium.com/@bluesh55/javascript-prototype-이해하기-f8e67c286b67">https://medium.com/@bluesh55/javascript-prototype-이해하기-f8e67c286b67</a>
참조에 의한 객체 복사	<a href="https://ko.javascript.info/object-copy#ref-265">https://ko.javascript.info/object-copy#ref-265</a>
ECMA6 호환 테이블	<a href="https://kangax.github.io/compat-table/es6/">https://kangax.github.io/compat-table/es6/</a>
배열과 reduce 메서드 활용	<a href="https://medium.com/@hongkevin/js-3-자바스크립트-배열-메서드-reduce-100-활용법-feat-egghead-io-97c679857ece">https://medium.com/@hongkevin/js-3-자바스크립트-배열-메서드-reduce-100-활용법-feat-egghead-io-97c679857ece</a>
자바스크립트와 액션케이בל	<a href="https://medium.com/@valentinowong/using-rails-action-cable-with-a-vanilla-javascript-front-end-1e00ed90067e">https://medium.com/@valentinowong/using-rails-action-cable-with-a-vanilla-javascript-front-end-1e00ed90067e</a>
Promise 설명	<a href="https://programmingsummaries.tistory.com/325">https://programmingsummaries.tistory.com/325</a>
로컬 스토리지	<a href="https://blog.logrocket.com/localstorage-javascript-complete-guide/">https://blog.logrocket.com/localstorage-javascript-complete-guide/</a>
방향키 입력 감지	<a href="https://stackoverflow.com/questions/5597060/detecting-arrow-key-presses-in-javascript">https://stackoverflow.com/questions/5597060/detecting-arrow-key-presses-in-javascript</a>
자바스크립트로 pong 구현	<a href="https://www.youtube.com/watch?v=KApAJhkkqkA">https://www.youtube.com/watch?v=KApAJhkkqkA</a>
data-attribute 세팅	<a href="https://stackoverflow.com/questions/13524107/how-to-set-data-attributes-in-html-elements">https://stackoverflow.com/questions/13524107/how-to-set-data-attributes-in-html-elements</a>
쿠키 사용방법	<a href="https://thereclub.tistory.com/59">https://thereclub.tistory.com/59</a>
SPA 초기 로딩속도 개선	<a href="https://medium.com/little-big-programming/spa-초기-로딩-속도-개선하기-9db137d25566">https://medium.com/little-big-programming/spa-초기-로딩-속도-개선하기-9db137d25566</a>

콜백, 프로미스 강의

<https://www.youtube.com/watch?v=s1vpVCrT8f4&list=PLv2d7VI9OotTVOL4QmPfvJWPJvkmv6h-2&index=11>

import & export

[https://www.youtube.com/watch?v=ohvckEuyZ\\_s](https://www.youtube.com/watch?v=ohvckEuyZ_s)

중복 이벤트 방지(1)

<https://hyeonguj.github.io/2020/02/27/double-click-problem-javascript/>

중복 이벤트 방지(2)

<https://stackoverflow.com/questions/5497073/how-to-differentiate-single-click-event-and-double-click-event>

중복 이벤트 방지(3)

<https://m.blog.naver.com/PostView.nhn?blogId=psj9102&logNo=221282045518&proxyReferer=https:%2F%2Fwww.google.com%2F>

## Backbone

백본JS 튜토리얼(1)

<https://auth0.com/blog/backbonejs-getting-started/>

백본JS 튜토리얼(2)

<https://github.com/jashkenas/backbone/wiki/Backbone%2C-The-Primer>

백본JS 튜토리얼(3)

<https://adrianmejia.com/backbone-dot-js-for-absolute-beginners-getting-started/>

백본JS 튜토리얼(4)

<https://www.jamesyu.org/2011/01/27/cloudedit-a-backbone-js-tutorial-by-example/>

백본JS 튜토리얼(5)

<https://www.tutorialspoint.com/backbonejs/index.htm>

백본JS 튜토리얼(6)

<https://thoughtbot.com/upcase/hands-on-backbone-js-on-rails>

백본JS 튜토리얼(7)

<https://www.youtube.com/watch?v=HOAU-nfy5Sc&list=PLT9miexWCpPUfPUGeQUMXQ9WS9rrbxI2D>

백본JS 기초 정리

<https://becommingprogrammer.wordpress.com/2020/07/20/backbone-js-정리-1-기초/>

TODO MVC 예제(1)

<https://backbonejs.org/docs/todos.html>

TODO MVC 예제(2)

<http://todomvc.com/>

백본 source

<https://backbonejs.org/docs/backbone.html>

백본 architecture

<https://hub.packtpub.com/architecture-backbone>

레이즈 + 백본 강의	<a href="https://thoughtbot.com/upcase/videos/basic-setup-and-coffeescript">https://thoughtbot.com/upcase/videos/basic-setup-and-coffeescript</a>
레이즈 + 백본 예제	<a href="https://github.com/thoughtbot/hands-on-backbone-js-on-rails-solution">https://github.com/thoughtbot/hands-on-backbone-js-on-rails-solution</a>
SPA 정의	<a href="https://en.wikipedia.org/wiki/Singlepage_application">https://en.wikipedia.org/wiki/Singlepage_application</a>
view render시 el의 역할	<a href="https://stackoverflow.com/questions/13659701/what-does-the-el-do-in-view-render-el">https://stackoverflow.com/questions/13659701/what-does-the-el-do-in-view-render-el</a>
새 경로 navigate	<a href="https://stackoverflow.com/questions/24009247/backbone-spa-best-practice-for-loading-new-routes">https://stackoverflow.com/questions/24009247/backbone-spa-best-practice-for-loading-new-routes</a>
일반적인 실수 8가지	<a href="https://www.toptal.com/backbone-js/top-8-common-backbone-js-developer-mistakes">https://www.toptal.com/backbone-js/top-8-common-backbone-js-developer-mistakes</a>
Beginner's mistake	<a href="https://xebia.com/blog/8873/">https://xebia.com/blog/8873/</a>
CSRF 방어	<a href="https://stackoverflow.com/questions/18124870/how-to-protect-against-csrf-when-using-backbone-js-to-post-data">https://stackoverflow.com/questions/18124870/how-to-protect-against-csrf-when-using-backbone-js-to-post-data</a>
View 렌더링에 대한 고찰	<a href="https://ianstormtaylor.com/rendering-views-in-backbonejs-isnt-always-simple/">https://ianstormtaylor.com/rendering-views-in-backbonejs-isnt-always-simple/</a>
Child Views 렌더링(1)	<a href="https://ianstormtaylor.com/assigning-backbone-subviews-made-even-cleaner/">https://ianstormtaylor.com/assigning-backbone-subviews-made-even-cleaner/</a>
Child Views 렌더링(2)	<a href="https://stackoverflow.com/questions/9271507/how-to-render-and-append-sub-views-in-backbone-js">https://stackoverflow.com/questions/9271507/how-to-render-and-append-sub-views-in-backbone-js</a>
Child Views 렌더링(3)	<a href="https://stackoverflow.com/questions/9337927/how-to-handle-initializing-and-rendering-subviews-in-backbone-js">https://stackoverflow.com/questions/9337927/how-to-handle-initializing-and-rendering-subviews-in-backbone-js</a>
뷰 제거	<a href="https://stackoverflow.com/questions/9079491/cleaning-views-with-backbone-js">https://stackoverflow.com/questions/9079491/cleaning-views-with-backbone-js</a>
Tip and Pattern	<a href="https://www.smashingmagazine.com/2013/08/backbone-js-tips-patterns/">https://www.smashingmagazine.com/2013/08/backbone-js-tips-patterns/</a>
백본JS를 레일즈 뷰로 사용하기	<a href="https://viblo.asia/p/using-backbonejs-as-rails-view-1VgZvEwpKAw">https://viblo.asia/p/using-backbonejs-as-rails-view-1VgZvEwpKAw</a>
백본 sync에 header 추가	<a href="https://stackoverflow.com/questions/10081728/add-request-header-on-backbone">https://stackoverflow.com/questions/10081728/add-request-header-on-backbone</a>

이벤트에 컨텍스트 전달	<a href="https://stackoverflow.com/questions/44869753/passing-context-of-view-on-event-in-backbone-js">https://stackoverflow.com/questions/44869753/passing-context-of-view-on-event-in-backbone-js</a>
listenTo vs on 비교	<a href="https://stackoverflow.com/questions/16823746/backbone-js-listento-vs-on">https://stackoverflow.com/questions/16823746/backbone-js-listento-vs-on</a>
history.start 메서드	<a href="https://stackoverflow.com/questions/19947281/where-to-call-backbone-history-start">https://stackoverflow.com/questions/19947281/where-to-call-backbone-history-start</a>
ejs 사용	<a href="https://stackoverflow.com/questions/40487162/what-is-the-meaning-of-in-ruby-on-rails">https://stackoverflow.com/questions/40487162/what-is-the-meaning-of-in-ruby-on-rails</a>
컬렉션 쿼리(1)	<a href="https://stackoverflow.com/questions/12315973/fetch-backbone-collection-with-search-parameters">https://stackoverflow.com/questions/12315973/fetch-backbone-collection-with-search-parameters</a>
컬렉션 쿼리(2)	<a href="https://stackoverflow.com/questions/6659283/backbone-js-fetch-with-parameters">https://stackoverflow.com/questions/6659283/backbone-js-fetch-with-parameters</a>
컬렉션 URL 동적 생성	<a href="https://stackoverflow.com/questions/6263539/how-to-set-a-collections-url">https://stackoverflow.com/questions/6263539/how-to-set-a-collections-url</a> Stack Overflow
컬렉션에서 모델 제거하기	<a href="https://stackoverflow.com/questions/15068036/remove-model-in-collection-and-fire-remove-event-backbone-js">https://stackoverflow.com/questions/15068036/remove-model-in-collection-and-fire-remove-event-backbone-js</a>
underscore의 template	<a href="https://stackoverflow.com/questions/11893966/underscore-js-template-if-check-on-a-variable/11894563">https://stackoverflow.com/questions/11893966/underscore-js-template-if-check-on-a-variable/11894563</a>
서브 뷰에서 부모 뷰 가져오기	<a href="https://stackoverflow.com/questions/19398530/backbone-js-access-parent-view-from-child-view">https://stackoverflow.com/questions/19398530/backbone-js-access-parent-view-from-child-view</a>
sync 과정	<a href="https://ohgyun.com/458">https://ohgyun.com/458</a>
클릭한 위치에 레이어 띄우기	<a href="https://zzznara2.tistory.com/621">https://zzznara2.tistory.com/621</a>
윈도우 객체와 이벤트 연결	<a href="https://stackoverflow.com/questions/9110060/how-do-i-add-a-resize-event-to-the-window-in-a-view-using-backbone/9110145">https://stackoverflow.com/questions/9110060/how-do-i-add-a-resize-event-to-the-window-in-a-view-using-backbone/9110145</a>
fetch success callback	<a href="https://stackoverflow.com/questions/19713613/backbone-fetch-callback-the-proper-way">https://stackoverflow.com/questions/19713613/backbone-fetch-callback-the-proper-way</a>
router querystring 파싱	<a href="https://stackoverflow.com/questions/11671400/navigate-route-with-querystring">https://stackoverflow.com/questions/11671400/navigate-route-with-querystring</a>

