

FRONT
BACK
DESIGN

• DESIGN

• FRONT

• BACK

**피어개발
백서**

PEER

peer

CONTENTS

목차

발간사

- peer 운영대표 전준성 4
- peer 개발총괄 류한솔 5

백서의 목적, 목표

- peer 의 목적 6
- peer 서비스의 목표 7

Chapter.1 디자인&기획

- peer 기획 11
- peer UX 23
- peer 디자인 66

Chapter.2 프론트엔드 개발자

- | | |
|---------------|-----|
| @woorikim 김우림 | 94 |
| @jujeon 전준성 | 96 |
| @jeyoon 윤정연 | 99 |
| @hyunjung 정현섭 | 101 |
| @hyna 나현 | 102 |
| @hyeokim2 김현지 | 105 |
| @hujeong 정희호 | 108 |
| @hoslim 임호성 | 110 |

Chapter.3 백엔드 개발자

@haryu 류한솔
@hyeongki 김형찬
@juhyelee 이주현
@junssong 송준상
@jwee 위치혜

114
117
122
125
128

Chapter.4 디자이너

@Bori 이보람
@Yachae 양채윤
@Joah 조하연

132
133
134

Thank U for Contributors

· Thanks

135

• • •

19년 말 과학기술정보통신부 산하 이노베이션아카데미,
42서울(école 42 seoul camp.)이 설립되었습니다. 그리고 peer는
21년 3월부터 시작한 소규모 학술스터디가 시초가 되어, 23년 말 지금은
전체 연수생의 약 40%가 소속된 교내 커뮤니티로 자리매김 하였습니다.

école 42는 교수와 교재, 수업이 없는 3무교육으로 PBL(Project-Based-Learning),
동료학습(Peer-to-Peer-Learning) 시스템을 가지고 있는 혁신교육입니다. peer는
많은 연수생들이 PBL과 동료학습을 효과적으로 누릴 수 있도록 기여해왔습니다.

동료에게 배우는 동료학습의 특성상, 많은 동료를 만나 함께 학습하는 것이 중요합니다.
'어떻게 동료들과 학습할 것인가?', '어디서 동료들을 만날 것인가?'의 두 가지,
근본적인 질문을 해소하는 것이 중요합니다.

약 50명 규모의 자연스럽게 동료들과 만날 수 있는 교내 행사를 자주 개최해왔고,
현직 개발자들과 연수생들의 네트워킹을 위한 300 명 규모의 개발자 컨퍼런스
2023 Wassup 42를 개최하였습니다. 또한 약 200 여개의 학술스터디와
약 50여차례가 넘는 연수생들의 지식세미나도 진행되어 교내 연수생들의
집단 학습수준 향상에 기여해왔습니다.

무엇보다 이번 개발백서를 집필하게 된 이유는 기존에 노션에서 운영되어오던
커뮤니티 서비스를 웹 서비스로 개발해왔던 값진 과정을 알리고, 이를 통해서
저희가 경험했던 시행착오, 노하우 그리고 결과물이 다른 사람들에게 참고할 수 있는
디딤돌이 되길 기대하기 때문입니다.

부디 저희의 바람과 같이 새로운 프로젝트를 시작하거나, 이끌어갈 때 도움이 되기를
희망합니다. 아무쪼록 peer의 발자취에 힘찬 응원을 보내주시기 바라며, 앞으로도
peer는 개발자들을 연결하는 든든한 조력자로서 노력하겠습니다.

peer 운영대표 전준성

언덕을 따라 돈키호테에서 사자왕이 될 수 있을까?

peer 라는 동아리, 혹은 커뮤니티로 사람들과 개발자라는 꿈을 향해 달려 가는 중입니다. 20대의 마무리, 부족한 경험, 실력, 그런 상황에서 새로운 직업을 갖기 위해 밑바닥부터 새롭게 끌어 올려 무언가를 한다는 것은 정말 쉽지 않았습니다. 저를 보고 많은 사람들이 '돈키호테' 같다고 생각하지 않았을까 생각해보곤 합니다. 비아냥 거리지 않았을까? 자조적으로 생각도 해봅니다. 저의 20대를 지나 정말 마지막이라고 생각하고 시작한 도전은 '사실' 비현실적인 것이라고 불려도 할 말이 없을지 모르겠습니다.

그럼에도 허무맹랑해 보임에도, 저는 멈추고 싶지 않았습니다. 42서울에서의 동료학습은 저를 성장 시켰습니다. 오히려 더 충실히 컨셉(?)을 지키고자, 돈키호테가 더 돈키호테가 되기 위해 도전을 해보고 싶어졌습니다. 아직 배우지 못해 기회가 없던 분들에게도 함께 해보자고 제안 해보았습니다. 함께 배워서, 함께 세워내길 해보고 싶었습니다. 여러 일들이 있었습니다. 갈등도, 폭발 버그 엔딩도 몇 차례 치룬것 같습니다. 시간도 예상보다 더 많은 시간이 걸려 버렸습니다. 많은 일들이 있었지만, 드디어 peer 라는 이름에 걸맞는 웹 어플리케이션 구축에 성공할 수 있었습니다.

비록, 이 시작은 돈키호테가 그저 이제 조금 견습 기사가 된 것 같다는 생각이 들긴 합니다. 온전히 목표하고 지향했던 지점까지 가지는 못했다고 생각합니다. 하지만 결국 언덕을 따라 여기까지 올 수 있었습니다. 시작은 미비했지만 과정을 겪으면서 함께한 분들 모두가 성장을, 동료학습을 경험해주셨고 그 결과가 이렇게 '시작' 되었다는 것을 느낍니다.

언젠가 피어가 더 많은 동료학습의 장이 되길 바라며, 동료학습을 왜 이렇게 강조하며, 지금 이 순간까지도 peer 라는 이름에 집중하고 있는지 많은 분들이 느끼실 수 있길 기원합니다. 앞으로의 피어 개발-운영팀은 그런 순간을 경험하고 느낄 수 있는 기능들을 구현해보고, 서비스를 해보고 싶습니다. 그게 제가 피어 개발 총괄을 통해 얻은 너무나도 소중한 기억이자, 경험이자, 실력이 되었습니다.

peer 개발총괄 류한솔





이제 세상은 4차 산업혁명시대에 진입하여 이전에 맞딱뜨리지 못했던 새로운 문제를 해결하는 창조적 인재들이 필요합니다. 이러한 시대적요구에 세계 교육의 트렌드는 PBL(Project-Based-Learning) 과정을 빼놓고 말할 수 없게 되었습니다.

학생들이 정해진 답을 외우는 것이 아닌, 집단토의를 통해 스스로 문제를 해결 할 수 있는 아이디어를 떠올리고, 가설을 실험하고, 실패하고, 다시 반복해 프로젝트를 완성하는 것.

ICT분야에서는 수 십년에 해당하는 커리어 기간 동안 필요한 기술들을 정확히 예측하는 것은 불가능합니다. 따라서 새로운 상황에 대한 적응력, 새로운 문제를 해결할 수 있는 능력, 새로운 기술과 프로그래밍 언어에 대한 숙련도가 필수적입니다. 따라서 école 42라는 혁신교육은 기술적인 능력 습득을 넘어 학습능력을 연마하는데 초점을 맞추고 있습니다.

교수, 교재 그리고 수업이 없는 3무 교육인 école 42는 PBL(Project-Based-Learning)과 동료학습 (peer-to-peer-learning)의 교육시스템을 가지고 있습니다. 주어진 과제를 바탕으로 학생들이 서로 함께 배워야하는 교육과정인 것 입니다. 여기서 peer라는 커뮤니티는 42서울(école 42 seoul camp.)에서 보다 더 많은 동료들을 만나 학습을 시작할 수 있게 도와주는 매개체 역할을 해왔습니다.

개발자는 평생 학습하는 직군이라고도 불립니다. 끊임없이 새로운 기술을 배워야하는 직업 특성상 효율적으로 배우는 것이 중요합니다. 이는 다른 사람들과 함께 학습하는 것이 효과적입니다. 그럼 어떻게 적절한 사람을 찾을 수 있을까요 ? 이를 해결하는 것이 peer의 목적입니다.

단순히 스터디 그룹을 만들어서 숙제검사하듯이 학습하는 것이 아닌, 공동의 문제를 함께 해결하는 프로젝트를 진행하거나, 동료들과 토의하고, 서로 가르쳐주며 성장하는 개발자들의 커뮤니티로 자리매김 하는 것이 peer의 목표라고 할 수 있습니다.

PEER 서비스의

목표

'peer 웹 어플리케이션'은 이름 그대로 단순한 웹 페이지 서비스가 아닙니다. peer 서비스는 web application 의 조건이 되는 Progressive Web Application 기준을 준수하여 웹 푸시, 실시간이 필요한 영역에 대한 지원 등이 포함되어있습니다. 그렇기에 이른바 '웹 어플리케이션'이라는 카테고리의 서비스입니다. 이러한 서비스를 기획하고 프로젝트를 진행함은 아래와 같은 목표를 가지고 있기 때문입니다.

동료학습이란?

우리의 서비스의 기반이 되는 개념입니다. 이는 42 프랑스 - 42 서울로 이어지면서 계속 확장되어간 개념으로써, 전통적인 개념의 학습 구조를 벗어나서, peer to peer 를 만들고, 구성원이 서로가 서로에게 객관적 평가나 가이드 제시, 동시에 토론적 구조를 통해 문제 해결의 길을 만들어 나가는 것을 의미합니다.

개발자들을 위한 목표

1. 경험이 없어도, 실력이 부족해도 동료들과 함께 문제를 해결해 나가는 개발 프로젝트로 만들자.
2. 실제 상용 서비스라는 수준을 규정하고, 운영자들의 입장을 고려한 기획과 설계를 담은 프로젝트를 진행하자.
3. 실제 상용 서비스라는 수준을 규정하고, 이용자들의 입장을 고려한 기획과 설계를 담은 프로젝트를 진행하자.
4. 학생 수준을 넘어서기 위해 노력해보자.

운영자들의 목표

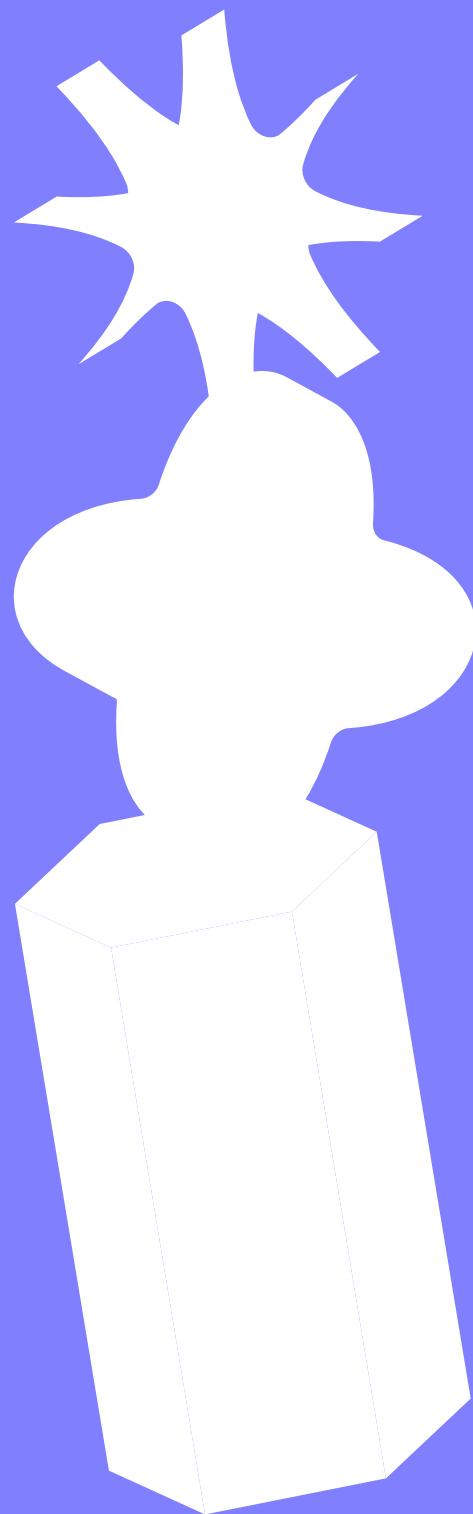
1. 기존 노선의 터전에서의 불편한 사용자 환경을 개선하자.
2. 동료 학습의 장을 제공해주기 위해, 기존의 작업의 고도화를 달성하자.
3. 최초 런칭의 기반이 되고, 다음 운영-개발진을 구성하기 전 단계의 목표들을 달성하자.

이용자들을 위한 목표

1. 동료 학습을 위한 터전으로의 역할을 명확히 하며, 사용자들의 동료 학습을 통한 스터디, 프로젝트 활성화를 핵심 제공 서비스에 담아낸다.
2. 동료 학습의 결과물을 남기고, 남긴 결과물들에 대해 공유하여, '결과'가 계속해서 서로에게 영향을 줄수있는 서비스를 제공하자.
3. 동료 학습을 진행하는 프로젝트, 스터디의 기능적 도구들을 제공해 줌에 따라 동료 학습 과정을 그저 '개념'적으로 끝나는 것이 아닌 '실용'적인 도구로 활용될 서비스를 제공하자.

제 1장

디자인 & 기획





peer 기획

시작은 무엇에서 비롯되었는가

커뮤니티에 대한 수요는 항상 많습니다. 자기들 만의 공간을 만들고, 그 곳에서 어떤 목적을 위해 연대하고, 네트워크를 통해 성장하거나, 조직 자체가 커지는 것은 사람으로써 할 수 있는, 자신과 주변에 힘을 키우는, 성장하는 방안이라고 할 수 있습니다. 개인의 힘 보다는 조직, 팀, 커뮤니티의 힘이 더 크다는 것을 증명하는 것이라고 생각합니다.

peer라는 커뮤니티도 마찬가지입니다. 동료학습의 좋은 사례들이 발굴되고, 좋은 사례들이 모이는 장소를 만들어져 갔습니다. 비록 부족하지만 동료 학습, 같이 성장이라는 주제를 가지고 고민하던 시간과 그 노력에 사람들은 함께 해주었고, 동료학습의 열정 만큼이나 42서울이라는 공간에서의 잊지 않는, 꺼지지 않는 생명력의 일부로 존재한다고 해도 과언이 아닙니다.

기획자로서 저는 이러한 현상을 가까이에서 보면서, '가능성'을 바라봤습니다. 하지만 아이러니하게도 동료학습을 위해 서로가 서로를 관리하는, peer 서비스의 개발에 필수 기본 요소들의 구현은 어렵지 않습니다. 그렇기에 이것이 peer라는 공간이 필요한가에 대한 의구심을 갖고 있었습니다. 기획을 하고, 이것의 필요성이 입증되며, 새로운 서비스로 만드는 과정이 정말 필요할까? 저는 이 논제에 대해 몇달 이상을 고민하였으며, 거기서 핵심이라고 할만한, 본질적인 물음에 계속해서 답을 내지 못하고 있었습니다. 왜냐하면 동료 학습이란 것이 사람과 사람 사이의 주체성으로 형성되는 것이며, 거기서 서비스가 가지는 의미는 '도구'라는 의미를 벗어나지 못한다면, 새로운 서비스가 필요한 이유는 없는 것이라고 생각했습니다. 이미 너무 많은 서비스들은 동료 학습, 커뮤니티를 표방하고 있으며, 그런 서비스들은 그 크기와 집중도의 여하를 막론하고, 동료들을 주선해주고, 알선함으로써 그 역할을 다합니다. 그 뒤부터의 서로의 관리는 직접적인 협업툴로 표방되는 깃, 깃허브, 지라, 노션과 같은 것들이 그 역할을 이미 충분히 잘 해주고 있었습니다.

시장 조사를 통해 피어 웹 서비스, 이것이 반드시 필요하다!를 소비자들에게 하여금 납득시키기가 어려웠습니다. 그렇기에 저는 이걸 해야할 이유와, 그 가능성을 측정하고 싶었습니다. 종래의 많은 서비스들 사이에서 어떤 강점을 가질 수도 없는데, 그런 서비스가 계속해서 성립되기도 어려울 뿐 아니라, 살아남기 위해선 비용적인 부분들을 고려해야 했기 때문입니다. 더불어 생각해보면, 도구 그 자체에만 충실했던 서비스가 이미 수천 수백개가 존재하는데 말이지요.

그렇기에 초기 peer 서비스를 위한 '웹 페이지 구축'에 대해서는 부정적인 답이 제 안에 내려진 상태였습니다. 그런 상태로 peer의 2022년도 마무리 되어가고 있었습니다. 오히려 저에겐 다른 기획들이 더 중요하게 느껴졌습니다. 하지만 동료학습을 하고, 42서울에서 배운 많은 것들, 그것들을 보다 멋지게 만들고 싶다는 생각은 분명히 있었고, 그런 것이 필요하다는 부름이 결국 peer라는 커뮤니티를 존속시킨 이유였기 때문에, '기획'을 그리고 서비스를 구축할 '동력'이 무엇인가 찾아 헤매기를 멈출수는 없었습니다. 그러던 와중 저의 눈길을 끈 것은 OpenAI에서 나온 LLM(Large Language Model) chatGPT를 필두로 시작된 거대 언어 모델형 AI 들입니다.

LLM에 대해서 여기서 디테일하게 설명드리긴 어렵습니다. 하지만 내용들을 다소나마 정리하고 종합해보면, '인간의 행위'의 평균값을 알려줄 수 있는 도구이며, 이미 이것의 능력을 모르는 사람은 없을 것입니다. 언어 데이터를 모으고, 사람의 행위의 확률적 계산한 뒤, 어떤 대화, 단어, 표현이 나올 때 어떤 다음의 표현, 단어, 대화가 나올지에 대한 확률의 이정표 모음입니다. 그렇다면 왜 이것이 피어와 연결 되는가? 어쩌면 이것이 현재 목표로 하는 피어 기획의 근간이라고 할 수 있습니다.

서비스의 대상은 누구인가

기획의 핵심은 역시나 수요를 가지는 대상에 대한 분석이었습니다. 기획을 해나가고, 그 기획의 형태를 구체화 하는 과정에서 **디자인 씽킹**이라는 방법론을 기반으로 생각했습니다. **사용자**의 페르소나(인격)를 정리하고, 그 대상이 어떤 '생각'으로 우리 서비스를 사용하는가를 고민하였습니다. 어떤 상황에서 서비스를 찾고, 어떤 상황에서 그들은 문제라는 것을 지정하고, 그들이 타당하다는 방식, 실마리로 선택하는 것이 무엇인가 등을 명확히 한다면 이는 바로 서비스를 구축하는데 핵심이 되고, 서비스를 사용자들의 선택이 되는 핵심이 되기 때문입니다.



이것이 중요하다고 생각한 가장 큰 이유는 peer의 커뮤니티 운영의 과정에서 경험적으로 얻었던 데이터 때문입니다. 스터디를 만들거나, 프로젝트를 진행한다고 할 때, 그 작업이 결과가 성공적이라는 기준은 다양할 것입니다. 그러나, **성공이라는 것들의 차이점을 덜어내다 보면 어느정도 일반화도 가능합니다.** 만족도가 어느정도 되는가? 팀원들 사이에서 학습의 수준은 어느정도 되는가? 지정한 일정에 최대한 맞춰 보았는가? 등 유사하고, 일반화 되는 그런 수준이 존재합니다

이런 기준에서 데이터들을 종합해본 결과 단순하게 성공을 하는 데 있어서 구성원들이 '잘 한다' 와 '못 한다' 와 같이 **실력적인 부분이 100% 완벽하게 성공을 보장하지 못하는 것이 밝혀졌습니다.** 잘하는 팀원들이 모였지만 좋지 못한 결과가 나오는 경우도 상당하며, 잘 못하는 이들이 모였지만 팀의 만족도, 완성 여부를 통해 공통의 '성공'이라는 기준에 상당히 높게 부합하는 경우가 존재했습니다. 그렇다고 비율적으로 잘하는 이들과 못하는 이들의 인원의 수 역시 성공을 보장해주지는 않았습니다.

예를 들면 대학생들에게 프로젝트나 스터디, 협업에 대해, 매우 강력한 효율 중심의 의사 결정이 필요하다는 것이 그들의 수요 밑바닥에 존재했습니다. 왜냐하면 42서울을 참여하는 사람들 기준으로 본다면 자신의 앞길을 걸어가야 하고, 이를 위해 대학교 생활 뿐 아니라 42서울이라는 과정을 함께 해보기를 결정했고, **그만큼 그들에게는 '고민하기 위한 시간' 자체가 죄악 내지는 비효율로 생각하는 자연스러운 의식이 생겨나는 것을** 볼 수 있었습니다.

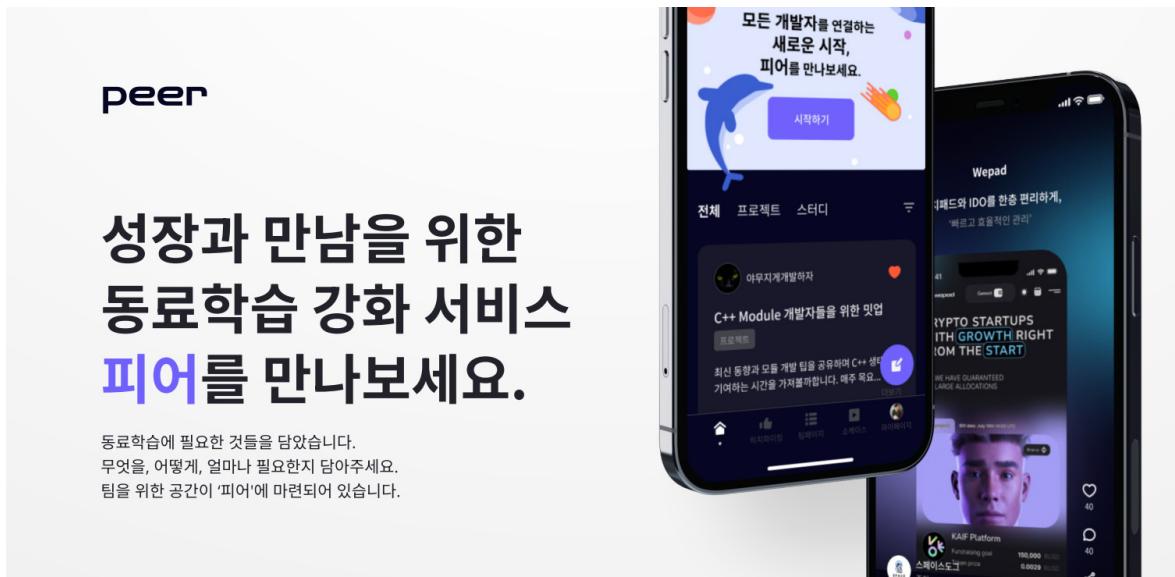
그렇기에 일반적으로 프로젝트, 스터디 등을 생각하는 열정적인, 피어에 대한 수요가 있을 대학생들은 과제, 프로젝트 등을 진행한다고 할 때 속도, 빠름을 상당히 중요시 하며, 다소 공략적인 접근 방법을 채택하는 경우가 대다수의 경우 많으며 특히 전공자들을 중심으로 그러한 움직임을 주도 하고 있었습니다. 그러나 보니 반대의 경우, 즉 학습 자체를 하는 것일 깊게 바라보거나, 고민을 하는 것이 필요하다고 느끼는 42서울 카뎃들을 만나면 서로의 목적에서 상충하는 고충을 peer 커뮤니티 내부에서 발생하는 것을 볼 수 있었습니다.

이와는 별도로 취업이나 창업을 목적으로 전일제(全日制)로 학습 내지는 프로젝트를 진행하는 입장은 이와는 크게 달랐습니다. **오히려 취업을 고려중인 입장에선 42서울의 과정, 프로젝트 등을 진행하는 과정이 곧 현실에서의 프로젝트와 유사할 거라는 나름의 믿음체계를 갖고 있었고, 그렇기에 동료학습을 통해 성장하는 모습을 바라보는 카뎃들은 '고민하는 시간을 줄인다' 는 말 자체를 이해하지 못할 뿐 아니라, 오히려 더 많이, 더 자세히 배우려는 인식이 그들만의 가치 체계로 측정 되었습니다.**

창업을 고려하는 이들은 또 다른 입장을 가지고 있었습니다. 전일제를 한다는점에서 동료학습의 가능성이나 긍정성은 상당히 높은 편에 속했지만, 이들의 경우 대다수의 사람들이 자기 자신을 위한, 자기가 꿈꾸는 목표를 만들어 가기를 바라는 성향이 다분히 강하다보니 **기술 자체의 근본적인 고민을 하기 보단 빠르게 원하는 기술의 관점을 이해하고, 새로운 것을 만들기 위해 기술을 빠르게 파악하고 섭렵하는 것에 높은 가치**를 두는 것을 볼 수 있었습니다. 그렇기에 동료 학습이나 42서울의 과제를 진행할 때의 중요 포인트는 매우 달랐으며, 성격이나 개인의 학업 상황 등이 연루된다면 정말로 n개의 가능성이 생겨나는 것을 유저 분석을 통해 알 수 있었습니다.

그렇기에 피어는 만약 웹 서비스로 자리를 잡아야 한다면 이러한 사람들의 요구들이 적절히 섞여 있어야 한다고 판단하였습니다. 그리고 이들은 공통적으로 어떤 조건이든지 간에 피어라는 서비스가 존재하길 원하며, 이러한 점을 제대로 접목시킨 서비스가 아직 부족하다는 판단하에 피어 서비스가 필요하다는 점을 다시 한 번 확실시 시킬 수 있었습니다. 이들을 유저로 지정하고, 사용자의 행동, 패턴에 맞춰 원하는 동료학습의 도구로 필요한 것들과 그들을 가로질러 필요 시 되는 핵심 기능을 기획 진행하였습니다

서비스 기획의 핵심 문제제기와 제공할 기능은 무엇인가?



peer를 사용할 사람들 중에는 당연히 목표 지향적, 그리고 시간을 아끼며, 고민하는 시간을 최소화 하는 것에 그 가치를 두고 있었습니다. 그러나 분석 과정에서도 드러났던 것처럼, 어떤 경우에는 학습의 경험도 중요시 하며, 상호작용 면에서 다양한 것들을 만나길 원하는 이들도 있습니다. 따라서 상대적으로 시간을 많이 쓰길 바라는 경우도 있습니다.

처음 보기엔 상당히 상충적인 요소들의 조합이란 생각을 했습니다. 그러나 peer 이용자들과 이야기하면서 알게 된 점은, 결국 '**효과적**으로 조건을 간추린다'는 점에서, 간추리고 나면 상충 요소는 사실 상충 요소가 아닌 '연관' 요소라는 생각을 할 수 있었습니다. 오히려 각 카뎃들은 **자신의 입장에서 '확실한 결정'을 하고 싶다**는 근본적인 마음은 동일하게 가지고 있다고 볼 수 있었습니다.

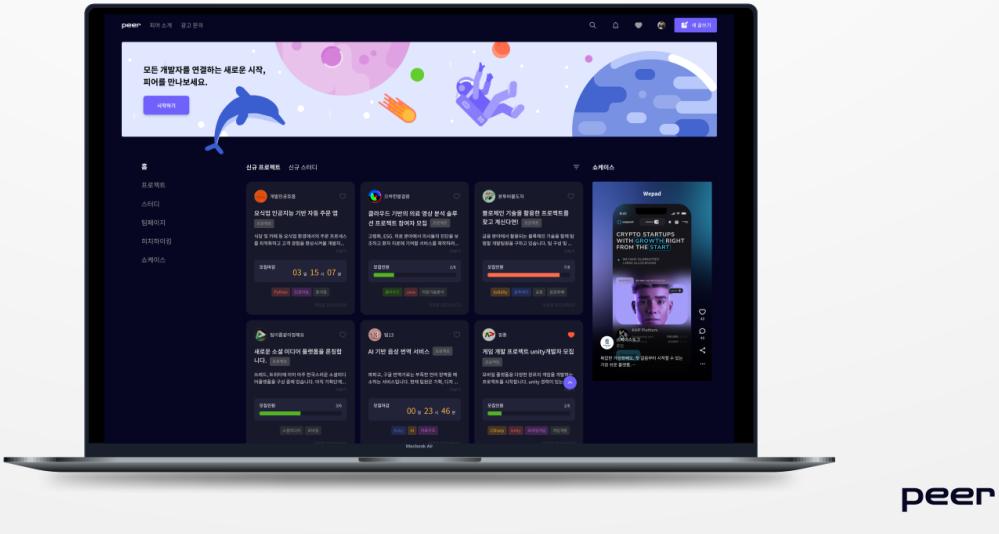
결국, 이러한 니즈들을 정리하다보면 몇 가지, 서비스가 필요로 되는 포인트를 생각해볼 수 있습니다.

1. 동료학습을 위하여 적절한 인원 찾기, 사람 파악하기가 어렵다. 스터디든 프로젝트든 상황에 맞게 세부적인 조건에 맞춰서 만들거나, 들어갈 수 있으면 좋겠다.
2. 검증된 사람을 구한다는 것은 제일 좋다. 하지만 사람에 대해, 커뮤니케이션에 대해 관심을 가질 상황이 많지 못하다.
3. 소프트 스킬의 인사이트 제공이 없다보니, 자기 자신을 객관화 하는 영역에서 사람마다 편차가 발생하며, 이러한 부분에서 긍정적인 개선이 쉽지 않다. 더불어 팀을 이끌거나, 팀 내에서의 발생하는 이슈들에 대한 '정답'에 대한 가이드가 있을 수는 없다보니 적절한 대처라는 것이 쉽지 않다.
4. 하지만, 3번의 조항에서 그걸 개선할 기회나, 방법을 제시한다고 할 때 이를 수용하겠다는 의견이 주를 이루었습니다. 이는 이미 성사된 프로젝트, 스터디를 성공적으로 끝내고 싶다는 점에서 기인했습니다.

구해야 할 것은 동료, 제일 어려운 것도 동료

동료학습, 프로젝트의 진행에서 항상 마주하는 고비이자 기회.

피어 레벨, personal color는 여러분이 원하는 팀, 스터디를 찾는데 도움을 줍니다.
완료된 프로젝트, 멋진 스터디가 있다면? 쇼케이스가 멋진 등대 역할을 해줄 것입니다.



현재의 개발을 학습하고, 깊이를 더하는 사람들에게 필요한 것은 효율성이라는 부분에선 차이가 없습니다. 더불어 취업을 준비하거나, 창업을 준비하는 등 일생의 큰 터닝포인트를 만드려는 이들은 '확실한 성장'과 '효과성', '효율성'이라는 키워드를 기반으로 프로젝트와 스터디를 진행합니다. 하지만 교육학적으로 보나, 인간의 성장 곡선등을 분석하는 많은 연구에선 기본적으로 '효율성'이 곧 '성장'이라고 직결시키지는 않습니다. **학습, 성장이라는 키워드는 보다 다른 영역과 마주하고 있으며**, 오히려 이러한 타 요소들이 없이 효과와 효율이라는 키워드가 연결되었을 때 생기는 '비효율성'은 많은 학자들로부터 문제시 되었으며, 선행 요소 이후에 연결 되었을 때, 비로소 제대로 그 진가를 발휘함을 시사하고 있습니다.

그러나 이러한 학술적 결과를 통해 서비스의 성공과 직결 되지 않는 것 역시 당연한 사실일 것입니다. 그렇다면 서비스는 어떻게 대처해야 하는가? 서비스에 사용자의 니즈는 실제 성장과는 별개의 길을 지향할 가능성이 있고, 그러는 와중에서 결과를 낼 수 있게 만들어 낼 수 있는 방법은 어디서 나오는가? 이에 대해 저희 peer 기획팀에서는 투 트랙 전략을 취하기로 합니다.

이용자를 따라가자

우선 이용자들의 요구사항은, 학문적, 이론적 성장과 직결되진 않습니다. 하지만 이용자의 수요를 만족시켜주지 않는다면 서비스는 성사되지 않습니다. 심지어 이러한 학문적, 이론적 기반으로 가능할 수 없는 부분, 사람의 가능성이라는 부분도 존재하고 있기 때문에, 무조건 이러한 논조를 따라 '옳은것'이라는 식으로 교조적(教條的)으로 기획을 정의 내릴 순 없었습니다.

그렇기에 우선은 사용자들의 니즈를 충족시킬 수 있도록, 현재의 상황에서 사용자가 원하는 부분들을 위하여 다양한 기능들을 준비 하였는데, 예를 들면 아래와 같습니다.

- 프로젝트, 스터디에 대한 세부 검색 기능 제공
- 프로젝트, 스터디에 대해 다양한 조건들 중 핵심이 될만한 내용들을 사전에 기록할 수 있도록 함으로써 조건에 맞는 글을 작성하거나, 찾아갈 수 있는 기능 구현
- 관심 리스트, 관심 키워드를 입력하고, 해당 내용에 맞는 제공해줄 수 있는 알림 기능 구현
- '히치 하이킹'이라는 이름으로 보다 조건에 합당하는 키워드를 가진 프로젝트 / 스터디를 찾을 수 있는 새로운 개념의 카드 UI 구축
- (개인정보 공유 등의 하에) 들어가고 싶은 팀, 팀에 합류하고 싶어하는 대상에 대하여 분석한 성격을 대변하는 피어 스코프의 기능으로 상대를 선 파악할 수 있게 하며, 학습, 프로젝트에 대하여 팀 전체를 통찰력있게 볼 수 있는 성향 분석 도구 '피어 레벨', '콜라보 레벨' 기능을 제공한다.
- 팀 협업의 특징을 살려, 도구들은 다른 것들을 사용하되 '허브' 역할을 할 수 있는 팀 페이지 기능 구현
- 팀 페이지에서 단순히 게시판 형태로만 사용되는 것이 아닌, 다양하고 복합적인 사용이 가능한 위젯 UX 구현

이용자들의 기본적인 욕구, 어쩌면 프로젝트, 스터디를 위한 서비스가 갖춰야 할 기본기라고 생각되는 영역이라고 기획팀에서는 생각했습니다. 특히나 여기서 신경을 가장 많이 썼던 부분은 '개발자'만의 영역으로 치부하고 끝나지 않도록 하는 것이었습니다.

개발자들은 개발이라는 특성을 가지고 있다보니 가지는 '개발자 스러움'이라는 독자적인 창조 행위에 대한 문화가 있습니다. 뿐만 아니라 개발자가 기획도, 디자인도 다할 수 있는 것도 아닌데 초기에 처음으로 개발의 프로젝트나 스터디를 할 때는 보통 이를 현실적인 조건 때문에 다 할 수 밖에 없다는 조건이 '치명적인' 프로젝트 / 스터디의 실패 요인이라는 분석도 있었습니다. 따라서, 이러한 부분에서 개발자 이외의 기획자, 디자이너도 함께할 수 있도록 만들기를 원했습니다

이용자를 도와주자

peer 팀의 생각, 기획, 목표를 보면서 여러 사람들이 모일 수 있었습니다. 특히나 42서울의 특징, 정말 다양한 기준 분야에서 몸 담고 계신 분들이 참여해주었습니다. 이 글을 작성중인 haryu를 포함하여, 교직에 근무하셨던 분, 그 밖에도 다양한 자문을 받았던 분들의 이야기를 들어왔습니다. 이를 통해 알 수 있는 프로젝트, 동료 학습과 같은 행위에서 '성공적일 조건'이라고 하는 것은 사실 '효율성'과 그 관련성이 명확하지 않다는 것은 다소 명확해 보였습니다.

그렇다면 사용자들을 설득해야 할까요? 사용자들에게 가르치면 해결될 것인가? 이것은 결코 좋은 관점이 아닙니다. 이용자, 소비자란 냉정한 법입니다. 이러한 방식은 서비스의 진가를 보여주는 방식은 최악의 접근법입니다. 자연스럽게 그들이 공감하도록 만들거나, 주장하는 가치가 그들의 과정에서 알게 모르게 침가 되어가며, 성공적인 사용자의 '이익'을 얻게 만듦으로써 오히려 그들이 서비스가 괜찮단 생각을 하도록 만드는 것이 이상적인 결론일 것입니다.

그렇기에 저희 기획은 위에서 우선은 이용자들을 따라가면서도, 그들이 제대로 방향성이 될만한 행동들이 자연스럽게 유도되도록, 특히 '선순환' 되어가는 기획이 되어질 수 있도록 peer 웹 어플리케이션의 형태를 가꾸어 갑습니다.

- 프로젝트와 스터디를 해나감에 있어 놓치기 쉬운 커뮤니케이션에 대한 조언이나 힌트, 대인 관계의 '멘토링'이 필요하다. 따라서 이를 자연스럽게 하는 LLM AI의 알고리즘 구축, 이를 위한 프롬프트, 데이터 체인 모델 등을 구축(피어스코프, 피어레벨, 콜라보 레벨과 연계)
- 작업의 내용을 보다 잘 알리고, 잘된 케이스들을 '홍보'할 수 있는 수단으로 '피어로그', '피어 쇼케이스' 기능 제공
- 추후 인공지능을 적극 활용하여, 문서 작업시 보조 역할을 하는 '독스 어시스팅'을 비롯 사용자의 상황에서 난감하거나, 어떤 방향성으로 접근하면 좋을지 난감해지는 영역에 맞춰 성하율 고려한 LLM AI의 어시스팅 기능 탑재 예정

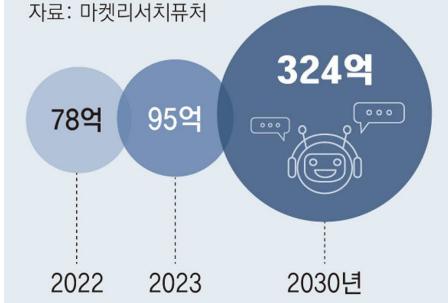
LLM 인공지능의 성장은 눈부십니다. 뿐만 아니라 초장에서 언급했듯 현재의 협업의 가장 핵심은 무엇인가? 성장은 어디서 오는가? 커뮤니케이션 능력, 성공과 그 트랜드를 제대로 본인의 프로젝트, 스터디에 녹아내기에는 누군가의 일방적인 가르치는 구조로는 해결하기 어렵습니다. 유저를 교조적으로 가르친다고 해결될 문제도 아닙니다. 하지만 이때 인공지능이 제공할 수 있는 지식 제공 능력과, 인문학적 접근법의 능력치는 이미 심리학자나 저명한 트레이너들의 그것 을 아득히 뛰어넘었습니다.

예를 들어보면, 사람으로 구성된 CS(Customer Service)팀을 구축하고, 이들에게 특정 내용을 사용자에게 전달하고자 하는 순간, 인간의 심리 기재는 놀랄만큼 다양하며, 동시에 사회적 상호작용이라는 형태에서 생기는 난감할 정도의 변수들이 발생합니다. 하물며 그러한 팀 구성원들 조차 사람이므로 자신의 감정, 정신 상태 등으로 매번 같은 답변, 적절한 대처가 가능한지에 대해서 매우 확실하게 '불가능' 하다고 볼 수 있습니다.

하지만, 거기서 성공적인 케이스들을 확률적으로 알고 있는 AI는 어떤 상황에서도 같은 답변으로 유도해냅니다. 하물며 이를 사용하는 사람도, 결국 '필요한 정보'를 얻기로 온 상태이며, 대상이 AI이자 자신에게 최적화된 방식으로 내용을 전달한다고 하면? 거기서 그들이 사람을 대한다, 사회적 상황에서 그릇된 반응이나, 폭발적인 반응이 발생할 일도 없습니다.