## CSS

CSS 선택자	<a href="https://www.w3schools.com/cssref/css_selectors.asp">https://www.w3schools.com/cssref/css_selectors.asp</a>
Complete Guide to Flexbox	<a href="https://css-tricks.com/snippets/css/a-guide-to-flexbox/">https://css-tricks.com/snippets/css/a-guide-to-flexbox/</a>
그리드 5분만에 배우기	<a href="https://ibrahimovic.tistory.com/23">https://ibrahimovic.tistory.com/23</a>
그리드 완벽 가이드	<a href="https://heropy.blog/2019/08/17/css-grid/">https://heropy.blog/2019/08/17/css-grid/</a>
1분 코딩 CSS Grid	<a href="https://studiomeal.com/archives/533">https://studiomeal.com/archives/533</a>
반응형 웹의 미디어 쿼리	<a href="https://velog.io/@bungouk6829">https://velog.io/@bungouk6829</a>
How to create toggle switch	<a href="https://www.w3schools.com/howto/howto_css_switch.asp">https://www.w3schools.com/howto/howto_css_switch.asp</a>
HTML5/CSS 지원 테이블	<a href="https://caniuse.com/">https://caniuse.com/</a>

## Restful API

API 응답 가이드	<a href="https://gist.github.com/subicura/8329767">https://gist.github.com/subicura/8329767</a>
REST API 튜토리얼	<a href="https://www.restapitutorial.com/">https://www.restapitutorial.com/</a>
DELTE 요청과 query	<a href="https://stackoverflow.com/questions/34939484/in-rest-api-can-delete-methods-have-parameters">https://stackoverflow.com/questions/34939484/in-rest-api-can-delete-methods-have-parameters</a>
API 관리 도구 모음	<a href="https://www.ciokorea.com/news/37168?page=0,1">https://www.ciokorea.com/news/37168?page=0,1</a>
SPA와 라우팅	<a href="https://poiemaweb.com/js-spa">https://poiemaweb.com/js-spa</a>

## Environment

ESLint, Prettier Setup(1)	<a href="https://gist.github.com/bradtraversy/">https://gist.github.com/bradtraversy/</a>
ESLint, Prettier Setup(2)	<a href="https://velog.io/@_jouz_ryul/ESLint-Prettier-Airbnb-Style-Guide로-설정하기">https://velog.io/@_jouz_ryul/ESLint-Prettier-Airbnb-Style-Guide로-설정하기</a>
VSCode ESLint 설정(1)	<a href="https://steadyzest.tistory.com/20">https://steadyzest.tistory.com/20</a>
VSCode ESLint 설정(2)	<a href="https://blog.echobind.com/integrating-prettier-eslint-airbnb-style-guide-in-vscode-47f07b5d7d6a">https://blog.echobind.com/integrating-prettier-eslint-airbnb-style-guide-in-vscode-47f07b5d7d6a</a>
레일즈 6의 웹패커 이해	<a href="https://prathamesh.tech/2019/08/26/understanding-webpacker-in-rails-6/">https://prathamesh.tech/2019/08/26/understanding-webpacker-in-rails-6/</a>
Webpack Performance(1)	<a href="https://webpack.js.org/guides/build-performance/">https://webpack.js.org/guides/build-performance/</a>
Webpack Performance(2)	

Webpack Performance(3)	<a href="https://slack.engineering/keep-webpack-fast-a-field-guide-for-better-build-performance/">https://slack.engineering/keep-webpack-fast-a-field-guide-for-better-build-performance/</a>
webpacker not found error	<a href="https://github.com/stephencookdev/speed-measure-webpack-plugin">https://github.com/stephencookdev/speed-measure-webpack-plugin</a>
package.json	<a href="https://stackoverflow.com/questions/57891751/webpacker-configuration-file-not-found-rails-6-0-0">https://stackoverflow.com/questions/57891751/webpacker-configuration-file-not-found-rails-6-0-0</a>
bundler practice	<a href="https://programmingsummaries.tistory.com/385">https://programmingsummaries.tistory.com/385</a> <a href="https://www.viget.com/articles/bundler-best-practices/">https://www.viget.com/articles/bundler-best-practices/</a>

## General

42 API	<a href="https://api.intra.42.fr/apidoc/guides/web_application_flow">https://api.intra.42.fr/apidoc/guides/web_application_flow</a>
객체지향 개념 요약	<a href="https://jongmin92.github.io/2019/02/10/Programming/object_oriented_facts_and_misunderstandings/">https://jongmin92.github.io/2019/02/10/Programming/object_oriented_facts_and_misunderstandings/</a>
객체지향 학습 참고도서 5	<a href="https://javarevisited.blogspot.com/2017/04/top-5-books-to-learn-object-oriented-programming.html#axzz6iyAEVI3E">https://javarevisited.blogspot.com/2017/04/top-5-books-to-learn-object-oriented-programming.html#axzz6iyAEVI3E</a>
CQRS 개념	<a href="https://www.popit.kr/cqrs-eventsourcing/">https://www.popit.kr/cqrs-eventsourcing/</a>
Oauth 소개	<a href="https://velog.io/@sonypark/OAuth2-인증">https://velog.io/@sonypark/OAuth2-인증</a>
MVC, MVP, MVVM 차이	<a href="https://beomy.tistory.com/43">https://beomy.tistory.com/43</a>
HTTP부터 웹소켓까지	<a href="https://medium.com/@chullino/http에서부터-websocket까지-94df91988788">https://medium.com/@chullino/http에서부터-websocket까지-94df91988788</a>
github reviewer vs assignee	<a href="https://stackoverflow.com/questions/41087206/on-github-whats-the-difference-between-reviewer-and-assignee">https://stackoverflow.com/questions/41087206/on-github-whats-the-difference-between-reviewer-and-assignee</a>
백엔드 인사이트	<a href="https://velog.io/@city7310/백엔드가-이정도는-해줘야-함-1.-컨텐츠의-동기와-개요">https://velog.io/@city7310/백엔드가-이정도는-해줘야-함-1.-컨텐츠의-동기와-개요</a>
pong 게임 구현시 주의사항(1)	<a href="https://blog.ifunfactory.com/2016/08/02/사례-연구-pong을-만들어보자/">https://blog.ifunfactory.com/2016/08/02/사례-연구-pong을-만들어보자/</a>
pong 게임 구현시 주의사항(2)	<a href="http://www.greened.kr/news/articleView.html?idxno=56447">http://www.greened.kr/news/articleView.html?idxno=56447</a>

## 부록 4-1 팀원 iwoo 인터뷰 | 42를 받아들이는 것

### 개발을 시작하기 전에 어떤 사람이었는지?

경영학과 출신의, 개발보단 재무와 전략에 관심이 많던 사람이었어요. 살아오면서 생존에 대한 불안이 늘 있었는데, 내가 몸담고 있는 조직이 사라지거나 갑자기 나를 자르더라도 잘 살아남을 수 있는, 자립적인 역량을 지닌 사람이 되고 싶다고 생각해왔어요.

그래서 전문적인 자격증을 따기 위해 회계사를 도전해보기도 했고, '안 되던 것들을 되게 만드는' 창업적 역량을 배워보려고 빠르게 성장하는 스타트업에서 일해보기도 했어요. 영어 회화를 가르쳐주는 사람, 배우려는 사람을 매칭하고 장소를 제공하는 플랫폼이었는데요. 스터디 장소를 관리하는 일을 주로 했어요. 장소 제공 업체와의 비용 협상, 예약 관리와 정산, 그리고 비용을 줄이기 위한 기획 등이 주요 업무였죠.

스타트업이다보니 CS 업무도 같이 해야 할 때가 많았고, 고객이나 파트너들과 커뮤니케이션하는 일이 많은데 그게 에너지가 들었어요. 목표로 했던 '스스로 살아남을 수 있는 역량'을 잘 달성해가고 있느냐라는 질문에 대해서도 흡족하지는 않았고요. 일을 하면서 스스로를 관찰했을 때 사람들과 협상하는 일보다는 비용을 줄이거나 수식을 다루는 일이 더 맞았어요. 그러다 보니 자립적 역량이 큰 직업이라고 생각해온 개발자로의 커리어 전환을 고려하게 되었어요. 학생 때부터 개발이 나랑 맞을 것 같다는 생각도 머리 한쪽에 있기도 했고요. 창업에 필요한 인재들을 보통 3H라고 하는데, 막상 스타트업에 들어가 보니 허슬러는 나랑 안 맞고, 힙스터는 더더욱 아닌데, 해커는 맞지 않을까 싶기도 했어요(웃음).

결혼했고, 가정이 있기 때문에 아내와 대화가 필요했어요. 다행히 몽골에 여행을 갔을 때 프루공이라는 자동차 안에서 창밖을 안 보고 5시간 동안 스도쿠만 풀고 있었는데, 그런 걸 보면서 아내가 '이 남자는 개발을 해야 했는데'라고 생각했었다고 하더라고요(웃음). 평소에 보여준 이런 면들 때문에 설득하기가 어렵지 않았어요. 노마드 코더라는 서비스를 통해 클론 코딩을 해보면서 내가 재미있고, 잘할 수 있는지 스스로 검증했을 때에도 성공적이었고요. 재미있고, 결과물이 빠르게 나오고, 성향도 INTP여서 여러모로 내가 개발자



랑 잘 맞겠다는 판단이 들었어요. 그래서 퇴사 후 개발자를 준비하게 되었죠.

### 퇴사하고 바로 42로?

퇴사하고 바로 온 건 아니고요. 42를 알고 퇴사를 한 게 아니었기 때문에 42 공고를 보기 전까지 2개월 정도의 텀이 있었어요. 개발을 어떻게 시작해야 할까 고민했고, 국비학원이나 부트캠프를 알아봤죠. 돈을 많이 들이더라도 좋은 개발자가 될 수 있다면 괜찮다고 생각해서 부트캠프 쪽으로 더 알아봤었어요. 아는 형도 부트캠프를 추천하면서 돈을 빌려줄 수 있다고 했고(웃음).

그러다가 지인을 통해 42서울을 알게 되었죠. 내가 운 좋게 아웃스탠딩한 사람이라고 치더라도 3~4개월 공부하는 것만으로는 룽런할 수 있는 깊이를 다지기 힘들 거라 생각하던 차에, 42서울은 매력적인 교육방식과 지원금 혜택이 있으니 오랫동안 나의 기초를 근본 있게 제대로 다질 수 있겠다는 기대가 생겼어요. 그래서 42서울에 도전하게 되었죠.

### 지난 1년은 어떠셨나요?

의식적으로 42에 집중했어요. 클러스터가 열려있는 동안에는 거의 매일 클러스터를 갔어요. 일주일일 있다면 내가 하고 싶은 공부를 하는 하루, 아내와 시간을 보내는 하루를 제외한 시간을 거의 42에 썼어요.

'개발' 안에서도 분야가 다양하게 나뉘는데, 내가 어떤 분야를 선택해서 더 깊게 팔지 답을 찾아가는 1년이었어요. 42에선 실습과 경험 중심의 학습이 가능하니 다양한 경험을 통해 해당 분야와 나의 fit을 검증해볼 수 있길 바랐죠. 실제로 cub3D를 하면서 나는 그래픽은 아니구나, 게임은 만드는 것보다는 즐기는 쪽이구나 하고 생각하게 됐고요(웃음). 쿠버네티스를 할 때는 한창 클라우드에 관심이 많아서 IBM 클라우드스에도 가입했었는데, 인프라가 생각보다 어렵고 코드 치는 게 당장의 나에게 더 재밌다고 느꼈어요. 또 자연어처리 관련 공모전을 통해 머신러닝이 정말 매력적이지만 목표한 기간에 취업이 가능할 만큼 역량 쌓기는 힘들겠다고 스스로 판단할 수 있었어요. 해커톤에서는 라즈베리 파이로 회로를 조립하는 프로젝트를 했는데, 임베디드나 iOS가 나에게 맞는지 고민해 볼 수 있는 시

간이었죠. 결국 마지막 서브젝트인 트랜센던스를 통해 웹 프론트엔드와 백엔드를 경험하면서 백엔드 개발자로 방향을 잡게 되었네요.

다만 세상을 바꾸고 있는 힘찬 기술들이 많은데, 내가 너무 기초만 파고 있는 게 아니냐라는 걱정이 있었어요. 매일 눈을 뜬 이후부터 잠들 때까지 개발 역량을 쌓으려고 노력했는데, 그런데도 당장 만들 수 있는 게 없어서 불안하더라고요. 한국에서 취업에 중요하게 여겨지는 기술이랑 42의 기술 스택도 많이 다르잖아요. 내가 열심히 노력한 시간이 인정받지 못 할까 봐 스트레스를 받았어요. 처음 선택하려고 했던 부트캠프와 계속 비교하게 되고, 현업으로 바로 가서 배우는 게 낫지 않느냐는 고민도 하게 되고요. 이 길이 정말 나에게 맞는 건지 의심도 들었죠. 그런데도 개발의 근본을 다지는 42의 가치, 이 길이 맞는 길 인지가 중요한 게 아니라 맞는 길이 되도록 하겠다는 다짐을 되새기면서 멘탈 관리를 해왔던 것 같아요.

총평하자면, 행복했어요. 오설록 녹차, 쿠키, lo-fi 음악과 함께 코딩하면 행복한 감정을 느꼈어요. 지금도 42를 진행하는 동안 느꼈던 불안감이 사라지지는 않았지만, 기초가 기초인 이유는 그것이 중요하고 어떤 개발을 하든 필요한 기반 지식이기 때문이라고 생각해요. 면접 질문을 보다 보면 결국 기초를 물어보기 때문에 제대로 된 기초를 다지는 이 시간은 헛되지 않고, 그저 제가 너무 조급했던 게 아닌가 싶어요. 사실 2년을 생각했던 게 저의 초심이고, 공통 과정이 끝났으니 부족했던 기술과 지식은 남은 1년 동안 채우면 되는 거니까요.

#### **42에서 무엇이 가장 좋았나요?**

솔직히 말하면 지원금(웃음). 음, 좋은 동료들을 알게 된 게 좋았어요. 자극을 많이 받았어요. 저만 열심히 사는 게 아니라 다들 열심히 살잖아요. 이런저런 것을 만들어서 슬랙에 공유하면, 거기에서 자극을 받을 수 있는 게 좋았어요. 덕분에 의욕을 잃지 않고 잘 갈 수 있었던 것 같아요.

지원금과 동료 이외에 좋았던 것은 과제를 어느 정도 수준까지 구현할지 스스로 결정할

수 있다는 거였어요. 보너스 같은 경우 필수가 아니니까 관심에 따라 구현 범위를 조절할 수 있고, 또 깊게 학습을 해서 통과를 하고 싶다면 시간을 들여서 그렇게 해도 되는 게 좋았어요. 예를 들자면 philosopher인데, 3일이라고 권장 시간이 되어있지만 2주 정도 시간을 썼거든요. 서브젝트에 언급되어 있지 않더라도 더 꼼꼼하게 좋은 품질의 코드를 만들고 싶었어요. 학습에 자율성이 있는 건 42의 큰 장점이라고 생각해요.

### 42에서 아쉬웠던 부분은?

코로나로 인해서 원격 동료평가를 해야 하는데, 정성적으로 오프라인으로 동료평가를 할 때보다 경험의 질이 떨어지는 부분이 있어서 아쉬웠어요. 물론 의욕 있는 분을 만나면 15분할 평가도 1시간을 진행하지만, 전체적으로 이 시국과 42의 포맷이 안 맞는 것 같은 느낌이 있었어요. 사실 42라기보다 시국의 아쉬움이죠.

### 트렌센던스는 어땠나?

어려웠던 건 액션 케이블이예요. 웹소켓으로 브로드캐스팅되는 메시지를 관리하는 것이 가볍지 않았어요. 기술이 어렵다기보다 발생하는 예외가 어려웠어요. 실제 운영이랑 배포를 경험하지 못한 것, 실제 유저의 이용으로부터 만들어지는 트래픽을 감당해보는 경험을 하지 못한 것이 아쉬웠고요. 저희 케이스에 상속이 따로 필요 없어서 쓰지 않기는 했지만, 객체 지향을 공부하면서 상속을 잘 써보고 싶었는데 쓸 기회가 없었던 점도 아쉬웠어요. Gemfile.lock 과 관련한 환경 이슈가 계속 있었는데, 결국 해결하긴 했지만, 확실히 알고 해결한 느낌이 들지 않아서 그것도 아쉽습니다.

좋았던 점은 성장하는 데 도움이 되었다는 거겠죠. 그 이유는 좋은 팀이랑 협업할 수 있었기 때문인 것 같아요. 의욕이 넘치고, 학습에 대한 욕심이 있어서 아무도 무임승차하지 않았어요. 더 좋은 방식으로 코딩하기 위해서 팀원들이 노력을 아끼지 않았다고 생각해요. 팀원들의 코드를 볼 때 더 좋은 코드를 만들기 위한 노력이 늘 숨어있었고, 그런 걸 발견하면서 배우는 즐거움이 컸어요. PR 단위도 크게 느껴지니까 적정 PR 단위를 고민해서 브랜칭이나 PR 전략을 변경하던 순간도 인상 깊어요.

또 이전에 프론트에 대한 개념을 많이 찾아봐서 알고는 있었지만, 몸으로 느끼지는 못했었어요. 우리가 역할을 fix하지 않고 학습을 위해 모든 역할을 경험하면서, 프론트와 백이 나뉘는 이유가 무엇이고 그 이점이 무엇인지를 몸으로 느낄 수 있어서 좋았어요. 특히 프론트와 백을 다 직접 해보면서 오랜 고민이었던 진로를 백엔드로 정할 수 있었고요. RSpec으로 스펙을 설계하고 test를 하나씩 통과할 때마다 게임을 하는 것 같은 쾌감이 있더라고요.

### 5명이 비대면으로 협업을 하는 과정이 힘들지는 않았나?

불편함을 크게 못 느꼈어요. 자연스러웠어요. 의식적으로 처음에 개발환경을 세팅하는 데 많은 시간을 쓴 데다가, 디스코드와 라이브쉐어를 잘 활용해서 정말 불편함 없이 협업했던 것 같아요. 비대면이 가지는 불편함이라고 하면 즉각적인 커뮤니케이션, 피드백이 어렵다는 점일 텐데 모두가 풀 커밋하고 있어서 그런 이슈도 없었고요. 이런 점이 늘 연결되어 있다는 불편함을 낳을 수 있는데, 디스코드에 오아시스나 옥상 같은 채널을 만들어서 휴식도 할 수 있게 환경을 세팅함으로써 잘 넘어갈 수 있었죠. 종일, 밤늦게까지 대화를 해야 하다 보니 굳이 불편함을 얘기한다면 아내에게 미안한 마음?

### 만약 트랜센던스를 처음부터 다시 한다면?

기본적으로 모든 시간이 다 필요했던 시간이었다고 생각하고, 도입을 못 하거나 늦어져서 아쉬웠던 점만 있어요.

포스트맨으로 요청하고 응답을 확인할 수 있어서 테스트케이스를 따로 만들지 않았는데, 코드를 변경하면 다시 확인해야 하는 비용이 있었죠. RSpec을 늦게 도입한 게 아쉬워요. 특히 모델이나 잡에는 테스트가 쌓였지만, 브라우저까지 포괄하는 통합 테스트 도입에는 실패했는데 이 점도 아쉬워요.

구현적인 관점에서는 더 SPA스러운 코드를 만들려고 할 것 같아요. 처음에 백엔드에서 프론트에 데이터를 가능한 한 다 넘겨주고, 그다음부터는 프론트에서 대부분의 요청을 처리하는 방식으로 구현해보려고 하지 않을까. 또 게임 화면과 채팅 화면을 한 화면에 놓는

방식으로, 게임을 하면서 채팅도 자연스럽게 가능하도록 시도해 볼 것 같아요.

### 어쩌다 트랜센던스까지 끝나게 되었는데?

트랜센던스를 시작할 때 백본, 루비, 레일즈 같은 기술 스택을 아무것도 몰랐지만 결국 해냈어요. 모르면 배워서 하면 된다는 걸 다시 한번 느꼈죠. 이름만 생각해보면 트랜센던스는 '초월'이라는 의미잖아요. 기초만 다지던 내가 상용 서비스를 만들기까지 깨야 할 벽 중 하나를 깨게 만든 서브젝트인 것 같아서 후련하고, 또 공통 서클을 빠르게 깬 속에 속하기 때문에 그것도 좋아요. 이제 벽 하나를 깬 거고, 깨야 할 벽은 더 많지만요.