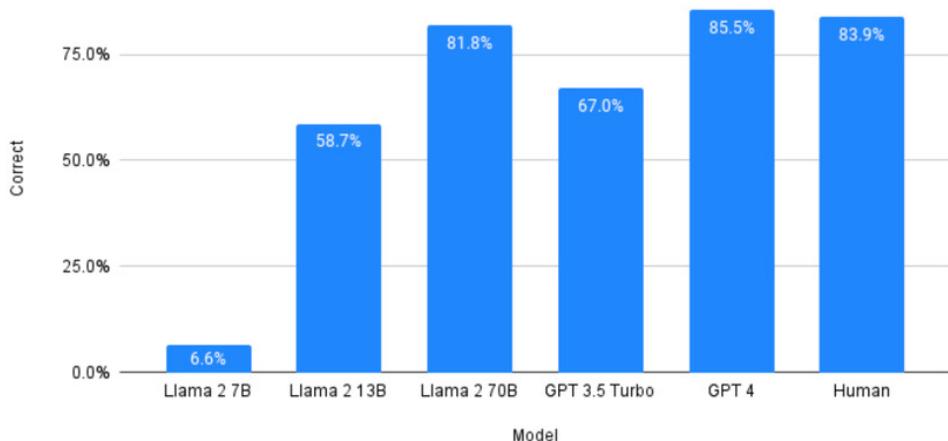
대화형 AI 서비스 시장 전망

단위: 달러
자료: 마켓리서치컴퍼니



이걸 증명하듯, 현재 심리 상담을 기반으로 하는 LLM, 대화에 최적화된 AI 시장은 이미 그 시장성을 증명한 상태입니다. 뿐만 아니라, 학습이나 프로젝트 등 무언가를 성취하기 위한 팀플레이 상황에서 심리적 제어나 조언, 이러한 것들을 통해 팀 내 갈등을 효과적으로 제어 가능하다면? AI가 가지는 가공할 특성은 단순히 피어가 프로젝트, 스터디를 구하기만 하는 서비스가 아니라, 향후 AI를 통해 프로젝트나 스터디, 구성원들 하나하나에게 최적화된 심리 컨설턴트가 되어 주는 것도 꿈이 아닙니다.

이미지 출처 : "AI와 대화하며 심리 상담"... '감성 대화' 서비스에 月 2억명 방문 - 동아일보



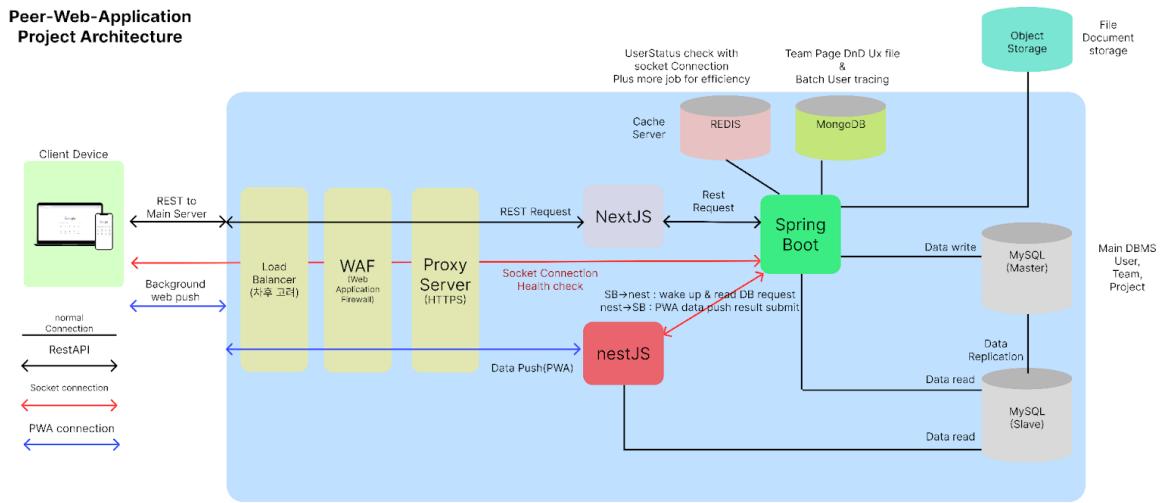
이미지 출처 : How Does Llama-2 Compare to GPT-4/3.5 and Other AI Language Models by Prompt Engineering Institute

뿐만 아니라 기술적 구현에 있어서도 OpenAI라는 거대한 플랫폼 하에 지배당하지 않을, 기술 부채면에서도 상당히 긍정적인 전망이 나오고 있습니다. 기존의 ChatGPT를 필두로 폐쇄적인 AI의 성능을 오픈소스에선 도저히 따라갈 수 없다고 생각했습니다. 하지만 오픈소스 학습모델들도 학습의 방식을 통해 그 정확도가 매우 급성장한 것을 알 수 있으며, 현재 인간의 평균 83.9%는 이미 준하는 수준까지 오게 되었고, 이러한 성장, 인공지능의 fine tuning은 상용 서비스에 충분히 구축이 가능한 수준까지 도달했습니다.

더불어 이를 구동하도록 만들 하드웨어 시장도 기존에는 병렬 연산 GPU 구동을 베이스로 삼고 있었지만, 점차 GPU의 가격 경쟁력의 부족과 더하여 CPU를 통한 병렬 연산처리, 메모리 사용 방식의 최적화 등의 다양한 기술적 변화 역시 AI의 성능, 구동의 시장성을 확보해나가고 있는 실정입니다.

전문적인 심리 기술들을 추가학습시키고, 저희가 만들어낸 인공지능 알고리즘을 활용하여 제공되는 기능들, 특히나 매우 싸거나 사실상 서버 비용을 제외하고 사용이 가능하다고 한다면, 이러한 양질의 어시스템은 개발자, 기획자 등을 비롯하여 프로젝트 / 스터디에서 동료학습을 보다 선명하게 강화시킬 수 있는 것입니다.

서비스 아키텍쳐와 기술제반 설명



기획이 아무리 잘 나와도 그것을 소화시키는 것은 기술의 제반합니다. 기술의 완성도, 구축을 위한 시스템 설계가 없이 기획이 성공적일 수는 없을 것입니다. 구현할 방법을 고민하지 않는다면 개발은 성사될 수 없을 것입니다.

이 부분에 시니어들의, 정말 제대로 아는 사람들이 해야 할 일일것입니다. 그렇기에 부족하고, 아쉬운 실력이겠지만 최선을 다했다는 부분을 먼저 알려드립니다. 또한 이 설계를 구현하는데 영감을 주신 Datus 의 우광명 이사님의 조언은 정말 감사 했습니다. 그분이 제시해준 '대규모 시스템 설계 기초'라는 서적은 정말 백엔드 계의 꽃이라고 부를 수 있는 서버 아키텍쳐의 구조와 어떤 내용들을 고려해야하는 부족한 시각을 넓혀주는데 정말 큰 도움이 되었습니다.

해당 아키텍처를 설계하면서 핵심은 '감'을 잡는데 있었습니다. 처음 백엔드 설계를 시작하고, 여러 사람들의 이야기를 들으면서 백엔드 개발을 한다고 하면서도, 시스템, 하드웨어의 성능이 얼마나 필요할지, DAU 목표를 기준으로 최소 어느정도 성능을 유지할 수 있으면 되는것인지. DB가 어느정도 하드웨어를 사용하며, 어느 수준의 연산 성능이 필요한가? 이러한 부분들은 전혀 감이 없는게 사실입니다. 그렇기에 아키텍처에 대한 학습을 하면서도 안정적이고, 빠른 서버를 구축하는 것이 어떻게 구조를 짜면 되는가? 또한 서비스를 위한 기획을 구현하기 위해 여러 기능이 필요할텐데 그러한 통신은 도대체 어떻게 짜야 하는가? 하물며 보안은? 그냥 단순히 입력 폼에 SQL인젝션으로 충분한 것일까? 등, '감'을 잡는 것은 이 아키텍처 설계에 있어서 핵심이라고 해도 과언이 아닐 것입니다.

우선, 구현의 핵심은 다음과 같은 기능 조건이 있었습니다.

- 기본적인 웹 서비스를 위한 구조를 가진다.
- 프론트 개발자가 백엔드 개발자 없이도 개발이 가능한 기능적 조건을 만들어내야 한다. 구체적으로는 팀 페이지의 위젯이 작성 가능하도록 만들어내야 한다.
- 보안을 고려해야 함과 동시에 PWA 어플리케이션을 지향하는 설계가 필요하다.

- 서비스에서 유저의 연결 상태를 파악할 수 있어야 하며, HTTP 통신으로 구현하여 생길 프론트 개발의 난이도 상승 내지는 개발의 유지 보수성을 개선할 방안을 마련해야 한다.
- 기본적인 웹 서비스용 서버의 부담이 가중되지 않으며, 자주 데이터 입출력이 존재하는 행위 등에 대해 처리는 가능한 한 해당 성격에 맞는 기술 방식을 접목시켜야 한다.
- DB와 같이 성능이 민감한 포인트, 보안 등을 고려하여 권한에 맞춰 계정 분배, DB 성능 최적화 기법을 활용한다.
- 기본적인 웹 서비스를 위한 구조를 가진다.
- 프론트 개발자가 백엔드 개발자 없이도 개발이 가능한 기능적 조건을 만들어내야 한다. 구체적으로는 팀 페이지의 위젯이 작성 가능하도록 만들어내야 한다.
- 보안을 고려해야 함과 동시에 PWA 어플리케이션을 지향하는 설계가 필요하다.
- 서비스에서 유저의 연결 상태를 파악할 수 있어야 하며, HTTP 통신으로 구현하여 생길 프론트 개발의 난이도 상승 내지는 개발의 유지 보수성을 개선할 방안을 마련해야 한다.
- 기본적인 웹 서비스용 서버의 부담이 가중되지 않으며, 자주 데이터 입출력이 존재하는 행위 등에 대해 처리는 가능한 한 해당 성격에 맞는 기술 방식을 접목시켜야 한다.
- DB와 같이 성능이 민감한 포인트, 보안 등을 고려하여 권한에 맞춰 계정 분배, DB 성능 최적화 기법을 활용한다.

기획이 아무리 잘 나와도 그것을 소화시키는 것은 기술의 제반입니다. 기술의 완성도, 구축을 위한 시스템 설계가 없이 기획이 성공적일 수는 없을 것입니다. 구현할 방법을 고민하지 않는다면 개발은 성사될 수 없을 것입니다.

이 부분에 시니어들의, 정말 제대로 아는 사람들이 해야 할 일 일것입니다. 그렇기에 부족하고, 아쉬운 실력이겠지만 최선을 다했다는 부분을 먼저 알려드립니다. 또한 이 설계를 구현하는데 영감을 주신 Datus 의 우광명 이사님의 조언은 정말 감사 했습니다. 그분이 제시해준 '대규모 시스템 설계 기초'라는 서적은 정말 백엔드 계의 꽃이라고 부를 수 있는 서버 아키텍쳐의 구조와 어떤 내용들을 고려해야하는 부족한 시각을 넓혀주는데 정말 큰 도움이 되었습니다.

해당 아키텍쳐를 설계하면서 핵심은 '감'을 잡는데 있었습니다. 처음 백엔드 설계를 시작하고, 여러 사람들의 이야기를 들으면서 백엔드 개발을 한다고 하면서도, 시스템, 하드웨어의 성능이 얼마나 필요할지, DAU 목표를 기준으로 최소 어느정도 성능을 유지할 수 있으면 되는것인지. DB가 어느정도 하드웨어를 사용하며, 어느 수준의 연산 성능이 필요한가? 이러한 부분들은 전혀 감이 없는게 사실입니다. 그렇기에 아키텍쳐에 대한 학습을 하면서도 안정적이고, 빠른 서버를 구축하는 것이 어떻게 구조를 짜면 되는가? 또한 서비스를 위한 기획을 구현하기 위해 여러 기능이 필요할텐데 그러한 통신은 도대체 어떻게 짜야 하는가? 하물며 보안은? 그냥 단순히 입력 폼에 SQL인잭션으로 충분한 것일까? 등, '감'을 잡는 것은 이 아키텍쳐 설계에 있어서 핵심이라고 해도 과언이 아닐 것입니다.

이러한 점을 종합하여 다음 형태를 구축하였습니다.

(1) <p>메인 서버는 Spring Boot 를 활용하였습니다. main 서버는 비즈니스 로직의 기본을 담당하며, 사용자의 소통 등으로 데이터의 입출력이 많은 쪽지 영역에 대해서는 가능한 성능적 최적화를 위하여 비동기-비봉쇄 방식을 추가하여 구현하였습니다.</p>	(2) <p>서브 서버인 NestJS 서버의 경우, 사용자의 활동에 따라 지속적으로 있을 PWA 알림 부분을 담당하게 되며, 이 역시 비동기 이벤트 방식으로하여 가능한한 많은 데이터 입출력을 상정한 기술 스택을 상정하게 되었습니다.</p>
(3) <p>더불어 1, 2번에서 사용되는 DB 계정은 다르게 하여, 쓰기를 담당하고, 마스터 역할을 하는 계정과, 알림 이벤트를 읽는 계정으로 구분하였습니다.</p>	
(4) <p>DB의 성능의 향상과 대용량 처리를 고려하는 설계를 위해 DB Replication 을 도입하고자 하며, Master DB 서버로 MySql 을 활용하고 Slave 서버를 한대 기본으로 두며, 최종 목표는 Auto Scalable 한 DB 구축을 해내는 것을 목표로 합니다.</p>	(5) <p>NoSQL 을 활용하여 팀 위젯 UX 를 구축함에 있어 데이터 형태, 프론트와 백엔드의 규약이 필요하지 않은 데이터 저장 공간을 구축하였습니다. 이를 통해 프론트엔드 개발자들이 백엔드 개발자들과의 논의 없이도, 사용자들을 위한 다양한 위젯 개발이 가능하도록 구축하였으며, 일부 API 논의가 필요한 위젯을 제외하곤 백엔드와 상관없이 개발이 가능해졌습니다.</p>
(6) <p>Spring Boot 서버는 사용자가 온라인, 오프라인 상태인지를 파악해야 하나, 이러한 영역에 대해 개발자가 직접 무언가 작업하지 않아도 되도록 Socket.IO 라이브러리를 활용하였습니다. 이를 통해 사용자의 상태를 Stateless 한 상태에서도 파악이 가능합니다. 뿐만 아니라 프론트에서 REST API 를 구축하는데 있어서 만약 사용자의 정보를 파악하고 알아야 하는 순간에 REST API로만 했을 때 생길 수 있는 보안 문제, 라우팅 되는</p>	매 페이지 마다 이를 위한 코드가 필요하다면, Socket 을 이용한 양방향 통신 구축으로 최상위 노드에서 기본적인 연결이 성사 된 이후에는 간단한 이벤트 처리와 수신 처리 만으로도 사용자에 대한 정보, 바로 필요한 정보들을 받아 볼 수 있도록 처리가 가능합니다. 이로써 Rest API를 따로 짜고, 데이터 관리하는 등의 번거로움을 최소화 시켰습니다.
(7) <p>Nginx 를 활용하고, Hosting 업체를 통해 신뢰할 만한 도메인을 할당 받았으며, 이를 통해 HTTPS SSL을 발급 받고, 각 서버들에 맞추어 적절한 포워딩 조치를 취해주었습니다.</p>	(8) <p>서버의 성능 향상을 위해 데이터 중 지속적으로 사용하는 데이터에 대해서는 Redis 를 통한 물리적인 보조기 역장치를 거칠 때 생기는 입출력 대기 시간을 최소화하고자 노력했습니다.</p>

<p>(9)</p> <p>그 외에도 입출력 성능향상 및 서비스 사용자의 사용 경험을 확보하고자 아마존 S3에 호환되는 Object Storage라는 nhn Cloud의 CDN 서비스를 활용하였으며, 구현되는 비즈니스 로직에 맞추어 적절한 기술을 적용하고자 노력하였습니다.</p>	<p>(10)</p> <p>이러한 내용을 전체를 포함하여 모두 기본 nhn Cloud에서 동작하게 구현 후, 온프로미스 하드웨어 서버를 구축하였습니다. 이를 통해 서버 비용을 최소화하고, 향후 지속적인 업데이트가 가능한 CICD를 GitHub 로 구축하였습니다</p>
<p>(11)</p> <p>향후에는 step 3 개발 포인트를 지정하고, 이를 위한 보다 복합적인 기술을 검증 및 적용할 예정이며, 최종적으로 auto-scaling 을 지원하는 쿠버네티스를 적용하고, 기존 서비스들중의 독립적으로 분리 가능한 서비스들을</p>	<p>분리해내는 MSA 아키텍처로의 전환을 예정하고 있습니다.</p>

피어 웹 어플리케이션이 가지는 의미

피어의 시작은 딱히 대단히 특별하지 않았습니다. 그걸 수행하는 사람들 중 저는 특히나 비전공자로써 그렇게 재능이 넘쳐나는 사람은 아닙니다. 그럼에도 42서울에서 배우고 경험하고, 동료학습의 장점과, 그걸 알리고 싶은 생각들, 그리고 그러는 와중에 생기게 된 기회들이 켜켜히 쌓여 드디어 하나 제대로 만들어보자! 라는 의지와, 생각으로 여기까지 올 수 있었습니다.

peer 웹 어플리케이션은 그렇다면 정말 잘 만드는 과정을 거쳤는가? 결과물이 참 괜찮은가? 사실 여기에 답은 부끄럽게도 No입니다. 부족하고, 열악했습니다. 배움이 짧았으며 고민이 짧았기에 개발의 이어지는 연기와 함께가던 동료들 중 떠나는 이들도 생기고, 새로이 합류하는 이들도 생기면서 지금의 상황까지 올 수 있었습니다.

기획은 온전하지 못했으며, 당초 계획보다는 다운사이징을 할 수 밖에 없는 기능들, 그리고 이를 구현한다고 해도 현실적으로 드는 서버의 비용등을 계산하면서 과연 실제 서비스를 구현해낸다는게 얼마나 힘든 일인가? 를 새삼 깨닫는, 그리고 동시에 이려는 와중에 동접자 10만명, 100만명을 구현하는 서비스들이 정말 얼마나 많은 노하우와 깊이를 알 수 없는 고민들이 들어갔을지를 생각하니 정말로 개발자 선배님들의 노고와 그 다음을 걸어가고 싶어하는 내 자신의 부족함을 배울 수 있었습니다.

부족함을 더 알고, 부족함 속에서 더 많은 욕구를 발견하고, 무엇보다 이걸 해나가기 위해 다수의 사람들의 다른 생각들 속에서, 이것들을 하나로 뭉치게 만든다는 것, 이런걸 위해 노력하는 이 시간들은 정말로 값진 경험이었습니다. 더불어 이렇게 해 나가는 과정에서 감사하게도 믿어주는 이들 덕에 리더로 세워지고, 리더들끼리 함께 해 나가면서 서로 힘들고 외로웠던 순간들, 그 순간들이 리더들을, 저를 더 강하게 성장시키는 순간들이 아니었나 생각해봅니다.

2월 5일 베타 런칭을 하게 되고, 드디어 새로운 지평이 열리게 됩니다. 본 서비스가 계속 운영될 수 있을지 없을지, 무엇보다 목적지로 찍은 그 지점까지의 개발이 달성될지도 잘 모르겠습니다. 하지만 확실한 것은 믿어주는 동료들과 함께 경험하고 성장했던 진실, 진심이 이걸 써서 변화와 성장하고 싶은 모든 분들의 기회로 자리매김 하기를 바라며, 함께한 동료들, 믿어주고 함께해준 모든 멘토분들에게 정말로 감사의 말씀을 전합니다.



peer UX/UI 가이드

Peer UI/UX 가이드를 만든 이유

프론트엔드 개발자에게 기술적인 면은 점점 더 그 중요성을 잃어갑니다. 웹 기술의 개발, 하드웨어 성능의 향상을 비롯한 다양한 라이브러리들의 발전은 웹이라는 가장 중요한 도구의 '제작'에서 생기는 제약을 개선하고, 더 이상 '창작'의 영역이 아니라 '편집'의 영역으로 만들고 있습니다.

거기서 우리가 생각해야 하는 것은 '방향'일 것입니다. 무엇을 목표로 삼을 것인가. 기술에 먹히는 인간이 될 것인가, 기술을 선도하는 인간이 될 것인가. 이미 우리의 시대는 자동화와 인간을 압도하는 효율성의 시대에 들어간 이상, 지향해야 할 바는 정해져 있습니다. 더욱이 가장 인간다운 행위를 하지 않을 수 없습니다.

도구의 제작, 편집이 프론트엔드 개발자의 영역에 들어와 버렸고, 거기서 인간의 감수성, 창작성, 무언가를 펴내는 행위라고 하는 것 이 제약이 생길 수록 우리는 보다 '치밀한' 가치를 이해할 줄 알아야 하며, AI가 하는 통계적, 확률적 세상에서 인간의 심리를, 인간의 '미'와 인간 자체에 대한 이해를 높인 디자인, 프론트적 가치를 추구하여서 여전히 '창작자'가 되는 것이 어쩌면 프론트엔드 개발자가 해나가야 하는 길이 아닐까 합니다.

결국, 우리는 가치를 논해야 합니다. 보다 디테일한 가설과 생각으로 UX/UI를 바라봐야 하며, 개발자가 가치는 가치와 철학을 계속 풍부하게 만드는 것이 가장 중요할 것입니다. 또한 현재 취업 시장이 더 이상 '기술'적 인간상을 원하는 것은 '기본'이 된지 오래입니다. 기술을 배우는 인간의 능력치는 정해져있고, 누구나 배울 수 있는 시대가 된 이상 천재적 기술력이 아닌 이상 우리의 가치는 동일할 수 밖에 없습니다.

거기서 확신하는 것은 다음과 같습니다.

자신이 하는 행위를 스토리로 묶어 낼 줄 아는 것, 어떤 문제를 깊이있는 고민으로 접근해본 경험이 있는 것, 그런 인간적 매력과 협응력이 뛰어나기에 보다 복잡하고 깊이있는 문제를 해결하는데 두려워하지 않는 개발자야말로 지금 이 시대에 '코더'가 아닌 개발자가 되는 길일 것입니다.

앞에 있던 장황한 내용을 모두 포함하여 저희 피어 프론트엔드 개발팀은 반드시 지향해야 할 것은 다음과 같습니다.

1. 인간 친화적 UI/UX 개발 경험을 만든다.
2. 이를 위한 문서화 작업을 통해 학생이 상상해보지 못할 고민을 실제로 해보는 것.
3. 그것을 묶어내 사람들에게 보여줄 무언가를 구체화 시키는 것.

이를 위한 디자인 가이드이며, 궁극적으로 '피어의 철학을 담은 디자인'을 만드는 것을 목표로 삼읍시다!

Peer 의 핵심가치

접근성 저희는 42서울 내에서 제대로 된 동료학습을 원하는 카뎃들이 부담없이 다가올 수 있게 해야합니다.	확장성 42서울 내부 사람뿐만 아니라 42서울에 관심을 갖고 있는 외부인들(현직자, 블랙홀에 빠진 전 카뎃 등등)도 사용할 수 있어야 합니다	생산성 스터디나 프로젝트에 필요한 필수적인 요소들을 제공하여야 합니다.
신뢰성 사용자의 니즈를 충족시킬 수 있는 신뢰를 주어야 합니다.	포용성 제로 베이스부터 현직자까지 모두가 같이 성장할 수 있는 환경을 제공하여야 합니다.	

UX/UI 가이드라인

Apple HIG를 많이 차용하고 참고해서 작성했습니다. 그 이유는 다음과 같습니다.

1. 사용성에 치중한 가이드라인

- 가독성 높은 텍스트
- 이해하기 쉬운 아이콘
- 중요한 요소를 강조하는 명확함
- 콘텐츠 이외의 요소가 더 부각되지 않게

이러한 요소들은 피어 개발자에게 중요한 가치로 생각했고 이를 잘 녹아든 Apple Human Interface Guide를 공부하면 되겠다고 생각했습니다.

2. 개발 과정에 적합한 가이드라인

모바일부터 구현하는 개발 과정과 일관성, 직접 조작, 피드백, 은유 등 이러한 디자인 원칙은 개발에 적용하기 적합하다고 생각했습니다. 개발자와 디자이너에게 명확한 지침을 제공함으로써 개발을 할 때 실제로 사용이 가능하다는 점과 애플이라는 하나의 기업이 오래동안 수정과 삭제를 거듭하면서 일관성 있는 디자인 철학과 원칙이 있기에 UX/UI 가이드라인을 처음 제작하는 저희에게는 좋은 교보재라고 생각했습니다.

0. 디자인 원칙

사용자 중심

사용자들의 쉽고 직관적인 인터페이스를 통해 불편함을 최소화합니다. 사용자의 관점에서 디자인을 생각하고 기능을 효과적으로 배치하세요.

접근성

모든 사용자들이 쉽게 사용할 수 있도록 접근성을 고려한 디자인을 도입하세요. 텍스트 크기, 색상 대비, 키보드 탐색 등을 고려합니다.

일관성

배경색상부터 버튼 스타일, 폰트 크기에 이르기 까지 일관된 디자인 시스템을 맞춰 사용자들이 익숙해지기 쉽게 만듭니다

단순하고 간결한 디자인

복잡하지 않은 디자인으로 사용자들이 원하는 정보나 기능을 빠르게 찾을 수 있게 해주세요. 불필요한 요소는 제거하고 중요한 기능이나 정보를 강조합니다.

피드백 제공

사용자의 행동에 대한 피드백을 제공하여 사용자에게 현재 상태를 명확하게 인식하도록 도와주세요. 예를 들어, 버튼 클릭 시 애니메이션으로 효과를 나타내거나, 진행 상황 표시 등입니다.

가이드 및 도움말 제공

사용자들이 헷갈리거나 어려울 수 있는 부분에 가이드 또는 도움말을 제공해주세요. 이를 통해 쉽게 서비스를 사용할 수 있도록 돋습니다.

1. 웹 / 앱 아키텍쳐

<손쉬운 사용>

능력이나 상황에 관계없이 모든 사람이 정보를 사용할 수 있도록 하는 것으로, 접근성과 단순성이라는 파트로 나뉩니다.

접근성

스캔 용이성: 사용자가 페이지를 빠르게 스캔할 수 있도록 중요 정보를 시각적으로 돋보이게 합니다.

정보의 가시성: 중요한 정보는 큰 글씨, 대조적인 색상, 볼드체를 사용하여 강조합니다.



NO

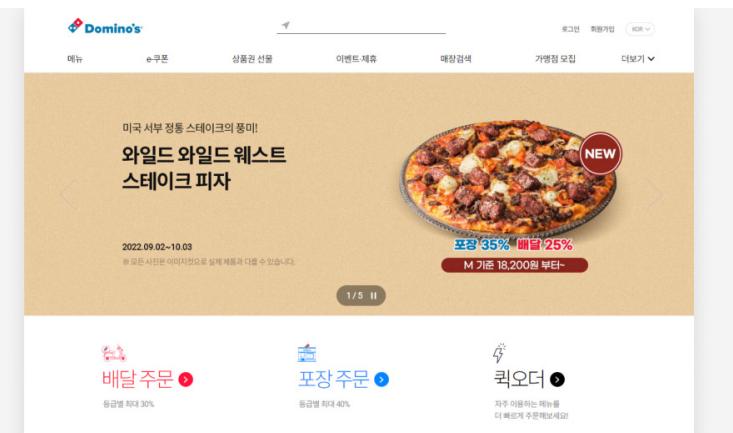
YES

(지만 아이콘이 애매하게 많음)

나를 괜찮은 구조지만
너무 단색의 구조

시각적 계층 : 중요한 메시지나 기능은 색상, 크기, 위치를 통해 강조합니다.

행동유도 : '핵심 행동'과 '부수적인 행동'을 나누어서 핵심 행동을 최대한 유도하세요. 즉 사용자가 자신이 원하는 것을 빠르게 이를 수 있는 행동을 명확하고 찾기 쉽게 만들면 됩니다. 핵심 행동은 색상, 크기, 위치를 통해 눈에 띄게 합니다. 스터디와 사이드 프로젝트의 참여를 유도해야 합니다.



배달주문과 포장주문을 유도



We will only send you love and sometimes some words to go with it.

Email Address

Submit

Email: carissapottercarlson@gmail.com

© 2017 Carissa Potter. Proudly created with WIX.com

submit을 유도하는 버튼

반응형 디자인 : 사용자가 어떤 환경에서도 피어에 접근할 수 있게 해야합니다. 다양한 화면 크기와 장치에서의 호환성을 고려합니다. 모든 사용자가 일관적인 경험을 할 수 있게 합니다.

접근성

복잡한 작업을 간단하게 수행할 수 있도록 친숙하고 일관된 상호 작용을 지원해야 합니다.

1. 직관적 상호작용

버튼과 필터 사용자가 선택할 수 있는 명확한 버튼과 필터를 제공합니다.

사용자가 원하는 결과가 빠르게 나올 수 있게 만드세요.

입력보단 '버튼'이 편하고, 검색보단 '필터' 적용이 간편합니다.

일관된 배치 사용자에게 익숙한 위치에 버튼이나 메뉴를 배치합니다.

사용자가 스스로 학습하고 스스로 익히기 편한 환경을 조성해야 합니다. 우측 하단의 탭바에 '내 프로필' 버튼이나 '설정' 버튼이 있는 건 매우 당연하고 익숙합니다. 이처럼 일관성 있는 위치에 버튼이 존재해야 합니다. 일상에서 쓰이는 규칙을 최대한 이용합시다.

적중 대상 크기 버튼과 대화형 요소는 충분히 크게 설정하여 사용 용이성을 높입니다.

모든 컨트롤과 대화형 요소(버튼, 토글 등등)에 충분히 큰 적중 대상을 지정합시다.

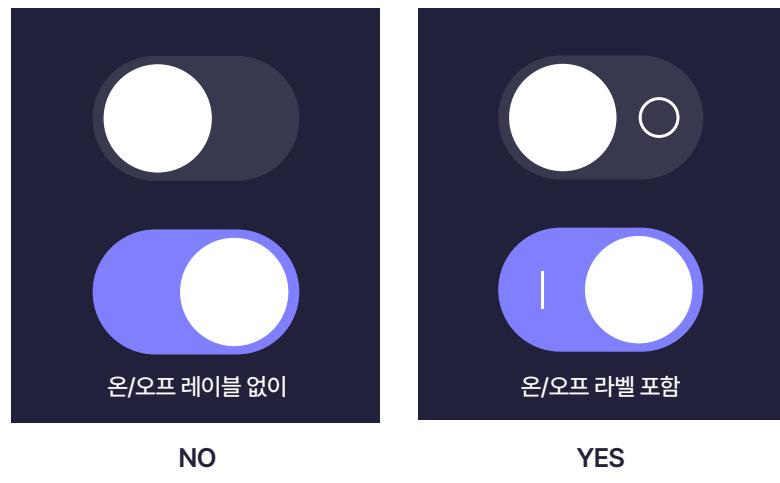
좋은 버튼의 크기는 7mm로, 웹에선 44px(2.75rem)입니다.

CTA 버튼 중요한 버튼은 눈에 띄게 디자인하고 수를 최소화합니다.

- 중요하고 핵심적인 버튼은 엄지존에 위치하게 하고 CTA(Call to Action)를 고려해서 색상을 넣습니다. CTA를 강조한 버튼의 개수는 적을수록 좋습니다.
- 강조를 위해 contained 속성을 이용하는 것도 방법입니다.

색상 사용 버튼과 대화형 요소의 구분을 위해 색상을 이용합니다.

- 배경과 구분을 주기 위해서 배색을 주어서 버튼을 만듭니다.
- 일관된 스타일 계층 구조를 사용하여 버튼의 상대적 중요성을 전달합니다.
- 삭제 버튼은 빨간색을 이용해서 표현합니다.
- 평소 흰색 배경의 버튼을 사용한다면 중요한 버튼은 색상을 넣어서 구현합니다.
- 시스템 제공 스위치를 선호합니다.(어플의 경우)
- 색상은 일관성을 위해 최대한 테마 색상을 사용합니다.
- 색상 외에 도형 등으로 시각적 표시기를 제공하세요.



NO

YES

2. 사용자 입력

텍스트 입력 최소화

- 사용자가 입력하는 걸 막지 말되 입력할 부분을 최소한으로 줄여봅시다. 입력이 필요한 경우에만 입력창을 두고 사용자가 직접 입력할 때 방해되지 않도록 레이아웃과 내부 항목을 정합니다.
- 사용자가 입력을 할 때 방해하는 요소를 최소화 해야합니다.
- 입력하는 키보드에는 자판 이외의 것들을 최소화 합시다.
- 작성 중, 완료 버튼을 위에 배치하여 의도치 않은 행동을 사전에 방지합시다.

텍스트 표시

- 디자인이 확장 가능하고 모든 글꼴 크기에서 읽을 수 있는지 확인하십시오. 이를 위해선 rem 단위를 이용해야 합니다.
- 글꼴 크기가 커지면 텍스트 잘림을 최소화하십시오.
- 글꼴 크기가 커질수록 의미 있는 인터페이스 아이콘의 크기가 커집니다. 인터페이스 아이콘을 사용하여 중요한 정보를 전달하는 경우 더 큰 글꼴 크기에서도 쉽게 볼 수 있는지 확인하십시오.

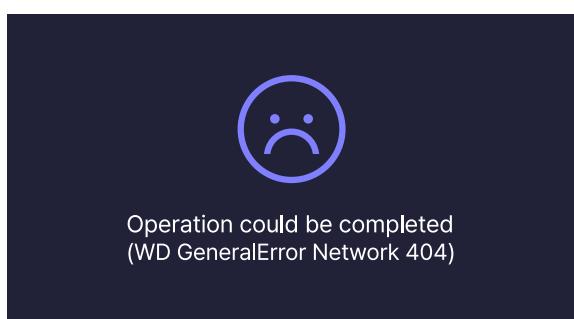


- 현재 글꼴 크기에 관계없이 일관된 정보 계층 구조를 유지합니다.
- 일반 또는 무거운 글꼴 두께를 선호합니다. 일반, 중간, 세미 볼트 또는 볼드 글꼴 두께를 사용하면 더 쉽게 볼 수 있습니다.
- 굵은 텍스트를 켰을 때 앱이 올바르게 응답하고 잘 보이는지 확인하세요.
- 글꼴 두께와 글꼴 명암으로 계층을 만드세요. 강조하기 좋은 방법입니다.

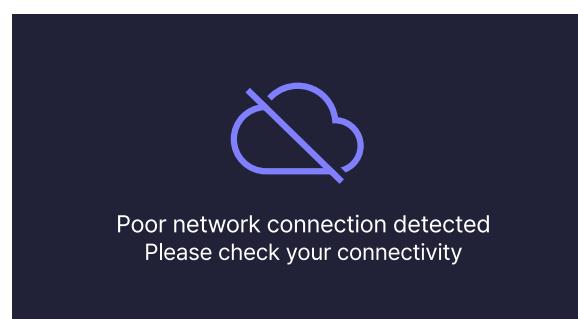
▶ 아래 부가 설명이 영상의 제목보다 덜 강조되는 걸 볼 수 있습니다.

3. 문제 상황 대응

- 사용자의 전 행동을 기억해서 문제 상황 이후 그 행동이 연속될 수 있게 합니다.
개설하기 → 로그인 → 메인화면 (X) / 개설하기 → 로그인 → 개설하기(O)
- 사용자가 덜 입력할 수 있게 합니다.
예제) 회원가입 시 입력하다가 잘못 눌러서 입력 칸이 다 비워지는 경우
- 문제상황이 생겼을 때 바로 해결할 수 있게 사용자를 자연스레 유도합니다.
적당한 계층 구조와 색상의 대비로 강조합시다. 피드백을 바로 줄 수 있게 합시다.
- 적절한 에러 문장으로 사용자가 어떤 오류에 나왔는지를 알리고, 이에 대한 다음 행동을 자세하게 유도합니다. 이때, UX writing이 필요합니다. (QA 단계에서 계속 수정하여 고쳐봅시다.)



혼란스러운 에러 메세지



정확한 에러 메세지

<로딩 화면>

사용자가 과정이 진행 중임을 알 수 있게 명확하게 표현하세요. 애니메이션을 넣는 방식이 대표적입니다.

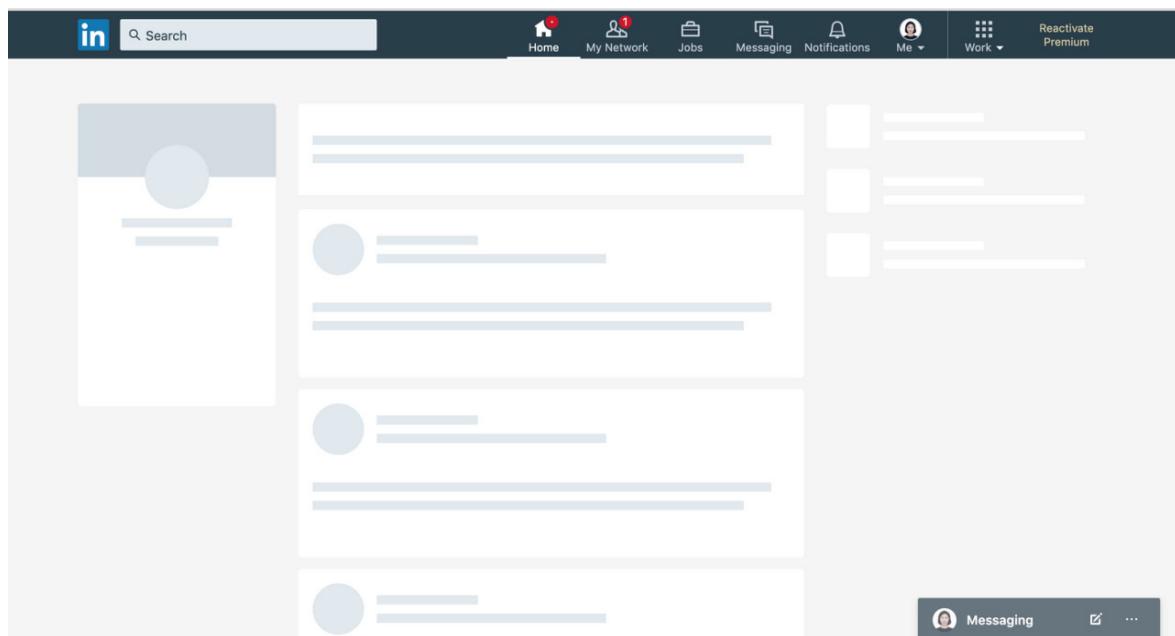
브랜드 컨셉을 전달하고 기다릴 때 주의를 산만하게 하는 훌륭한 방법입니다.

기본 원칙

시간대별 반응 중요한 버튼은 눈에 띄게 디자인하고 수를 최소화합니다.

- 0.1초 : 즉각적인 반응으로 인지합니다.
- 1.0초 : 자연으로 인지하며 직접 조작하는 느낌을 상실합니다.
- 10초 : 사용자가 집중할 수 있는 마지노선입니다. 이 이상의 자연동안 다른 작업을 할 수 있게 예상 완료 시간을 알려주는거나 또 다른 피드백을 제공해야 합니다.

스켈레톤 스크린 사용자가 인터페이스 로딩을 점진적으로 느낄 수 있게 하고, 점차적으로 진행되는 느낌을 줄 수 있는 좋은 도구입니다



<모달 뷰>

현재 페이지에서 콘텐츠의 진행을 잠시 중단하고 독립적인 기능을 제공하여, 사용자에게 무엇을 알리거나 입력하고 다시 원래의 흐름으로 돌아가도록 하는 요소입니다.

기본 원칙

1. 모달이 윈도우에 뜨면 나머지 화면 요소들은 비활성화 되어야 합니다.
2. 오류 메시지나 대기 중, 작업 성공을 알리기 위한 용도로 사용하지 않습니다.