팀원들한테 고마움이 많아요. 생각만 해도 한숨이 나온 에러들을 몇 시간씩 붙잡고 두려움 없이 부딪혔던, 커밋 메시지에도 신경을 써서 프라이드 있게 작업해줬던 동료들한테 고마움을 많이 느껴요. 집에서 아주 시끄러웠는데 배려해준 아내한테도 고맙고요.

### 이후 계획은?

운영과 배포에 대한 경험을 42에서 많이 하지 못했고, 학습을 목적으로만 어플리케이션을 만들었기 때문에 실제 유저가 있는 서비스를 만들어서 운영과 배포를 해보고 싶어요. 그 과정에서 개인 브랜딩도 해보고 싶네요.

백엔드로 진로를 생각하게 된 만큼, 그쪽으로도 더 파볼 생각입니다. 백엔드 개발자에게 필요한 DB 튜닝, 인프라적 지식, 백엔드 프레임워크, 디자인 패턴 같은 것을 중점적으로 학습해보려고 하고 있어요. 42 아우터 서클 중에 흥미로워 보이는 것들이 있어서, 시간이 되면 해보고도 싶어요. 물론 가정의 생계를 위해 올해에는 취직해야 하겠지만요.

### 이제 막 카뎃이 된 사람들에게 해주고 싶은 말이 있다면?

42의 컨셉은 3~6개월 만에 바로 현업에 투입 가능한 개발자를 만드는 게 아닌 것 같아요. 비전공인 사람도 와서 2년의 시간을 통해 컴퓨터 공학을 전공한 사람처럼 기초를 쌓고, 통런할 수 있게 도와주는 곳이라 생각합니다. 그 외의 것은 스스로 챙취해야 합니다.

42가 포커싱하지 않는 방향의 교육을 기대하고 오면 실망할 수 있어요. 42의 컨셉을 이해하고 받아들이면 편해지는 것 같아요. 42가 제공하는 것이 기대하는 것과 다르더라도 결국 다 필요한 지식이라고 생각해요. 혹시라도 제가 도울 수 있는 부분이 있다면 편하게 slack으로 DM 주세요

@iwoo

[github.com/humbleEgo](https://github.com/humbleEgo)

## 부록 4-2 팀원 sanam 인터뷰 | 트랜센던스로 발견한 포지션

### 42 이전의 삶은 어땠나?

금융을 공부했어요. 열심히 했고, 잘했어요. 원래 계획은 대학원에 가는 거였죠. 실무에서 중요하다고 말하는 자격증도 200만 원 넘는 돈을 들어서 땀어요. 학생 신분에서 진짜 큰 돈이었는데(웃음). 그렇게 대학원을 준비하다 원하는 대학원의 교수님과 연락이 닿아서, 어떤 걸 준비하면 좋겠냐고 여쭙봤어요. 코딩을 애기하셨어요. 금융도 손으로 계산을 하는 게 아니고, 프로그램을 쓰면 숫자를 다루기에 유리하다고요. 그런 이유로 4학년 때부터 컴퓨터공학 수업을 듣게 됐어요.

듣다 보니까 예전에 했던 금융 계열의 공부랑 너무 다르더라고요. 이전에는 책에 있던 내용만 공부해오고, '나중에 어떻게 써먹을 수 있을까?'라는 고민을 해왔어요. 사실 금융만이 아니라 모든 공부에서 평생 그랬죠. 그런 와중에 C언어 수업을 듣다가 가위바위보 과제를 했는데 그게 너무 재밌더라고요. 너무 기초적인 거였지만, 그걸 만들자마자 컴퓨터 공학과로 진학한 고등학교 친구한테 바로 문자를 보냈어요. 이렇게 재밌는 걸 왜 혼자만 했냐고. 공부가 재밌다고 느끼면서 4학년 동안 컴퓨터와 관련된 수업을 들었어요.

열심히 금융을 공부해왔지만, 미래가 잘 보이지 않고 회사의 부품이 되는 미래만 잘 그려졌어요. 어느 순간 회사에서 나가라고 하면 짐을 싸야 하고, 나가서 능동적으로 할 수 있는 게 없는 그런 미래가 그려졌어요. 그런 와중에 코딩이 재미있고, 이 계열도 전망이 유망하고, 컴퓨터 공학과 교수님도 잘 아껴주시고, 성적도 잘 나와서 그렇게 마음을 바꾸게 됐어요. 이 길로 가야겠다.

그러다 어느 날 OKKY라는 사이트에 들어갔는데, 공부만 하면 100만 원을 준다는 광고가 보였어요. 바로 걸심했죠. 나는 여기 가야겠구나. 2년간 제대로 공부도 할 수 있고, 돈도 주는구나. 그게 42였죠. 특별할 건 없었어요. 열심히 살아왔지만, 부품은 되길 싫었고 마침 기회가 온 거죠.

### 고민은 안 했나?

금융 쪽이지만 대학원에 가려고 했었으니까, 대학을 졸업하고 2년은 더 쓰려고 했어서 시간이 부담스럽지는 않았어요. OKKY라는 사이트를 자주 들어갔었기 때문에 국비지원과 싸피를 비교하는 글을 많이 봤었는데, 싸피는 시험을 봐야 해서 부담스러웠어요. 학교 수업에도 시간을 써야 했던 상황이었기 때문에.

### 4학년 2학기가 끝나고 바로 왔나?

운이 좋게도, 코로나가 터져서 4학년 2학과 42에서의 첫 3개월을 병행했어요. 마지막 학기라 수업도 적고 원격이라 시간에 여유가 있었거든요.

### 카뎡으로 1년을 어떻게 보냈는지?

그렇게 성실하지만은 않았던 것 같아요. 원래 제가 성실의 아이콘이었는데(웃음). 코로나 때문에 대부분 집에서 했잖아요. 변명일지는 모르겠지만, 집에서 하게 된 것 때문에 게을러진 것 같아요. 공부 시간이 목표했던 것보다 엄청나게 많지 않았어요. 일요일 빼고는 10~12시간 정도씩 공부는 했는데, 집에서 하다 보니까 유혹도 많고 집중을 잘 못해서 공부의 질이 떨어졌어요. 누구도 지켜보지 않으니까 스스로 통제하기도 힘들었고요. 그래서 중간중간 클러스터가 열릴 때마다 가서 집중하려고 했는데, 그 외의 시간에는 자신의 노력에 대해 아쉬운 부분이 있어요. 42는 소통이 중요한데, 그런 부분도 아쉬웠어요.

### 소통이 아쉬웠다고?

대학교에서 랩실에서 생활할 때는 오고 가는 사람들이랑 친하게 지냈었어요. 42 클러스터에서도 비슷한 생활을 기대했어요. 오가는 사람들이랑 많이 대화하고, 함께 코딩하면서 성장해야겠다는. 그런데 클러스터에 직접 가지 않고 원격으로만 하다 보니까, 사람들과 그런 커넥션이 만들어지지 않은 거죠. 커넥션이 일어날 수 있는 과정은 평가밖에 없는데, 제가 그런 부분에서 미숙해서였는지 평가를 통해서 사람들과 관계를 만들기가 생각보다 쉽지 않았어요.

### 무엇이 가장 좋았나?



앞으로 어떤 걸 해야겠다는 생각하게 된 것, 그게 가장 좋았어요. 원래는 임베디드를 하고 싶었어요. 기계를 만져보고 싶었고, 멘토님과 상담을 할 때도 그랬어요. 웹서버까지만 해도 웹은 생각도 안 했어요. 그런데 트랜센던스를 하면서 앞으로 프론트를 해야겠다는 방향을 정하게 되었고, 그게 42를 돌아볼 때 가장 좋은 점이예요. 프론트에 페이지를 만들고, 백엔드와 연결시키는 실제적인 서비스를 만들어보면서 자연스럽게 결심이 됐어요. 프론트를 더 공부해야겠다. 개발을 시작하면서 계속 고민을 해왔던 주제였는데, 트랜센던스를 하기 전까지도 그 고민이 풀리지 않던 게 가장 큰 스트레스였거든요. 그런데 마지막 과제인 트랜센던스를 하면서 이것이 좋은 기술이고, 이걸 잘한다면 나도 성공할 수 있겠다는 자신이 서더라고요.

### **왜 프론트였나?**

기술이 이렇게 많은 시대에서 풀스택을 꿈꾸는 것은 어불성설이고, 하나를 집중적으로 해야 할 것 같은데 눈에 보이는 작업을 한다는 게 프론트의 큰 매력이더라고요. 처음에는 편견이 있었어요. 프론트 개발자는 진정한 개발자가 아니라는 편견이요. 부끄러운 말이지만, 처음 개발 공부를 시작하면서 그런 편견이 저를 지배하고 있었어요. 웹서버를 할 때까지도 마찬가지였어요. 트랜센던스를 하면서 그런 편견이 와장창 깨졌죠. 기대하지도 않았던 게 엄청난 것으로 느껴지면 빠져버리는 게 있잖아요. 그런 게 저한테 프론트였어요. 아무것도 아니라고 생각했는데 막상 해보니까 너무 대단했던 거죠. 앞으로 몇십 년을 개발자로 살아야 할 텐데, 클라이언트 사이드의 스펙이 좋아지면서 점점 프론트엔드가 맡을 수 있는 책임이 커지겠다는 기대도 있었죠.

트랜센던스도 쉬운 프로젝트였다면 42에 실망했을 거예요. 웹서버까지도 실망을 하고 있었거든요. 그런데 그게 아니었고, 트랜센던스는 정말 고마운 프로젝트가 되었어요.

### **트랜센던스 얘기를 더 해달라.**

힘들었던 것부터 말씀을 드릴게요. 처음에 실시간 처리를 하기 위해 액션 케이블을 다룰 때 너무 큰 스트레스를 받았어요. 내 마음대로 되는 게 아무것도 없었어요. 구독하지도 않

있는데 왜 자동으로 구독이 되는 거야? 또 액션 케이블을 사용하면 프론트와 백엔드가 언제든지 먼저 메시지를 줄 수 있는 양방향 통신을 할 수 있게 되는데, 이게 제가 알고 있는 상식선에서 와닿지 않았어요. 처음에는 백엔드에서 먼저 프론트로 메시지를 줄 수 있게 하는 기술이라고 생각했는데, 양방향으로 메시지를 주고받을 수 있고 그런 사실에서 파생되는 개념과 현상들을 이해하는 게 가장 힘들었어요.