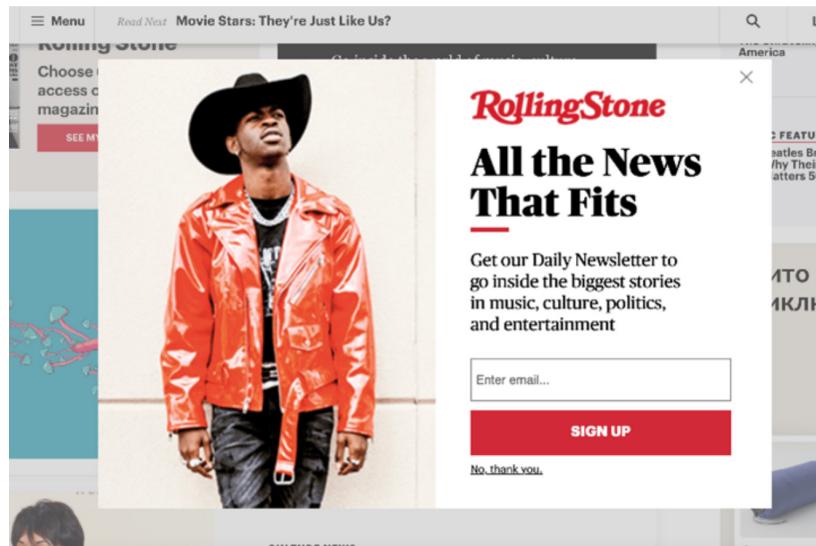
오류 메시지 : 별도의 창으로 보여주거나 그 맥락 안에서 해결하는 게 좋습니다.

로딩 : 로딩 창에서 보여주는 것이 일반적입니다.

작업 성공 : 같은 위치에서 반응하거나 별도의 페이지로 넘어가서 보여주는 것이 좋습니다

3. 주의해서 사용합시다.

그 이유는 시스템에 의해 유발되는 모달 창때문입니다. 사용자에 의해 유발되는 모달 창은 사용자가 어떤 버튼을 누른다든가 할 때 실행됩니다. 하지만 시스템에 의해 유발되는 모달 창은 사용자가 하고 있는 작업을 방해합니다.



회원가입을 유도하는 모달

4. 가장 안전한 방식은 사용자의 행동과 직접적으로 관련된 경우나 정말 중요한 메시지를 보여줄 때만 사용하는 것입니다. 예를 들어 글을 삭제하거나 회원을 탈퇴할 때가 있습니다.

5. 사용자들에게 해야 할 일을 알려주는 용도로 활용합시다.

6. 시스템적으로 유발된 모달창이라면 나타난 이유를 그대로 알려줘야합니다. 이 때는 모달창 창 제목과 기본 확인 버튼의 텍스트를 '동일하게' 설정하는 것이 좋습니다.

7. 모달창 안에서의 콘텐츠와 기능적인 요소들에 '우선순위'를 두어야 합니다.

8. 모달창 안에서 스크롤을 내려야 하는 상황을 피합니다.

9. 모달창 안의 텍스트는 '1~2개의 문장'으로 끝냅니다.

10. 동일한 모달창에서 2가지 이상의 행동을 요구하지 않습니다.

11. 닫기 버튼을 잘 보이게 만듭니다. 언제나 사용자가 빠져나갈 수 있어야 합니다.

12. 창의 크기를 적당하게 만듭니다.

13. 모달창에 시선이 집중되어야 합니다.

14. 사용자들이 시선이 위치한 곳에 모달 윈도우를 직접 보여줍니다.

15. 모달 창을 배경의 메인페이지와 시각적으로 확실하게 구분되어야 합니다.

16. 데스크톱 사용자들을 위해 키보드를 이용해서 모달 창의 콘텐츠로 이동할 수 있게 해야 합니다.

17. 키보드를 이용해서 모달 윈도우의 기능을 컨트롤할 수 있게 합니다.

18. 모달 창을 중첩해서 사용하지 않습니다.

2. 시작 요소

<적응성(반응성) 및 레이아웃 >

CSS Grid의 중요성

이젠 CSS Grid의 방식을 이용해서 레이아웃을 만들고 우리의 사이트에 반응성을 줄 겁니다. 그리드 시스템을 택한 이유는 크게 세 가지입니다.

1. 현업에서 사용하는 CSS 시스템입니다.
2. 어떤 해상도에서든 반응하는 반응성을 줄 수 있습니다.
3. 일관성 있는 레이아웃을 만들 수 있게 도와줍니다.

특히 대표적으로 많이 사용하는 '8px 그리드 시스템'을 이용합니다. 8px 그리드 방식 중 하드 그리드를 통해 좀 더 견고하고 일관성 있는 레이아웃을 짜는 것이 좋습니다.

반응형 레이아웃

분기점



- mobile/pc의 분기점: 480px
- 480px 이하일 경우 모바일 뷰를 적용합니다.
- 480px 초과일 경우 PC 뷰를 적용합니다.

< 애니메이션 >

기본 원칙

1. 요소들이 상태나 위치를 변경할 때, 애니메이션의 지속 시간은 사용자에게 변화를 느낄 수 있도록 충분히 주어야 하지만, 동시에 지루함을 못 느낄 정도로 충분히 빨라야 합니다.

2. 최적의 속도는 0.2~0.5초(200~500 ms)입니다.

3. 기기별 최적의 속도

모바일 : 기본 0.2~0.5초

태블릿 : 기본 0.4~0.45초

컴퓨터 : 0.05~0.3초 이내

4. 애니메이션의 시간은 오브젝트의 크기에 따라 달라져야 합니다. 변화가 크지 않은 요소나 애니메이션은 더 빠르게 움직여야 합니다. 따라서 크고 복잡한 요소들은 짧게 지속될 때 더 나아 보입니다.

5. 오브젝트의 충돌을 고려해서 균등한 뷔를 나눠야 합니다. 바운스 효과는 주의를 산만하게 합니다.

6. 오브젝트들의 움직임은 명확하고 날카로워야 합니다. 즉 모션블러를 사용하지 맙시다.

7. 리스트 아이템은 이들의 등장 간격이 매우 짧게 지연되어야 합니다. 새로운 요소 각각의 발생은 0.02~0.25초 지속되어야 합니다

Erasing (완화기법)

- Easing은 오브젝트의 움직임을 더욱 자연스럽게 만들어줍니다.
- 자연스러운 애니메이션을 위해 가속과 감속을 이용하면 좋습니다.

선형 운동

- 어떠한 물리적인 힘에도 영향을 받지 않는, 예를 들어 우주에 떠 있는 우주선과 같은 오브젝트들은 선형으로 움직입니다. 즉 일정한 속도로 움직입니다.
- 애니메이션 곡선을 이용해서 속도와 위치를 표현합니다.

동일한 인터랙션

- 사용자의 주의를 같은 방향으로 이끄는 하나의 흐름을 만듭니다.

선형 운동

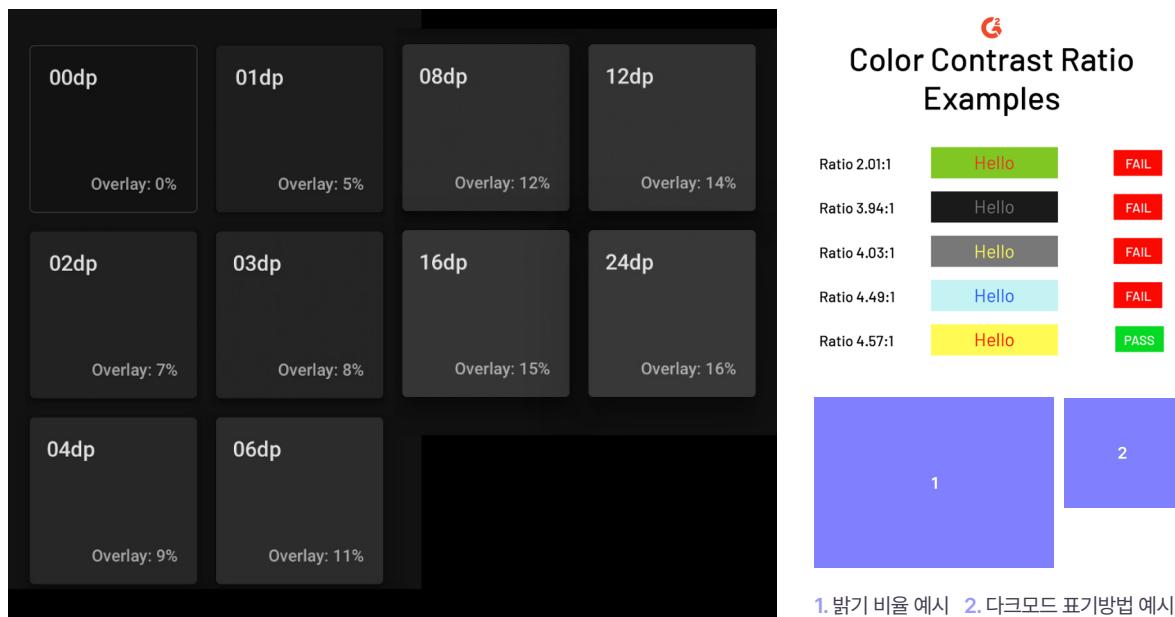
- 사용자의 모든 주의를 이끄는 하나의 특정 오브젝트를 가져야 합니다. 그리고 다른 모든 요소들도 그것에 종속되어야 합니다. 요소들 간 위계를 느끼게 하고 주요 컨텐츠에 집중할 수 있도록 합니다.
- 오브젝트가 불균형하게 사이즈를 변경하면 직선보다 이차곡선을 그리면서 움직이게 됩니다. 반대로 사이즈가 균형있게 변화하면 직선을 그립니다.

- 커브곡선을 따르는 오브젝트의 이동경로는 스크롤 인터페이스의 주축과 일치해야 합니다.
- 만약 움직이는 오브젝트의 경로가 다른 것과 교차된다면, 서로를 통과해서 움직일 수 없습니다. 충분한 여유 공간을 만들고 움직이게 합시다.
- 움직이는 오브젝트가 다른 오브젝트 위로 떠오를 수도 있습니다. 그러나 다른 오브젝트를 통해서 지나거나 분해하지는 않습니다.

< 다크모드 >

기본 원칙

1. 다크 모드는 기본적으로 기기의 설정에 따르되, 사용자가 필요하면 설정할 수 있게 공개되어야 합니다.
2. 다크 모드는 '저조도' 모드이므로, 완전한 검정색을 사용하는 것은 지양합시다.
3. 색상의 대비를 위해 밝기 비율은 4.5 : 1을 유지합시다.
4. 밝기 비율은 색의 최대 밝기에서 최소 밝기를 나눈 값입니다.
5. 하지만 이 비율은 상황마다 다를 수 있습니다. 알맞는 비율을 위해 다양한 비율을 적용해봅시다.
6. Elevated UI의 고려
7. Elevated UI란 두 가지 레이어가 중첩되는 형식의 UI로 기본 배경 위에 올려지는 Bottom sheet 같은 UI를 의미합니다.
8. 일반 모드에선 그림자로 표현되지만, 다크 모드에선 불가능해서 다른 방식이 필요합니다.



3. 아이콘

< 웹/앱 아이콘 >

기본 원칙

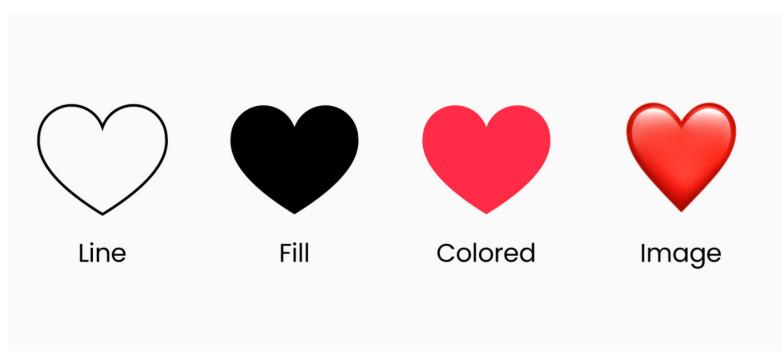
1. 아이콘의 주된 목적은 '빠른 정보 전달' 입니다.

2. 스타일

Line, Fill : 기본적인 형태로 레이아웃에 배치됩니다.

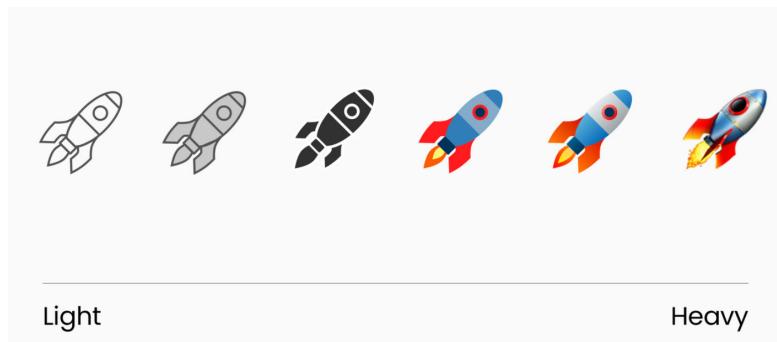
Colored : 행동에 대한 피드백을 주거나 주목도를 높이기 위해 사용합니다.

Image : 높은 주목도가 필요할 때 사용합니다.



3. 무게

텍스트가 주된 콘텐츠라면 무거운(Heavy) 계열의 아이콘을 통해 다채로움을 줍니다.



4. 속성

Thickness : 모바일 환경을 고려해서 굵기를 지정합니다.

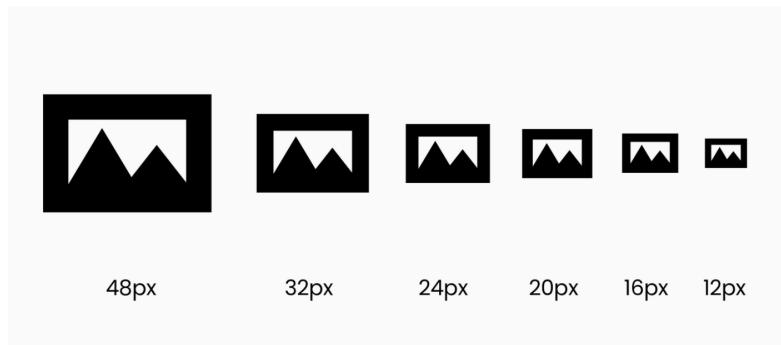
Ends & Join : 부드러움을 표현하기 위해 라운딩을 줍니다.

Radius : 라운딩을 이용해서 표현합니다



5. 크기

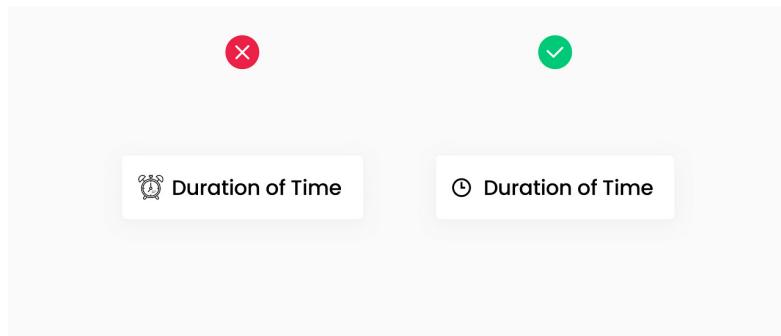
아이콘의 크기는 '8의 배수'를 베이스 라인으로 사용합니다. 8의 배수는 모든 해상도를 지원하기 때문입니다.



5. 사용자 편의

명확성 : 아이콘의 디자인은 사용자가 기능에 대해 빠르게 이해할 수 있는 최소한의 요소로 이뤄져야 합니다. 너무 많은 의미를 내포하는 것은 오히려 복잡함을 주게 됩니다. 가장 좋은 방법은 이미 시중에서 사용하는 디자인을 최대한 이용하는 방식입니다.

디테일 : 좁은 영역 안에 디테일을 최소화 하는 것이 좋습니다. 특히 좁은 영역에 들어가는 시스템 아이콘의 경우, 어떤 내용을 전달하는지 알 수 있는 최소한의 요소만 남기고 다른 장식적인 요소는 제거하는 것이 좋습니다.



일관성 : 하나의 서비스에서 같은 기능들 하는 아이콘은 같은 스타일로 표현합니다. 사용되는 맥락에 따라서 Fill과 Line을 함께 사용할 수 있지만, 동일한 맥락 안에서는 통일시킵시다.

원근법 : 정면에서 바라본 모습으로 제작합니다.

익숙함 : 플랫폼에 상관없이 동일한 아이콘을 사용합니다.

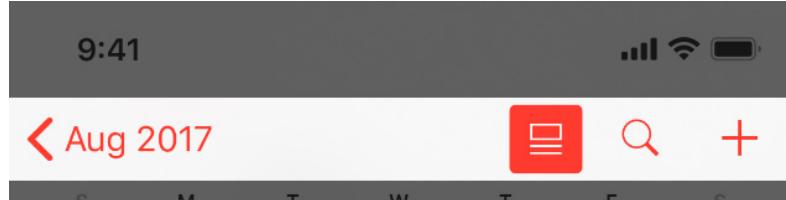
< 아이콘의 저장과 파일명 >

기본 원칙

1. 모든 아이콘의 확장자는 가급적이면 svg로 저장합니다. (* png까지는 괜찮습니다.)
2. 특히 다크모드와 라이트모드 적용을 위해서 svg icon를 이용하는게 좋습니다.
3. 파일의 이름은 명시적인 형태를 기준으로 정합니다.
예를 들어 스톱워치 아이콘의 경우 Stopwatch로 합니다.

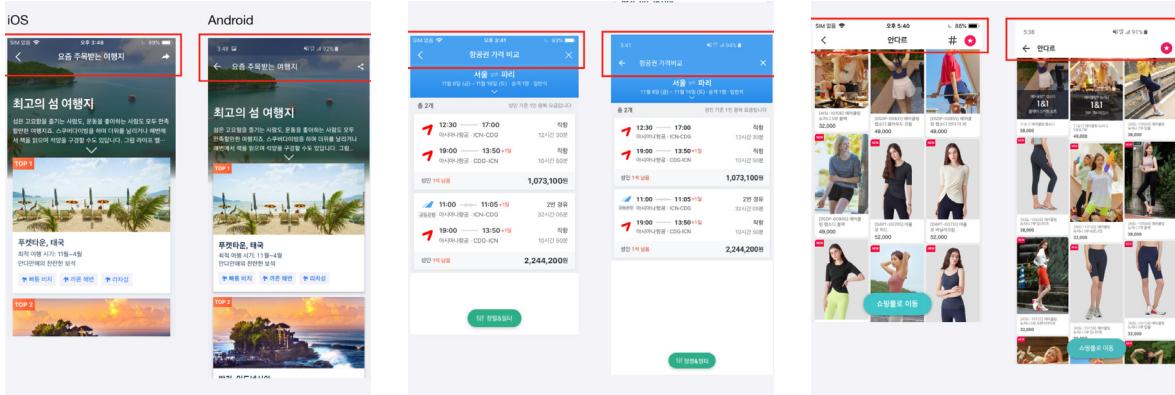
4. 바

< 네비게이션 바 >



기본 원칙

1. 네비게이션 바는 앱 화면의 최상단 혹은 상태 바의 아래에 위치하고, 일련의 체계에 따라 화면을 바꾸어야 합니다.
2. 네비게이션 바에는 너무 많은 컨트롤 요소를 활용하지 않습니다. 일반적으로 현재 뷰의 제목, 돌아가기 버튼, 현재 컨텐츠를 관리하는 버튼보다 더 많은 요소를 포함시켜서는 안됩니다.
3. 화면 왼쪽 상단에는 이전 페이지로 돌아가는 '뒤로' 버튼이나 다른 적절한 버튼을 배치합니다.
4. 화면 우측 상단에는 동작 버튼 또는 메뉴 표시를 추가할 수 있습니다.
5. 네비게이션 바의 배경은 사용자의 시선을 끌지 않도록 선택해야 합니다. 흰색이나 라이트 블루와 같은 밝은 색상을 추천합니다.
6. 오른쪽 상단에는 주로 추가 작업을 수행하는 '+' 아이콘이나 편집 작업을 수행하는 연필 아이콘 등을 사용합니다.



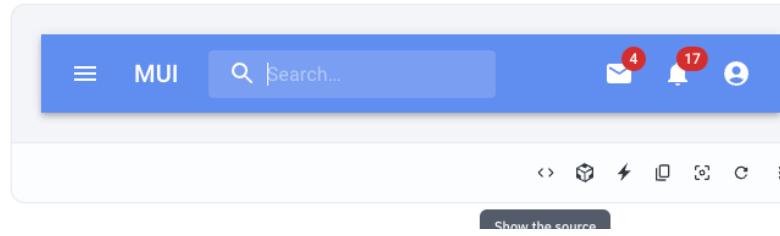
iOS와 안드로이드의 UI 차이

추가 고려 사항

- 사용자가 원하는 정보를 빠르게 찾을 수 있도록 공간 활용 및 위치배치를 잘 고려하여 검색 기능을 제공하세요.

App bar with a primary search field

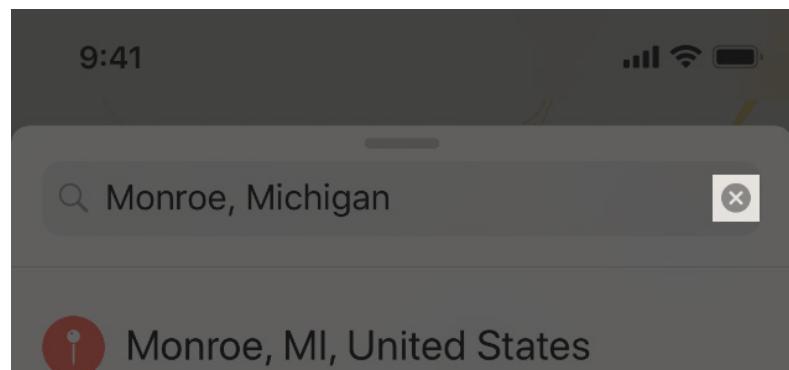
A primary searchbar.



- 유저의 행동 흐름에 따라서 네비게이션을 변경, 유지, 축소할 수 있습니다.

<검색창>

검색 창은 필드에 텍스트를 입력해 수많은 값들 중 원하는 값을 검색할 수 있게 만드는 바입니다. 검색 창은 단독으로 보여질 수 있고 네비게이션 바나 컨텐츠 뷰에도 포함될 수 있습니다.



기본 원칙

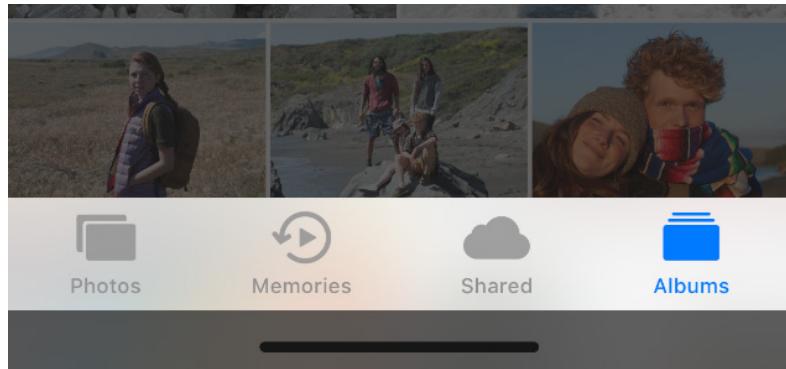
1. 검색 창의 높이는 44포인트로 설정하여 사용자가 쉽게 텍스트를 입력할 수 있을 정도로 충분한 공간을 제공합니다.
2. 검색어 입력 필드와 검색 버튼을 제공합니다.
3. 검색 활동을 종단할 수 있는 취소 버튼을 활성화 합니다.
4. 검색 창의 배경색은 일반적으로 하얀색 또는 회색 계열의 밝은 색상을 사용하는 것이 좋습니다.
5. 테두리는 검색 창이 눈에 잘 띄도록 해당 배경에서 도드라지는 색상을 사용하며, 둑근 모서리 스타일로 디자인 합니다.

추가 고려 사항

- **bad case :** 사용자가 쉽게 찾을 수 없는 위치에 있거나, 텍스트 입력 영역이 정확히 보이지 않는 경우

<탭 바>

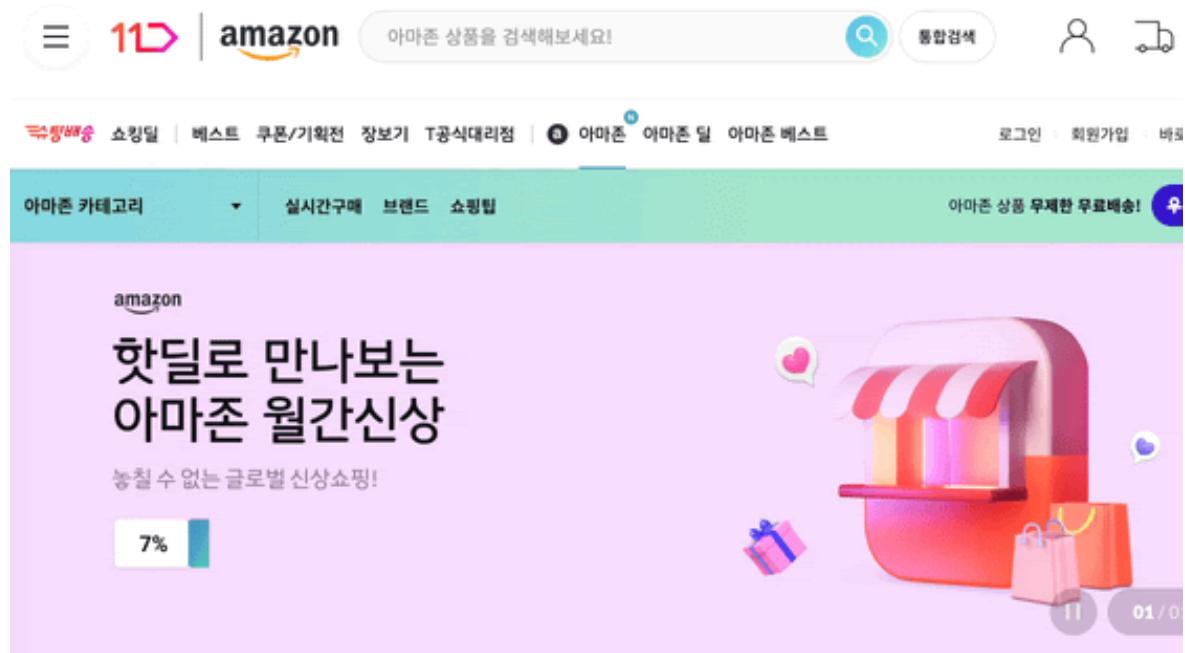
탭바는 앱 화면 하단에 나타나 앱의 여러 개의 탭 아이템으로 빠르게 이동할 수 있도록 해줍니다. 탭바는 기본적으로 투명하나 배경 색상을 가질 수도 있습니다. 키보드가 나올 땐 가려집니다. 주요 기능을 빠르게 액세스할 수 있는 중요한 UI 요소이기 때문에, 사용자가 쉽게 이해하고 사용할 수 있도록 쉽고 명확하게 디자인하고 배치해야 합니다.



기본 원칙

1. 각 탭은 앱의 주요 기능 또는 독립적인 내용 구조를 나타내야 합니다.
검색어 입력 필드와 검색 버튼을 제공합니다.
2. 사용자가 탭을 클릭하면 해당하는 탭의 컨텐츠 또는 뷰가 업데이트 되어야 합니다.
3. 탭 바의 배경색은 애플리케이션의 전반적인 테마와 일관성을 유지하기 위해 보통 흰색, 회색 계열 또는 브랜드 색상을 사용합니다.
4. 선택되지 않은 아이콘과 텍스트의 색상은 연한 색상으로 설정하여 선택된 탭과 구분이 되게 합니다. 선택된 탭은 보다 진한 색상 또는 브랜드 색상을 사용하는 것이 좋습니다.
5. 직관성을 위해 레이블은 5개 이내로 제한하는 것이 좋으며, 전체적인 단어 수를 최소화 하는 것이 좋습니다.

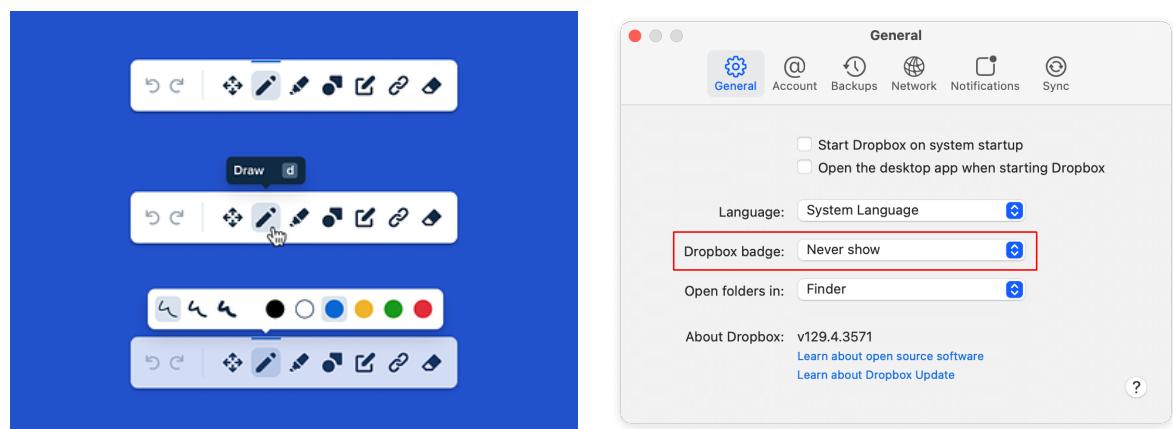
추가 고려 사항

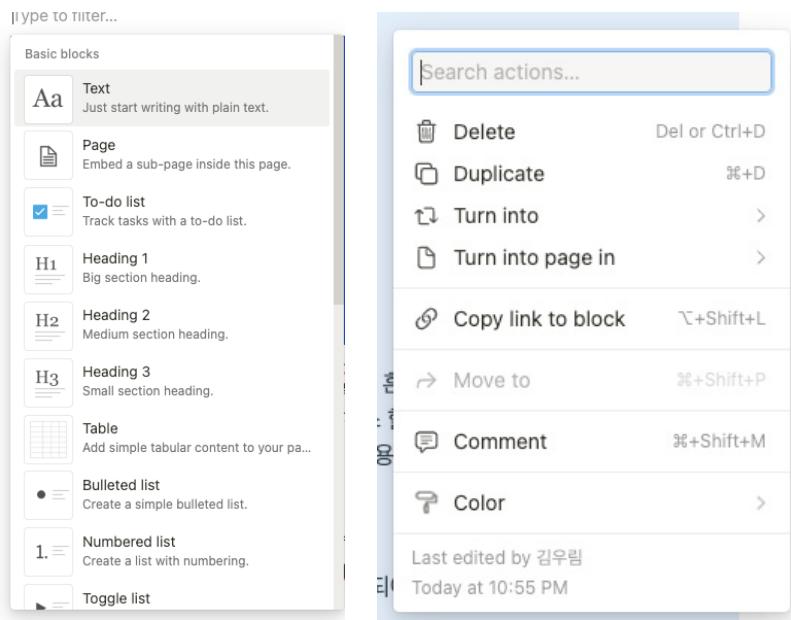


- 위 gif처럼 각 탭의 이름을 명확하고 구체적인 워딩을 사용해 사용자의 콘텐츠 이해를 쉽게 합니다.
- 컨텐츠 내의 일관화가 필요하다면 그룹화 합니다. (11번가 상품판매와 아마존 상품판매를 나눈 예시)

<툴바>

애플리케이션 또는 웹 페이지에서 사용되는 UI 요소로, 흔히 상단이나 측면에 위치합니다. 툴바는 사용자가 자주 사용하는 기능 또는 메뉴 항목에 빠르게 액세스 할 수 있게 해주는 버튼, 드롭다운 메뉴 등을 포함합니다. 배경색 및 구분선 등 다양한 디자인 요소를 활용해 툴바와 다른 요소가 명확히 구분되게 합니다.





기본 원칙

1. 툴바에는 사용자가 자주 사용하는 기능이나 메뉴 항목이 포함되어야 합니다.
2. 툴바의 버튼이나 아이콘은 한눈에 알아볼 수 있도록 명확해야 합니다.
3. 툴바의 기능이 너무 많거나 복잡하면 드롭다운 메뉴를 사용해 관련 기능을 그룹화하여 사용자가 덜 혼동되도록 합니다.
4. 툴바의 배경색은 애플리케이션의 전반적인 테마와 일관성을 유지하기 위해 보통 흰색, 회색 계열 또는 브랜드 색상을 사용합니다.
5. 텍스트와 아이콘의 색상은 배경색과 대조되어 선명하게 보이게 설정합니다. 보통 검은색 또는 브랜드 색상을 사용합니다.

툴바와 탭바의 차이점

탭 바는 앱의 기본 네비게이션 요소입니다. 여러 가지 탭이 표시되며, 각 탭을 선택하면 해당 뷰 컨트롤러로 이동합니다. 일반적으로 탭 바는 앱의 메인 화면이나 핵심 부문의 하위 항목에 사용됩니다.

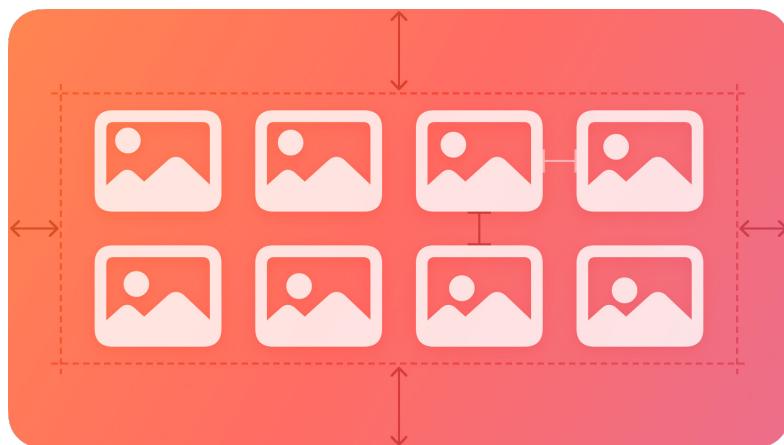
반면에 툴바는 일반적으로 앱에서 특정 작업을 수행하는 데 사용되는 작은 버튼의 집합을 포함합니다. 툴바는 앱의 많은 화면에서 일관된 위치에 유지되며, 선택적으로 활성화하거나 비활성화할 수 있습니다. 툴바가 사용되는 경우 버튼을 클릭하면 해당 작업이 수행됩니다. 따라서 탭 바는 주로 앱 내부의 다양한 뷰 컨트롤러 사이를 전환하는 데 사용되고, 툴바는 특정 작업을 실행하는 데 사용됩니다.

툴바를 다루는 동작을 자연스럽고 직관적으로 하기 위하여 애니메이션 효과를 적절히 사용합시다.

5. 뷰

< 컬렉션 뷰 >

컬렉션은 정렬된 콘텐츠 집합을 관리하고 이를 사용자 지정 가능하고 시각적인 레이아웃으로 제공합니다. 컬렉션은 이미지 기반 콘텐츠를 표시하는 데 이상적입니다.



기본 원칙

1. 가능하면 그리드 레이아웃을 사용합니다. (규칙적인 레이아웃을 유지해야 합니다)
2. 그리드의 경우, 보통 pc는 12단, tablet는 6단, mobile는 4단 그리드를 사용합니다.
3. 항목을 쉽게 선택할 수 있어야 합니다.
4. 콘텐츠가 겹치지 않도록, 이미지 주위에 적절한 여백을 제공합니다.

추가 고려 사항

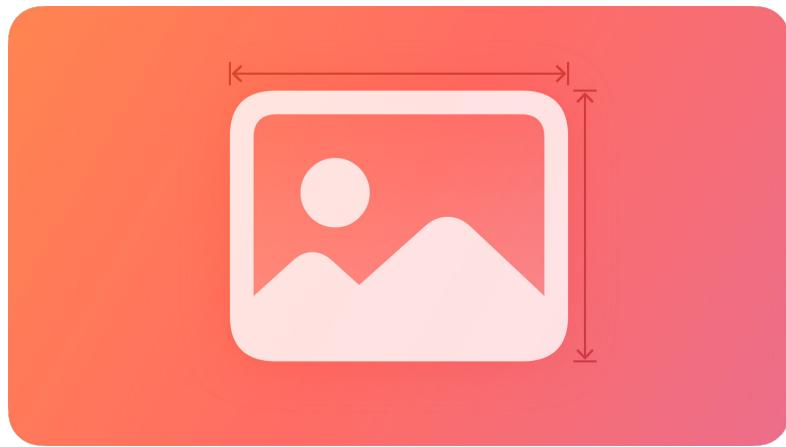
- 텍스트만을 컬렉션 해야한다면 컬렉션 뷰가 아닌 테이블을 활용합니다.
- 필요한 경우 사용자 지정 상호 작용을 추가합니다.
 탭하여 선택, 길게 터치하면 편집, 스와이프하면 스크롤 등등...
 하지만 이런 부분은 모바일에서만 가능한 옵션이기 때문에, pc일때는 어떻게 처리할지도 고민해야 합니다

< 이미지 뷰 >

이미지 보기는 투명하거나 불투명한 배경에 단일 이미지(또는 경우에 따라 일련의 애니메이션 이미지)를 표시합니다.

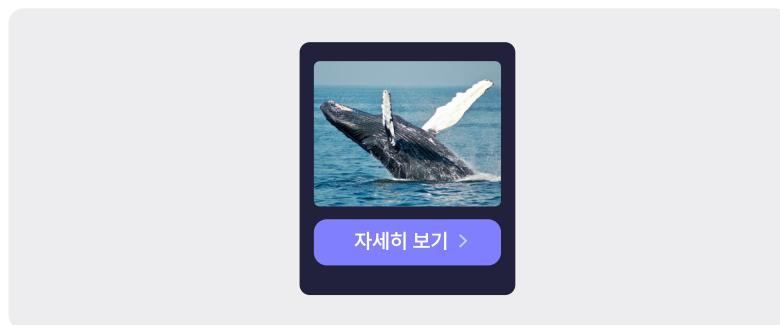
상세 기능

이미지 보기 내에서 이미지를 특정 위치에 늘리거나 크기를 조정하거나 고정할 수 있습니다. 이미지 보기의 일반적으로 대화형(액션을 주고 받는 기능)이 아닙니다.



기본 원칙

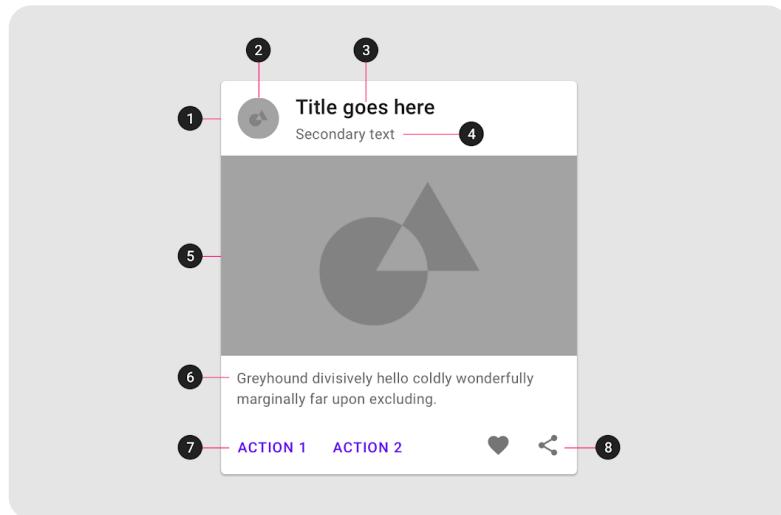
1. 아이콘을 표시하고 싶다면, 아이콘 사진을 이미지 보기에 넣는 대신 아이콘 컴포넌트를 사용합니다.
2. 이미지 보기 대신 이미지 버튼을 사용하여 클릭 가능한 이미지를 만듭니다. 따로 버튼을 만들어두지 않습니다.
3. 단지우기나 특정 행동을 위한 버튼은 표시를 줘야 합니다.
4. 이미지를 클릭해서 이미지 보기 가능하도록 해야합니다. 따로 이미지 보기 버튼을 만들지 않습니다.