가장 좋았던 건 '어피어런스 뷰'를 작업할 때였어요. 사람들이 접속할 때, 게임 중으로 상태가 변할 때, 로그아웃했을 때 사람들의 접속 상태를 변경하는 실시간 처리를 하는 작업이었어요. 모델을 만들고, 모델에 이벤트 핸들러를 등록해서 처리하는 일련의 과정들이 좋았어요.

#### **5명이 비대면으로 협업하는 게 힘들지는 않았나?**

협업이란 게 어려운 거잖아요. 저는 상대적으로 팔로우하는 입장에서 참여했는데, 제가 생각했을 때는 잘 된 협업이었다고 생각해요. 오히려 대면이 아니라 온라인이었기 때문에 잘 되었다는 생각도 들어요. 서로가 언제든지 온라인으로 만날 수 있었고, 또 얼굴을 보지 않았기 때문에 더 편하게 대화를 할 수 있었던 것 같아요. eunhkim님이 잘 리딩하고 iwoo님이 잘 기록하는 등 커뮤니케이션이 전반적으로 원활했던 것 같아요. 나중에 제가 팀을 꾸려나간다고 하면 막막할 것 같아 걱정이네요(웃음).

#### **만약 트랜센던스를 처음부터 다시 해야 한다면?**

학습 시간이 길었던 것 같아요. 제 문제일지는 모르겠는데, 학습하라고 하면 허송세월하게 되는 느낌이 있었어요. 자바스크립트, 백본, 루비 등 45일간의 학습 기간에 얻은 게 많지 않은 느낌이에요. 학습 기간을 줄이고 নিজ 프로젝트를 늘렸을 것 같아요.

#### **설계나 구현을 바로 들어가는 게 아니라 নিজ을 늘리자고?**

아예 모르는 상황에서 설계하고 구현을 한다는 건 많이 어려운 것 같아요. 제대로 된 웹페이지를 만들어 본 적이 없는 상황에서, 또 SPA라는 새로운 형태의 애플리케이션을 바로 구현한다는 건 말이 안 된다고 생각해요. 그러니까 학습 기간을 줄인다기보다 নিজ 프로

젝트를 통해 실습을 거쳐 학습하는 방향으로 가는 게 맞는 것 같아요.

### 설계나 구현 단계에서 달라지는 게 있을까?

설계는 저희가 했던 것으로도 만족스러워요. 구현 단계에서는 할 얘기가 많은데요. 서브젝트에서 요구한 바는 이미 모두 넘어섰다고 생각하고, 서브젝트와 무관하게 SPA는 빠른 속도, 즉각적인 반응을 추구해야 한다고 생각해요. 그런데 저희 서비스를 보면 어피어런스 뷰에서만 그런 느낌을 많이 받아요.

다시 돌아간다면 모델과 컬렉션, 액티브 잡, 액션 케이블을 훨씬 적극적으로 이용해서 서비스 전체적으로 실시간 연동에 신경을 많이 쓸 것 같아요. 유저가 새로고침 버튼을 누르거나 페이지에 다시 접근할 필요 없이, 특정한 페이지를 렌더링하는 데 필요한 데이터의 상태가 변경될 때마다 실시간으로 해당 뷰를 다시 렌더링하는 거죠. 앱 전체에서요.

위 모듈을 작업할 때에도, 워타임에서 일어나고 있는 전투 상태는 실시간으로 계속 변경을 하면서도 전쟁 현황에 대한 정보는 실시간 렌더링을 하지 못한 게 너무 후회가 돼요. 다이렉트 메시지 역시 해당 유저와의 대화창을 켜지만 도착한 메시지가 있다는 걸 확인할 수 있는데, 열지 않아도 새로운 메시지나 읽지 않은 메시지를 확인할 수 있게 처리를 했다면 좋았을 것 같고요.

마지막으로 랭킹 역시도 지금은 유저가 요청할 때마다 인덱스를 계산하는 방식인데, 10분이나 30분 단위로 주기적으로 job을 통해 랭킹을 서버 사이드에서 업데이트하고, 프론트로 전송해서 정보를 업데이트하는 것도 해보면 좋겠어요.

### 어쩌다 트랜센던스까지 끝나게 되었는데?

마지막 과제는 괜찮았지만, 마지막 과제까지 버티는 과정이 험난했어요. 고비도 많았고, 하기 싫었고, 내가 이걸 왜 하나 싶었어요. 제가 앞으로 어떤 분야로 가야 할지 구체적으로 선택하지 못했고, 42에 오면 빠르게 정할 수 있을 것으로 기대했는데 마지막까지 잡히지 않아서 스트레스였어요.

마지막 과제 전까지는 정말로 불만족스러웠지만 지원금으로 버텼고, 마지막 과제로 인해 그래도 42 과제가 할만하다고 생각하게 되었어요. 다만 마지막 과제도 이왕이면 많이 쓰는 기술들, 리액트나 node.js를 썼으면 더 좋았겠다 싶지만요.

### 이후 계획은?

프론트 엔드 개발자가 되기 위해 앞만 보고 달릴 겁니다. 아직 HTML/CSS에 대한 두려움이 남아있어서 그걸 없애는 것, 자바스크립트의 언어적 특성이나 문법적인 영역을 학습하는 것 등이 요즘의 과제예요. 드림코딩 엘리 채널의 강의를 보면서 역량도 다질 거고요. 개인적으로 하고 싶은 프로젝트가 생기면 해보고도 싶어요. 그러다 자연스럽게 반년 뒤에 취업하면 좋겠다고 생각하고 있고요.

### 이제 막 카뎃이 된 사람들에게 해주고 싶은 말이 있다면?

저처럼 살면 안 돼요. 저처럼 살았던 게 어떤 거냐면, 커리큘럼에만 올인하는 거예요. 다른 프로젝트를 하나씩 계속해야 해요. 누군가 시키는 걸 하는 게 아니라, 내가 원하는 걸 찾아서 하나씩 구현해가면서 학습을 하는 게 더 좋아 보여요. 그게 즐거움도 있고 진로를 정하는 데도 큰 도움이 되는 것 같아요. 간단한 것이라도 좋으니까 프로젝트를 계속 병행하면 좋지 않을까 싶어요.

제가 42 교육과정에 대해 아쉬움을 많이 얘기했는데, 사람에 따라서는 좋은 과정일 수도 있을 것 같아요. 취업이 문제가 아니라 C언어나 유닉스 관점에서 컴퓨터를 바라보고, 그런 깊이를 추구한다면 이 교육 과정도 충분히 좋을 수 있다고 생각합니다.

@sanam

github.com/simian114

## 부록 4-3 팀원 yohlee 인터뷰 | 최선을 다한다는 것

### 어떻게 살아왔나?

개발을 시작하기 전에도 42에서처럼 열심히 살아왔어요. 클래식을 전공했어요. 피아노를 전공해서, 예고로 가려다가 잘 안 됐죠. 내신이 안 좋았거든요. 공부에 대한 갈증을 그때 제대로 느꼈고 그 뒤로는 열심히 공부를 하게 됐어요. 누나가 오르간을 전공해서 음악을 하면서 독일에서 살고 있는데, 비용이 많이 들어서 부모님이 제가 음악 하는 것에 대해 부담을 느끼셨어요. 그래서 저는 돌아가더라도 공부를 해서 좋은 대학을 가고, 독문과로 간 다음에 독일의 음악대학원으로 유학을 가려고 했어요. 운이 좋아서 좋은 대학을 갔고, 대학을 가서도 음악에 대한 갈증이 계속 있었어요. 꿈이 다른 쪽에 가 있다 보니 학술적인 공부는 잘 안 하고 피아노를 많이 쳤어요. 연주회도 많이 하고요.

그러다 군대를 갔는데, 군악대는 음대생을 뽑기 때문에 저한테는 기회가 안 왔어요. 일반적인 병과로 가다 보니 음악을 할 수 없는 상황이 너무 힘들었어요. 2년이라는 시간이 짧은 시간이 아니니까, 전역했을 때는 너무 늦어버렸다고 생각하면서도 클래식을 하고 싶다는 마음은 계속 남아있었어요. 두 마음 사이에서 답을 찾다가 지휘로 방향을 틀었죠. 음악은 재능이 차지하는 비중이 큰 분야이고, 2년이란 시간 동안 음악을 놓았다가 다시 시작하는 상황이기 때문에 한 톨에 고민은 계속 남았지만요.

지휘에 대한 공부를 계속하고 레슨도 받다가 우연히 4차 산업혁명에 관한 심포지엄에 참여하게 됐어요. 유명한 교수님들이 하나의 기술을 놓고 어떻게 인간적으로 사용할 수 있는가에 대한 방법론적 이야기들을 많이 나누셨어요. 음악만 쫓으면서 살아왔던 저에게 큰 충격이었어요. 인공지능을 더 찾아보다가 자연어 처리(NLP)라는 분야를 알게 되었고, 저도 어쨌든 언어가 전공이다 보니 배우면 잘하겠다는 생각이 들었어요. 그러다 마침 뮤직 제네레이션(AIM)이라고 인공지능이 음악을 만드는 영역을 보면서 큰 가능성을 봤죠. 나중에 역량이 쌓여서 NLP와 AIM을 합치는 작업을 해보면 좋겠다. 그런 이유로 마침내 진로를 변경하게 했어요.

## 왜 42였나?

솔직히 얘기하면 이쪽 분야에 정보가 없었어요(웃음). 42를 하기 전에는 학교 수업 정도로만 인공지능을 접해봤었고, 코딩을 어떻게 해야 할지 잘 몰랐어요. 처음부터 코딩을 배우고 싶다는 갈증이 있었죠. 친구들도 다들 문과생이다 보니 당시의 저에게 다른 선택지는 없었던 것 같아요.

## 지난 1년의 카뎃 생활을 돌아본다면?

학부 과정이 끝나지 않았기 때문에 1년 동안 휴학을 하고 42에만 집중했어요. 주말에는 신다든가 하는 루틴을 두지 않고 매일매일 했어요. 하루에 쓰는 시간은 달마다 달랐는데, 처음 몇 달은 개인적으로 힘든 일이 있어서 8시간 정도씩만 앉아있었고, 다섯 달쯤 지난 후부터는 꼬박꼬박 12시간 이상씩 한 것 같아요. 중간중간 클러스터가 열릴 때는 매일 클러스터에 갔었고, 예약에 실패했을 때는 출입제한이 풀리는 저녁 11시에 가서 아침 9시에 집에 왔어요. 통학에만 3시간이 걸리는 거리였지만, 집이 쉬는 공간이 아니게 되는 게 싫어서 어떻게든 클러스터에 갔죠. 클러스터가 닫힌 기간에는 어쩔 수 없이 집에서 코딩했지만요.