5. 이미지에 대해 일관된 크기를 사용해야 합니다.
6. 지정된 높이와 크기에 따라 이미지를 보여줘야 합니다.
7. 추상적인 개념에는 그림을, 구체적인 개념에는 사진을 사용합니다

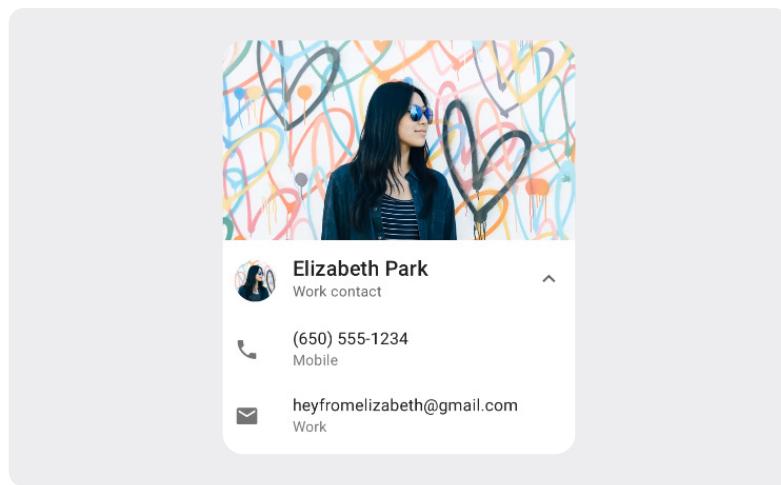
< 카드 >

카드에는 단일 주제에 대한 내용과 작업이 포함됩니다.

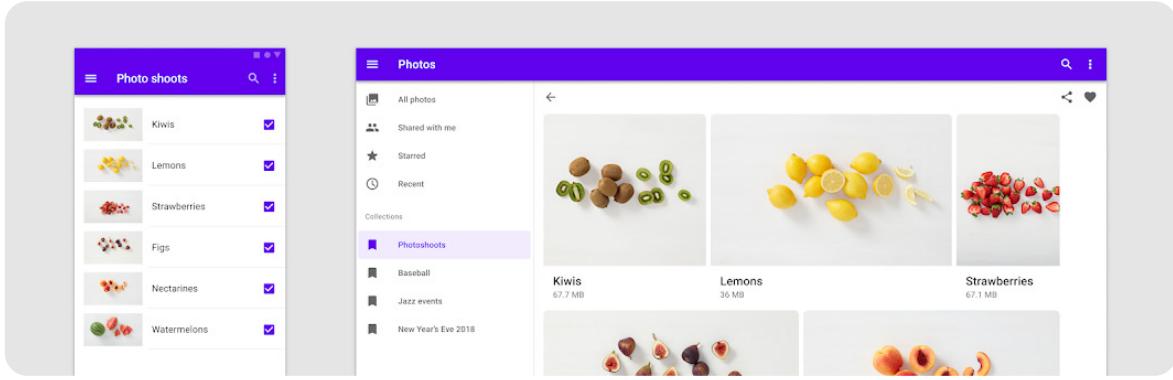


기본 원칙

1. 카드는 무언가를 감싼 하나의 단위로 인식 되어야 합니다.
2. 카드는 다른 카드와 병합하거나 여러 카드로 나눌 수 없습니다.
3. 필요한 내용만 포함시킵니다. 사진 속 요소들을 다 넣을 필요는 없습니다.
4. 카드를 클릭하면 디테일 페이지로 넘어가거나, 추가정보를 드롭다운을 통해 밑에 보여줄 수도 있습니다.

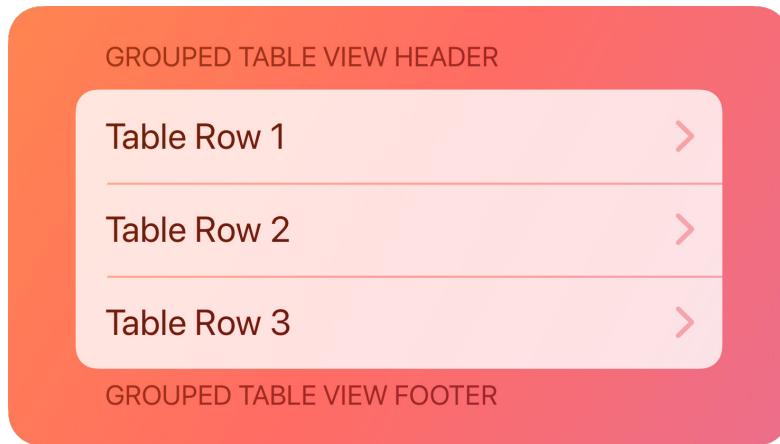


5. 모바일인지 PC인지에 따라 카드의 모양을 다르게 할 수도 있습니다
6. PC일때는 넓은 화면을 고려한 카드를, 모바일일때는 좁은 화면을 고려한 카드를 만드는 것도 좋습니다.



< 테이블 >

테이블은 하나 이상의 행 열에 데이터를 표시합니다.



상세 기능

테이블은 그룹 또는 계층 구조로 구성된 데이터를 나타낼 수 있습니다. 또한 테이블의 데이터에 대해서, 선택, 추가, 삭제 및 재정렬(sort)과 같은 사용자 상호 작용을 지원할 수 있습니다.

사용자 상호작용 예시

1. 확인란, 정렬, 필터, 페이지네이션 등행에 호버 시 해당 행의 배경색 표시

<input type="checkbox"/>	Status	Signal Name	Severity	Stage	Lapsed Time	Team Lead
<input type="checkbox"/>	Offline	Astrid: NE shared managed-features	Medium	Triaged	10:12	Chase Nguyen
<input type="checkbox"/>	Offline	Cosmo: prod shared vm	Huge	Triaged	12:45	Brie Furman
<input type="checkbox"/>	Offline	Phoenix: prod shared lyra-managed-features	Minor	Triaged	13:06	Jeremy Lake
<input type="checkbox"/>	Offline	Sirius: prod shared ares-managed-vm	Negligible	Triaged	13:18	Angelica Howards

2. 행에 호버 시 해당 행의 배경색 표시에 호버 시 해당 행의 배경색 표시

<input type="checkbox"/>	Status	Signal Name	Severity	Stage	Schedule	Project Team Le...
<input type="checkbox"/>	Online	Astrid: NE shared managed-features-provider-heavy	Medium	Triaged	0:33	Chase Nguyen
<input type="checkbox"/>	Offline	Cosmo: prod shared ares-managed-features-provider-heavy...	Huge	Triaged	0:39	Brie Furman
<input type="checkbox"/>	Online	Phoenix: prod shared lyra-managed-features-provider-heavy...	Minor	Not triaged	3:12	Jeremy Lake
<input type="checkbox"/>	Online	Sirius: prod shared ares-managed-features-provider-heavy...	Negligible	Triaged	13:18	Angelica Howards

3. 열 레이블에 호버 시 전체 열이름 표기

	Severity	Stage	Schedule	Project Team Le...
<input type="checkbox"/>	Medium	Triaged	0:33	Chase Nguyen
<input type="checkbox"/>	Huge	Triaged	0:39	Brie Furman
<input type="checkbox"/>	Minor	Not triaged	3:12	Jeremy Lake
<input type="checkbox"/>	Negligible	Triaged	13:18	Angelica Howards

유형

1. 리스트 테이블

1컬럼	2컬럼
1열	목록 텍스트텍스트 목록 텍스트텍스트

2. 목록 선택 리스트 테이블

<input type="checkbox"/>	1컬럼	2컬럼
<input type="checkbox"/>	1열	목록 텍스트텍스트 목록 텍스트텍스트
<input type="checkbox"/>	1열	목록 텍스트텍스트 목록 텍스트텍스트
<input checked="" type="checkbox"/>	1열	목록 텍스트텍스트 목록 텍스트텍스트
<input type="checkbox"/>	1열	목록 텍스트텍스트 목록 텍스트텍스트
<input type="checkbox"/>	1열	목록 텍스트텍스트 목록 텍스트텍스트

3. 이미지(아이콘) 리스트 테이블

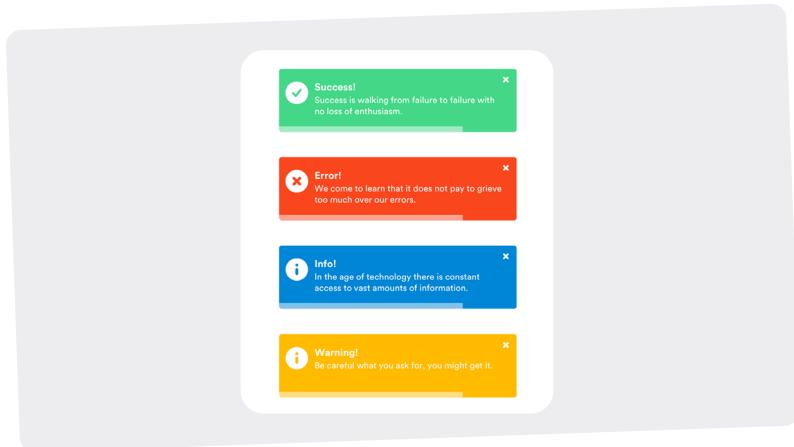
1컬럼	2컬럼
1열	목록 텍스트텍스트 목록 텍스트텍스트

기본 원칙

1. 주로 텍스트를 표시합니다. 항목의 크기가 다양하거나, 이미지가 많이 들어간다면 컬렉션을 고려해봅시다.
2. 만약, 이미지를 꼭 넣어야한다면, 이미지의 높이와 너비를 고정시켜야 합니다
3. 사람들이 목록 항목을 선택할 때 적절한 피드백을 제공해야 합니다. 예를 들어 옵션을 나열하는 테이블의 경우, 항목이 선택되었을 경우 해당 항목의 배경색을 바꾸어 선택되었음을 강조시킬 수 있습니다.
4. 내용을 읽기 쉽도록 항목 텍스트를 간결하게 유지해야 합니다.
5. 잘림과 줄바꿈을 최소화합니다.
6. 어쩔 수 없이 텍스트가 너무 길다면 테이블에 그 내용을 노출시키기 보다는, 디테일 페이지에서 보여줍니다.
7. 아니라면, 잘릴 수 있는 텍스트의 가독성을 유지하는 방법을 고려합니다. 예를 들어 말줄임표 (text-overflow: ellipsis)를 이용하거나 경우에 따라 텍스트 중간에 말줄임표 이용하는 것도 좋습니다

< 노티피케이션 >

알림을 사용하여 사용자에게 시스템 상태의 업데이트 또는 변경 사항을 알려주는 구성 요소입니다.



기본 원칙

1. 메세지는 간결해야합니다. 노티피케이션은 금방 닫히기 때문입니다.
2. 텍스트 영역을 초과하는 경우, 더보기란을 제공합시다.
3. 닫기 버튼 선택 시 알림을 닫습니다. 선택하지 않으면, 정해진 시간 뒤에 자동으로 닫힙니다.

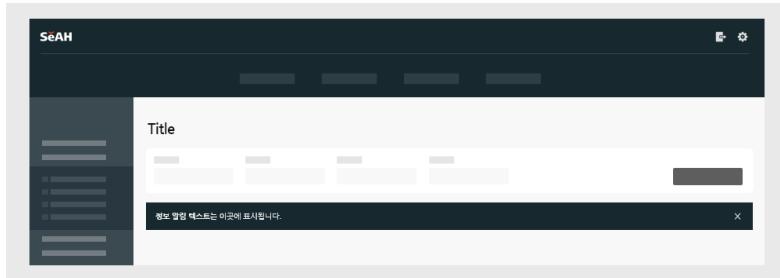
유형

1. 색상에 따른 알림유형

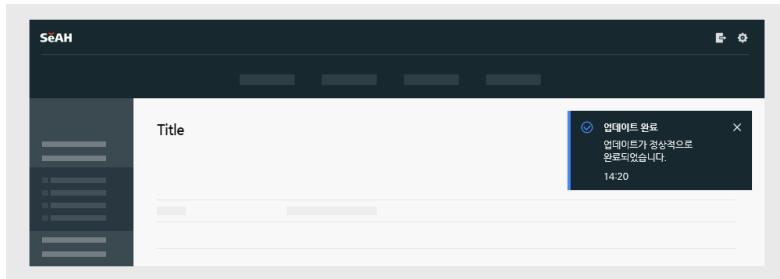
- **초록색** : 성공
- **빨간색** : 에러
- **파란색** : 정보
- **노란색** : 경고

2. 모양에 따른 알림유형

인라인 알림 : 내용이 들어가는 영역에 알림 사항이 있는 경우 사용합니다



토스트 알림 : 업데이트 완료 메시지 등 알림 사항이 있는 경우 화면 위에 레이어 형태로 나타납니다.

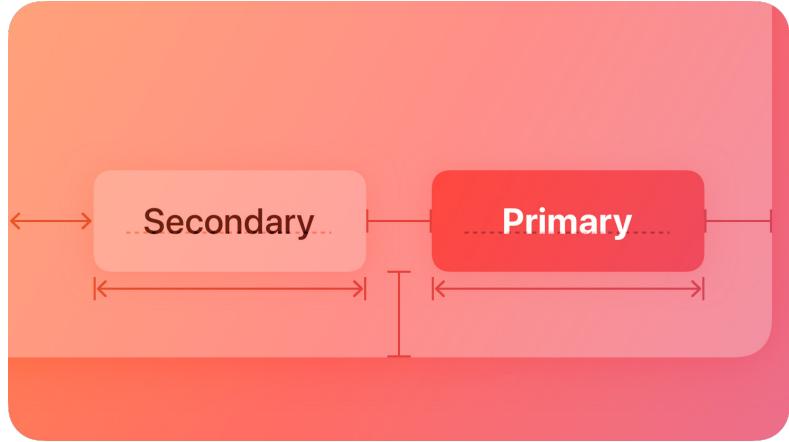


6. 컨트롤

6-1. 기본 컨트롤

<버튼>

버튼은 즉각적인 작업을 시작합니다. 일반적으로 버튼은 기능을 명확하게 전달하기 위해 세 가지 속성을 결합합니다



속성

1. 스타일

크기, 색상, 모양

콘텐츠

버튼의 기호나 텍스트

2. 역할

Normal : 특별한 의미가 없습니다.

Primary : 사람들이 선택할 가능성이 가장 높은 버튼입니다.

보통 Primary 컬러(브랜드 컬러)를 사용합니다.

Cancel : 현재 작업을 취소합니다.

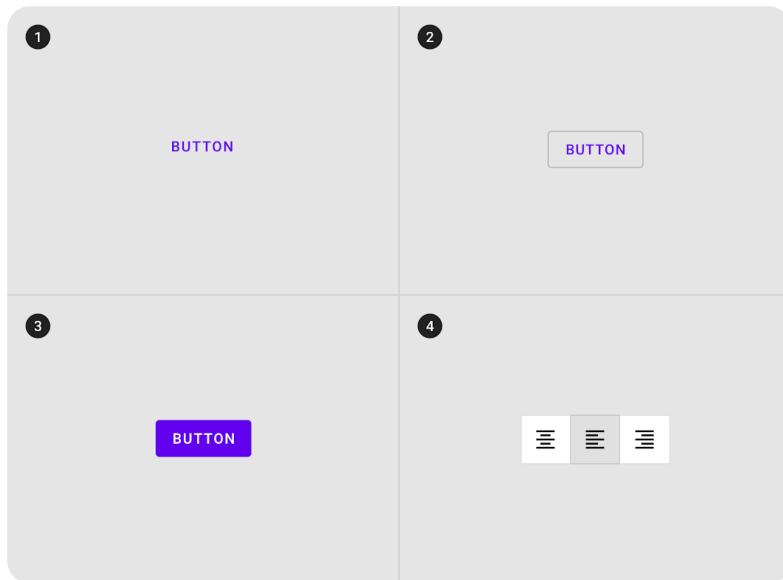
Destructive : 데이터를 파괴할 수 있는 작업을 수행합니다. 주로 빨간색을 사용합니다.

기본 원칙

1. 사람들이 사용하기 쉬운 버튼을 만드십시오
2. 버튼을 식별할 수 있도록 버튼 주변에 충분한 공간을 줄 수 있어야 합니다.
버튼 주변에 충분히 margin을 줍시다.
3. 버튼에 충분한 공간을 주는 것도 버튼을 선택하기에 도움이 됩니다. 버튼에 충분한 padding을 줍시다.

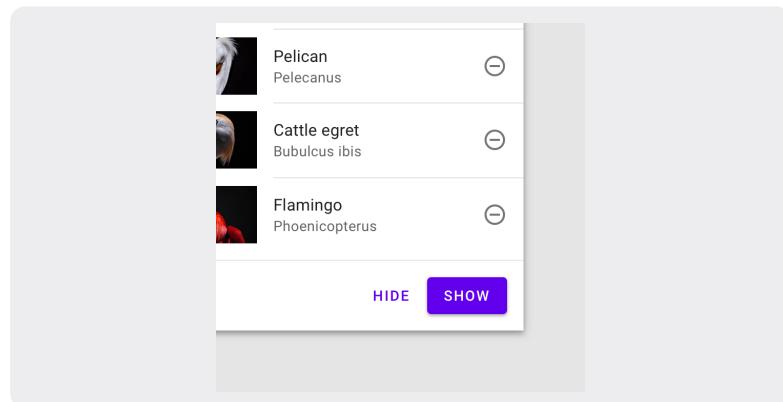
스타일

1. 배경 색이 있는 버튼은 가장 눈에 띄기 때문에 사람들이 가장 원하는 작업 or 가장 눈에 띄기 원하는 작업을 넣습니다.

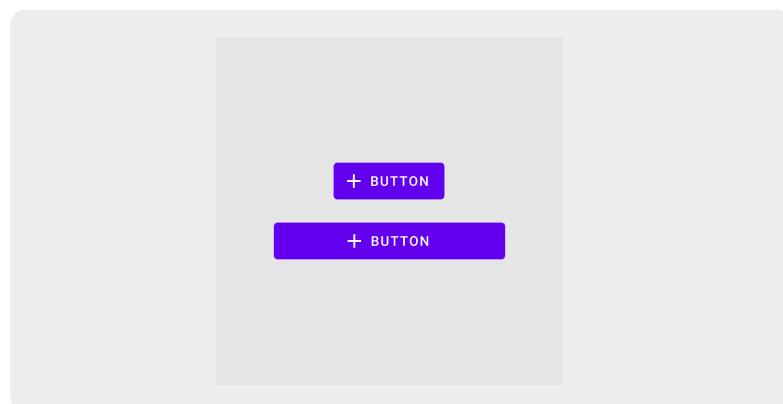


2. 시각적으로 눈에 띠는 버튼의 수를 뷔 당 1~2개로 유지합니다. 눈에 띠는 버튼이 많으면 시선이 분산됩니다.
3. 크기는 같지만 색깔은 다르게 사용하여 여러 옵션 중에서 선호하는 선택을 시각적으로 구별합니다.

동일한 크기의 버튼을 2개 이상 나열하면, 이 버튼들이 연관 버튼이라는 생각을 하게 만들기 때문입니다



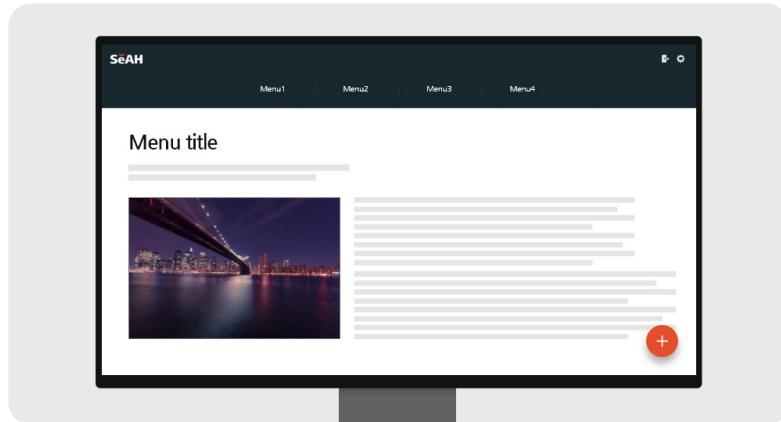
4. 핵심 버튼은 오른쪽에 배치합니다. 부정하는 버튼은 좌측에 놓습니다.
5. 버튼은 적절하게 확장되도록 합니다. 모바일일땐 작았던 버튼도, PC에서는 넓게 만들 수 있습니다



6. 아이콘은 필요할 때만 넣습니다.

콘텐츠 및 아이콘과 텍스트

- 버튼에는 아이콘 혹은 텍스트, 혹은 둘 다 들어갈 수 있습니다.
- 아이콘을 연상시킬 수 있는 작업을 수행중이라면 아이콘을 사용하는 것도 좋습니다. 그러나 '장바구니에 추가' 같은 짧은 텍스트가 아이콘보다 더 명확하게 전달할 수 있다면 텍스트를 사용합시다.



플로팅 버튼

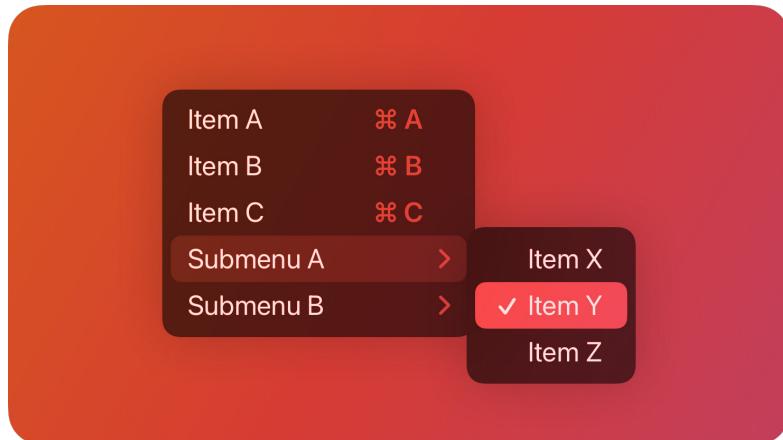
플로팅 버튼은 화면에 떠있는 버튼입니다. 화면의 핵심이 되는 태스크를 수행할 수 있도록 제공합니다. 보통 하단 가장자리에 배치합니다.

기본 원칙

- 한 개만 제공하는 것을 권장합니다.
- 긍정적인 버튼만 플로팅 버튼으로 사용합니다.
- 플로팅 버튼에는 핵심적인 작업을 설정합니다. 중요하지 않은 작업은 플로팅 버튼에 넣지 않습니다.

<메뉴>

사람들이 메뉴와 상호 작용할 때 옵션을 표시합니다. 앱이나 게임에서 명령을 표시하는 효율적인 방법입니다.



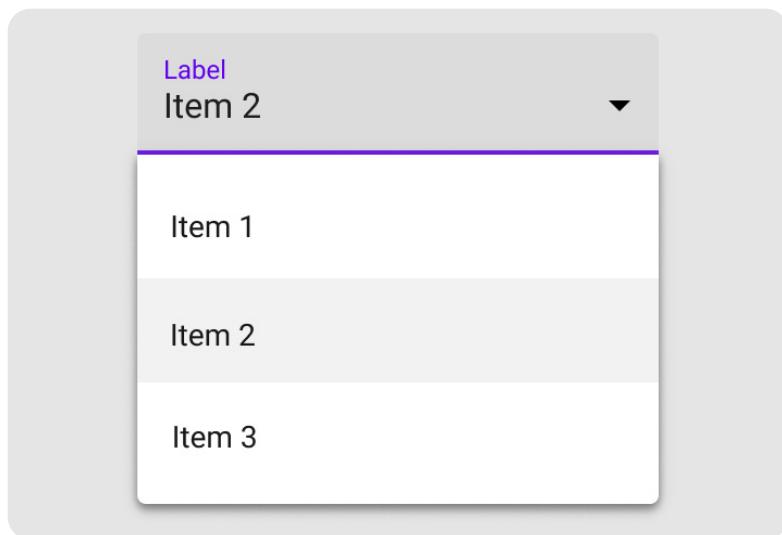
기본 원칙

1. 메뉴 항목은 너무 많으면 사용자가 한 번에 스캔하기 힘드므로 적게 유지합니다.
2. 메뉴 항목이 많아야 하는 경우, 메뉴를 스크롤할 수 있도록 지원합니다.
3. 메뉴가 화면에서 잘리지 않도록 해야합니다. 잘리지 않도록 적절하게 위에서 열리거나, 아래에서 열리도록 해야합니다.
4. 전체에서 일관되게 상황에 맞는 메뉴를 지원합니다. 어떤 곳에는 컨텍스트 메뉴가 있는데 어떤 곳에서는 없는 경우, 사람들은 혼란스러워 합니다.

유형

1. 드롭다운 메뉴

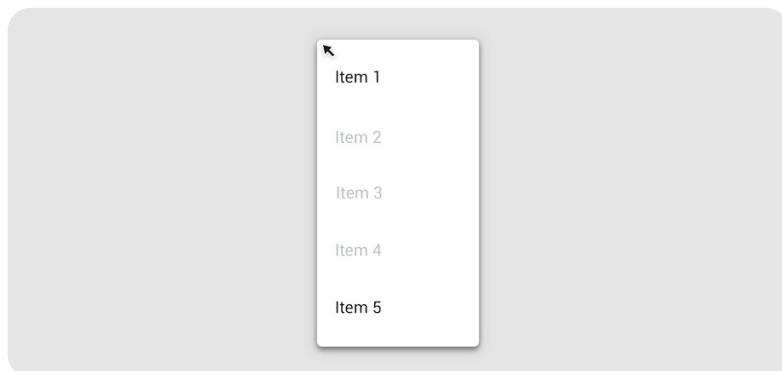
드롭다운 메뉴는 아이콘, 버튼 또는 동작으로 트리거되는 옵션 목록을 표시합니다



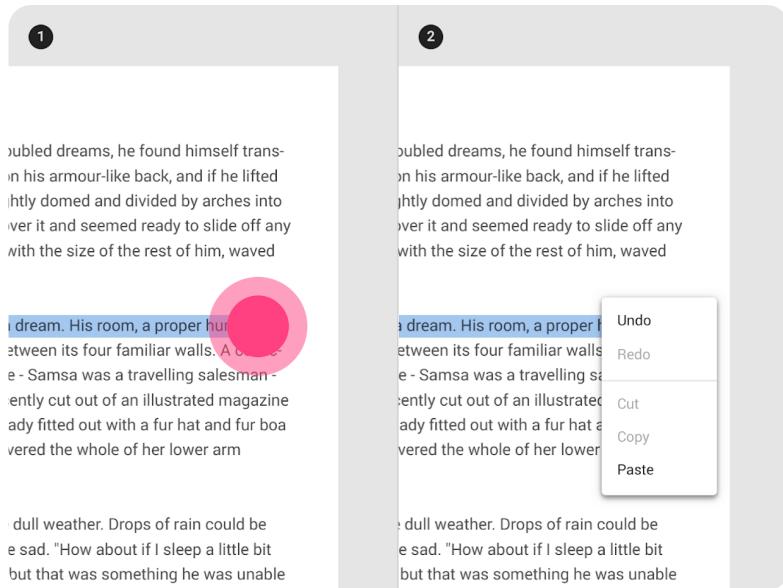
2. 컨텍스트 메뉴 (상황에 맞는 메뉴)

컨텍스트 메뉴는 일관된 UI 요소에 의해 트리거되지 않습니다.

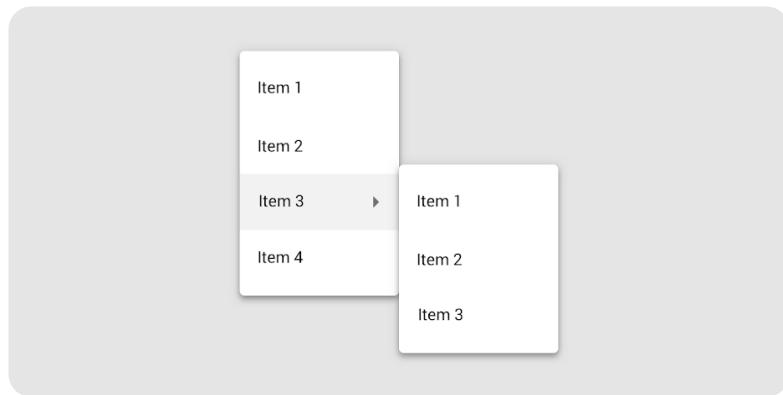
사용자가 탭하는 위치 옆에 표시되며 탭 대상에 따라 동작이 달라질 수 있습니다.



글을 선택했을 때 일정한 컨텍스트 메뉴가 나와야 합니다.



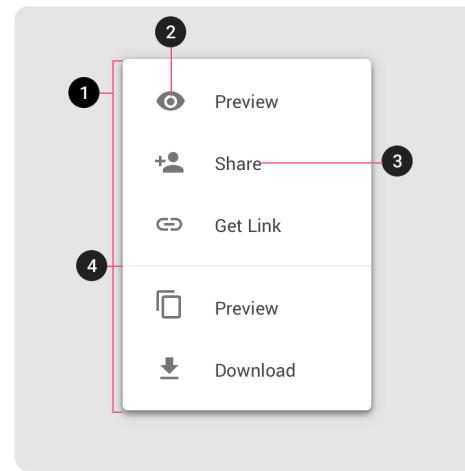
하위 메뉴는 한 수준으로 (1 depth)로 유지합니다. 더 깊이 들어가면 복잡하고 인지하기 힘듭니다.



가장 자주 사용하는 메뉴 항목을 가장 위에 배치합니다

아이콘을 사용해 기능을 명확하게 만들면 더 좋습니다

구분선을 사용하면 사람들이 메뉴를 더 빨리 헤어볼 수 있도록 도울 수 있습니다.



▶ 4번이 기준선

< 피커와 날짜선택 >

피커는 사람들이 선택할 수 있는 하나 이상의 스크롤 가능한 고유 값 목록을 표시합니다



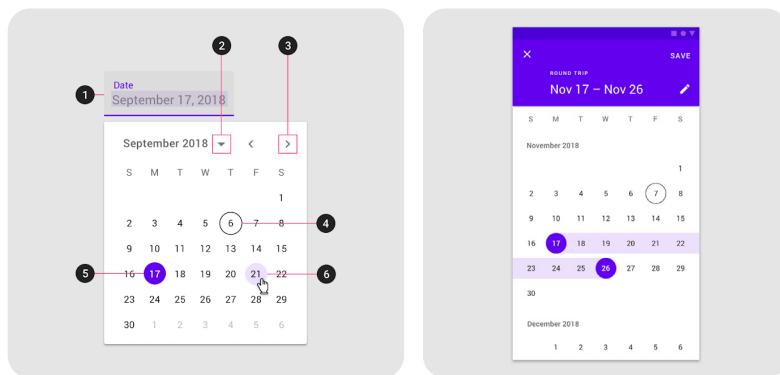
기본 원칙

1. 피커를 통해서 긴 항목 목록을 제공합니다.
2. 만약 선택목록이 짧다면, 드롭다운 버튼을 사용합니다.
3. 매우 많은 항목의 집합을 표시해야 하는 경우는 테이블을 사용합니다.
4. 알파벳 순이나 숫자순 등 예측 가능하고 논리적으로 정렬된 값을 사용합니다.

유형

1. 날짜 선택기

날짜를 선택할 수 있으며, 필요에 따라 날짜 범위 선택을 제공할 수 있습니다.



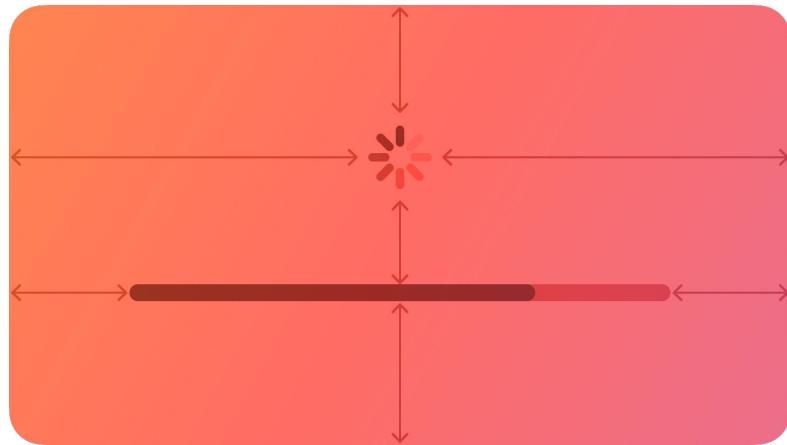
2. 시간 선택기

- 시간을 선택할 수 있습니다.
- 분을 지정할 때 더 적은 세분성을 제공할 수 있습니다. 예를 들어 분 목록에 0~59를 제공할 수도 있지만, 사용자들은 15분 간격 (0,15,30 및 45)을 원할 수도 있습니다.



< 활동 지표와 진행 바 >

진행률 표시기는 콘텐츠를 로드하거나 긴 작업을 수행하는 동안 앱이 중단되지 않았음을 사람들에게 알려줍니다.



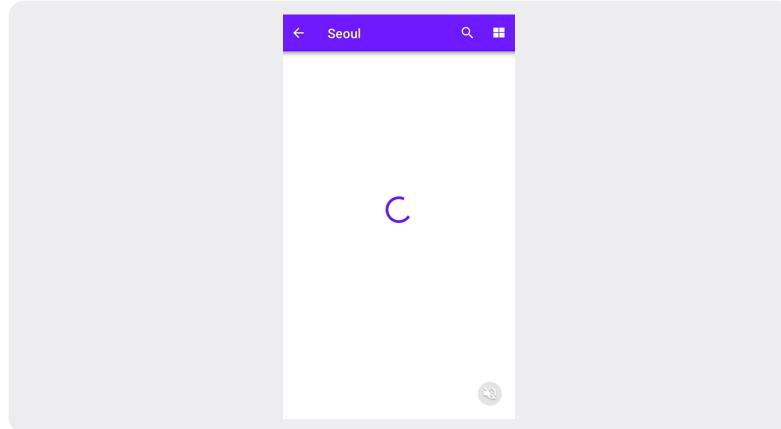
기본 원칙

1. 결정된 진행률 표시기에서 진행률을 보고할 때 가능한 한 정확해야 합니다.
2. 사람들이 작업을 완료하는 데 필요한 시간에 대해 확신을 가질 수 있도록 진행 속도를 균등화해야 합니다.
5초만에 진행률 90% 띄워놓고 100%를 채우는데 5분이 걸리면 안됩니다.
3. 사람들이 어떤 일이 계속되고 있음을 알 수 있도록 진행률 표시기를 계속 움직입니다.
4. 일관된 위치에 진행률 표시기를 표시합니다
5. 가능하면 사람들이 처리를 중단할 수 있다면 중단할 수 있도록 합니다. 예를 들어 취소 버튼을 눌러 취소할 수 있게 합니다.
6. 프로세스를 중단할 경우, 부정적인 결과 발생이 예측된다면 이 점을 알려줍니다.
7. 앱의 모든 버튼에 진행률 표시기를 적용하지 마세요. 사용자의 작업을 불필요하게 방해할 수 있습니다.

유형

1. 화면 중앙에 나타나는 형태

화면 중앙에 나타나는 경우 콘텐츠의 초기 로드를 나타냅니다.



2. 선형 및 원형



3. 확정 및 불확정 형태

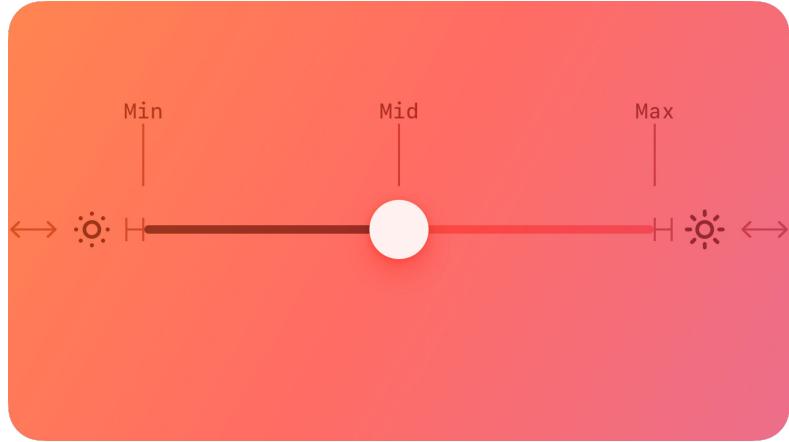
확정: 진행률 표시기가 0에서부터 100까지 채워집니다

불확정: 확정되지 않은 진행률 표시기는 고정된 트랙을 따라 이동합니다.

6-2. 슬라이더와 스태퍼

<슬라이더>

사람들이 최소값과 최대값 사이에서 조정할 수 있는 엄지라고 하는 컨트롤이 있는 수평 트랙입니다



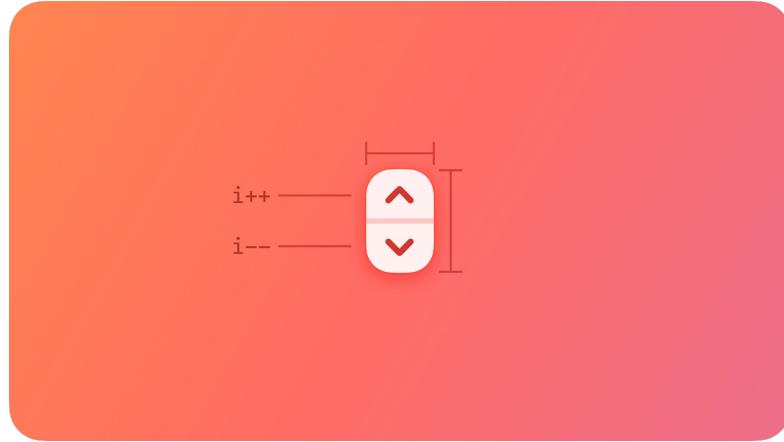
기본 원칙

1. 친숙한 슬라이더 방향을 사용합니다. 가로 슬라이더면 최소값을 앞에, 최대값을 뒤에 배치합니다. 세로 슬라이더라면 최소값을 아래에, 최대값을 위로 배치합니다.
2. 텍스트 필드와 스텝퍼로 슬라이더를 보완하는 것을 고려합니다. 왜냐하면 슬라이더만 주면 정확한 슬라이더 값 을 맞추는 것을 어려워할 수도 있기 때문입니다. 직접 입력할 수 있는 칸을 주거나 1씩 조절할 수 있는 스텝퍼를 두는 것을 고려하세요.
3. 슬라이더 값이 변경될 때 실시간 피드백을 제공하는 걸 고려하세요.
4. 글자 크기를 정하는 슬라이더라면 그 옆에 슬라이더의 변화에 따라, 글자가 실시간으로 변하는 것을 보여줘야 합니다.
5. 슬라이더에 균일한 간격의 단위를 표시해 사용자의 선택을 용이하게 만들 수 있습니다.



<스텝퍼>

스텝퍼는 사람들이 충분 값을 늘리거나 줄이는 데 사용하는 2세그먼트 컨트롤입니다.



기본 원칙

1. 스테퍼 자체는 값을 표시하지 않기 때문에 스테퍼는 현재 값을 표시하는 필드 옆에 있어야합니다.
2. 스테퍼가 영향을 미치는 값을 명확하게 표현합니다. 예를 들어 스테퍼에 옆에 스테퍼에 따라 변하는 값을 텍스트 필드로 보여주는 방법을 사용할 수 있습니다. 이는 사용자가 값을 텍스트 필드로도 입력할 수 있도록 합니다.

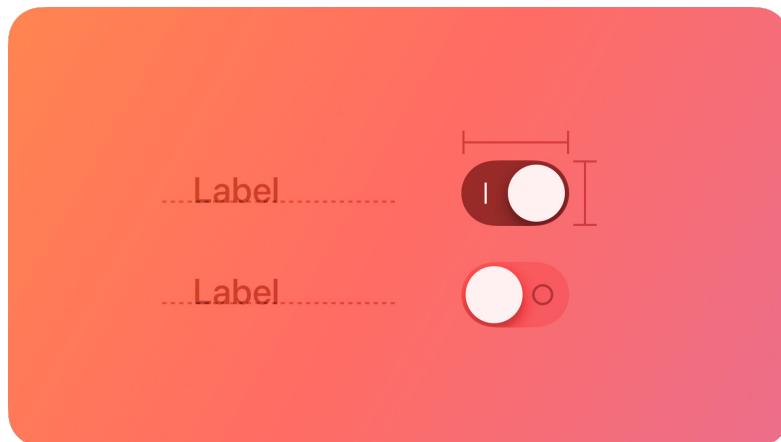
6-3. Selection Controls

기본 원칙

1. 스위치는 두 개의 상반된 선택지가 있을 경우 사용합니다.
2. 확인란 같은 것은 체크박스를 사용합니다.
3. 2개 이상의 선택지를 제시하는 경우는 라디오 버튼을 사용합니다.
4. 그러나 5개 이상의 옵션을 제시해야 하는 경우라면, select(드롭다운) 사용을 고려합시다.

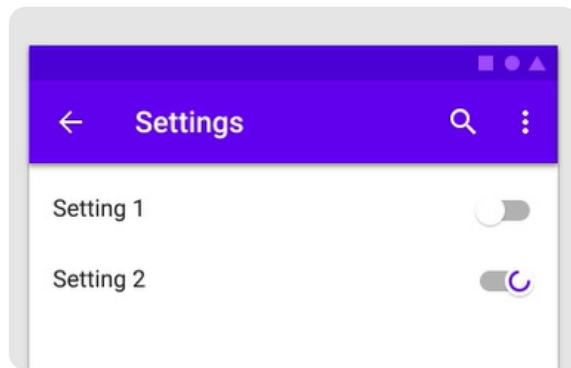
< 스위치(토글) >

스위치는 반대의 상태 중 하나를 고를 수 있도록 합니다. 예를 들어 켜거나 끄는 것처럼, 각각의 상태를 보여주기 위해서 다른 모양을 사용합니다.



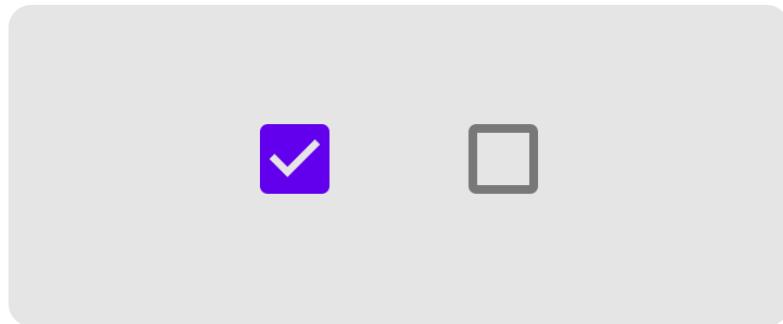
기본 원칙

1. 스위치는 두 개의 상반된 선택지가 있을 경우 사용합니다.
2. 모바일 및 태블릿에서 단일 항목을 켜거나 끕니다. PC에서는 클릭하기 더 좋은 체크박스를 더 추천합니다.
3. 무언가를 즉시 활성화 또는 비활성화시킵니다.
4. 그래픽 자체 내에 "on" 및 "off" 텍스트를 포함하는 스위치를 피합니다. 그래픽 자체만으로 on/off 상태를 파악할 수 있게 하세요.
5. 상태 변경이 지연되는 경우, 처리 상태 애니메이션을 사용하여 지연을 표시하면, 사용자가 이를 인지할 수 있습니다.



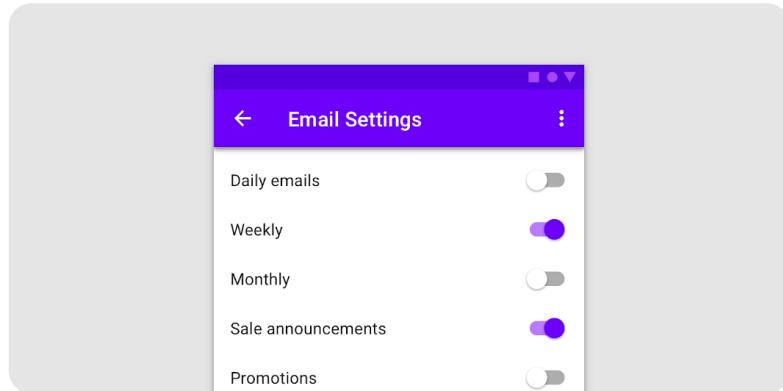
< 체크박스 >

사용자가 세트에서 여러 개의 옵션을 선택할 수 있습니다. 체크박스를 사용함으로써 아이템을 끄거나 켜 수 있습니다.



기본 원칙

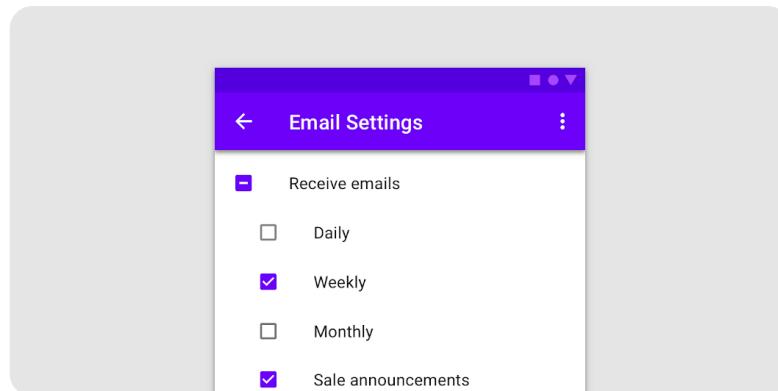
1. 리스트에서 하나 이상의 옵션을 선택할 경우 체크 박스를 사용합니다.
2. 하위 선택을 포함하는 리스트를 보여줄 수 있습니다.
3. 데스크탑 환경에서 아이템을 켜거나 끄기에 유리합니다.
4. 여러 개의 옵션이 포함된 리스트의 경우, 스위치 대신 체크박스를 사용하세요.
5. 체크박스는 연관된 아이템들을 암시하고, 시각 공간을 덜 사용합니다.



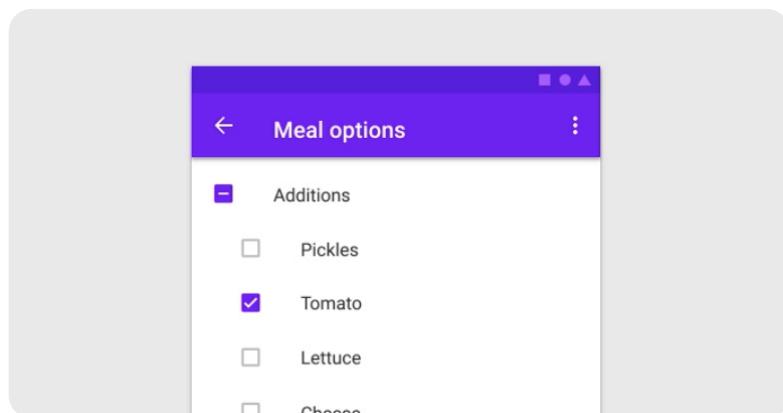
▶ 토글과 체크박스의 비교

6. 체크박스는 다른 체크박스와 부모-자식 관계를 가질 수 있습니다.

- 부모 체크박스가 선택 되면, 모든 자식 체크박스가 체크됩니다.
- 부모 체크박스가 선택 해제 되면, 모든 자식 체크박스가 선택 해제됩니다.

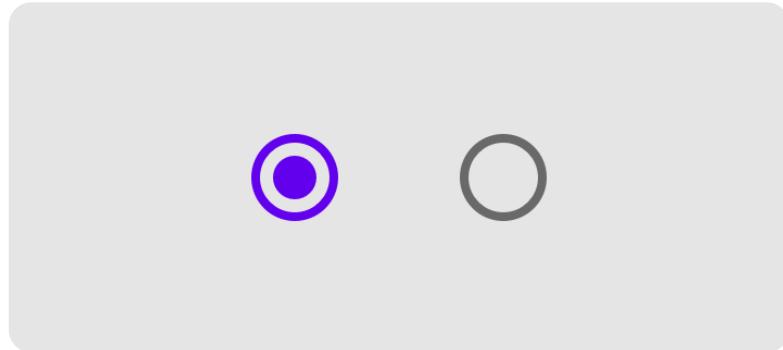


7. 자식 체크박스가 일부만 선택 되어 있을 경우, 부모 체크박스에 기본 체크 표시가 아닌 다른 표시를 사용하여 일부만 선택 되었다는 것을 보여줍니다.



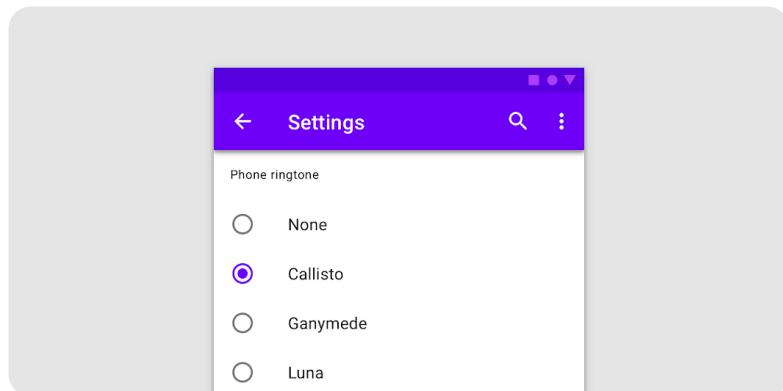
< 라디오 버튼 >

사용자가 옵션 목록에서 하나의 옵션을 선택할 수 있습니다.

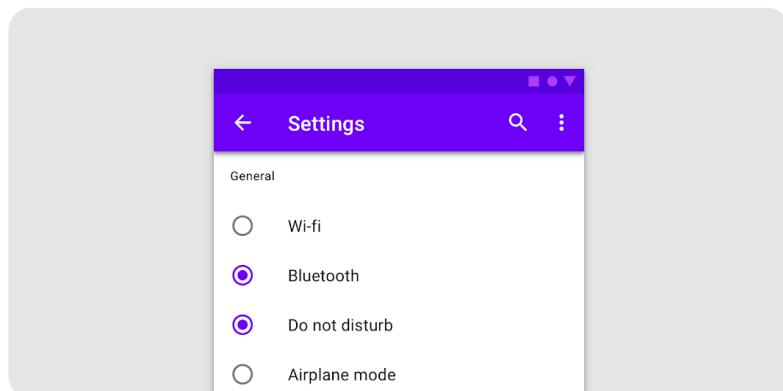


기본 원칙

1. 옵션 개수가 적어서 사용자가 선택할 수 있는 모든 옵션을 한 번에 보여줘도 되는 경우
라디오 버튼을 사용합니다.
2. 옵션 개수가 많다면 공간을 덜 차지하는 Select(드롭다운)를 사용하는 것이 좋습니다.
3. 목록에서 여러 개의 항목을 선택해야 하는 경우, 라디오 버튼 대신 체크 박스를 사용합니다.



4. 라디오 버튼을 사용하여 항목을 켜거나 끄도록 하면 안됩니다. 해당 기능이 필요하다면, 토글을 사용합시다.



7. 폰트

<글씨 크기 >

- 모든 폰트에는 일반체와 볼드체가 있습니다.
- 적절한 폰트 크기를 사용해야 합니다.
- 대표적인 텍스트 크기

제목 : 24px 소제목: 20px 본문: 15px

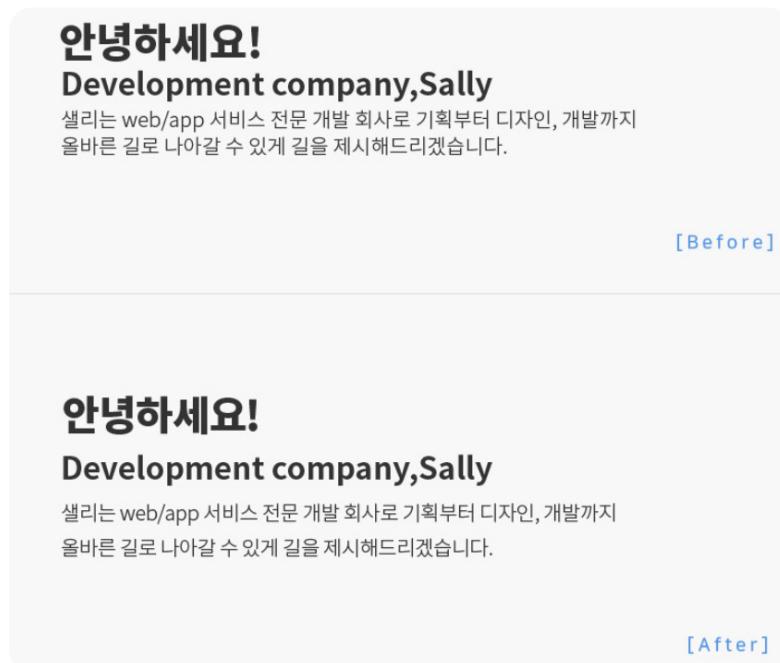
HIERARCHY	WEIGHT	SIZE	LINE HEIGHT
Headline-Emphasis	Semi Bold	32	150%
Headline	Regular	32	150%
Title1-Emphasis	Semi Bold	24	150%
Title1	Regular	24	150%
Title2-Emphasis	Semi Bold	20	150%
Title2	Regular	20	150%
Title3-Emphasis	Semi Bold	18	150%
Title3	Regular	18	150%
Body1-Emphasis	Semi Bold	15	150%
Body1	Regular	15	150%
Body2-Emphasis	Semi Bold	14	150%
Body2	Regular	14	150%
Caption-Emphasis	Semi Bold	12	150%
Caption	Regular	12	150%
Tag	Regular	11	150%

▶ 실제 사이트에서 사용한 폰트 위계

기본 원칙

1. 인터페이스 주 요소와 텍스트 계층을 이해하기 쉽게 만들어야 합니다.
2. 계층 구조를 설정할 때는 폰트 크기, 굵기, 색상 등을 사용해 중요한 정보를 강조할 수 있도록 해야합니다.
3. 사용자의 디바이스에 따라 최적화된 가독성을 제공하기 위하여 반응형 디자인을 제공합니다.

< 글간과 행간 >



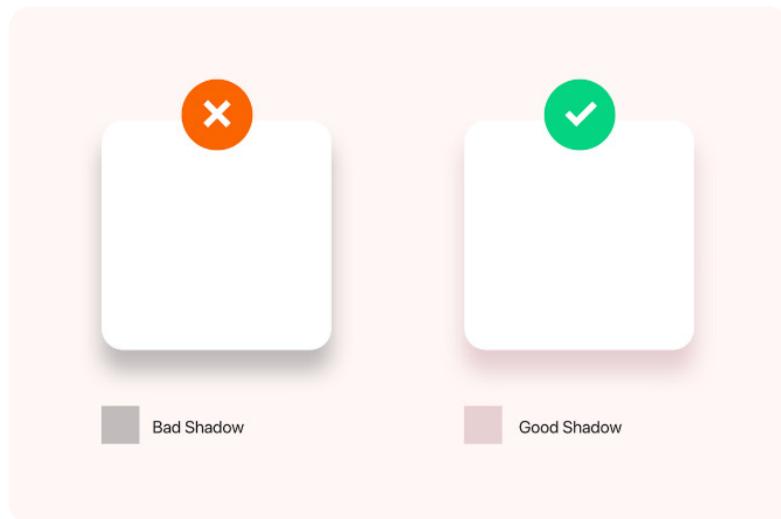
기본 원칙

1. 기본적으로 읽기 쉽고 명확한 행간을 설정합니다. 일반적으로 본문 텍스트 행간은 150%로 설정합니다.
예시로 본문 텍스트가 17포인트인 경우 20.4포인트 or 25.5포인트 사이에서 행간을 설정합니다.
2. 제목과 부제목은 각 텍스트 요소에 따라 적절히 행간을 설정해야 한다. 일반적으로 제목의 행간은
좁게 설정하고, 부제목의 행간은 본문 텍스트와 유사하게 설정 합니다.
3. 행간 설정에 따라 텍스트 가독성이 영향받을 수 있기에 적절한 행간을 유지하여 텍스트 가독성을
높일 수 있도록 조절 합니다.
4. 다양한 디스플레이 크기 대응해야 합니다. 기기와 화면 해상도에 따라 행간이 달라질 수 있으므로 이를
고려하여 다양한 디스플레이 크기에서도 균일한 행간 값을 유지하도록 설정 합니다.

8. 컬러

기본 원칙

1. 일관된 색상 테마를 적용합니다. 메인 색상과 보조 색상, 그리고 강조되어 사용되는 색상들이 서로 잘 어울리고 조화를 이루는 테마를 사용하면, 전체적인 UI가 깔끔하고 친숙하게 보입니다. 확인란 같은 것은 체크박스를 사용합니다.
2. 적절한 그림자 색상을 활용하세요. 그림자라고 모두 검은 것은 아닙니다.



< 메인 색상 >

peer의 메인 컬러는 #060623 입니다.

#060623 Color Hex



기본 원칙

1. 브랜드 색을 사용합시다. 메인 색상은 앱의 브랜드를 강조하는 데 중요한 역할을 합니다. 기업 로고나 브랜드의 기본 색상을 선택하여 앱의 메인 색상으로 사용하세요.
2. 상호작용성을 보여주기 위한 대표 컬러 사용을 고려합니다. 예를 들어 메모 앱에서는 상호작용 컬러가 노란색이고, 달력에서는 빨간색입니다. 상호작용을 보여주는 대표 컬러를 정했다면, 다른 색상이 이 대표 컬러와 겹치거나 방해하지 않도록 합니다.

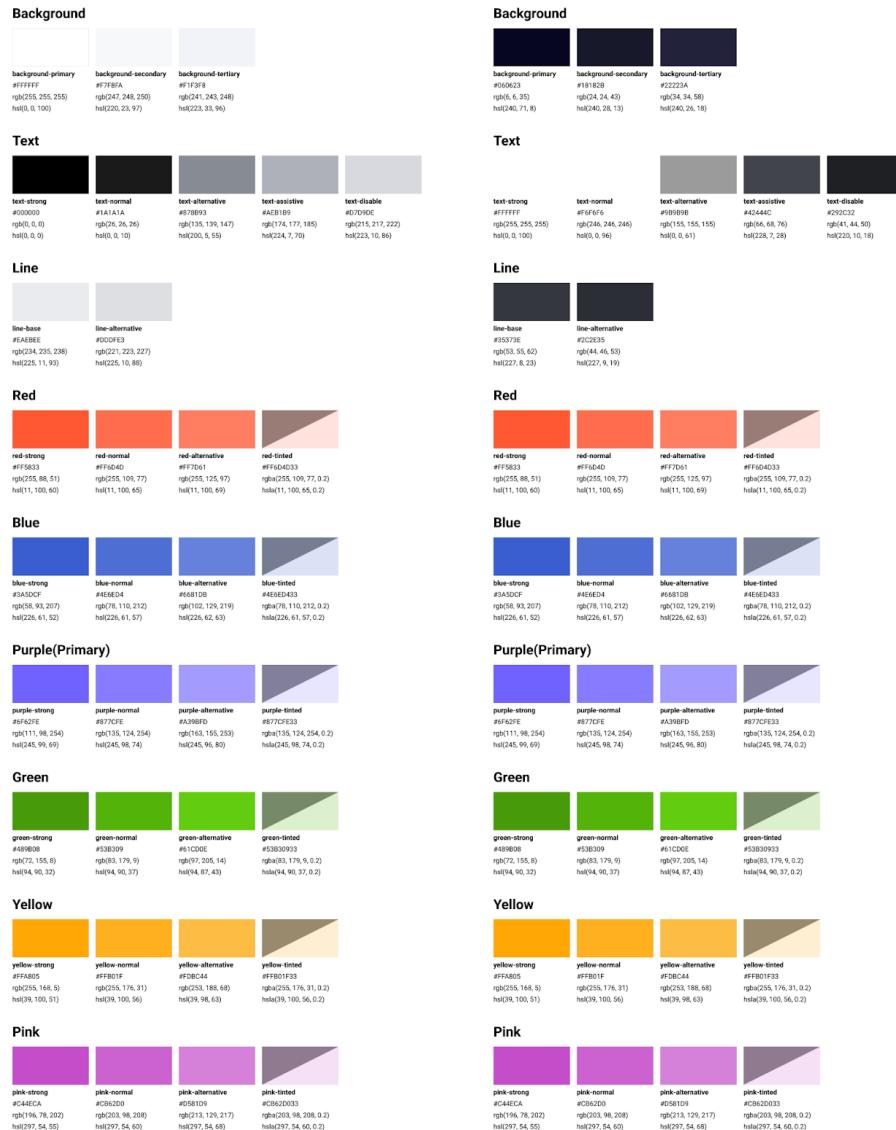
< 보조 색상 >

기본 원칙

- 메인 색상과 상호보완적인 색상을 선택하거나, 브랜드의 이미지와 일관성을 유지할 수 있는 다른 색상을 사용 합니다. 보조 색상은 메인 색상과 조화를 이룰 때 가장 효과적입니다.
- 보조 색상은 메인 색상과 대조적이면서도 조화롭게 어울려야 합니다. 높은 대비와 낮은 대비를 적절하게 혼합 하여 앱에서 아이콘, 버튼, 링크 등의 요소를 강조하세요.
- 보조 색상을 선택할 때 다양한 디스플레이 크기와, 밝은 모드와 어두운 모드에 대한 대응도 고려하세요. 두 가지 모드에서 모두 잘 작동하는 색상을 설정하세요.

< 팔레트 >

라이트모드 / 다크모드



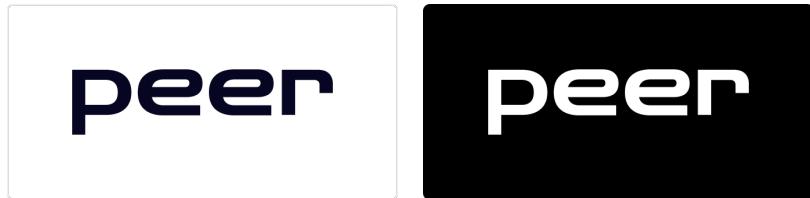


Peer 디자인

Peer LOGO

BRAND LOGO

로고는 주로 디지털 및 인쇄 매체에서 사용됩니다. 또한 웹사이트, 소셜 미디어, 문서, 프로모션물 등 다양한 매체에 적용됩니다. 피어 로고는 전체 시그니처 색상 (background-primary / #060623) 또는 전체 흰색(text-strong / #FFFFFF)으로만 표시할 수 있습니다.



Minimum Clear Space and Minimum Size

로고 주변의 여백은 peer의 p 높이와 같거나 그 이상이어야 합니다.



Display

서비스명/브랜드명은 로고에 표시된 대로 소문자로만 표시할 수 있으며, 다른 방식으로는 사용할 수 없습니다.

CORRECT peer

MISUSE Peer X pEEr X PEER X peeR X

NOT Allowed

로고를 수정하거나 왜곡하거나, 색상을 변경하거나, 추가 요소를 추가하지 마세요.



App Icon

앱 아이콘은 피어의 기능 및 서비스를 나타내는 데 사용할 수 있습니다. 둑근 모서리는 구글 및 애플스토어의 동적 적용을 따라갑니다.



NOT Allowed

모든 로고 데이터는 원본 디자인의 가로 세로 비율을 변경하지 않고 축소/확대할 수만 있습니다.
로고를 사용할 때 왜곡, 추가 또는 색상 변경은 허용되지 않습니다.

1. Color

로고의 색상을 변경하지 않습니다.



2. Shape

로고의 모양을 왜곡하거나 변형하지 않습니다.

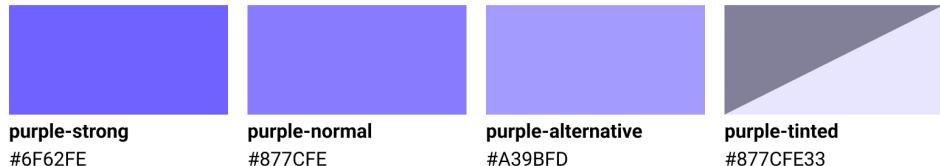


Peer Design Foundation

Colour

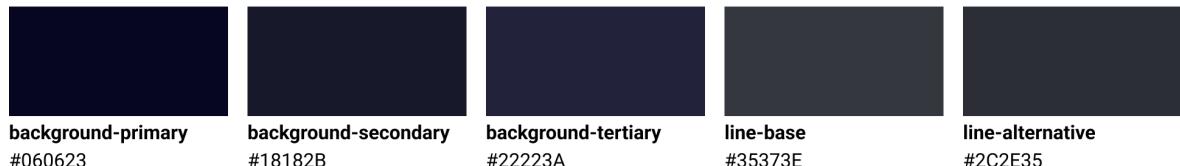
BRAND COLOUR

PURPLE

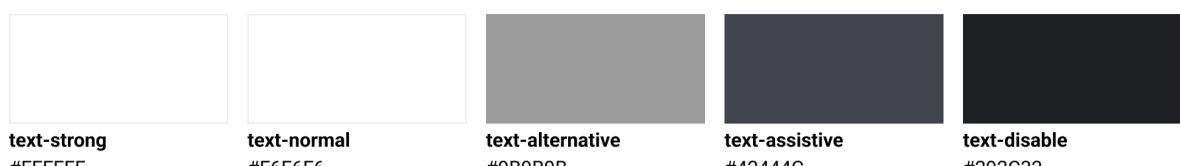


DARK MODE

BACKGROUND

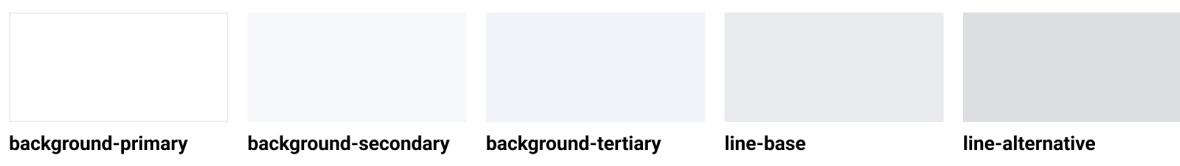


TEXT

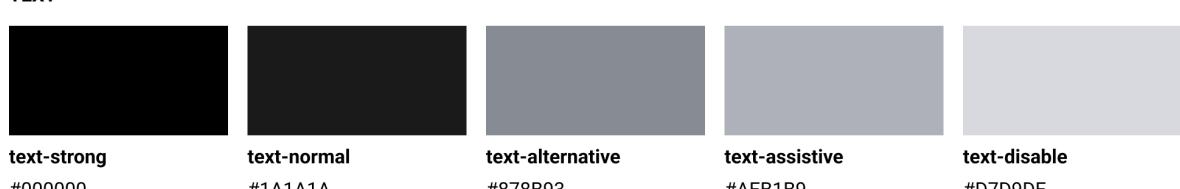


LIGHT MODE

BACKGROUND



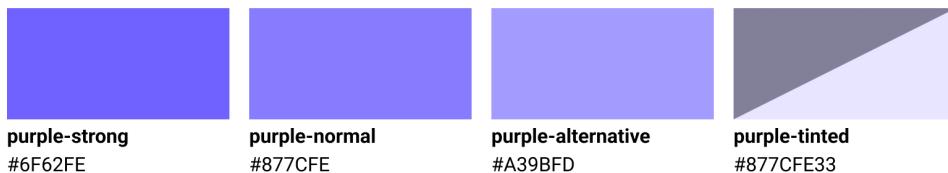
TEXT



Colour

SEMANTIC COLOUR

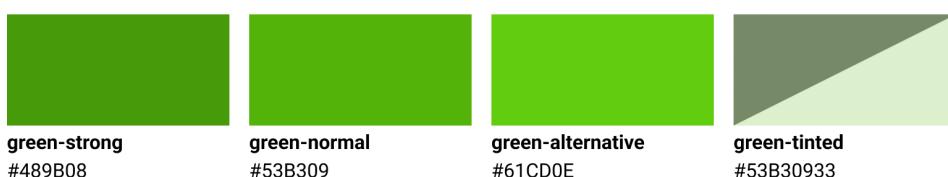
PURPLE : CHECKED OR HIGHLIGHT OR TAG



RED : ERROR OR TAG



GREEN : TAG



YELLOW : TAG



Typography

FONT

Pretendard Vatiabile

USAGE

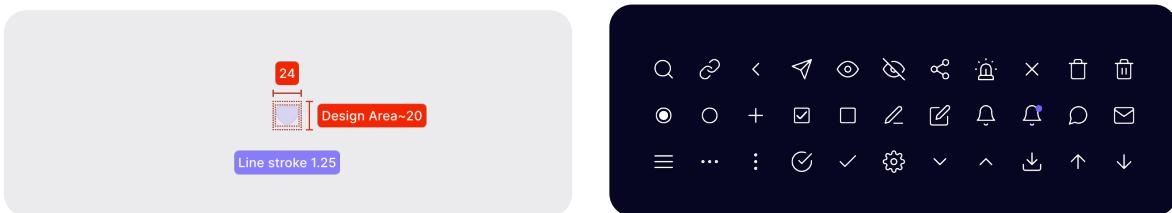
HIERARCHY	WEIGHT	SIZE	LINE HEIGHT
Headline-Emphasis	Semi Bold	32	150%
Headline	Regular	32	150%
Title1-Emphasis	Semi Bold	24	150%
Title1	Regular	24	150%
Title2-Emphasis	Semi Bold	20	150%
Title2	Regular	20	150%
Title3-Emphasis	Semi Bold	18	150%
Title3	Regular	18	150%
Body1-Emphasis	Semi Bold	15	150%
Body1	Regular	15	150%
Body2-Emphasis	Semi Bold	14	150%
Body2	Regular	14	150%
Caption-Emphasis	Semi Bold	12	150%
Caption	Regular	12	150%
Tag	Regular	11	150%

Iconography

Type & Usage

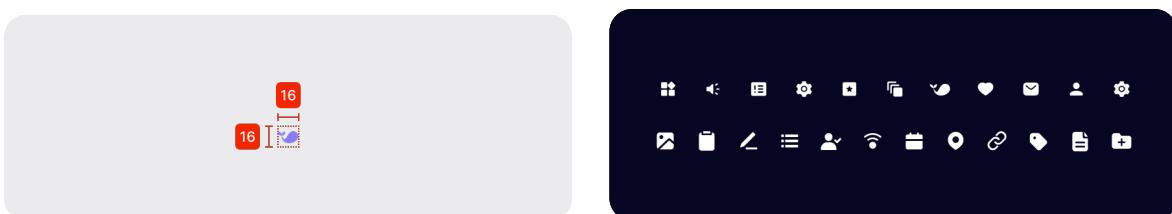
1. System Icon

시스템 아이콘은 앱의 메뉴 또는 기능을 담습니다. 서비스 전반에서 흔히 사용되므로 무게감이 적은 Line 스타일로 구분합니다.



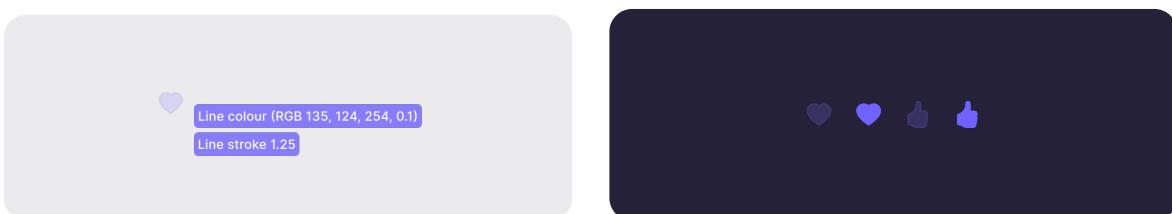
2. Symbol Icon

심볼 아이콘은 텍스트보다 의미를 빠르게 전달하기 위해 사용합니다. 사용자의 고민 시간을 늘릴 수 있는 불필요한 디테일을 최소화하고 간결하고 명확해야하며 면을 채운 Solid 스타일로 구분합니다.



3. Etc

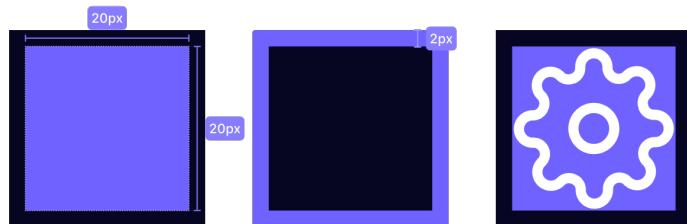
하트(관심리스트)와 땀즈업(좋아요) 아이콘은 사용자의 시선을 사로잡고 행동을 유도하기 위해 예외적으로 면을 채운 선 스타일을 적용합니다.



Icon Creation

1. Grid

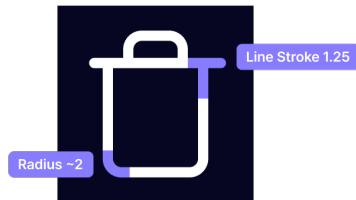
24 X 24 px 안에서 작업합니다. (디자인 영역 20 X 20 px, 패딩 2 px)



2. Line Stroke & Radius

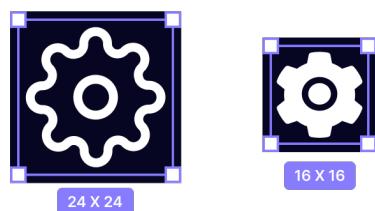
라인 스타일을 사용하는 시스템 아이콘의 두께는 1.25로 합니다.

Radius 값은 0 ~ 2 사이값으로 하고, 때에 따라 소수점을 허용합니다.



3. Rules

아이콘은 예측 가능성�이 크게 작용하는 시각 요소이므로, 널리 쓰이는 패턴(멘탈 모델)을 따라가야합니다. 또한 시스템 아이콘인지 심볼 아이콘인지 사용 목적을 분명히 하고 디자인합니다. 기본 그리드는 24 X 24 px를 따르지만, 8의 배수로 늘어나고 줄어들 수 있습니다. (예시. 16 X 16 px)



Layout

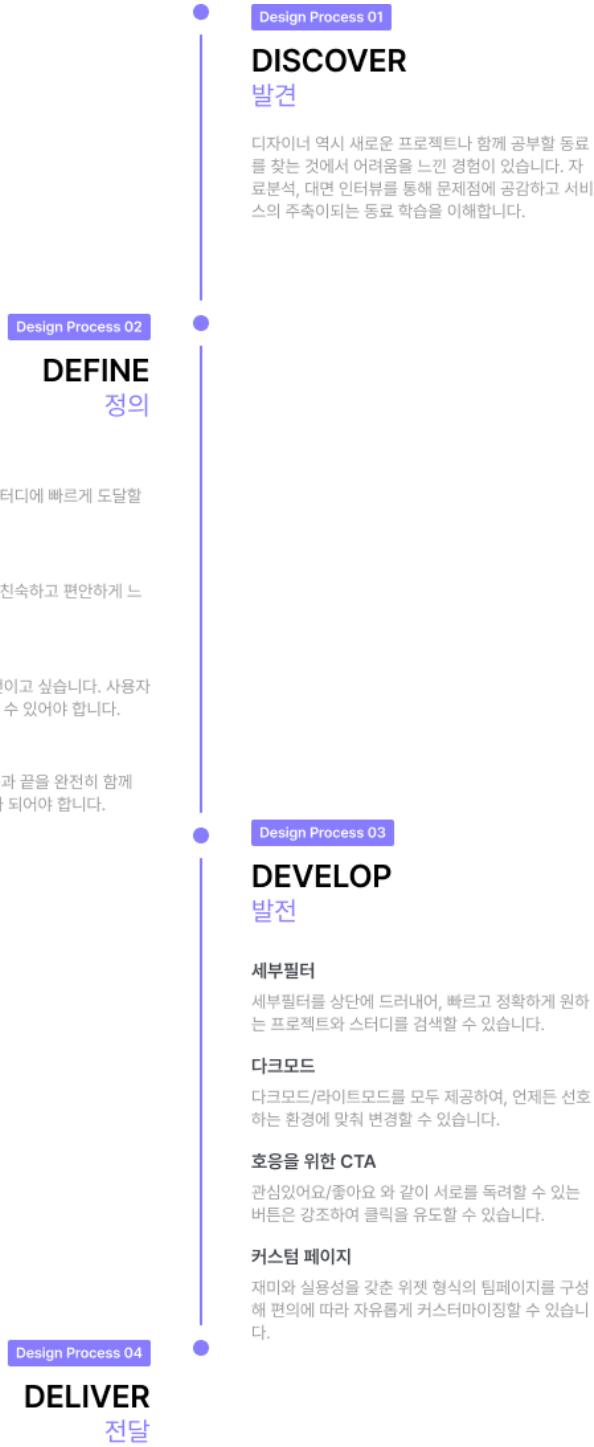
1. Resolution

웹 해상도는 1920, 모바일 해상도는 360을 기본으로 작업하였습니다. (브레이크 포인트 480)

2. Content width & margin

웹은 콘텐츠 영역 1280을 중심으로 유연하게 반응합니다. 모바일은 좌우 16px 고정 마진값을 가지고 유연하게 반응합니다.

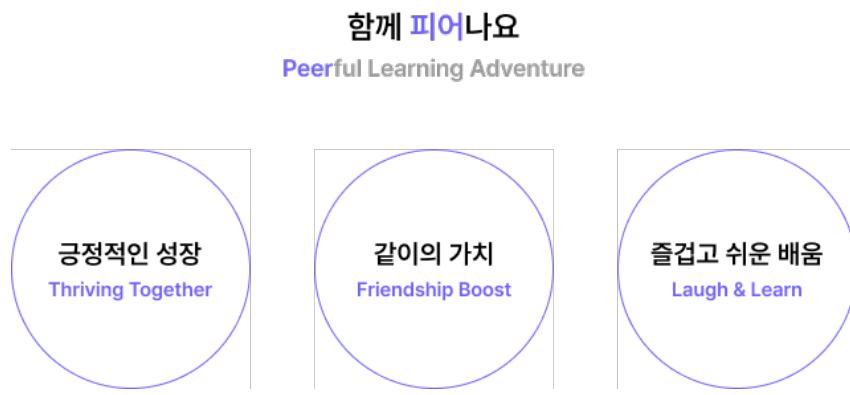
Peer Design Journey



Peer UX Writing

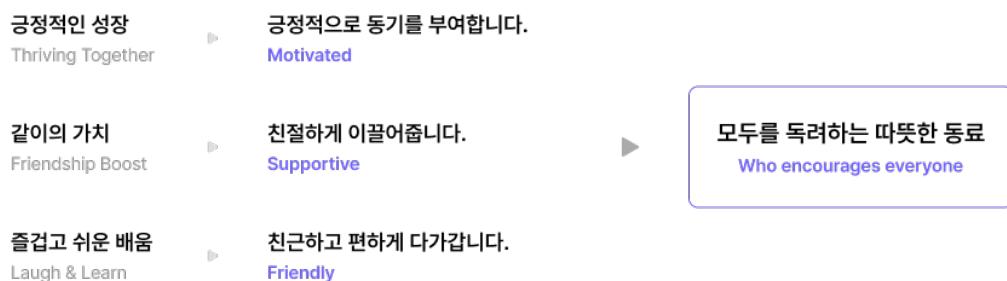
Core Value

피어에서 가장 핵심이 되는 고유한 가치입니다.