책으로는 거의 학습을 하지 않았고, C++ 직전 과제들까지는 책을 한 번도 안 읽었어요. 모든 것을 웹서치로 해결했어요. webserv부터 웹서치로 부족하다는 생각이 들어서 책을 읽기 시작했죠. 서브젝트 자체는 절대적인 것으로 받아들였기 때문에 모든 사항을 꼼꼼하게 지키려고 노력했고, 딥러닝을 빨리 공부하고 싶다는 마음이 점차 강해져서 보너스는 웬만하면 하지 않았습시다. 그룹 과제들을 제외하고는 혼자 학습했던 것 같아요.

## 무엇이 가장 좋았나?

저는 42의 시스템이 영국 엘리트 교육을 모방했다고 계속 생각해오고 있었는데, 유럽의 잘 사는 애들이나 받는 교육이 한국에 들어와서 잘 정착이 될까라는 생각이 있었어요. 영국의 엘리트 교육이라고 하면 큰 특징이 자기주도적 학습인데, 한 마디로 혼자 알아서 하라고 하는 거잖아요. 제가 인상 깊게 들었던 얘기 중 숲에 아이들을 들어가라고 한 뒤에 선생님들이 그냥 집에 간다는 얘기가 있어요. 이런 게 한국에서는 절대 안 통할 거라는 선

입견이 있었어요. 좋지 않은 시선으로 42를 바라보고 있었는데, 막상 교육 체계 안에 편입이 되어보니 너무 좋고 신기하더라고요. 반전이었죠. 모르는 사람이라 인사하고, 아무렇지 않게 다가갈 수 있다는 것도 특히 좋았어요.

또 작년 여름부터 밴드 동아리를 했는데, 합주를 6번밖에 못했지만 그때가 되게 재밌었고요. 작년 8월쯤에 자연어 처리 스터디를 하고 공모전에 나갔는데 그때도 기억에 강하게 남아있어요.

### 아쉬웠던 부분은?

제일 큰 건 좋은 코드를 짜지 못했다는 거예요. 시행착오가 있어서 printf만 두 달을 했는데, 다른 1기 1차 분들보다 진도가 느려져서 자꾸 비교됐어요. 결국 속도를 맞추기 위해 코드의 품질에 대해 타협을 했죠. 더 많은 사람을 만나려고 개인적인 노력을 하지 않았다는 것도 아쉬운 부분이에요.

### 트렌센던스 얘기도 듣고 싶다.

전체적으로 좋은 점들이 많았어요. 다른 팀원분들이 열정이 많으시다는 게 좋았어요. 열정맨들(웃음). 프로젝트 일정에 대한 팀원들의 의견이 안 맞으면 눈치를 자꾸 보게 되잖아요. 마음껏 열심히 할 수 있다는 것, 서로 자극을 받고 더 열심히 하려고 한다는 것, 프로젝트 기간이 길어지더라도 누구 눈치를 안 봐도 된다는 게 무엇보다 좋았죠.

또 설계를 제대로 하고 프로젝트를 했다는 것을 빼놓을 수 없는 것 같아요. 이렇게 정교하게 협업을 하는 것 자체가 20대 통통어서 처음이었거든요. 설계를 잘하고 들어가는 게 좋다고 듣기만 했었는데 실제로 해볼 수 있어서 좋았고, 그 과정에서 루시드 앱, adobe xd, 포스트맨 같은 다양한 툴들을 썼던 것도 좋았던 포인트였어요.

구현 과정에서는 range나 progress bar와 같은 UI 컴포넌트를 렌더링할 때, 내가 이런 소스 코드들을 잘 가져와서 쓸 수 있게 되었구나 싶어서 좋았어요. validates 옵션이 아니라 validator 클래스를 따로 만들어서 유효성 검사를 처리할 때에도 보람찼어요.

아쉬운 부분이 있다면 에러 처리예요. 서비스 에러 객체를 만들고, 각 모델별로 개별적인 예외를 처리하는 에러 객체를 서비스 에러 객체로부터 상속받아 생성하려고 했었는데 잘 안 됐어요.

또 팀원들의 열정이 많은 게 장점이었지만, 열정이 너무 크다 보니 아침부터 시작해 새벽까지 할 때도 있어서 작업 일정이 체계적으로 관리되었다면 더 좋았겠다 싶었어요. 쉬는 시간이나 필수적 작업 시간, 선택적 작업 시간이 체계적으로 관리가 되었다면 쉬고 싶을 때 편하게 쉴 수 있지 않았을까 싶네요.

끝으로 후반에는 작업 일정이 맞아떨어지지 않아서 jujeong님이나 sanam과 주로 팀을 이루어서 작업했는데, iwoo님과는 밀도 있게 같이 작업해보지 못해서 아쉬웠어요.

#### **5명이 비대면으로 협업을 하는 과정이 힘들지는 않았나?**

딱히 힘들지 않았어요. 오프라인으로 했을 때의 장점도 있겠지만, 오프라인이 아니었을 때의 단점도 크게 느끼지 못했어요. 대시보드 관리도 잘 되고, 설계도 잘해놓고 진행을 했기 때문에 오프라인이 가능했더라도 그게 더 좋았겠다는 생각은 안 드네요.

#### **만약 트랜센던스를 처음부터 다시 한다면?**

사전학습에 대한 기간을 줄이고, 처음부터 নিজ 프로젝트를 진행할 것 같아요. 저희가 처음 45일간 공부만 했을 때는 공부를 하면서도 뭐가 뭔지 잘 몰랐어요. নিজ 프로젝트를 했을 때야 '공부했던 것들을 이렇게 활용하면 되겠구나'라는 감이 생겼어요. 그 감은 빠르게 잡을수록 좋은 것 같고, 기술을 잘 모르더라도 튜토리얼들을 무작정 따라가도 감은 잡히는 것 같아요. 괜히 프로젝트 학습이라는 말이 있는 게 아닌가 봐요.

নিজ 프로젝트 이후로는 비슷하게 갈 것 같아요. 설계 자체는 이번이 너무 좋은 경험이었기 때문에 똑같이 갈 것 같고요. 구현에 관해서는 백본의 모델과 컬렉션을 더 적극적으로 활용해보려고 할 것 같아요. CSS 역시 반응형에 더 힘을 실어보고 싶고, 백엔드에서 모델의 수를 줄이거나 연관 관계를 더 디테일하게 설정하려고 할 것 같아요. 백엔드에서 어떤



데이터를 찾으려고 할 때 쿼리가 복잡해지는 경우가 많았던 게 불편했거든요.

### 어쩌다 트랜센던스까지 끝나게 되었는데?

아직 얼떨떨해요. 홀리그래프에 들어가면 원래는 다음에 진행해야 할 서브젝트 한두 개에만 흰색 테두리가 있었는데, 지금은 바깥 서클의 많은 서브젝트가 흰색 테두리로 둘러싸여 있어서 신기해요. 이제 본격적으로 시작된 느낌이라 마냥 좋다고만 할 수는 없지만, 어쨌든 큰 하나가 끝났으니까 기분은 좋습니다.

### 이후 계획은?

아우터 서클의 10시 방향에 AI와 머신러닝 관련된 서브젝트들이 있는데, 해당 서브젝트들을 먼저 하려고 합니다. 사실 이미 register했어요(웃음). 규모가 큰 것들이 아니어서 2주 안에 4개는 할 수 있을 것 같아요. 해당 분야에서는 약간 예전 것들이고, 관련 기술의 공식 홈페이지 첫 번째 챕터에 있는 내용들이거든요. 42의 AI 커리큘럼이 상당히 기초적인 편이라 빠르게 끝내고 코테와 취업을 준비하려고 합니다.

### 이제 막 카뎃이 된 사람들에게 해주고 싶은 말이 있다면?

이번에 4기로 친구들 2명이 들어왔어요. 1차인 한 명은 인터뷰하는 지금 final exam을 보고 있고, 2차인 다른 한 명은 다음 주에 피썬을 시작해요. 제가 이 둘에게 매일 하는 얘기가 '개 같이 하라'예요. 열심히 하라고. 이곳은 친절하지 않아요. 누구도 먼저 알려주지 않죠. 스스로 할 수 있는 최선을 다하는 게 이곳에서 살아남을 수 있는 길이에요..

대신에 비전공자라고 너무 걱정할 필요 없다는 얘기도 꼭 하고 싶어요. 저도 해냈으니까. 이번에 트랜센던스를 함께 통과한 5명 중에서 비전공자가 4명인데, 사실 42가 추구하는 방향의 산 증인들인 것 같아요. 저희가 다른 분들보다 빠르게 공통 과정을 끝냈다는 것 자체가 응원의 메시지가 될 수 있지 않을까요.

@yohlee

github.com/l-yohai

## 부록 4-4 팀원 jujeong 인터뷰 | 전공자가 바라본 이너서클

### 이 팀의 유일한 전공자라고?

네. 컴퓨터 공학을 전공했어요. 학교가 특이한 게, 컴퓨터 공학과와 소프트웨어 학과로 나뉘어요. 소프트웨어는 코딩 위주이고, 컴퓨터 공학과는 코딩 반 하드웨어 반이에요. 저는 하드웨어 쪽에 관심이 많다 보니 C나 파이썬을 주로 사용해서 아두이노나 아두이노보다 더 답하게 쓰는 기계들 위주로 코딩을 했어요. 이론 수업은 많이 들어서 기본적인 CS나 알고리즘 이론은 학습이 된 상태였지만, 졸업하고 보니 친구들보다 취업하기에 코딩 쪽이 부족한 것 같아서 추가적인 공부만 필요하다고 생각했어요. 혼자 하는 것보다 나을 것 같아서 42에 들어오게 됐어요.

### 왜 42였나?

대학교를 졸업하고 코딩을 준비해야겠다고 생각했을 때, 형의 교수님이 마침 42서울을 알려주셨어요. 공공롭게도 테스트 당일 아침 8시에 알려주셨는데, 당일이어서 테스트나 봐보자는 생각으로 봤어요. 테스트를 잘 본 것 같지는 않았는데 다음 날 합격했다는 문자가 와버렸죠. 다른 프로그램들은 비용적으로도 비싸고 시간 준비도 많이 들 것 같아서 42로 들어오게 됐어요.