Persona

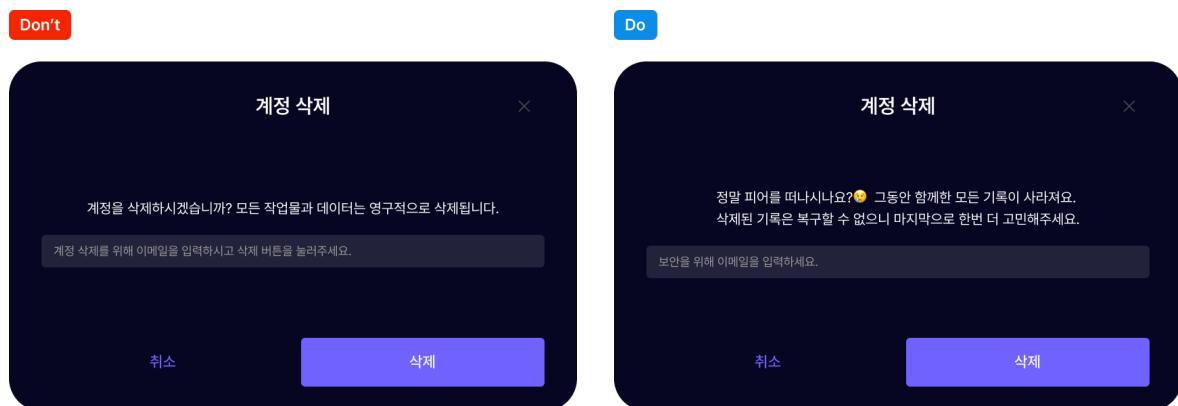
핵심 가치를 통해 찾아낸 피어만의 목소리를 정의합니다.



Writing principle

1. 친절하게 알려줍니다

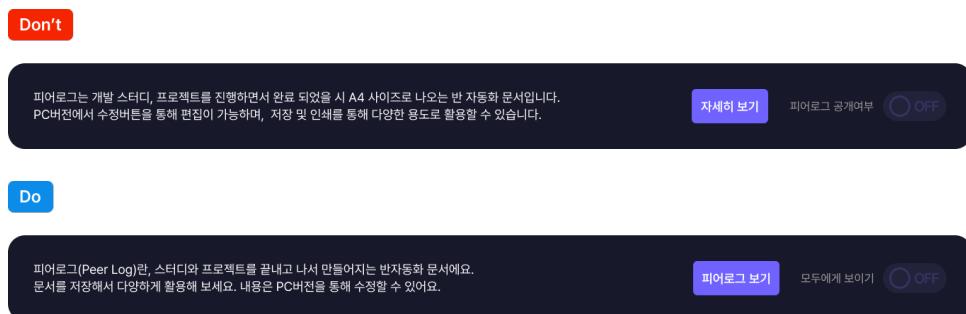
- 편안한 느낌을 줄 수 있도록 [해요체]를 사용합니다.
- 이해를 도울 수 있도록 한 번에 한가지 정보를 제공합니다.
- 사용자의 상황에 대한 충분한 설명을 제공합니다.



피어의 핵심 가치는 모두가 배움의 동료라는 것입니다.
유저들에게 늘 곁에 있는 동료처럼 친근한 말투로 다가갑니다.

2. 쉽고 간단하게 알려줍니다

- 문장을 길게 풀어서 적기보다 한 눈에도 읽기 쉬운 문장을 적습니다.
- 이해하기 쉬운 직관적인 표현을 사용합니다.



피어로그라는 새로운 개념을 설명합니다. 익숙하지 않은 정보는 알기 쉽게 적습니다.

3. 올바르게 알려줍니다

- 국립국어원을 참고해 올바른 맞춤법을 사용합니다.
- 한자나 문어체, 외래어 사용을 지양합니다.

Don't

맞춤 스터디를 빠르게 찾아요. ^

기술스택 프레임워크를 입력해주세요.
Angular X Node.js X Javascript X

기간 활동기간을 선택해주세요. ▾

활동지역 활동지역을 선택해주세요. ▾

초기화

작업 단계 모집전 모집완료 진행중 진행완료

활동방식 온라인 오프라인 혼합

Do

맞춤 스터디를 빠르게 찾아요. ^

개발언어 찾고 있는 개발언어를 입력하세요.
Angular X Node.js X Javascript X

작업 기간 원하는 기간을 선택하세요. ▾

활동 지역 활동지역을 선택하세요. ▾

모두 지우기

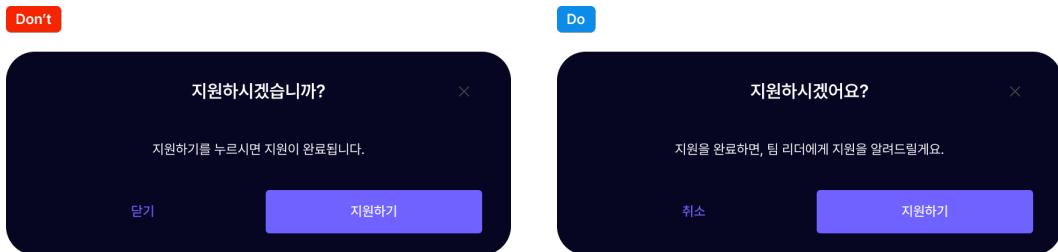
작업 단계 모집 전 모집 완료 진행 중 진행 완료

활동 방식 대면 비대면 혼합

개발의 특성상 많은 외래어가 사용됩니다. 불가피한 상황을 제외하고는 한글로 변경합니다.

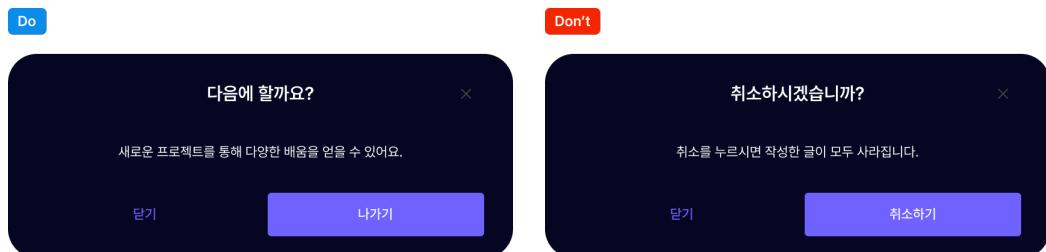
4. 사용자에게 공감합니다

원하는 프로젝트나 스터디를 지원할 때 설렘과 긴장감이 공존합니다. 지원을 하고나서 어떤 과정들을 거쳐야 할지 알 수 있다면, 그 긴장감을 줄여줄 수 있습니다. 또한 지원을 취소하는 경우에도, 프로젝트/스터디 지원을 통해 얻는 이점을 알려주며 한번 더 선택할 수 있는 기회를 제공합니다.



원하는 프로젝트나 스터디를 지원할 때 설렘과 긴장감이 공존합니다.

지원 후 과정을 알 수 있다면, 그 긴장감을 줄여줄 수 있습니다.



지원을 취소하는 경우에도, 프로젝트/스터디 지원을 통해 얻는 이점을 알려주며

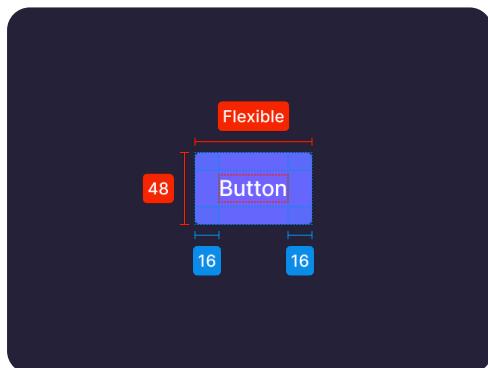
한번 더 선택할 수 있는 기회를 제공합니다.

Peer Components

Buttons

Box button

박스 버튼은 가장 보편적으로 사용되는 CTA(Call To Action)버튼입니다. 버튼이 위치하게되는 영역에 따라 가로방향으로 유연하게 확장 및 축소할 수 있습니다.



Button colour	<input checked="" type="radio"/> Purple/purple-strong
Text colour	<input type="radio"/> Text/text-normal
Font	Pretendard Variable Medium
Weight	500
Size	15px
Line height	150%

Box Button Variation

박스 버튼은 2개 이상이 함께 사용되는 경우 아래 세 가지 스타일을 활용해 중요도를 구분합니다.
(중요도. primary>secondary>territory)

Box button_big_primary

Property 1 ◆ Box button_big_primary

Box button_big_secondary

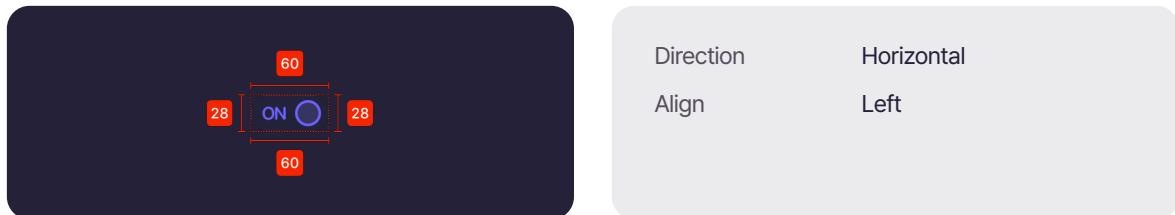
Property 1 ◆ Box button_big_secondary

Box button_big_teritory

Property 1 ◆ Box button_big_teritory

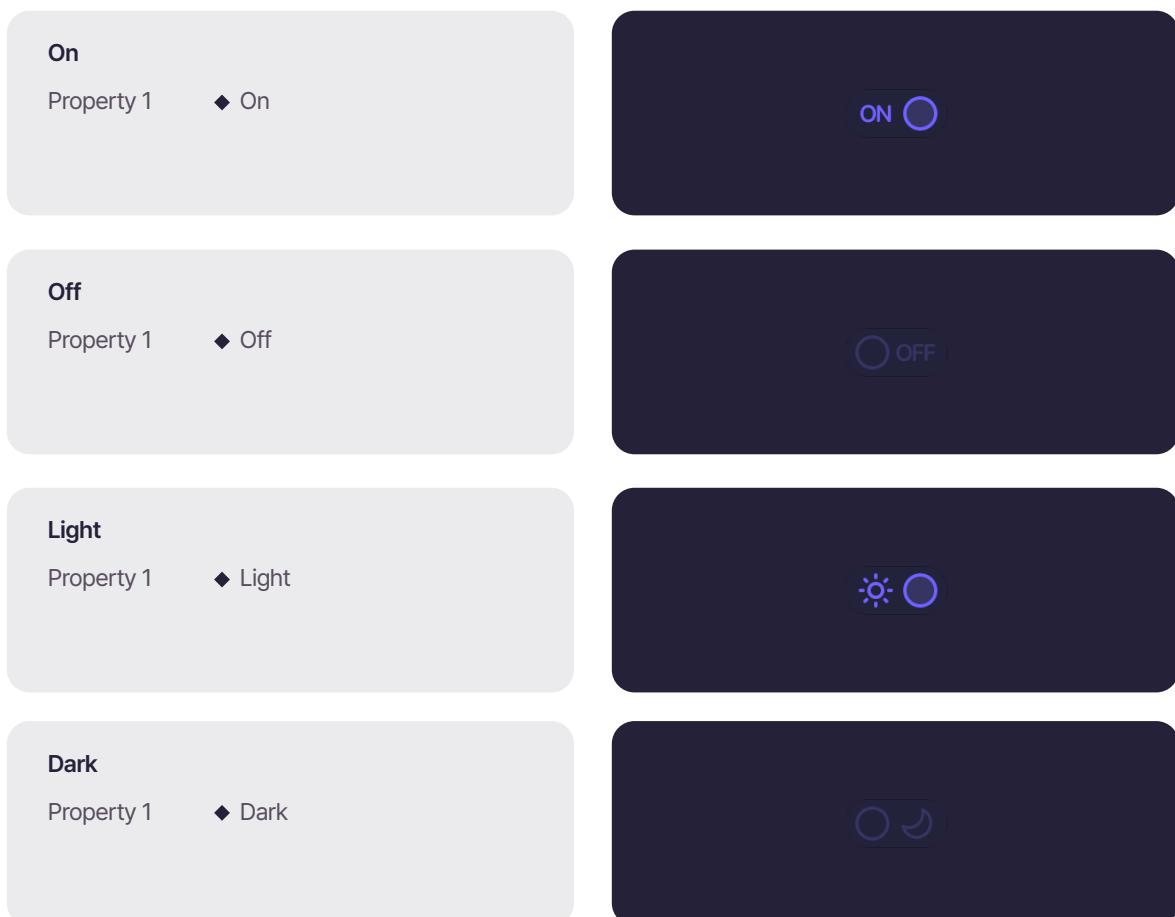
Toggle

토글은 두 가지 선택지 중 하나를 반드시 선택해야하는 경우 사용하는 버튼입니다. 누르는 즉시 상호배타적으로 작동하는 것을 특징으로 합니다.



Toggle Variation

현재 사용하는 토글의 종류는 아래와 같습니다.

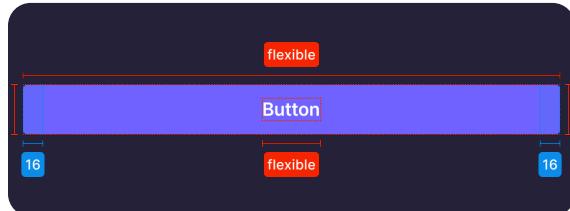


Other buttons

그 외 다양한 버튼을 사용합니다.

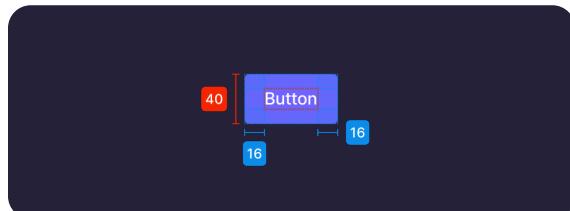
Box button_small_long

Property 1 ◆ Box button_small_long



Box button_small_short

Property 1 ◆ Box button_small_short



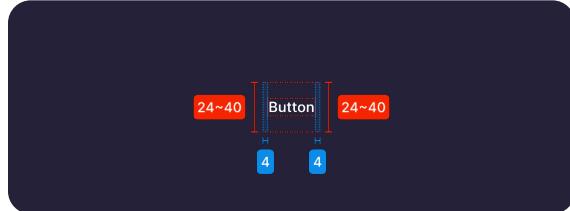
Outlined button_small

Property 1 ◆ Outlined button_small



Text button

Property 1 ◆ Text button



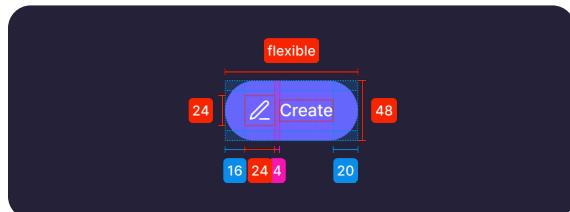
FAB_default

Property 1 ◆ FAB_default



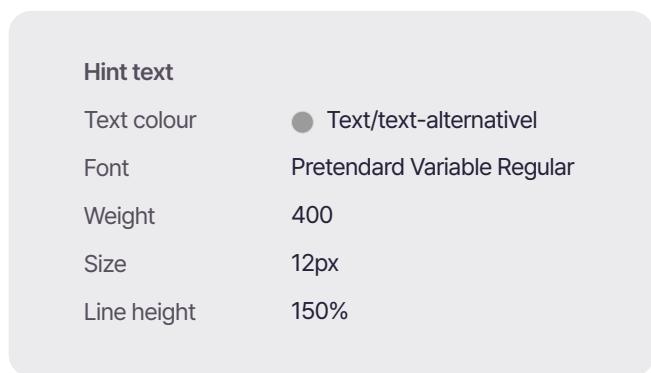
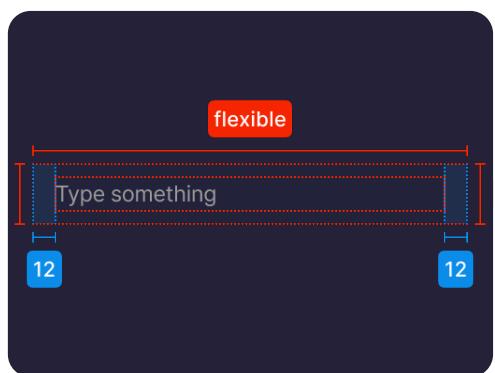
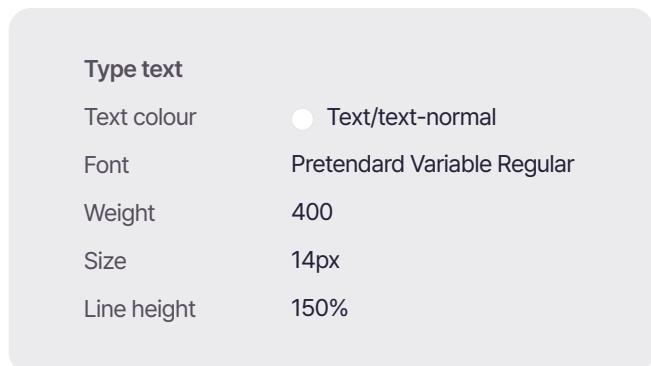
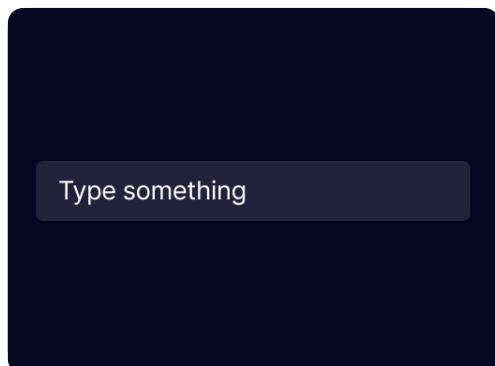
FAB_extended

Property 1 ◆ FAB_extended



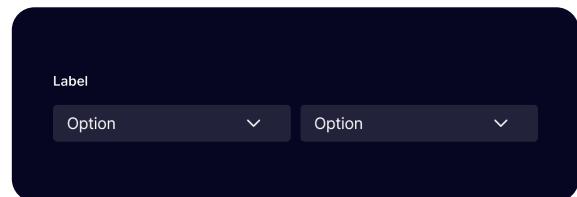
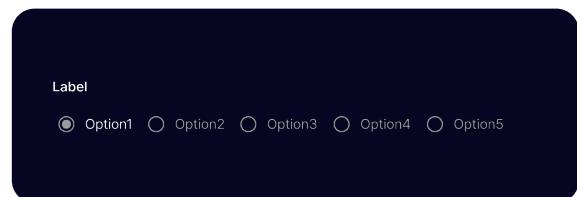
Text Inputs

텍스트 인풋은 글을 작성하거나 옵션을 선택하여 정보를 입력할 때 사용합니다.



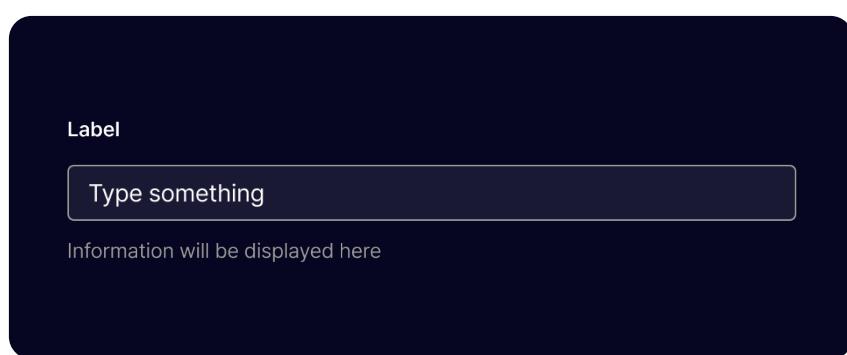
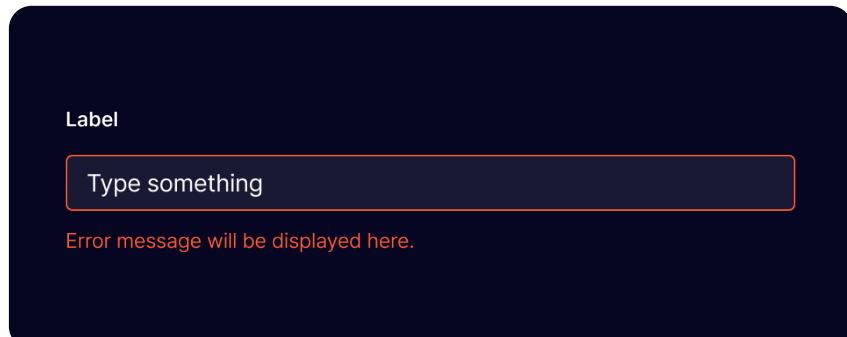
Variation

텍스트 인풋은 텍스트 박스 높이를 유지하고 아이콘 & 버튼 등을 추가하여 다양하게 변형할 수 있습니다.



Status

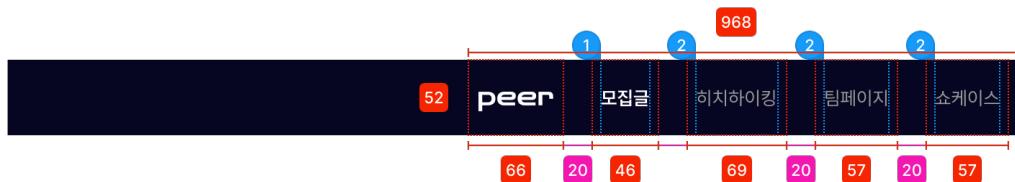
오류(Error) 또는 안내(Information) 상황에서는 아래를 따릅니다.



Navigation Bar & Tab Bar

Web Navigation bar

웹 화면 상단에 위치하는 네비게이션바는 유저가 자신의 위치를 파악하고 원하는 서비스로 빠르게 이동할 수 있도록 돕습니다.



1 현재 페이지

Fill Style	● Text/text-normal
Font	Pretendard Variable Semibold
Weight	400
Size	14px
Line height	150%

2 페이지 이동

Fill Style	● Text/text-alternative
Font	Pretendard Variable Regular
Weight	400
Size	14px
Line height	150%



3 검색

4 알람

5 관심리스트

6 프로필사진 (마이페이지CTA)

Mobile Tab bar

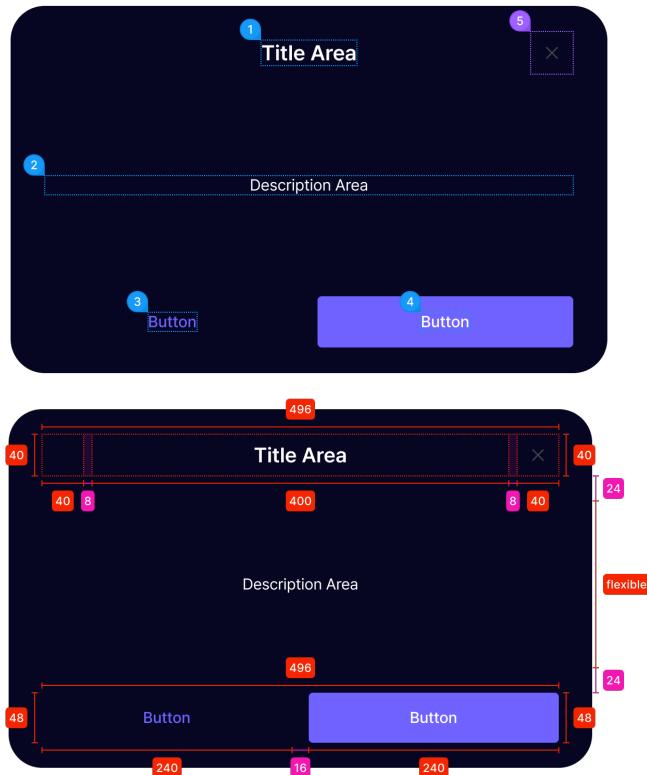
웹 네비게이션바는 모바일 사이즈로 반응시 탭바(Tab Bar)로 일부 변경되어 모바일 상/하단에 위치됩니다. 각 탭은 유저의 이해를 돋기위해 라벨을 함께 배치합니다.



Modal

Web Modal

모달은 정보 입력, 작업 완료 안내, 수정 및 설정 등에서 다양하게 활용합니다. 모달을 구성하는 콘텐츠 양에 따라 모달 사이즈는 세로방향으로만 확장될 수 있습니다. (단, 예외를 허용합니다.)



1 Title Area

Colour	● Text/text-normal
Font	Pretandard Variable Semibold
Weight	600
Size	20px
Line height	150%

2 Description Area

Colour	● Text/text-normal
Font	Pretandard Variable Regular
Weight	400
Size	14px
Line height	150%

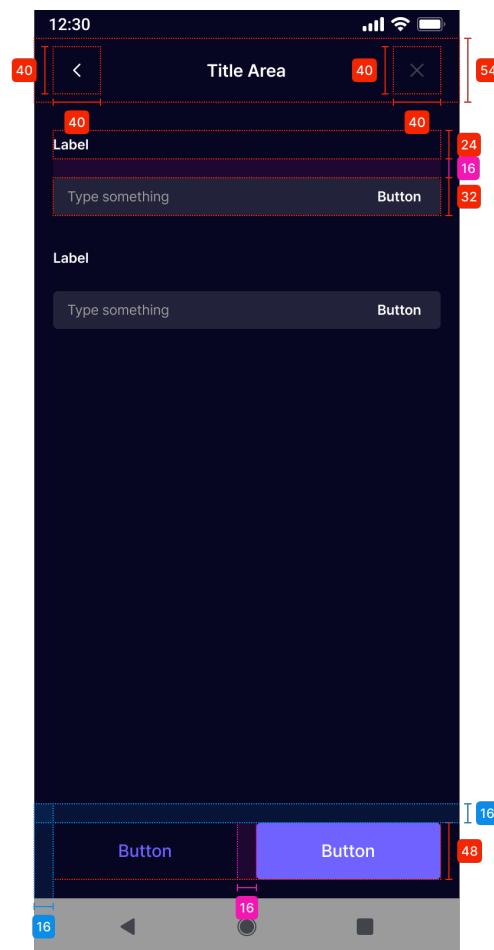
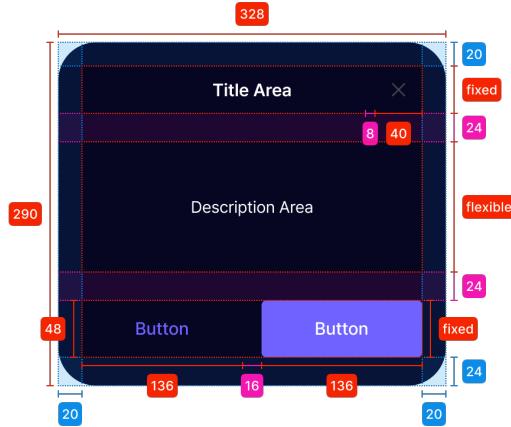
3 Territory Box Button

4 Primary Box Button

5 Icon Area

Mobile Modal

모달은 모바일 사이즈로 반응시 가로방향으로 사이즈가 축소(560px > 328px)됩니다. 또한 예 / 아니오로 응답하는 경우와 정보를 입력해야 하는 경우를 구분하여 후자의 경우화면 전체를 채우도록 합니다.



Web Toast

키워드 알림, 에러 알림은 토스트를 활용합니다. 되돌리기 기능이 필요한 경우 되돌리기 버튼도 함께 배치합니다.



Background colour	#201E4F
Corner radius	12px
Text colour	#201E4F
Button colour	#201E4F
Text Size	13px

- 1 알림 또는 에러 안내 메시지
- 2 액션 되돌리기
- 3 토스트 닫기

Mobile Toast

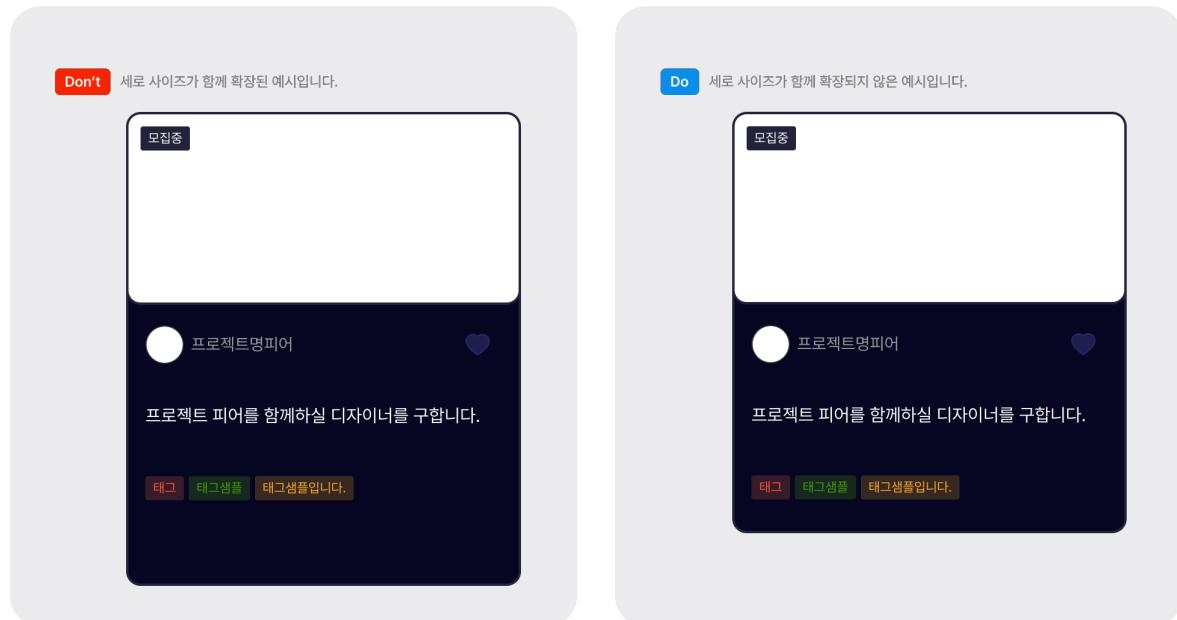
모바일 토스트는 디바이스 화면에서 좌/우 각 16margin을 띄우고 화면을 채웁니다. 내부 메시지는 최대 2줄까지 노출되고 이하 생략됩니다. (예시. '...')



Cards

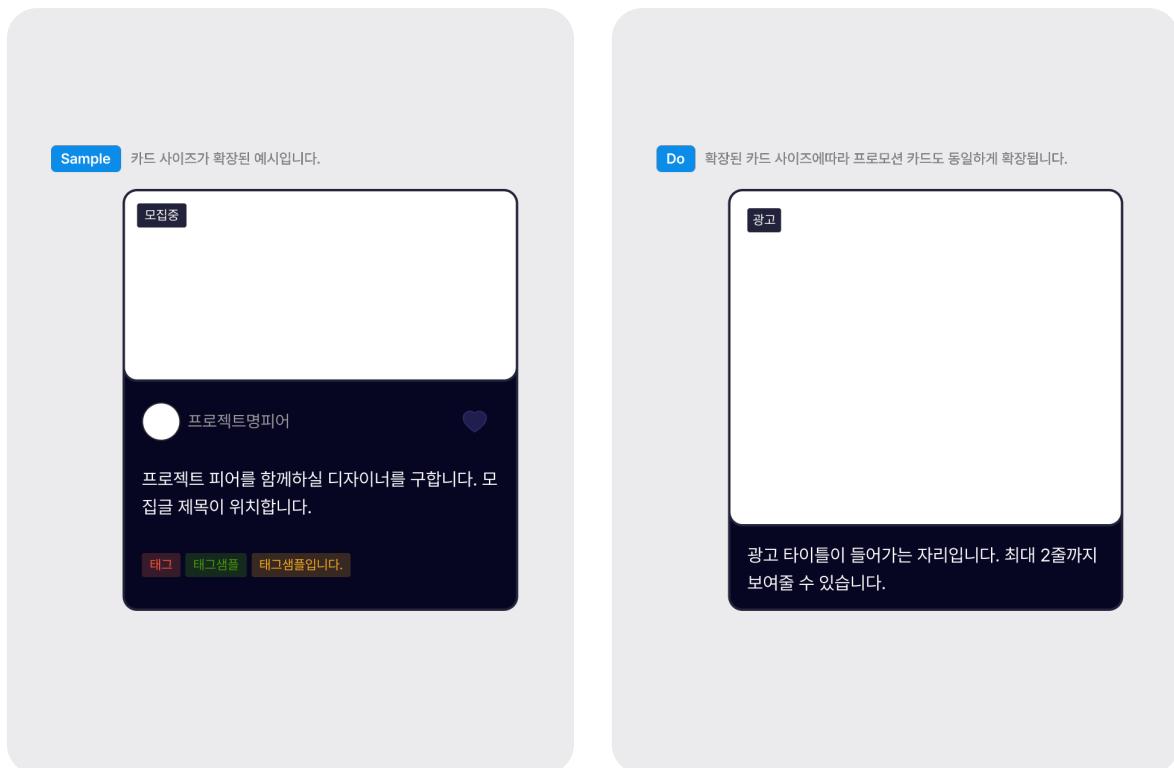
Study / Project Card

스터디 / 프로젝트 카드는 모집글을 찾는 홈, 히치하이킹, 쇼케이스 등 다양한 화면에서 사용합니다. 디바이스 혹은 카드가 위치하게 되는 영역에 따라 카드의 크기와 구성은 유지하되 가로방향으로만 확장할 수 있습니다. 카드는 세부 필터의 모든 옵션을 포함하므로, 옵션 변경이 필요하다면 세부 필터도 함께 고려해야 합니다.



Promotion Card

광고 카드는 스터디/프로젝트 카드와 자연스럽게 노출될 수 있도록 같은 크기와 형태를 따릅니다. 카드가 확장되는 경우 광고 카드 역시 동일한 크기로 확장되어야 합니다.



제 2장

FRONT

FRONT-END

개발자들의 이야기

@woorikim	김우림	94
@jujeon	전준성	96
@jeyoon	윤정연	99
@hyunjung	정현섭	101
@hyna	나현	102
@hyeokim2	김현지	105
@hujeong	정희호	108
@hoslim	임호성	110



@ woorikim 김우림

Peer다웠던 Peer에 참여하며

안녕하세요, Peer웹사이트 개발에 프론트엔드 파트로 참여한 woorikim입니다. Peer 웹 개발 설명회를 들었던 날부터 프로젝트의 마무리가 보이는 지금까지 시간이 정말로 빨리 흘러간 것 같습니다. 이렇게 우연한 기회에서 시작한 여정을 돌아보니, 배운 것도, 아쉬움도, 그리고 앞으로 나아가야 할 방향도 많이 보입니다.

저는 올해 처음으로 프론트엔드 공부를 시작했었고 다뤄본 것이라고는 html, css, JavaScript뿐이었습니다. 그래서 Peer 개발진 모집 공고를 봤을 땐 '이 얇은 경험으로 어떻게 프로젝트에 기여할 수 있을까?' 하고 의문이 들어 망설였던 것 같습니다. 하지만 우연히 Peer 웹 기획 설명회에 참석하게 되었고, 이 프로젝트가 '진짜 내가 사용하고 싶은 서비스'라는 생각이 들어 욕심을 내어 참여하게 되었습니다.

우선 지금까지 맡은 부분을 해낼 수 있었던 것은 동료학습을 추구하는 Peer다웠던 개발 과정이 있었기 때문이라고 생각합니다. 먼저, 프론트엔드 개발의 팔로우업 팀인 참고래 팀의 리액트 기초 스터디에서 시작해, 파이널 미션을 허겁지겁 완성했던 시간들이 떠오릅니다. 떨리는 마음으로 미션 결과물을 발표하며, "확신이 없는 코드를 공유한다는 것은 정말 힘들구나, 채워나가야 하는 부분이 너무 많다"는 것을 여실히 느꼈습니다. 그 후 본격 개발에 들어가면서 "웹 접근성", "next.js 심화", "drag-and-drop 라이브러리" 등 hoslim님이 준비해 주셨던 주제를 가지고 스터디를 더 진행했고, 이때서야 데모코드들을 작성하면서 점점 자신감도 붙여 가고, 팀원들과의 질문 공유를 통해 소통하는 방법도 익숙해져 갔습니다. 이 과정이 있었기에 지금처럼 개발 중 문제상황을 바로바로 공유하거나 편하게 도움을 주고받을 수 있게 되었다고 생각합니다. (여담이지만 이 때 Peer의 동료학습 철학을 체감하고, 이 커뮤니티의 일원으로 남고 싶다는 마음이 들어 운영진에도 참여하게 되었답니다...)

물론, 개발 과정에서 겪은 어려움도 많았습니다. 단순하다고 생각했던 로그인 기능조차 토큰과 쿠키 관리, Axios interceptor, OAuth 로그인 등에서 예상치 못한 오류들을 마주했고, 업데이트 자료가 많지 않은 Next.js 13환경까지 맞물려 해결하는 데 꽤 많은 시간을 할애해야 했습니다. 특히, 쿠키를 다루는 부분에서는 CORS error 때문에 develop 브랜치에 머지를 해서 배포가 되어야만 테스트를 진행할 수 있는 경우가 대부분이었는데, 그 과정에서 곳곳에 쓰이는 axios interceptor가 무한 요청을 시도하거나, 로그인에서부터 전반적인 오류가 났던 당시 상황들을 떠올리면 아직도 아찔합니다. 42와 Google OAuth 로그인 파트에서는 계정 연동 기능까지 포함해 백엔드와 수차례 상의하며 리다이렉션 페이지를 갈아엎고 또 만들고, API문서

를 수정해야 했습니다. 지금까지도 프로젝트의 마무리 단계인데 해결해야 할 문제들이 남아있어 마음이 편치
만은 않고, 대학교와 42과제를 병행하며 매달 블랙홀에 허우적거리느라 Peer에 더 많은 시간을 투자하지 못
한 것도 아쉬움으로 남습니다.

그럼에도 불구하고, 10명이 넘는 팀원들과 함께하며 배운 것들은 앞으로도 정말 크게 남을 것 같습니다. 매
번 다른 팀원들의 코드, 이슈, 풀리퀘스트 리뷰들을 보며 제 자신의 부족함을 깨닫고 있고, 다들 바쁜 와중에
긍정적인 분위기를 유지하는 것을 보면서 대단함을 느낀 적도 한두 번이 아닙니다. 리더님들은 전체 일정에
딜레이가 생기거나 인력에 문제가 생겼을 때도 항상 모두에게 피드백을 받고 실질적인 방법을 찾으려고 하셨
고, 그 덕분에 부담 없이 어려움을 나누고 의견을 낼 수 있었습니다.

큰 팀에서 협업의 경험이 없었던 저로서 이 Peer 프로젝트는 기술적 성장을 이루는 것뿐만 아니라 좋은 팀
의 좋은 구성원이 되는 방법을 알아가는 시간이었다고 생각합니다. 회의실, 랩실에 모여 개발을 하면서 즐거
운 추억들도 생겼고, 앞으로도 42서울 안에서 함께할 수 있는 좋은 동료들을 얻은 기분이라 기쁩니다. 팀원
들 다들 모르셨을 수도 있겠지만 제가 진짜 많이 의지했고, 많이 배웠습니다. 모두 고맙고 고생 많으셨어요!



@ jujeon 전준성

나의 peer 가 아닌 우리의 peer

peer 프로젝트를 진행하면서 가장 기억에 남은 말은, 개발총괄인 류한솔님의 한마디입니다.“이렇게 인복이 좋았던 때가 없다.” 저로서는 정말 감사한 일입니다. 이 말을 꺼냈던 류한솔님 조차 저에게는 귀인입니다. 짧지 않은 기간동안 프로젝트를 진행하면서 이렇게 좋은 사람들과 화기애애한 분위기로 개발을 할 수 있다니 감사한 마음이 컸습니다.

22년 첫 교내 커뮤니티로 peer를 만들었을때는 저 혼자서 모든 일을 다 처리했습니다. 혼자서 고민하고 혼자서 행동했죠. 그러다 옆에서 보다못한 류한솔님이 운영진과 함께 해보라며 조언을 해주었고 그 길로 저만의 peer가 아닌 우리의 peer가 시작되었습니다.

조직을 이끌면서 배우게 된 성공한 조직을 만드는 법

초기에는 6명 내외의 인원으로 구성된 운영진과 함께 움직였습니다. 그러다 피어 웹 개발팀을 꾸리고 난 이후에는 3명의 디자이너를 포함한 약 15인 내외의 개발팀과도 함께 움직이게 되었습니다. 이제는 더 이상 작은 팀이 아닌 여러 팀을 거느린 조직으로 성장한 것입니다.

저는 팀 구성원들의 시너지를 잘 이끌어낼 수 있는 능력이 강점이예요. 팀원들의 생각과 감정을 캐치하고 모두가 만족할 최선의 방향으로 결론내리고 이끌어가는 걸 잘 해왔습니다. 이걸 잘 해내지 못하면 $1 + 1$ 이 2가 아니라 0.5가 되버리는 경험을 많이 하게 됩니다. 반대로 이걸 잘 해내면 2가 아닌 3의 결과를 얻어낼 때도 많죠. 그 덕에 많은 연수원 동료들에게서 좋은 평판을 유지하고 있는게 아닌가 생각하고 있습니다.

그러나 이번은 작은 규모의 팀이 아니라 많은 인원이 포함된 조직입니다. 어떤 방향이 성공적인 조직으로 만드는데 최선일까 고민했습니다. 인원이 많다는 건 해야할 일도 많다는 것이고, 1명이 할 수 있는 일은 제한적일 수 밖에 없습니다. 결국은 누군가에게 위임을 해야하는 것이고, 그 과정에서 가장 최선은 가장 잘 할 수 있는 사람에게 맡기는 것, 믿어주는 것. 그리고 부족함이 보여도 온전히 그의 힘으로 풀어나갈 수 있게 기다려주는 것이었습니다.

개발자, 기획자, 디자이너와 협업경험

école 42에서 개발을 배울때는 개발자들과만 협업할 수 있었기에 실무처럼 기획자와 디자이너와 협업할 기회가 드뭅니다. 운이 좋게도 전직 기획자 출신 분들 그리고 현직 디자이너분들과 함께 협업하게 되었습니다.

다. 이 과정에서 단순히 개발만 한다고 전부가 아니라는 걸 깨닫게 되었습니다.

개발자는 기획과 디자인에 맞춰 개발만 하면 된다고 생각하기 쉽지만, 무엇을 위해서 이런 기획과 이런 디자인의 모습이 되었는지 이해하고 있어야 합니다. 그렇지 않으면 고생해서 개발했는데 기획과 디자인이 갑자기 바뀌게 되면 다시 처음부터 개발해야하는 불상사가 생기기 쉽습니다.

예를 들어 제가 담당했던 모집글쓰기 파트에서 이미 구현이 끝났는데, 기획팀에서 역할 추가 기능에서 이전에 없던 팀리더를 자동으로 추가해달라는 요청이 왔습니다. 정말 난감했던 건 로직상 많은 부분을 뜯어고쳐야하는 상황이어서 상당히 많은 시간비용이 예상되었습니다. 거기서 잘 했던 건 모집글을 볼때 왜 프론트엔드, 기획자 등과 같은 역할과 함께 팀리더라는 역할이 보여져야하는지 함께 고민했던 것입니다. 사용자 입장에서 팀리더라는 역할이 보여지는 건 크게 의미가 없고, 안보여지는 것이 더욱 자연스럽다는 결론에 도달할 수 있었습니다.

또, UI/UX 디자이너분들과 협업하면서 느꼈던 중요한 점은, “어디까지 가능하고, 얼마나 비용이 더 드는지 잘 설명하는 것이 중요하다”입니다. 디자이너 분들이 고민하는 건 기획의 바탕위에서 자연스러운 동선과, 최선의 사용자경험을 디자인에 녹여내는 것 입니다. 그렇기 때문에 디자인요소도 정답은 없습니다. 얼마든지 상황에 따라 여러가지 디자인을 적용할 수 있습니다. 대화를 통해서 좀 더 적은 개발 비용을 들이면서, 최선의 디자인 요소를 만들어 낼 수 있습니다.

얼핏보면 별거 아닌 일 같지만, 혼자서 만들어 나가는게 아닌 결국 사람과 사람 사이에서 다함께 만들어 나가는 것이기에 기획, 디자인, 개발 모두 적절히 조율해나가는 능력이 중요하다고 느끼게 되었습니다. 이를 위해서 기획자를 이해하기 위해 프로젝트 기획단계부터 함께 참여해서 기획에 대한 이해를 고취시켰고, 여러 UI/UX 가이드를 참조하며 피어만의 UI/UX 디자인 가이드를 정립해보며 UI/UX디자인에 대한 이해를 함양하려 노력했습니다.

프론트 엔드 개발자로서 배운 점, 아쉬웠던 점, 고찰

프론트엔드를 선택하게된 이유는 명확했습니다. 사용자가 애용하는 서비스가 되려면 사용자 경험이 정말 중요하기 때문입니다. 요즘같은 세상에 아무리 기능이 좋더라도 버벅이거나, 사용자 경험을 고려하지 않은

플로우로 설계가 되어있으면 다시 돌아오는 사용자의 비율은 현저히 낮아지게 됩니다. 이 부분을 직접 해결하고 싶어서 프론트엔드를 선택하게 되었습니다.

개발하면서 신기했던건 프론트엔드 개발자마다 각기 다른 코드취향을 많이 볼 수 있었다는 것. 또 수 많은 코드리뷰를 거치며 조금 더 깔끔하고 읽히기 쉬운 코드작성에 신경쓰게 되었다는 점입니다. 아쉬운 점은 당장 구현해야 할 업무에만 치중되어서, 어떤것이 더욱 리액트스럽게, 타입스크립트스럽게, 넥스트스럽게 코드를 작성하는 건지 깊은 고민이 부족했다는 점입니다. 분명 성능적으로 개선점이 존재할 텐데 이 부분은 추후 꼭 개선해보고 싶습니다.

또, 선택한 기술스택이 피어 웹 서비스를 구현하는데 최적의 도구를 선택한 것이 아닌, 팀원들이 가장 많이 다루는 언어 및 프레임워크로 선택한것이 아쉽습니다. 훌륭한 엔지니어의 덕목은 언어와 프레임워크라는 도구에 종속되는 것이 아닌, 목적에 적합한 도구를 사용할 줄 아는 것이라고 생각합니다. 그런 점에서 더더욱 아쉬움이 남네요. 피어를 운영해나가면서 필히 새로운 이슈들이 발생할텐데 그때는 가장 적절한 도구를 선택해서 문제를 해결해보고 싶습니다.



@jeyoon 윤정연

발자국을 쫓던 사람에서 함께 찍는 사람이 되기까지

또 다시 프로젝트

안녕하세요. peer 웹 서비스의 프론트엔드 개발을 하고 있는 jeyoon입니다. 런칭을 한 달 정도 앞두고 팀원들과 함께 열심히 달리는 중인데요. 저번에 팀원들과 얘기하면서 나왔던 “할 일이 무한 스크롤처럼 계속 로딩되는 느낌이다.”라는 것을 체감하고 있습니다. 남은 일도 많은데 해결된 줄 알았던 기능들에도 계속 이슈가 생기는 중이라 “이거 진짜 끝낼 수 있을까...?”하는 생각이 들기도 하지만 열심히 이슈를 닫다 보면 언젠가는 이슈 개수 0을 볼 수 있는 날이 오겠죠? 팀에 합류하자마자 정신없이 개발하다 보니 얼마 전이지만 합류하던 시점의 마음을 잊게 된 것 같았는데 이렇게 글로 생각을 정리할 수 있는 기회가 생겨 기쁩니다!

peer 웹 서비스는 1기 개발진을 모집할 때부터 눈여겨보던 프로젝트였습니다. 오랜 기간 42 서울에서 공부하면서 가장 부족하다고 느꼈던 것이 “자연스럽게 다양한 동료들을 만날 수 있는 계기”였는데요. peer 동아리가 동료들이 모일 수 있는 계기를 만들어주는 좋은 매개체가 되어 준다고 생각했거든요. 그리고 종종 peer를 통해 결성된 스터디에 참여하면서 느꼈던 노선의 불편을 해결할 수 있는 자체 웹 서비스를 만든다는 소식에 더욱 관심을 두게 되었습니다. 아쉽게도 1기 개발진을 모집하던 시기에는 하고 있던 다른 프로젝트가 있어 지원하지 못하고 건너 건너 들려오는 소식만 듣고 있었는데요. 마침 참여하던 프로젝트가 끝난 시점에 운이 좋게도 개발진 충원 모집이 열렸습니다. 사실 이제 프로젝트 경험은 충분하다는 느낌이 들었기에 프로젝트 참여 보다는 취업 준비를 해야겠다는 생각을 굳게 가지고 있었는데, 지원 마감일이 다가오니 제가 진짜 필요하다고 느끼는 서비스에 기여해보고 싶다는 생각이 강하게 들더라고요. 그래서 많은 고민 끝에 지원하게 되었고 감사하게도 1.5기로 참여할 수 있게 되었습니다.

나의 역할에 대한 고민

친절한 팀원들이 중간중간 궁금한 점 없냐며 물어봐 주긴 했지만, 합류 시점이 1차 스텝 마무리와 2차 스텝 기능 기획으로 모두가 바쁜 시기였기에 열심히... 자체적으로 온보딩을 했던 기억이 있습니다. 정말 합류 직후 2주 동안은 새로운 사람들과 개발 프로세스에 익숙해지느라, 기능 파악하느라, 전 팀원의 코드를 파악하느라 정신이 하나도 없었네요. 고백하자면 정신이 없었던 데에는 팔로워 역할이 조금 낯설었기 때문도 있었던 것 같습니다. 지금까지의 프로젝트에서는 표면적으로는 아니든 다른 팀원들을 이끄는 역할을 많이 맡았는데요. 이번에는 결성되어 함께 작업한지 오래 된 팀에 합류한 것이기도 하고, 명확히 리더가 존재하는 팀의 팀원이 되었기에 최대한 기존의 분위기에 맞춰가려고 노력했습니다. 제목을 정하다가 좋은 비유가 생각났는데 마치 앞서서 팀원들이 찍어 둔 발자국에 제 발자국을 맞춰나가는 것처럼 정해진 규칙과 작업 방식들을 잘

지키고, 어딘가 의문이 생겨도 “이렇게 결정된 데에는 이유가 있겠지.” 하는 생각에 최대한 이해하고 수긍했고요. 코드 리뷰를 할 때에도 기존의 작업 분위기라던가, 각자의 취향을 잘 모르니 코멘트를 달 때 되게 조심스러웠던 기억이 나네요.

그러던 중에도 “이렇게 정해진 것에 맞추는 것이 올바른 것인가?”라는 생각과 함께 제 역할이 어떤 마음가짐을 가져야 하는지가 잘 정리되지 않아서 은근한 스트레스를 받고 있었는데요. 문득 (저는 선발 뒷이야기를 전-혀 모르므로 100% 제 생각이긴 하지만) “지원자 중에서 나를 뽑은 데에는 내가 앞서서 겪은 프로젝트 경험들에 대한 기대가 있기 때문이 아니었을까?”라는 생각이 들었습니다. 그리고 그제야 팔로워의 역할에 대해 잘못 생각하고 있었다는 것을 깨달았습니다. 잘 따르는 것도 중요하지만, 이해되지 않는 것에 대해서는 의견을 내서 더 나은 쪽으로 나아 갈 수 있게 돋는 것도 중요한 팔로워의 역할이더라고요. 그래서 의견을 낼 때 너무 주저하지 않기로 했습니다. 그리고 저를 포함한 팀원들의 의견들이 모인 결과 말로만 진행하던 스크럼 기록을 자세히 남기기 시작했고 코드 관리 방식이나 단위 결정 방식에 대해서도 팀원들과 논의해 볼 수 있었습니다. 그리고 몇 번의 인수인계 경험으로 클린 코드의 중요성을 체감한바... 코드 리뷰를 할 때 가독성이나 리팩토링에 대해서 적극적으로 의견을 내고 고민도 나눠 보았습니다. 이제 와서 생각해 보니 제가 리더였을 때에도 의견을 잘 내주는 팀원들이 있었기에 프로젝트를 잘 진행할 수 있었던 것 같더라고요. 프로젝트는 모든 팀원들이 힘을 합쳐 한 걸음씩 완성을 향해 나아가는 것인데 이 뻔한 사실을 종종 잊게 되네요. 이번 프로젝트를 하면서 팀에 도움이 되는 팀원의 역할을 조금이나마 깨닫게 된 것 같습니다. 프론트엔드 팀장님에게도 제가 힘이 되는 팀원이라고 느껴졌다면 좋겠네요!!

멋진 마무리를 위해서!

다른 분들의 프로젝트 후기를 듣다 보면 종종 팀원들 사이의 갈등 때문에 힘들었다는 이야기를 듣게 되는데요. 저는 감사하게도 지금까지의 프로젝트에서도, 이번 프로젝트에서도 너무 좋은 팀원들을 만나 행복하게 팀 프로젝트를 할 수 있었던 것 같습니다. 저희도 물론 크고 작은 갈등들이 있었겠지만, 최대한 각자의 생각을 이해하고, 존중하려고 노력하면서 이야기했기 때문에 큰 문제 없이 넘어갈 수 있었던 게 아니었나 싶어요. 앞으로 남은 작업도 서로서로 존중하고 도와가며 잘 마무리할 수 있었으면 좋겠습니다. 그리고 비록 저는 2달 정도의 시간만 들였지만, 우리 팀원들이 애정을 담아 거의 반년 정도의 오랜 시간을 들인 peer 웹 서비스가 42 서울 카뎃들에게, 더 나아가서는 많은 사람에게 사랑받는 서비스가 되었으면 좋겠다는 바람입니다



@ hyunjung 정현섭

도전부터 배움까지, 그리고 함께 성장하기

강남 한 가운데 클러스터의 작은 공간에서 처음 모였던 순간이 아직도 생생하다. 대부분 생면부지인 카뎃들과 어색한 인사를 주고받으며 peer 웹사이트 개발에 대한 설명회에 참석하였다. 좁은 다용도실에 새로운 peer 웹사이트 개발에 관하여 관심 있는 수십 명의 사람들이 몰려들었다. 내 생각보다 더 많은 사람들이 동료학습이라는 가치에 대하여 많은 관심과 열정을 가지고 있다는 것에 조금 놀랐다. 그렇게 열정 하나만으로 시작된 peer의 웹사이트 개발에 대하여 잘 준비된 ppt와 타고난 언변 능력을 갖춘 리더들의 설명이 이어졌다.

이후 개발 리딩을 맡게 된 담당자들과 인터뷰를 진행하여 프론트엔드 팀에 합류하게 되었고, 42서울에서 과제를 진행할 때 자주 사용해 팀원을 모집하여 익숙한 플랫폼의 자체 웹사이트를 개발하는 큰 개발 프로젝트에 처음으로 일조하게 되었다. 웹 프론트엔드 개발 경험이라고는 2년 전 작은 토이 프로젝트를 만들 때 활용해 본 것이 다인데 짧은 시간 동안 프론트엔드 진영에서는 또 많은 기술들이 발전해 나갔고, 2년 전의 얇은 지식으로는 큰 규모의 개발을 하기에는 지장이 커서 팀원들과 함께 새롭게 React 및 Next.js에 대하여 학습했다.

단기간 내에 학습한 내용을 실제로 개발을 하며 사용하려니 만만찮은 러닝커브가 존재하였다. 처음 Next.js를 학습할 때 12버전을 기반으로 하는 강의가 대부분이라 12버전을 학습하였지만, 실제로 개발에서 사용할 버전은 메이저 버전이 업데이트된 13버전을 사용하게 되었다. 꽤 큰 변화가 이루어진 13버전을 사용하며 이전 버전과는 다른 디렉터리 구조 등에 적응한다고 꽤나 애를 먹었다.

또한 최신 업데이트인 탓에 chatGPT의 도움도 무용지물이었다. 믿을 거라곤 구글링과 내 옆의 동료에게 물어 문제를 해결하는 수밖에 없었다. 또한 깃허브를 활용한 협업도 처음에는 꽤나 적응하기 어려웠다. 대부분 깃을 소스 저장 용도로만 많이 사용하고, 제대로 된 협업을 해보지 않은 상황에서 Fork를 이용한 깃허브 전략도 처음에는 PR 한 개 생성하기까지 꽤나 오랜 시간을 허비하였지만 언제나 그렇듯 내 옆 동료와 함께 머리를 맞대어 문제를 해결할 수 있었다.

지난 2년간 42서울에서 동료학습이라는 학습 방식에 익숙해져 이전보다 문제에 대한 해결 방안을 더 빠르고 효율적으로 찾아내어 적용하였다. 이 방법이 처음엔 조금 낯설고 때론 나의 무지함에 좌절을 수없이 겪기도 하였지만, peer를 개발하는 동안 지난 시간에 비하여 조금이라도 발전한 나의 모습을 마주치며 2년 간의 시간이 무의미하게 흘러간 시간이 아니라고 느낄 수 있었다. 또한 개발 기간 동안 수많은 물음에도 항상 친절함과 함께 해결 방법을 찾아준 동료들한테 이렇게라도 감사의 인사를 전한다.



@ hyna 나 현

아는 것만을 활용한 차력 쇼에서 새로운 것을 받아들이기까지

안녕하세요, 프론트엔드 팔로업 팀 리드를 맡은 hyna입니다. 꽤 오랜만에 붙이는 수식어 같네요. 어떤 이야기를 쓸지 고민을 많이 했는데, 결국 통틀어서 제가 어떻게 발전했는지에 대해 이야기를 해볼까 합니다.

그림 그리는 사람들 사이에서 농담조로 하는 말이 있습니다.

'만화 그리기에 최적화된 클립 스튜디오를 두고, 포토샵으로 차력 쇼한다.'

'low level의 도구, 혹은 용도에 맞지 않는 도구를 가지고 작업물을 만들어냈다'라는 의미로도 쓰이기는 합니다. 그러나 '조금만 더 배우면 더 쉽고 빠르게, 작업물을 만들어낼 수 있었지만, 새로운 것을 배우지 않고 기존의 도구를 놓지 않는다'는 그런 자조적인 농담에 가깝기도 하지요.

저는 그림을 그리지는 않지만 그림을 업으로 삼은 친구들에게 그런 부류의 농담을 자주 들었고, 제 개발 방식이 이와 비슷하다고 생각했습니다. 왜냐하면 새로운 것을 배우지 않고 제가 알고 있는 것만으로 항상 결과물을 만들어왔기 때문입니다. 새로운 것을 만나더라도 제대로 알아보지 않고 일단 써보고, 잘 되는 것 같으면 넘어가고는 했죠.

제가 파이널 미션 때 새로운 라이브러리를 접할 때도 마찬가지였습니다. 이쁘다니까 MUI라는 걸 활용해서 컴포넌트를 가져다 쓰고, 쉽다니까 'react-hook-form'을 가져다 쓰고요. 그러나 공부라는 과정이 없으니 기본적인 기능을 가져다 쓸 줄만 알고, 복잡한 기능을 만들어내지 못했습니다. 이후로도 개발을 진행할 때 공식 문서를 제대로 읽어보지 않고 진행했던 것 같습니다.

처음 그 한계에 부딪혔던 때는 프로필의 링크 에러처리를 할 때였습니다. 잠시 부연 설명을 하고 넘어가자면, 'react-hook-form'이라는 라이브러리에서는 입력값에 대하여 규칙 입력 시 자동으로 유효성 검사를 해주는 기능이 있는데요. 기존의 닉네임 입력 칸 같은 경우 항상 필수이기에 단순하게 'required'라는 규칙을 주면 해결이 되었지만, 링크란을 편집할 때는 이게 어려웠습니다. 링크 등록 시에는 '링크 제목'과 '링크 주소'가 한 몸처럼 움직이며, 둘 다 비어 있을 때는 문제가 없었지만, 둘 중 하나라도 채워져 있으면 다른 입력 칸이 필수가 되어야 했거든요. 정말 어떻게 해야 하나 하고 고민했습니다. 우선은 제출 전 유효성 검사를 하는 코드를 작성하여 'toast'로 에러 메시지를 띄웠으나 만족스럽지 않았습니다. 다른 입력 칸은 입력 칸에 문제가 있으면 가시적으로 표현해 주었는데, 링크에 대해서는 그러지 못하여 매우 아쉬웠습니다. 제가 생각하는 UI/UX의 가장 중요한 부분은 통일감 있는 사용자 경험과 디자인인데, 그렇지 못하였거든요.

우리의 영원한 친구 ChatGPT에 물어봐도, 구글링을 해봐도 해답을 찾을 수 없었습니다. 그런데 해답은 (매우 당연하게도) 공식 문서에 있었습니다. 공식 문서를 뜯어보니 'setError'라는 게 보이더라고요. 이거였습니다. 물론 유효성 검사는 여전히 제가 해야 했지만, '링크 제목'이나 '링크 주소'가 없는 경우 'setError'를 사용하여 '이 입력 칸은 필수 칸인데, 해당 입력 칸이 비어있어 에러가 발생했다'는 문제를 통일감 있게 알릴 수 있었습니다.

이 사건을 통하여 저는 공식 문서가 왜 중요한지 깨달았고, 새로운 라이브러리를 만날 때마다 일단 공식 문서를 쭉 한 번 가볍게 읽어보는 시간을 가졌습니다. 공식 문서를 읽어봐도 잘 모르겠다면 해당 라이브러리에 대해 잘 설명하고 있는 블로그 글이라도 읽고 공식 문서를 다시 보았던 것 같아요.

그렇게 작업 방식을 바꾼 후, 전에 작업해 본 적 없는 새로운 기능을 만났을 때는 어려움을 느낄지언정 막막함을 느끼지 못했습니다. 특히 제가 공을 들여 작업했던 히치하이킹은 관련 모집 글을 틴더처럼 카드 형식으로 가볍게 보고 넘기는 재미 위주의 페이지인데요. 물론 제가 화려한 애니메이션이 있는 기능에 대해 작업하는 것을 좋아하기는 합니다만, 개발 과정에 약간의 어려움이 있었습니다. 원래 카드를 넘기고 되돌리는 기능은 'react-tinder-card'라는, 아주 쉽게 작업할 수 있는 라이브러리를 활용하기로 했는데요. 약간의 문제가 있어 해당 라이브러리를 사용하지 않고 'react-framer-motion'이라는 애니메이션 라이브러리를 활용하기로 하였습니다.

다행히 공식 문서도 친절하고, 처음부터 끝까지 잘 설명되어 있는 블로그 글이 있어 기능을 구현하는 데에는 어려움이 없었습니다. 그 대신 여기서 어려움을 느꼈던 부분은 CS 적인 부분이었습니다. 모바일에서는 드래그와 클릭이 구분되는데, PC에서는 구분이 안 되더라고요. 그래서 MDN등의 사이트에서 onClick 이벤트가 언제 발생하는지 찾아보고 다시 작업했던 것 같아요. 해당 작업을 하면서 자바스크립트 이벤트에 대한 공식 문서가 정말 간절했던 것 같습니다.

앞으로는 자바스크립트에 대해서 쭉 공부할 것 같습니다. 정확히 기억하지는 못하더라도 나중에 다시 어렵뜻이 떠올렸을 때 키워드를 기억할 수 있게 하는 게 목표입니다.

사실 저는 부족한 리더였다고 생각합니다. 특히나 능력도 있는데 하루 종일 노력하는 하류 님이나, 꿈을 꿀

줄 아는 주전님이나, 본질을 볼 줄 아는 호슬림님, 그리고 소통 능력이 뛰어난 제이위님을 보면 나도 저렇게 해야 하는데 싶으면서도 그렇게 하지 못했던 것 같아요. 그런데도 제가 무언가 부탁하거나 하자고 이야기했을 때 잘 따라와 주고, 무언가 어려움이 있을 때 저에게 찾아와 이야기 해준 팀원들에게 정말 감사하다고 이야기하고 싶습니다. 그리고 하류님과 주전님, 호슬림님, 제이위님도 감사합니다. 여러분을 보며 저도 조금씩 성장한 것 같아요. 정말 감사합니다.



@hyeokim2 김현지

도전부터 배움까지, 그리고 함께 성장하기

peer에서 나쁜 습관을 고치다

저는 peer를 시작하기 전에, 프론트엔드 경험이 있었습니다. 작년부터 프론트엔드 개발자가 되고 싶다는 생각을 가지고, 조금씩 프론트엔드 관련 공부를 하다가, 작년 9월부터 프론트엔드 실무를 도울 일이 있었습니다. 거의 1년이 넘어가는 기간동안 했던 경험은 분명 도움이 되었지만, 시간에 쫓기면서 코드를 작성하다 보니, 상급자의 코드 수정 요구에 의문을 제기하지 않고 그냥 코드를 작성하는 게 대부분이었습니다. 그래도 작업 중인 웹페이지는 괜찮아 보였기 때문에 문제가 없다고 생각했습니다. 그러나 저의 부족한 지식으로 인해 문제가 발생할 수 있다는 불안감은 한켠에 계속 있었습니다. 그 생각은 42 과정을 진행하면서 더 커져갔고요. 42과제는 평가를 받을때, 평가자가 저에게 질문을 던집니다. 그럼 저는 질문에 대한 답을 하고, 서로의 지식을 주고받게 되죠. 그 일련의 과정을 겪다보면, 제가 얼마나 부족했는지 깨닫고, 어떤 부분을 채워야 할지 공부하게 됩니다. 하지만 제 프론트 작업에는 질문을 해주는 사람이 없었습니다.

그래서 42peer 개발자 모집을 보았을 때, 좋은 기회라고 생각했습니다. 좋은 서비스를 만드는 게 가장 우선순위에 있었지만, 동시에 나와 프론트엔드 기술에 대해 고민하고 공유할 수 있는 사람들과 함께하고 싶었습니다. 그리고 부족한 지식을 채워나가고자 42peer에 지원하게 되었고, 약 6개월 동안 함께하게 되었습니다.

6개월이 지난 지금, 목표를 모두 이루었다고 말하기는 어렵습니다. 그러나 제가 어느 정도 부족한지를 깨달았습니다.

남이 보면 나쁜 코드

저는 배열에 map을 사용할 때마다 파라미터의 이름을 'v'로 설정하는 버릇이 있었습니다. map 메소드의 첫 번째 파라미터의 이름을 항상 'v'로 정하는데, value라는 의미로 작성했었죠. 어떤 배열을 돌리든, 모두 v로 작성했었습니다. 그리고 어느날 코드리뷰에서 이렇게 작성하면, 정확한 의미를 파악하기 어렵다는 코멘트를 받았습니다. 놀랍게도 저는 그 부분을 인지한 적이 없었지요. 그리고 이 습관은, 아마도 예전 프로젝트에서 다른 사람이 파라미터에 'v'를 사용하는 것을 따라하면서 굳혀졌던 것 같습니다. 지금 생각해보면, 그 전까지는 제 코드를 남이 볼 일이 별로 없었으니 생각없이 작성했던 건데, 제 코드가 남에게는 읽기 어려운 코드가 될 수 있다는 걸 깨달았습니다. 그 이후로는 코드리뷰를 받는 사람이 이해하기 쉽도록 코드를 작성하려고 노력하는 중입니다. 다른 사람들에게는 여전히 복잡하게 느껴질 수 있지만요.

고민하는게 괴로워요

SSR 때문에 Next.js에서 애를 먹은 적도 있었습니다. 메인 페이지는 서버에서 데이터를 받아와서 SSR로 렌더링하도록 만들려고 했는데, 로그인 여부에 따라 화면의 일부가 다르게 렌더링되어야 했습니다. 로그인 여부를 판단하는 토큰은 localStorage에 저장되어 있어서 서버 사이드에서 가져올 수 없었습니다. 백엔드에서 httpOnly 쿠키를 설정해준다는 말을 듣고 Next.js에서도 가져올 수 있다고 생각했습니다. 하지만 가져오지 못했습니다. 놀랍게도, 저는 이전까지 Next.js와 API 서버가 서로 공유되어 있다고 생각했습니다. 이 사실을 깨달은 것은 나중이었습니다. 그래서 거의 일주일 동안 삽질을 했던 것 같습니다. 마지막에 팀원들과 함께 고민한 결과, 결국 Next.js에서 접근 가능한 쿠키를 따로 만들어야 한다는 결론에 이르렀고, 문제를 해결할 수 있었습니다. 사실 문제해결을 하는 과정 내내 힘들었습니다. 그리 어렵지 않은 문제인 것 같은데, 인터넷에 있는 해결방안은 제가 적용했을 때는 안되고, 도대체 뭐가 문제인지 알 수가 없어서 답답했습니다. 하지만 그 과정이 있어서 많은 공부를 할 수 있었습니다. 예전 같았으면 상급자에게 해결 방안에 대해 질문하고, 아는 게 없으니 그대로 답변을 수용했을 것입니다. 하지만 지금은 그런 사람이 없으니 끝없이 찾아야했고, 아주 명쾌한 해결방안을 찾은 건 아니었지만 그래도 스스로 해결하는 경험을 할 수 있어서 좋았습니다.

알고 있는데 몰라요

그런가 하면, 종종 누군가가 저에게 질문을 던졌을 때 답변하지 못하는 경우가 있었습니다. 예를 들어, axios와 fetch의 차이에 대해 질문을 받았던 적이 있었는데, '그리게... 차이가 뭐지...?' 라는 생각만 하고 있었습니다. 이 사실에 꽤 충격을 받았습니다. 프론트엔드 개발자로서는 axios 또는 fetch를 사용하는 것이 필수인데, 그 둘의 차이점도 제대로 알지 못하고 사용하고 있다는 사실이 갑자기 부끄러워졌습니다. 이와 같은 일은 peer를 개발하면서 종종 발생했습니다. 누군가 질문을 하면 답변은 했지만 여전히 의문이 남는 경우가 많았습니다. 위에서 언급한 에피소드에서도 비슷한 상황이 있었습니다. 백엔드에서 설정한 쿠키를 Next.js에서 가져올 수 있는지 물어보고 싶었는데, 쿠키, 세션, 토큰에 대해 제대로 알지 못한 상태에서 말을 더 이상 이어나가지 못하는 저를 발견했습니다. 지식의 공백을 많이 느꼈던 것 같습니다. 그리고 이런 과정에서 자존심이 상하기도 했습니다. 그래도 나름의 프론트엔드 경험이 있었기 때문에 아는 게 많다고 생각했는데, 실제로는 아는 게 없다는 걸 깨닫는 순간이 많았기 때문에 더 그랬습니다.

그래서 저는

어느 순간부터는 모르는 것을 계속해서 메모하기 시작했습니다. 모르는 것을 하나씩 쓰다보니 그새 50개가 넘어가고 있는 중입니다. 그 목록에서 야금야금, 하나씩 공부하는 중이긴 한데, 언제 다 공부할지는 모르

겠습니다. 그래도 열심히 공부해보려고 합니다. 그 어느때보다 공부를 더 열심히 하고 싶은 때입니다.

그 열정에 불을 지핀 건 당연 peer입니다. 저 혼자서 코드를 짜고, 확인받고, 생각할 겨를도 없이 피드백에 맞춰 코드를 수정하던 때에는, 의문을 딱히 가지려하지 않았습니다. 그러므로 공부할 일도 적었죠. peer에서 처음으로, 사람들과 함께 코드를 공유하고 코드에 대해 논의하게 되면서, 어쩌면 제가 드러내기 싫었던 제 부족함을 드러낼 수 있었던 것 같아요. 그 실체는 꽤 충격적이었고요. 그럼에도 불구하고, 그 과정이 즐거울 수 있었던 건, 역시 좋은 팀원들 덕분입니다. 헛소리도 잘 받아주고, 말을 개떡같이 해도 찰떡으로 알아들어서 감사했습니다. 코어타임 내내 즐거웠어요. 정말 peer에서 겪었던 일들 모두, 좋은 기억, 경험으로 남을 것 같습니다. 모두 고생하셨어요.



@hujeong 정희호

개발 후기

peer 프론트에서 회원가입 파트를 맡은 hujeong입니다.

처음 피어 프로젝트를 참여하게 된 가장 큰 이유는 제가 사용해 보고 싶었던 서비스였기 때문이었습니다. peer 노션을 통해 스터디를 참여해 본 적이 있었는데, 노션으로 되어 있어 사용할 때 조금은 불편하게 느껴졌습니다. 노션에 익숙하지 않은 탓도 있었지만, 서비스로 개발되면 피어를 더 잘 사용할 수 있겠다라는 생각이 들었습니다. 그래서 개발에 참여하면 뿌듯하고 배울 것도 많을 것 같아서 지원을 했고, 감사하게도 개발에 참여할 수 있었습니다.

프로젝트를 시작하기 react 학습과 nextjs를 참고해 팀원들(hyna, woorikim)과 같이 공부를 했고 7월에 학습한 내용을 바탕으로 간단한 게시판을 만들어보는 온보딩 과제를 진행했습니다. 그동안 c언어를 학습하다가 react와 nextJs 같은 프레임워크를 다루기 시작하니까 쉽지 않았습니다. c언어는 단순하고 저수준이라 동작이 좀 더 이해가 잘 되었던 것 같은데, 프레임워크나 라이브러리는 추상성이 높아서 동작이 이해하기 어려웠습니다. 그리고 처음 레이아웃은 어떻게 작성해야 하는지 nextjs 디렉토리 구성은 어떻게 해야하는지 모르는 것도 많고 라우팅 에러도 계속 발생하고, 쉽지 않았습니다. 강의에서 코드를 따라 치기는 했지만 지식을 소화하기에는 짧았던 기간이라 어려웠던 것 같아요. 그래도 당시 멘토 역할을 해주신 hyunjung님의 도움으로 어찌저찌 과제를 제출하긴 했습니다.

본격적인 개발에 들어가기 전에 리액트나 nextJs 공식문서를 하면서 부족한 점을 채우려고 했습니다. 특히 nextJs가 리뉴얼 되면서 신뢰할만한 자료가 많이 없어서 공식문서를 많이 보고 이해해보려고 했습니다. (잘 이해는 못했던 것 같습니다.) 본격적으로 프로젝트를 진행할 때는 회원가입, 회원탈퇴 등 파트를 맡아서 react-hook-form 라이브러리를 사용해서 사용자의 입력값이 유효한 입력값인지 검증을 했고, axios와 swr을 사용해서 백엔드와 통신을 해보았습니다. 그리고 프로젝트를 하면서 가장 좋았던 점, 깃허브를 통한 협업을 했습니다. 그전에는 깃허브를 단순히 저장소로 사용했다면, 이번에는 팀원들과 이슈도 파고드 PR을 통해 리뷰를 주고받는 협업의 용도로 사용해 볼 수 있어 좋았습니다.

여기까지가 제가 참여한 개발 과정이고 11월 이후부터는 개발에 참여하지 않았습니다. 그래서 제가 peer에서 맡은 부분은 회원가입과 회원탈퇴, 간단한 개인정보 페이지 까지만 구현을 하고 나왔습니다.

팀을 나오게 된 가장 큰 이유는 잘해야 한다는 스트레스가 컸습니다. 처음이니까 잘 하지 못한 것은 당연한 일이지만 저는 이 부분에 대해 스트레스가 컸습니다. 제가 팀에 기여를 한다고 느껴야 뿐만 아니라 계속 참여하는 원동력이 되었을 것 같은데, 제가 코드를 짜는 게 팀원들에게 도움이 되는 것 같지도 않았습니다. 팀원 분들이 좋은 분들이라 격려도 많이 해주시고 했지만, 제가 짜는 코드에 대해 제 스스로 확신이 없으니까 매일 코드를 짜는 게 큰 스트레스로 다가왔습니다.

팀원들과 이 부분에 대해 미리 이야기하고 해소 했으면 좋았겠지만 적극적으로 소통을 하지 못했습니다. 온보딩 미션을 할 때부터 다른 팀원에 비해 학습하는 속도도 느린 것 같고, 코드를 짜는 것도 익숙하지 않아서 계속 주눅이 들었습니다. 나만 아무것도 모르는 것 같아서 소통이 부끄럽고 망설여졌는데, 리더분들이나 팀원분들과 적극적으로 소통하면서 개발을 진행했으면 좀 더 나은 결과가 있지 않았을까 싶어서 아쉬운 점이 많습니다.

또 당장 필요하지 않은 내용까지 공부까지 하려고 했던 점도 방해 요소였던 것 같습니다. 프로젝트를 본격적으로 돌입하기 전에는 next.js에 대해서 공부를 했는데, next.js에 대해 깊이 있는 학습도 물론 중요하겠지만, 제가 맡은 회원가입 파트에서는 react-hook-form이나 axios, swr 같은 데이터패칭 라이브러리에 집중해서 공부를 했어야 하는데, 공부를 다 하려다 보니 이 부분에는 조금 소홀했던 것 같습니다. 처음 해보는 프론트였고, 공부할 양이 많아서 무엇을 공부할 지 갈피를 못 잡았는데 이것도 저것도 집중을 못했던 것 같습니다. 현재 개발하고 있는 부분을 집중해서 공부를 했으면 더 좋지 않았을까라는 아쉬움이 남습니다.

코드도 아쉬운 점이 있는데 컴포넌트간에 의존성이 높았습니다. 회원가입에서 이메일이 인증되기 전에 다음 필드를 입력하지 못하게 막게 설계를 했었는데, 이 부분에서 입력 필드 간의 의존성이 높아지고 코드가 복잡해지는 결과를 낳았던 것 같습니다. 코드가 복잡하니까 개발할 때 동안 회원가입 부분에 매달렸던 기간이 늘어났습니다. 백엔드와 설계를 할 때 인증과정을 좀 더 단순화 했다면 더 좋지 않았을까하는 아쉬움도 있습니다. 처음 해본 대형 프로젝트라 미숙한 점이 많았습니다. 저는 끝까지 마무리하지 못하고 나왔지만, 하고 계시는 분들, 앞으로 하실 분들이 좋은 결과 만들어가셨으면 좋겠습니다.