### 지난 1년은 어떤 방식으로 학습을 했는지?

42에만 집중했어요. 빨리 취업을 하고 싶은 마음에 처음부터 엄청나게 달렸죠. 놀고 싶을 때 놀긴 했지만, 생산적인 일에 관해서는 다른 것을 하지 않고 42만 했어요. 학교에서 C를 많이 했기 때문에 초반 과제들은 어렵지도 않았고요. minishell에서 제동이 걸렸죠. 같이 할 팀원이 없어서 기다리는 시간이 있었고, 그다음부터는 계획과 다르게 시간이 많이 늦춰졌어요. 아무래도 혼자 하는 과제들이 아니니까요. 예상했던 것보다 기간이 길어져서 중간에 이 과정을 계속해야 하나 싶은 고민도 있었지만, 이왕 시작한 거 이너 서클까지는 끝내보자는 생각으로 계속했어요.

책은 따로 안 읽고, 일단 만들면서 테스트하는 스타일이라 직접 해보면서 학습을 했어요.

printf나 minishell도 그런 방식으로 학습을 했죠. 쿠버네티스나 이런 새로운 기술들이 나올 때도 웹서치로 해결했고, 웹상에 자료가 많아서 42 전체 교육 과정에서 특별히 웹서치로 부족하다고 느꼈던 적은 없는 것 같아요. 굳이 꼽으려면 cub3d인데, mlx를 어떻게 사용해야 하는지 감도 안 잡히고 사례도 많이 없어서 힘들었었죠.

### 무엇이 가장 좋았나?

원 없이 코딩을 할 수 있는 환경이에요. 제가 딱 원했던 거예요. 제 취업에 도움이 된다고 할 수는 없지만, 코딩을 실컷 하고 다른 사람들과 협업을 할 수 있다는 건 분명히 장점이예요. 꾸준히 과정을 따라가다 보면 자연스럽게 사람들과 협업을 해야 한다는 시스템이 좋은 것 같아요.

### 아쉬웠던 부분은?

마지막 트랜센던스 과제도 그랬지만, 저희가 학습하기를 원하거나 취업 시장에서 중요하게 여겨지는 기술 스택이랑 42 서브젝트의 기술 스택이 아주 다르다는 점이 아쉬웠어요. 또 저 같은 경우에는 학교에서 C를 많이 했기 때문에, C 기반 과제들이 공통 서클에 너무 많아서 시간이 아까운 부분도 있었어요. 그래서 싸피가 저랑 더 맞지 않았을까 하는 생각도 해봤죠. 42 자체에 문제가 있다기보다 비전공자에게는 42가 더 좋고, 전공자에게는 다른 프로그램이 더 맞지 않겠냐는 생각을 해요.

### 트랜센던스 얘기를 해보자.

저는 졸업과제도 하드웨어를 이용해서 진행했어요. 굳이 백엔드와 프론트엔드를 결합해서 하는 것이 아니라, 백엔드만 간단하게 만들어놓고 프론트엔드는 넘어온 데이터만 보여주는 식이었죠. 그런데 프론트와 백엔드가 데이터를 주고받고, 그 과정에서 실시간 처리도 해보면서 많은 도움이 되었어요. 협업하면서 다른 분들의 코드들을 많이 보다 보니, 지금까지 제가 해왔던 구현들에 대해서 더 좋은 방법은 없었는지 돌아볼 수도 있었고요.

초반의 학습 기간이 너무 길었던 건 아쉬웠어요. 저는 직접 해보면서 배우는 스타일이라, 학습 기간 때 많은 걸 배웠다기보다 나중에 직접 개발을 하면서 많이 배웠던 것 같아요.

너무 완벽하게 학습을 하고 진행하려고 하지 않았다. 처음에는 라이트하게만 학습했다가, 실습한 뒤에 경험을 가지고 다시 한번 학습을 하는 게 좋은 것 같아요.

#### **전공자라도 5명이 100일간 비대면 협업을 하는 건 처음이었을 것 같은데.**

처음에는 힘들었어요. 다른 것보다 집에만 있다 보니 작업능률이 잘 안 오르고, 초반 학습 기간에는 특히 집중이 잘 안 됐던 것 같아요. 그런데 저희가 어느 순간부터 무조건 디스코드에 모이게 되었잖아요. 의무적으로 디스코드에 접속해있어야 하고, 설계를 시작하고 서부터는 또 음성 채널에 대해 상시접속을 유지하기로 하면서 능률이 확 오른 것 같아요. 디스코드 강제 참여가 가장 크게 도움이 되지 않았나 싶어요(웃음).

#### **만약 트랜센던스를 처음부터 다시 해야 한다면?**

저는 속도 위주예요. 무조건 되지만 하게 만들자는 거죠. 그래서 처음에 모든 기능을 가능한 한 빠르게 구현하고, 수정해가는 스타일이에요. 일단 구현하고, 구현 과정에서 발생하는 문제들을 수정하고, 구현이 끝나면 나중에 있을 법한 문제들을 수정해가는 방식이죠. 그런데 제가 이전에 겪어왔던 협업은 2-3명이 소규모로 할 때 통용되었던 방식이었고, 수많은 사람과 제대로 협업을 하려면 계획을 제대로 해야 한다는 걸 이번에 느꼈어요. 또 제대로 된 프로젝트에서는 코딩 못지않게 대화하는 시간도 많이 필요하다는 것도 느꼈고요. 그래서 다시 처음부터 해야 하는 상황이 되더라도 제 성향보다는 지금처럼 계획적인 협업을 진행할 것 같아요. 코드에 대해서는 아쉬운 부분이 딱히 없어요. 저는 작동만 잘 되면 된다는 스타일이라(웃음).

#### **어쩌다 트랜센던스까지 끝났는데?**

정말 후련해요. 웹서버가 끝나고 트랜센던스를 시작할 때까지 딜레이가 되었던 시간을 포함하면 4달이 넘었는데, 긴 시간이었어요. 학교에서도 DB 설계까지 꼼꼼히 하지는 않았거든요. 설계 과정을 꼼꼼하게 진행해본 게 큰 도움이 되었어요. 취업을 준비하는 데에도 큰 도움이 되고, 현업에 가서까지 좋은 영향을 줄 것 같아요.

#### **이후 계획은?**

1순위는 취업입니다. 대학교 졸업하고 벌써 1년이 넘었어요. 처음 42서울에 들어올 때는 6개월에 이너 서클을 통과하고 6개월 안에 취업하는 게 목표였거든요. 계획보다 많이 늦어졌어요. 포지션에 대해서는 고민이 여전히 많고, 솔직히 이너서클은 포지션 결정에 큰 도움이 되지 않았어요. 그나마 트랜센던스를 하면서 프론트나 백의 재미도 느꼈고, 싫은 포인트들도 느꼈죠. 백엔드를 생각하고 있지만 프론트도 아예 생각이 없는 건 아니라, 취업 준비를 하면서 많이 생각해 볼 것 같아요. 둘 다 해보고 싶은 마음이 있어서 취업 공고의 수요를 보면서 생각이 바뀔 수도 있을 것 같고요. 우선 코테부터 준비하고, CS는 공부도 학부 때 했지만 바로 대답할 수 없을 만큼 까먹었기 때문에 리마인드도 해야 할 것 같아요. 토이 프로젝트는 당장 계획이 없지만 좋은 기회가 생긴다면 혼자서 A부터 Z까지 해보고 싶은 욕심도 있습니다.

#### **이제 막 카뎃이 된 사람들에게 해주고 싶은 말이 있다면?**

전공자 입장에서 얘기를 드리는 게 좋겠네요. 42는 당장 취업이 목표가 아닌, 기초를 다지고 싶은 비전공자에게 더 적합하지 않나 생각합니다. 저는 이너 서클을 뚫고 취업을 하겠다는 오기가 있었고, 또 이미 시작했는데 중간에 끝내버리기에는 아쉬운 부분이 많아서 끝까지 왔어요. 만약 전공하셨거나 그에 준하는 지식이 있으시다면 이너서클은 큰 도움이 되지 않는 것 같아요. 대학을 1년 더 다니는 느낌이었거든요. 또 취업을 생각하고, 취업을 목표로 42서울에 오셨다면 이너 서클을 뚫는 것만으로는 도움이 되지 않는 것 같아요. 아무래도 취업을 목표로 최신 기술만 열심히 파는 프로그램이 많으니까요.

@jujeong

github.com/ungchi

## 부록 4-5 팀장 eunhkim 인터뷰 | 우리의 선택에 대한 신뢰

### 자기소개 좀 해달라.

6년 차 교육기획자였습니다. 사회적 경제 영역, 구체적으로는 혁신 교육 분야의 사회적 기업들에서 주로 일해왔고요. 처음에는 교육 프로그램 기획/운영을 주로 하다가, 교육 콘텐츠나 학습 방식이 가지는 한계를 느껴서 교육 게이미피케이션 업무로 넘어갔어요. 그러다 대면 교육의 한계를 느껴서 교구 개발 및 사업 운영에 관한 업무들을 작년까지 했고요. 결국 교육 문제 해결의 열쇠는 기술에 있다는 생각이 들어서, 직접 원하는 서비스를 만들어보려고 개발을 시작하게 됐어요. 무엇이 되었든 만드는 거라면 다 좋아하기도 하고, 장기적으로 봤을 때 개발은 동료에게 위임할 역량이 아니라고 판단했어요.

### 왜 42였나?

시스템 때문이죠. 제가 일하던 섹터에서 42는 교육 혁신의 대표적인 사례이기 때문에 모를 수 없는 곳이고, 저도 2015년에 에콜42를 알게 된 이후로 계속 지켜봐 왔어요. 사실 다른 것보다 42의 LMS나 시스템적 혁신성을 체험해보고 싶었던 마음이 가장 컸어요. 개인적으로 흥미로운 교육 모델을 몸으로 경험하는 것만큼 즐거운 일이 없거든요. 또 전반적으로 교육에 대해 신뢰를 잘 가지는 편이 아니기 때문에, 해외에서 검증된 42 모델이 주는 신뢰도가 있었고요.

### 지난 1년은 어떤 방식으로 학습을 했는지?

평일이나 주말을 가리지 않고 개인적인 약속이 있을 때를 제외하고는 매일매일 했어요. 밥 먹는 시간 빼고는 대부분? 친한 동료들이 별로 없어서 클러스터가 열리든 말든 집이나 카페에서 코딩했는데, 동료학습 관점에서 볼 때 모범적인 카뎃은 아닐 거예요(웃음).

공부한 건 반드시 정리하는 편입니다. 노선에 하다가 gitbook으로 옮겼습니다. 학습 방식으로는 영상보다는 문서를 많이 봅니다. 책을 읽기보다 웹 서치를 주로 하는 편인데, 웹사이트와 트랜센던스는 책들을 같이 봤어요. 돌아보니 도커나 쿠버네티스는 한 권 사뵈으면 좋았을 것 같네요.

### 무엇이 가장 좋았나?

피션에서 BSQ를 할 때, 둘 다 개인 과제에 대한 욕심을 내려놓고 마지막 주를 BSQ에 쏟았었던 거요. 정말 좋았어요. 카넷이 되어서 피션 평가자로 참여했던 것도 기억에 남고요. 간간히 열렸던 글로벌 코딩게임 대회들, 소인수분해 대회, 두 번의 42 해커톤 같은 이벤트들도 즐거웠어요.

기본적으로 42의 학습 환경을 경험하는 것 자체가 멋진 일이었어요. 제가 피션 때 모든 러시아에서 각각의 이유로 0점을 맞았는데, 끝내주는 경험이었죠. 그중 두 명의 평가자가 도비와 김수보 멘토님이셨었는데, 평가해주시는 태도에서 감동을 많이 받았어요. 업이 교육이어서인지 모르겠지만, 가만히 슬랙만 보고 있어서 '이런 시스템이 한국에서 가능하다'는 증명'이 실시간으로 진행되고 있는 것 같아 몽클랄 때가 많아요.

서브젝트만 놓고 얘기했을 때는 두 가지인데, 하나는 트랜센던스예요. 아무래도 웹에 관심이 많았으니까 실제적인 역량, 협업 경험을 쌓을 수 있어서 좋았어요. 그보다 좋았던 건 C++ module 00의 ex02였는데, 맥락에 기반한 스토리텔링이 있었다는 점에서 42 이너서클을 통틀어 가장 좋은 과제라고 느꼈습니다.

### 서브젝트만 한 것 같지는 않다.

42라는 영역 안에서 재미있는 건 다 하려고 했어요. 42 해커톤이 각각 한 달과 3일에 걸쳐 두 번 열렸는데, 둘 다 참여해서 각각 슬랙앱과 ios 앱을 만들어봤어요. 일주일씩 있었던 코딩 게임 대회도 두 번 다 참여했구요.

그 외의 사이드 프로젝트는 대부분 42 과제와 관련한 것들이었어요. c++ 과제에 반복적인 작업이 많아, 불필요한 작업을 극단적으로 줄여주는 터미널 기반 익스텐션(c++)을 만들었고요. webserv는 튜토리얼을 만들었고, 트랜센던스는 이 인터뷰가 들어갈 플래닝 가이드를 만들고 있네요(웃음). c++, webserv, transcendence가 시간을 많이 잡아먹는 프로젝트들인데 제가 남긴 콘텐츠들을 잘 활용하신다면 시간을 줄이는 데 큰 도움이 되지 않으실까 싶어요. 제 github profile에서 확인하실 수 있어요..

### 아쉬웠던 것도 있을 것 같은데.

서브젝트와 평가 문항이 서로 다른 말을 하는 경우가 있어요. 서브젝트에서는 나오지 않았던 항목이 평가에서 등장하면 당황스럽고, 또 한쪽이 업데이트되면서 다른 한쪽은 이전 버전을 유지할 때 논리가 어긋나는 부분들도 더러 보였습니다. C++의 문제들이 특히 말도 안 되는 오류가 많은데, 프랑스에 문의해도 반응이 오지 않아서 답답했어요.

본 과정 초기에는 보컬들이 학습자들을 연결하거나, 문화를 촉진하는 이벤트들을 많이 열었는데 어느 순간 사라진 점도 아쉽습니다. 이제는 코알라시옹 리더들이 해야 할 일 같은데, 기대하는 만큼 혹은 필요한 만큼 되고 있지는 않은 것 같아요.

### 트렌센던스는 어땠나?

하고 싶은 걸 마음껏 해볼 수 있어서 좋았어요. 이제 여러 사람이 같이 하나의 웹 서비스를 개발할 때 어떤 과정을 거쳐야 할지 구체적인 흐름이 조금씩 잡히는 것 같아요. 아무래도 학습, 설계, 구현 과정에서 필요하다고 생각하거나 쓰고 싶은 툴, 하고 싶은 것들을 원 없이 해볼 수 있었기 때문이겠죠. 사실 그런 시도를 마음껏 해보고 싶어서 팀장을 자원했는데, 다행히 팀원들이 좋은 경험으로 받아들여 줘서 감사했어요.

제가 처음으로 다녔던 회사가 루비 온 레일즈로 플랫폼을 개발했었고, 개발자는 아니었지만 당시에 레일즈 튜토리얼을 했었기 때문에 정겨웠어요. 웹도 좋아하고 게임도 좋아하기 때문에 전체적으로 언어나 영역, 콘텐츠가 모두 즐거웠던 프로젝트였다고 말할 수 있겠네요. 실력도 많이 늘었고, 웹 프레임워크도 나름대로 잘 써봐서 과제를 통과했다는 기쁨이 아니라 이 기술을 어느 정도 소화했다는 기쁨이 특히 큰 것 같아요.

### 비대면 협업은 괜찮았나?

개인적으로 비대면은 환경이 반, 사람이 반이라고 생각해요. 일하고 돈을 받는 회사에서도 온라인 커뮤니케이션 역량이나 피드백 속도 면에서 동료들한테 아쉬운 적이 많았기 때문에 큰 기대를 안 했어요. 그런데 동료들이 모두 열정이 넘치고, 저희가 환경 구성을 워낙 잘해서 문제도 없고 오히려 오프라인보다 더 나은 협업을 했던 것 같아요. 그간 많은



종류의 협업을 해왔다고 생각하는데도 정말 손꼽을만한 협업이었어요.

### 만약 트랜센던스를 처음부터 다시 해야 한다면?

그때 그때 할 수 있는 최선을 다했던 것 같아요. 지금의 지식과 경험을 가지고 다시 한다면 파일 구조, 네이밍, 나열 순서, 뷰 형식 등 스타일 가이드를 더 꼼꼼하게 세울 것 같아요. 추상화를 충분히 해서 필요한 상속들은 가능한 쓰고, 예외처리나 에러 로그를 처음부터 꼼꼼하게 남겨야겠죠. 레일즈 컨트롤러와 모델의 책임을 명확하게 나누고, Rspec을 처음부터 제대로 도입해서 TDD를 하고 싶어요. 모델과 컬렉션을 더 적극적으로 활용하고, dom, css, view와 관련해서도 가능한 컴포넌트화를 하고 싶어요..

### 어쩌다 트랜센던스까지 끝났는데?

정말 어쩌다 끝났네요. 아쉬운 점은 학습 기록이 산발적이라는 거예요. 저는 책을 쓰듯이 기록하는 걸 좋아하고, 그러자면 아예 튜토리얼 형식의 글을 쓰거나 기술 자체를 소개하는 글을 써야 하는데 그걸 잘하지 못했어요. 산발적으로 코드나 지식을 정리해서, 나중에 봤을 때 체계가 없어서 직관적으로 원하는 내용을 찾기 어려운 경우가 많았죠. 다행히 웹 서버와 트랜센던스는 그걸 비교적 제대로 한 케이스고요.

또 애써 학습해서 통과한 프로젝트도 금방 까먹다 보니 어셈블리나 mlx는 가물거리고, 도커와 쿠버네티스는 이력서에 쓸 만큼 자신이 없어요. 시간이 지나도 자유롭게 기술을 사용할 수 있을 때 진짜 학습이 일어났다고 생각하는데, 그런 면에서 애써 익힌 기술도 잊어버렸다는 게 가장 큰 아쉬움입니다. 어쩌면 처음부터 제대로 학습하지 못했던 것일 수도 있겠죠. 반면 새로운 기술을 학습하는 역량과 개발, 좋은 코드에 대한 감각이 많이 늘었다는 건 유의미하게 느껴집니다. 새로운 기술에 대한 겁이 많이 사라진 것 같아요.

스스로 비판을 했지만, 자축하고 싶은 점도 있어요. 이너 서클의 모든 보너스를 다 했다는 거예요. 심지어 보너스 점수를 주지 않는 C++의 보너스들도 전부 했어요. 이너 서클을 온전하게 경험하고, 가능한 한 많이 학습하고 싶었거든요. 무조건 보너스까지 다 한다는 개인적 목표가 있었는데, 중간에 타협하지 않고 지키게 되어서 스스로 칭찬해주고 싶어요.

## 이후 계획은?

온라인 교육 서비스부터 시작해서 제가 원하는 서비스들을 만들면서 살아가고 싶어요. 좋은 경영자가 되는 게 삶의 목표이자 계획이고, 지금은 그에 필요한 주요 역량 중 하나를 다져나가는 시기라고 생각해요. 그래서 취업을 어떤 회사의 어떤 직무로 하게 될지 아직 잘 모르겠지만, 개발자로 취직한다면 그게 뭐가 됐든 제 미션이나 비전과 관련해 배울 점이 가장 많은 조직, 포지션으로 가지 않을까 싶네요.

우선 평범하게 코딩 테스트와 면접 준비를 하는 한편으로, 취업 시장의 요구와 상관없이 다양한 기술들을 학습하고 흥미로운 프로젝트를 많이 해보려고 합니다.

## 이제 막 카뎃이 된 사람들에게 해주고 싶은 말이 있다면?

42를 오랫동안 동경해왔기 때문에, 이곳에서 개발을 시작한 게 개인적으로 영광이에요. 이유는 많지만, 그중 하나는 역사를 몸으로 경험하는 곳이기 때문이에요. shell과 c부터 시작하여 시계의 태엽을 한 바퀴씩 감아내면서 이전 기술의 한계를 다음 기술이 어떻게 혁신하는지, 시간에 따른 기술의 변화를 몸으로 학습하잖아요. 제가 바라보는 42는 옛날 기술이 아니라 기술의 변화에 대한 흐름을 학습하는 곳이고, 그 감각을 기반으로 불편함에 더 민감하고 혁신에 더 익숙한 사람들이 자라나는 곳이에요. 개별 기술이 한국의 취업 시장에서 얼마나 쓸모 있는가가 아니라, 이전 서브젝트와 다음 서브젝트가 맺고 있는 관계에 집중해서 학습하다 보면 42에 감사한 마음이 들 때가 많아요.

또 42가 왜 잘 정리된 레시피를 따라 하는 클론 코딩이 아니라 실패와 시행착오를 통한 학습을 중요시하는가도 흥미해볼 주제인 것 같아요. 결국 우리는 현장에서 레시피가 없는 문제들을 풀어가게 될 텐데, 두려움 없이 부딪히면서 마침내 자기만의 크리에이티브한 접근에 도달하는 과정에서 만들어지는 근육은 꽤 질기다고 느끼거든요. 우리가 자신의 선택을 더 신뢰하고, 열린 마음으로 겪어간다면 42는 정말 멋진 선택이 될 거예요.

@eunhkim

github.com/eunhyulkim