@hoslim 임호성

익숙하지만 새로운 경험

저는 피어 웹사이트 개발팀에서 프론트엔드 리더이자 벨루가팀장 hoslim, 임호성입니다. 이번 개발백서를 쓰면서 피어 프로젝트에 참여한 소감을 한줄로 요약하면 어떨까란 생각을 했고 나름 나온 한줄이 위의 제목 이네요... 피어 프로젝트를 해온 과정을 회고하면 개발이라는 익숙한 과정 속에 수많은 새로운 경험이 있어서 더욱 더 성장할 수 있었던 계기가 되었지만, 동시에 정말 많은 고통과 인내의 시간이었습니다. 그 이야기를 하나하나씩 적어보려고 합니다.

처음 피어 프로젝트에 합류를 하고 처음 보는 사람들과 인사를 하면서 많은 걱정을 했습니다. 저는 극 I로써 (팀원 분들은 안 믿겠지만) 낯을 상당히 많이 가리는 성격이라 “팀에 잘 어울릴 수 있을까”라는 걱정을 많이 하게 되었습니다. 더불어 처음 맡게 되는 리더인 만큼 팀을 잘 이끌고 “모두의 성장을 도모하면서 프로젝트를 무사히 마칠 수 있을까?” 하는 걱정까지, 정말로 많은 걱정이 일을 진행하는 순간순간 저를 괴롭혔습니다. 그럴 때마다 제가 할 수 있었던 건 “그냥 해보자”라는 생각과 공부, 그리고 주변의 응원이였습니다. 팀원들이 저에게 주는 응원과 격려는 “이게 맞을까”라는 의문을 “이게 맞다”는 확답을 바꾸게 했습니다.

피어의 개발 과정에서 리더로서 팀의 성장을 이끌 수 있는 방법을 끊임없이 고민했고 2가지 만큼은 확실히 해야 겠다고 생각했습니다. 첫 번째로는 “팀의 분위기를 좋게 유지하자” 였습니다. 학습 단계에서는 경험자와 비경험자가 나누면서 암묵적인 상하관계가 생겼다고 생각하고 이는 정보의 불균형이 불러온 어쩔 수 없는 현상입니다. 하지만 개발 과정에서도 상하관계가 유지되고 자신의 의견을 자유롭게 말할 수 없는 구조가 된다면 팀이 아니라고 생각했습니다. 이런 분위기를 조성하기 위해 아침 스크럼을 통해 오늘 자신이 어떤 것을 할 예정이고 같이 의논할 점에 대해서 이야기하는 시간을 만들고, 코드리뷰를 팀의 핵심 문화로 삼고 최소 두 명의 리뷰어가 승인을 해야 다음 개발을 진행할 수 있게 했습니다. 특히 코드 리뷰의 경우 개발이 밀릴 수 있지만 그래도 서로가 서로에게 검증하는 절차가 빠르게 성장을 도모할 수 있는 방법이라 생각했습니다. 두 번째는 학습 방법에 대해서 많은 고민을 했습니다. 피어 개발 과정에서 이뤄지는 학습으로 심화적인 기술을 배우는 것과 실제 기능으로 탑재될 기술 스택을 검증하는 두 가지가 있습니다. 심화적인 부분은 이미 익숙한 42서울의 학습 방법을 차용했습니다. 42 서울에서의 학습 방법은 과제를 풀면서 서로의 생각을 코드와 토의를 검증하는 방식으로 진행됩니다. 이를 다시 구현하기 위해 과제의 서브젝트를 순수 만들었습니다. 템플릿을 만들고 과제의 챕터를 나누면서 핵심 키워드를 쭉 정리해서 이를 구현하는 방식으로 구성했고, 이후 하나의 팀을 만들어서 서로가 서로에게 검증할 수 있는 시간을 가졌습니다. 기술 검증의 경우에는 전담팀을 만들어서 진행하고 팀에서 내린 결론을 문서화하고 발표하는 자리도 만들어서 다른 팀원에게도 학습할 수 있게

했습니다. 그렇게 인력 손실을 줄이고 기술 검증의 속도와 퀄리티를 낼 수 있게 하였습니다. 물론 기존에 생각했던 만큼 안된 부분도 있었고 예상 외의 일들이 일어나도 했지만 오히려 팀원들에게 피드백 받고 이를 수용하면서 개발을 끝까지 마칠 수 있다고 생각합니다.

피어 개발을 하면서 정말 좋았던 것은 기획부터 개발까지 모든 과정을 밟을 수 있는 새로운 경험을 할 수 있다는 점이었습니다. 서비스의 재정의, 벤치마킹, 사용자 페르소나 같은 개발 전 과정부터 개발 이후 사이트 운영이나 유지보수까지 하나의 프로젝트에서 이 모든 사이클을 경험할 수 있었다는 것이 너무 신기했습니다. 할 일도 많고 개발과 기획으로 지칠때로 지쳐도, 개발자로서 자신의 의견을 적극 반영할 수 이 기회를 놓치고 싶지 않았습니다. 그리고 18인이라는 대규모 팀에 일을 할 수 있는 경험도 정말 뜻깊었습니다. 개발진 15명과 디자이너 3명으로 구성된 사이드 프로젝트 팀이 장기간 일을 하는 과정에서만 느낄 수 있는 경험과 겪을 수 있는 일을 제대로 느낄 수 있었습니다. 모두가 피어 웹사이트 구현이라는 하나의 목표를 갖고 이야기를 하고 코드를 만들 수 있다는 게 정말 재밌는 과정이었습니다.

반대로 아쉽고 안타까운 것도 있었습니다. 기술적으로는 TDD(Test-Driven Development)를 계획했지만 그 당시 판단으로는 일정적인 부분이나 러닝 커브적으로도 무리라고 판단되어 과감하게 포기한 점입니다. 물론 lint를 이용한 CI/CD로 대체하였어도 직접 테스트 코드를 쓰고 이를 고쳐나가는 과정을 느낄 수 없다는 것이 아쉬웠습니다. 학습 과정에서 너무 이론적인 부분을 다룰려고 한 거 같아 너무 아쉽습니다. 좀 더 개발 할 때 사용할 기술 위주나 패키지 위주로 진행한다면 목적이 명확해서 덜 혼란스럽지 않았을까 싶습니다.

마지막으로는 우리 피어 개발팀원들에게 이야기하자면 부족한 저를 리더로 세워준 리더진과 운영진께 감사하고, 저를 믿고 따른 프론트 엔드 팀원분들께 정말 감사합니다. 2023년 5월, 우연히 본 피어 프로젝트에 지원해서 우연히 프론트엔드 리더가 되고 우연히 좋은 팀원들과 개발을 할 수 있었다는 게 저에겐 매우 큰 행운이었습니다. 특히 리더라는 자리를 처음 맡아서 많은 일을 벌이고 저의 리드를 좋게 봐주신 프론트엔드 팀원들과 리더분들, 그리고 개발을 총괄해주신 류한솔님께 정말로 감사합니다.

제 2장

BACK

BACK-END
개발자들의 이야기

@haryu 류한솔	114
@hyeongki 김형찬	117
@juhyelee 이주현	122
@junssong 송준상	125
@jwee 위치혜	126

@haryu 류한솔

가재에게 위로받으며 성장하는 나

뇌 과학자 장동선 교수님의 쇼츠를, 정말 개발 막바지에 봤습니다. 교수님은 가재가 참 신기하다고 하셨습니다. 왜냐면 척추동물은 바깥이 부드럽고, 안에 단단한 뼈가 있으니 성장의 형태가 대단히 자연스럽습니다. 하지만 그에 비해 갑각류, 가재와 같은 것들은 외골격으로 내부가 오히려 부드럽고, 외부는 단단하기에 자연스럽게 커질 수가 없습니다. 그런데 그렇다면 도대체 어떻게 성장을 할 수 있는가? 라고 궁금해하셨습니다. 그에 대한 답은 다들 알다시피 탈피를 시도하고, 걸 껌질을 벗어던집니다. 그리고 그 순간 갑각류들은 매우 취약한 상태가 되며, 힘도 소진하여 그저 하염없이 벗어던진 외피보다 더 단단한 외골격이 마르고, 단단해지길 기다립니다. 기진맥진하게, 죽음의 위기를 느끼는 그 순간이 성장의 순간인 것이라고 교수님은 이야기 하셨습니다. 아무리 강하고, 아무리 덩치가 커져 있던 존재 조차 그러한 순간을 거쳐서 다시 한 번 강해져 갑니다.

내가 강해질 수 있는 순간은 언제였나 생각해보았습니다. 그리고 과정 과정마다 괴롭고 힘들던 것이 떠올랐으며, 기획을 다듬는 순간들, 부족한 실력에 밤을 새던 날들, 런칭 직전 등등.. 정말 많이 괴로웠습니다. 해내기 위해 노력했고, 고민했고, 참았으며 이 개발 과정에서 배울 수 있었던 것들, 무력하게 느꼈던 많은 감정들이 스쳐 지나갔습니다. 특히나 가장 쉽지 않았던 것은, 내 안의 외로움이었으며, 무언가를 달성하기까지 공부하고 고민했던 그 순간 순간들이었습니다. 올곳이 혼자서 해결해야만하는 그 순간들. 사실 그때는 정말 힘들다는 생각이 앞섰지 성장한다고 생각하진 않았던 것 같습니다.

교수님은 말씀하셨습니다. 갑각류들, 외골격을 가진 것들의 강해지는 과정을 바라보면 마치 사람의 마음의 성장과 비슷해 보인다고 말씀하십니다. 아키텍쳐를 설계하고, 서버 비즈니스 로직을 만들고, 기획을 하며, 팀 내의 사람들간의 갈등 사이에서 조율을 해보고, 내 스스로의 외로움과 힘듦과 싸워 지금 가장 많이 지쳐 있는 이 순간. 방금까지만 해도 우울했지만, 어느새 마음의 위로와 다시한 번 다음의 '성장'을 경험하고 있습니다. 함께 해주신 모든 분들에게 감사하며, 동시에 제 자신의 성장을 느끼며, 다음을 바라봅니다. 비록 부족할 수 있고, 아쉬울 수는 있지만 이번 피어의 개발은 저에게 다시 한 번 성장이란 이런 거라는 것을 상기시켜 주는 것처럼 보였습니다. 이제 마무리까지 얼마 안 남았습니다. 이 기간을 버티고, 웅크리지만 확실하게 성장해 줄 것이라고 오기를 부려 봅니다. 비록 좀 아쉬워 보여도, 비록 좀 부족해 보여도 참고 앞으로 나가고 싶습니다. 가재 같은 사람이 되고, 가재 같이 생각하고 이 산을 넘어 다음 언덕을 다시 목표로 삼고 나아가고자 합니다.

그렇게 생각이 정리 되고 나니 마음 한 켠이 후련함과 동시에 자신감이 생깁니다. 세상은 너무나 빠르게 급변하고, 따라가기도 벅찬 순간들이 계속해서 오는 것 같습니다. 배워도 끝이 없고 배우면 새로운 걸 요구하는 세상에 다소 암울한 생각이 들 때도 있습니다. 하지만 피어의 개발, 피어의 개발 전체를 바라보면서 그럼에도 불구하고 사람하는 일에 결국 줄다리기 같이 버티고, 꾸준하게만 해 나간다면 언젠가 결국 결론에 도달한다는 중요한 사실이 제 안에 남았습니다.

이번 백엔드 개발 과정은 정말 쉽지 않았지만 동시에 도전에 연속이었습니다. 어떤 기능을 구현하기 위해 단순히 있는 라이브러리, 패키지만을 쓰는게 아니라 이걸 위한 답을 찾기 위해서 직접 구현해보기도 하고, 특히나 개발 과정에서 생기는 개발 외의 조건들을 어떻게 타파할 수 있는가?와 같은 것들이 정말 많이 배울 수 있었습니다. 서버와 데이터베이스의 관계, 연결 방법에 대한 고민 등은 빼놓을 수가 없었고, 프론트엔드 개발자들이 자유롭게 작업이 가능하도록 NoSQL 을 활용하는 것은 특히나 고민이 많이 되고, 배울 수 있는 기회의 역할을 했습니다.

그리고 앞서 언급한 것처럼 사실 개발 자체의 다양한 기술적 접근, 배움도 중요하겠지만 팀 전체가 어떻게 코드의 퀄리티를 유지하며, 소통하며, 루즈해지거나 늘어지는 상황에서도 어떤 식으로 대응하면 될지를 정말 많이 고민할 수 밖에 없었습니다. 왜냐하면 개발자는 개발자이기 때문입니다. 고민하고 생각 하긴 하겠지만 전체 그림을 그리는 입장이 아닌 이상 지금 당장 앞전에 놓인 일에 최선을 다할 수 밖에 없는 것이 당연한 것이며, 코드 외의 것들까지도 넓게 보기에는 쉽지 않기 때문입니다. 그러다보니 팀원들 전체 속에서 생기는 느슨함, 불안감, 진척 속도에 대한 다양한 문제 등등... 이런 부분들에 대해 어떻게 조율하면 좋을지를 여러 방면으로 배울 수 있었습니다.

깃허브의 컨벤션을 지정하고, 깃 전략을 기존의 포크 정책에서 브랜치 전략으로 변경, 코드 리뷰를 개선하기 위해 코드 리뷰(PR)의 양을 API 단위로 제한하고, 양이 많을 경우 비즈니스 로직의 메서드 단위로 분할, 코드 분석 툴을 접목시켜 코드 스멜이나 버그들이 발생할 부분을 PR 단계에서 기계적으로 검출하도록 만들어 보았습니다. 그 결과가 이상적인 개발 문화를 만들었다는 생각하진 않으나, 이를 통해 성장하고, 문서화를 신경 쓰며 코드 퀄리티를 고민하는 팀원들의 문화를 만들 수 있었다고 생각이 듭니다.

앞으로, 개발자로서의 내가 되기 위해 이번 피어의 과정은 정말 큰 성장통이었다고 생각됩니다. 런칭이 되고, 향후 계속해서 업데이트 및 협업이 가능할지는 아직 확실하진 않으나, 개발 총괄이라는 말도안되는 직분을 맡고 밤을 새가며 여기까지 왔던 이 기억들은 앞으로 더 빠르게, 더 많이 바뀌는 개발의 세상에서 제가 살 아남을 힘을 키워줬다고, 특히 팀원들이 없이 이렇게 오긴 힘들었을 거라고 새삼 느낍니다. 몸소 동료학습, 협업이 왜 중요하고 성공적인 협업이 얼마나 대단한 일들을 이룰 수 있는지, 꿈꿀 수 있는지를 배울 수 있던 훌륭한 시간이었다고 말씀 드리고 싶습니다. 동료학습을 위한 커뮤니티가 되길 피어를 통해 바라는 만큼, 제가 느낀 이 감정과 이 경험, 이 고민들이 다음 피어 개발자들과 피어 커뮤니티 구성원들에게 계속해서 전달 될 수 있기를 기원해봅니다.



@hyeongki 김형찬

Peer가 피어나기까지

처음 슬랙에 올라온 peer 개발자 모집 공고를 보고 든 생각은 바로 '이거다'였다. 42서울에서 과제 외의 팀 프로젝트 경험이 필요했던 시기였는데, 42서울의 주요 정신 중 하나인 동료 학습을 도와주는 서비스라는 점에서 의미 또한 컸기에 너무 마음에 들었던 프로젝트였다. 그래서 큰 고민 없이 지원을 하고 미팅을 가진 후 최종적으로 개발진에 합류하게 되었다. 서초 클러스터 주피터 회의실에서 처음 다 같이 모였던 순간이 기억 난다. 그때부터 이 글을 쓰고 있는 현시점까지 많은 우여곡절이 있었지만 끝까지 책임감을 가지고 일해주신 peer 개발진분들이 있었기에 peer가 여기까지 올 수 있지 않았나 싶다. 동시에 그 과정에서 정말 여러방면에서 큰 성장을 이뤘다고 생각한다. 이 글의 초점은 어떤 우여곡절들이 있었고, 그 과정에서 과거와 비교하여 어떤 점들이 성장했는지에 대해 맞춰보려고 한다.

눈떠보니 최후의 상괭이가 되어버린 건에 대하여

최초에 peer 개발진의 구성은 기존에 웹 개발 경험이 있는 사람들로 이루어진 팀과 웹 개발을 완전 처음 시작하는 사람들로 이루어진 팀으로 이루어졌다. 백엔드에서 기존에 경험이 있던 사람들로 이루어진 팀의 이름은 상괭이였고, 두세 번의 웹 개발 경험과 더불어 peer의 백엔드 스택인 자바와 스프링에도 경험이 있던 나는 상괭이에 속하게 되었다. 원래는 총 3명으로 이루어진 팀이었지만, 제목에서도 알 수 있다시피 현재로선 나 혼자 유일하게 남아있게 되었다.

나간 팀원들 모두 각자의 사정과 이유가 있었지만 공통적으로 말했던, 그리고 나 또한 느꼈던 가장 큰 문제점은 소통의 부재였다. 본격적으로 프로젝트 개발에 들어가기 전, 웹 개발을 처음 접하는 팀원들은 토이 프로젝트를 진행하고 상괭이는 프로젝트에 필요한 스택들을 공부하는 시간을 가지고 있었는데, 팀원들 각자 이 스터디를 바라보는 시점이 달랐었던 것 같다.

당시에 나는 ft_transcendence라는 42서울 공통과정의 마지막 과제를 진행하고 있었지만, 다른 두 명의 팀원은 학교 졸업 프로젝트를 마치고 peer에 옮인을 하려는 상황이었다. 그런 상황에서 나는 개발에 본격적으로 들어가기 전 여유롭게 준비하는 시간이라는 인식이었지만, 옮인을 하고 있는 다른 팀원은 아니었던 것 같다. 그 팀원은 그 과정에서 굉장히 조바심을 느꼈지만 당시에 나는 몰랐다. 우리가 좀 더 소통이 활발했다면, 서로가 바라는 니즈를 조금만 더 궁금해했다면 해결이 될 문제였다. 그렇지만 우리 셋 다 모두 미숙했던 것 같다.

다른 한 명은 우리 팀, 그리고 백엔드의 리더였는데, 한 명을 떠나보내 놓고도 그 친구와 좀 더 적극적으로 소통을 하지 못했다. 와이어프레임이 지체되고, 백엔드 개발이 엄청 지연되는 상황에서 리더로서 엄청나게 압박을 받았었던 것 같은데, ft_transcendence를 끝낸지 얼마 안되어 peer에 올인한지 얼마가 안된 나는 그렇게까지 조급함을 느끼지 못했고, 그 친구가 그렇게까지 압박감에 시달리는지도 몰랐다.

소통? 별거 없다. 바라는 게 있으면 이렇게 해달라고 말하면 되는 거고, 마음에 안 드는게 있으면 그건 고쳐 달라고 말하면 된다. 사실 소통이 시작되기만 한다면 해결될 많은 문제들이 있지만, 가끔 우리는 소통을 시작하지 조차 않아 상황을 망쳐버리곤 한다. 나 또한 내 의견을 적극적으로 표현하지 않았고, 다른 팀원들이 어떤 상황인지에 대해 궁금해하지도 않았다. 개발자는 로봇이 아니라 사람인데, 개발자로서 협업하기 이전에 사람으로서 소통이 우선시 되어야 한다는 것을 배웠다. 그랬다면 최후의 상괭이가 되지 않았을지도...?

기술에 집착하지말고 체급에 집중하라

나는 이번 프로젝트를 진행하면서 기술적으로 성장한 점도 많았지만, 내가 42서울 통해 얼마나 개발자로서 성장했는지를 실감하기도 했다. 한 3년 전쯤 프로그래밍을 처음 배울 시절 진행했던 팀 프로젝트도 Java와 Spring으로 진행을 하였으나, 그때 당시에는 Java라는 언어를 사용하면서도 객체지향을 제대로 이해하지 못하고 활용도 하지 못했다. 또 Spring이라는 프레임워크를 사용하면서도 프레임워크가 무엇인지 설명조차 하지 못했으며, Spring의 3대 요소인 IoC, AOP, PSA는 이해하기를 시도할 엄두조차 내지 못했다. 그런 상태에서 진행했던 당시 프로젝트가 얼마나 엉망진창 이었는지는 가끔 옛날 코드들을 뒤적거릴때마다 느끼고는 한다,

분명 나의 Java와 Spring은 그 엉망진창이었던 3년 전에 멈춰있었다. 나는 그 이후로 프로그래밍을 아예 하지 않았던 시기도 길었고 42서울에 들어와서는 C와 C++만 주야장천 사용했다. 그런 내가 peer에 참여하며 오랜만에 Java와 Spring을 들춰보며 느낀 건 놀라움이었다. 앞서 나열했던 개념들과 그 필요성까지 모두 단숨에 이해가 되던 것이었다.

내가 아무 생각 없이 선언하던 Controller와 Service들, 그것들이 어떻게 객체화가 되고 관리가 되는지는 궁금하지도 않았고 사실 객체화가 되어야 한다는 사실을 인지조차 못했다. 그 모든 것을 해주는 IoC Container의 존재와 필요성을 느끼자 Spring의 위대함이 느껴졌다. 그와 더불어 DI는 당연한 것이었다. IoC Container 내부 객체들 간의 관계는 개발자가 정의해 주어야 한다.

AOP는 코드를 한층 깔끔하게 만들어주는 고마운 녀석이었다. 핵심 로직과 보조 로직의 분리를 통해 개발자가 비즈니스 로직에만 집중할 수 있게 해주며 만약 협업 과정에서 두 로직의 개발자가 다르다면? Spring AOP에게 절을 할 수밖에 없는 상황이 온다. 나는 peer에서 사용자와 그 행동들을 추적하는 파트도 맡았는데, 해당 파트를 개발할 때 AOP는 나에게 빛과 같은 존재였다.

내가 JDBC를 쓰던 때 사용하던 @Transactional을 어떻게 JPA 환경에서도 쓸 수 있는지 궁금증이 생겼을 때 비로소 PSA가 이해됐다. PlatformTransactionManager로 추상화된 인터페이스에 JDBC든 JPA든 구

현체를 가져다 끼우면 된다. 이는 JPA 추상화와도 이어지는 개념이다. 추가로 이 모든 마법이 가능해지게 해주는 객체지향... 잘 살펴보면 이 Spring의 주요 개념들은 모두 객체지향의 5원칙과 밀접한 연관성을 가지고 있으며 추상화와 상속, 다형성을 적극 활용하고 있다. 이러한 객체지향과 Spring에 대한 이해는 Spring Security를 이해하고 사용할 때도 많은 도움이 되었다.

나는 지인들과 이야기를 할 때 '개발 체급'이라는 표현을 자주 사용한다. 격투기의 체급과 마찬가지로 각종 기술에 대한 숙련도도 중요하지만, 더욱 중요한 건 어떤 기술이든 쉽게 이해하고 사용할 수 있게 해주는 기초 체력과 체급이 중요하다는 의미이다. 나는 분명 Java와 Spring을 공부하지 않았음에도 Java와 Spring이 늘어있었다. 이는 42서울을 통해 나의 '개발 체급'이 한층 성장했기 때문이 아닐까? 평소 머리로만 생각하던 것을 peer를 진행하며 몸으로 느끼고 나니 앞으로 어떻게 공부하며 성장해나가야 할지도 보였다. 기술에 집착하지 말고 체급에 집중하라. 해를 거듭할수록 급진적으로 발전하는 기술들 사이에서 살아남는 개발자는 누구보다 튼튼한 기본기와 거대한 체급을 가진 개발자가 아닐까?

마치 된 것 같아 개발자

peer를 개발하며 경험했던 것 중 가장 좋았던 점은 단순히 기술 공부용으로 진행하는 프로젝트가 아니라 실제로 서비스를 출시해야 하는 프로젝트를 경험했다는 것이다. 실제로 서비스를 출시하기 위해서는 기존에 굴러가게 만들고 땡 치면 되는 프로젝트들과는 달리 고려해야 할 것들이 많았다.

우선 Logging이 필요했다. 배포가 된 서비스는 로컬 개발 환경처럼 실시간으로 로그를 확인할 수 없기에 로그를 파일 형태로 남기는 것은 서비스를 운영하는데 필수적인 요소이다. 성능상의 이유로 Spring Boot에서 기본적으로 사용하는 logback 대신 log4j2를 사용하도록 설정하고 xml 파일을 통해 로그 파일 rolling 또한 구축하였다. 추가로 로그에 유의미한 데이터가 남아야 하기에 Spring AOP를 통해 모든 API들의 Request와 Response, Error까지 로그를 남기도록 구현하였으며 로그 레벨별로 각자의 파일에 rolling하도록 하여 사용성을 높였다.

CI/CD 파이프라인 또한 필수적이다. 실제로 서비스를 운영하는데 중요한 것이 배포할 버전의 안정성 검사와 자동화된 배포다. CI를 통한 빠른 코드 병합으로 개발 편의성을 향상시키고 빌드와 테스트를 수행함으로써 배포할 버전의 안정성을 검사할 수 있다. 그리고 CD를 통해 버전 관리와 배포에 소모될 인적 비용을 줄이는 것 또한 운영에 있어서 핵심적인 부분이다. 이전에 다녔던 회사에서 팀장님께 Jenkins을 이용한 CI/CD 파이프라인 과정에 대해 설명을 들은 적이 있었는데 당시에는 정말로 하나도 이해가 가지 않았고 저게 왜 필요한지도 몰랐다. 하지만 이번 프로젝트에서 Github Action을 통해 직접 CI/CD를 구축하다 보니 이게 얼마나 유용한지와 그 과정 또한 완벽히 이해할 수 있었다.

서비스가 운영이 되려면 인프라 또한 중요하다. 서비스를 운영하지 않는다면야 내 컴퓨터 로컬 환경에서 돌려보면 끝이지만 서비스가 운영이 되려면 24시간 돌아가는 서버에 배포가 되어야 한다. 그러한 서버, 즉 인프라에 대한 이해도 이번 프로젝트를 통해 크게 늘었다. AWS를 사용해 본 적은 있지만 블로그에서 하라는 대로 따라 해본 게 전부였던 나에게 NHN Cloud를 통한 인프라 구축은 매우 골치 아픈 과제였다. NHN

Cloud는 주로 기업을 상대로 서비스를 제공하던 CSP였기에 인터넷에 자료 또한 거의 없었다. 그나마 다행인 점은 한국 기업이라 한글로 된 Document를 천천히 따라가며, 그리고 제품들의 특징이 AWS와 많이 흡사했기에 AWS와 비교해가며 밤새 테스트 서버 환경을 구축했던 기억이 난다. 추가로 NHN Cloud에서 진행했던 교육에도 팀원들과 참가하며 더욱 인프라, 특히 클라우드에 대한 지식을 쌓아갔다. 현재는 비용 문제로 NHN Cloud에서 On-Premise 환경으로 이전을 준비하는 과정에 있다. On-Premise 환경에 대한 이해 까지 늘어난다면 어디 가서 당당히 “나 DevOps야”라고 외칠 수 있지 않을까?

이처럼 peer를 진행하며 실제로 서비스를 출시, 운영하기 위해 새롭게 경험한 것들이 많았다. 진정한 개발자로서의 첫걸음은 직접 만든 서비스가 세상에 선보이는 순간이 아닐까란 생각을 한다. 드디어 학생을 넘어 개발자로서 발돋움을 하는 태동의 순간에 있는 게 아닌가라는 생각을 한다.

이번 생에 팀원은 처음이라

프로젝트가 거의 마무리가 되어가는 이 시점에 내가 깨달은 것이 하나 있다. 그건 바로 잘 이끄는 것도 중요하지만 잘 따르는 것도 중요하다는 것이다. 지금까지 경험했던 프로젝트들에서는 다 리더나 그에 준하는 역할을 맡았었지만 이번 프로젝트는 팀원으로 참여하게 되었다. 팀원으로 참여하게 되었을 때, 부끄럽지만 내 첫 심경은 ‘편하겠는데?’였다. 지금까지 프로젝트를 하면서 리더로서의 고충은 충분히 느껴보았기 때문이었다. 그래서 솔직히 말하자면 처음에는 그냥 따라가기만 하면 되겠지, 시키는 것만 하면 되겠지라는 마음이 짐이었다. 내가 지금까지 경험했던 프로젝트들에서는 리더가 개발 상황 파악과 태스크 분배를 해주면 팀원은 개발만 잘 하면 됐었다. 하지만 그건 내가 했던 프로젝트가 모두 많아야 4~5명 정도의 소규모 프로젝트였기 때문에 가능했던 것이 아니었나 싶다. 거진 열댓 명의 개발자가 포함된 이런 대규모 프로젝트에서 그건 불가능하다는 것을 몰랐다. 그리고 팀원으로서의 입장도 처음이다 보니, 내가 어느 정도까지 능동적으로 의견을 내도 괜찮은지 모르기도 했고, 편하게 따라가고 싶은 마음도 크긴 했다. 지금 생각해 보니 내가 했던 프로젝트들에서도 내가 리더를 맡기는 했지만 팀원들 모두가 적극적으로 의견도 내주고 나를 도와주었기 때문에 내가 리더로서의 역할을 잘 할 수 있었던 것 같다. 그런데 나는 이번 프로젝트에서 그렇게 능동적으로 참여하며 ‘리더에게 많은 도움을 주는 팀원이었나?’라는 물음에 아니오라는 답을 할 수 밖에 없는 것 같다.

솔직히 아직도 모르겠다. 좋은 팀원이란 무엇이고, 어떻게 하면 리더에게 도움이 되는 팀원이 될 수 있는지. 어떻게 보면 좋은 리더가 되는 것보다 추상적이고 어려운 문제라고 생각한다. 취직을 한다면 이끄는 입장보다는 따르는 입장이 될 것이다. 그런 상황 앞에서 좋은 팀원이 되려면 어떻게 해야 할까라는 고찰을 얻었다는 것만으로 나는 큰 걸 배웠다고 생각한다. 옛날에 했던 프로젝트에서 우리 팀원들이 어떻게 해줬는지부터 곰곰이 떠올려봐야겠다.

중요한 건 꺼이지 않는 마음

가끔 개포 클러스터 랩실을 가득 채운 팀원들을 보면 놀라웠다. 저 많은 사람들이 한 목표를 위해 모여 달려가고 있다는 것은 너무나도 놀랍고 대단한 일이다. 책임감을 가지고 이끌어준 리더 분들과 항상 열심히 일해준 팀원들이 있었기에 수많은 우여곡절 사이에서도 peer는 무너지지 않고 완성을 향해 달려나갈 수 있는 게 아닌가 싶다.

포기하면 실패의 이유가 되지만 포기하지 않으면 성공의 밑거름이 된다고 생각한다. 우리가 겪었던 문제들, 쉽지만은 않은 과정이었지만 결국에 peer라는 서비스를 완성해 세상에 선보인다면 모두 우리의 경험으로 축적되어 우리를 한 단계 더 성장시켜줄 것들이다. 중요한 건 꺾이지 않는 마음이라는 말을 참 좋아한다. 한 번 문건 끝장을 봐야 직성이 풀리는 성격이다 보니 웬만한 건 죽이 되든 밥이 되든 끝까지 해내려고 하는 편인데, 그렇게 끝장들을 보다 보니 느낀 건 어느 상황에서도 마음만 꺾이지 않고 버텨내면 결국 무엇이든 남긴 남는다는 것이다. 그 남은 것들이 쌓여 나를 한층 더 성장 시킨다는 생각으로 하루하루를 버텨내다보면 결국 우리가 바라는 모습이 되어 있지 않을까란 생각을 한다.

peer를 하면서 기술적으로도 많이 성장을 했지만, 무엇보다 개발자로서, 사람으로서 여러가지를 배워가는 것 같아 정말 뜻깊은 프로젝트였다고 생각한다. 정말 좋은 리더분들, 팀원분들을 만난 것 같고 여러분들 덕분에 많은 걸 배웠다고 생각한다. 다들 고생 많았고 peer에서의 경험을 밑거름으로 성장하여 원하는 바 모두 이루기를 바란다.



@juhyelee 이주현

첫 걸음

Peer 개발 프로젝트를 알게 된 것은, Peer에서 이미 활동 중인 haryu 님을 통해서였습니다. 당시 저는 로그라이크 게임을 개발하는 팀 프로젝트에 참여하고 있었고, 팀 프로젝트는 막바지에 이를 때였습니다. 당시 저는 해당 게임 프로젝트가 첫 팀 프로젝트였습니다만, 소프트웨어 개발자가 아닌 디자이너로서 참여했다는 점이 아쉬웠습니다. 때문에, Peer 개발 프로젝트는 저에게 아주 매력적인 프로젝트였고, 소프트웨어 개발자로서 첫걸음이기도 했습니다.

Peer 개발 프로젝트에서만큼은 정말 디자인과 아무런 관련이 없는 것을 해보고 싶었습니다. 개인적으로 Peer 프로젝트가 게임 개발 프로젝트였으면 얼마나 좋았을까 하는 생각이 들었지만, 개발이면 무엇이든 괜찮을 것 같다는 생각이 더 컸던 것 같습니다. 그리고 지금까지 디자인의 영역을 조금이라도 걸치고 있었다면, 단 한 번도 걸치지 않은 영역. 그리고 정말 소프트웨어 개발자라는 그 직업이 저에게 맞는지 시험해 보고 싶기도 했습니다. 그래서 백엔드를 선택했고, 그 안에서 살아남기를 바랐습니다.

팀 덕분에

생각했던 것보다 처음부터 모든 것이 쉽지 않았습니다. 백엔드 팀과 함께 스터디를 한 이후에 프런트 팀의 한 명과 협업해서 파이널 미션을 진행했습니다. 모든 것이 낯설고, 익숙하지 않은 상황의 연속이었습니다. Java라는 언어와 Spring이라는 프레임 워크를 처음 배워야 했고, JPA를 잘 사용하기 위해서는 SQL과 관계형 데이터베이스에 대해서 공부해야 했습니다. 백엔드 팀원분들이 말하는 단어들(가령, WAS나 servlet, middleware, CI/CD 같은 용어들)과 프런트 엔드 팀원분들이 말하는 단어들을 이해해야 했고, GitHub를 사용하는 방법, 문서화, 프런트 엔드와 밸걸음을 맞추고 협업을 하는 방식을 찾아야 했습니다.

이상한 '책임감'이 있었던 저에게 이러한 상황은 오히려 저를 구석으로 가두었습니다. 모든 것을 혼자 해결 하려고만 했고, 그 누구에게도 물어보려 하지 않았습니다. 그러나 그럴 수 없었습니다. 그 이기적인 '책임감' 때문에 오히려 화를 부르는 일이 많았습니다. "Peer"라는 그 이름과 철학에 맞게 스터디부터 비롯한 모든 과정을 함께 진행하고, 함께 이루어 나갔습니다. Java 스터디를 비롯해서, jwee 님과 junssong 님과 함께 진행한 JPA와 웹 서비스의 구조에 대한 스터디를 함께 진행했습니다. 그리고 당연한 말이지만, 스터디 이후 파이널 미션과 실제 개발 과정 자체도 백엔드 팀과 더불어 프런트 엔드 팀과 함께 동기화하며 진행했고, 협업하는 과정에서도 여러 어려운 문제들, 감이 잡히지 않는 문제들을 함께 해결하며 진행했습니다. 덕분에 그 이상 한 '책임감'은 잘못된 것이라는 것을 알게 되었습니다.

개발을 진행하면서 Spring에 대한 이해도가 점점 깊어졌습니다. Spring에서 그저 프레임워크에서 제공해주는 줄로만 알았던 - 마치 마법처럼 동작했던 어노테이션을 직접 구현하는 것을 hyeongki 님의 코드를 리뷰하면서 알게 되었고, Java로 비동기적으로 동작하도록 하는 방법이나 CI/CD에 대해서도 완전히 이해되지는 않았지만, 어떤 식으로 설계하는지도 보게 되었습니다. 모든 경험들이 저에게 엄청난 득이 되었습니다. 놀라웠던 것은 프로젝트를 하며 42 과제 때문에 OSTEP이라는 책을 봐야 했었는데, 병행성에서 Translation 내용이 나와 금방 쉽게 이해할 수 있었다는 것입니다. 만일, 이러한 프로젝트와 경험이 없었다면 병행성으로 인한 문제를 이해하는 데 수일은 걸렸을 것입니다.

팀 때문에

함께 한다는 것은 서로 의지할 수 있다는 말이기도 했지만, 서로 다른 가치관이 충돌하는 것이기도 했습니다. 저희 팀도 좋은 순간이 많았다면, 좋지 않은 상황을 겪을 수밖에 없었습니다. 너무나도 독립적인 개개인이 동일한 목표와 목적을 가지고 톱니바퀴처럼 딱 맞아떨어지게 서로 움직인다는 것이 결코 쉬운 것은 아니었습니다. 각자의 성향과 각자가 상대에게 바라는 것들, 그리고 각자의 마음 상태 등 여러 가지가 맞물리지 않았습니다. 때문에 몇몇 인원은 프로젝트를 떠나게 되었고, 상황에 대해서 정리할 필요가 있었습니다.

개발 초기에는 프런트와 백엔드가 동기화되지 않는 문제가 발생하기도 했습니다. 백엔드는 프런트에서 어느 정도 진행되고 나서 본격적으로 개발이 시작되고, 프런트는 디자인과 목업이 나오기까지 대기할 수밖에 없습니다. 물론, 그동안 마냥 앉아 기다리는 것이 아니라 스터디를 진행하거나 기획을 바탕으로 데이터베이스를 설계하기도 했습니다. 한참 기획이 나오고 프런트에서 개발이 어느 정도 진행되었을 때, 백엔드와 프런트는 서로의 말을 맥락적으로 이해할 수 없었습니다.

이러한 불화들은 소통의 부재로 인한 문제이기도 했고, 때문에 소통으로만 해결할 수밖에 없었습니다. 이러한 상황을 바라보며, “나였으면 어떻게 했을까?” 하는 생각이 많이 들었습니다. 저로서는 상상도 하지 못 할 일들이었고, 전혀 예상하지 못한 일들이었기 때문입니다. 그리고 그러한 문제들을 잘 해결할 자신도 없었습니다(그런 의미에서 리더들이 대단하다 생각 듭니다). 물론 제가 해결한 문제들은 아니지만, 이러한 상황을 보면서 소통의 중요성을 다시 알게 되었고, 소통을 통해 결론을 도출해야 한다는 것이 중요하다는 것을 알게 되었습니다.

시작

Peer 개발 프로젝트가 저에겐 끝이라고 생각하지 않습니다. 이제 소프트웨어 개발자로서 첫걸음을 내디뎠고 한 걸음씩 차근차근 걸어가야 할 것입니다. 42 과정을 압축해서 배웠다고 생각될 정도로 정말 많은 것들을 배웠지만, 앞으로도 배울 것이 정말 많습니다. 특히나 CS, 운영체제에 대한 것들에 대한 부족함을 알고, 수많은 훈련과 수많은 노력이 필요하다는 것을 알게 되었습니다. 특히나 커뮤니케이션 능력에 있어서는 많은 것들이 발전했을지라도 저는 이제서야 아주 '기초적인' 것들만 숙지했을 뿐입니다.

때때로 만일 내가 Peer를 하지 않았다면 어떻게 되었을까 하는 생각을 합니다. 물론, Peer 이외에도 다른 프로젝트를 기회로 삼아 또 다른 때의 시작을 했을 수도 있겠지만, 현시점으로 보았을 때는 여전히 제자리걸음을 했을 것 같다는 생각이 들었습니다. 다사다난했던 프로젝트의 거의 모든 문제에는 '혼자'가 있었고, 그 해결 과정에는 늘 '팀'과 '동료'가 있었습니다. 그러한 상황을 겪고 함께 간다는 것이 중요하다는 것을 깨달은 저와 그것을 여전히 알지 못하는 저에게는 큰 차이점이 있습니다.

아쉽게도 Peer 개발 프로젝트에서 1차 개발을 마무리하고 잠시 나와야 했습니다. 저의 성실하지 못함으로 인해 42 과제들이 밀렸기 때문입니다. 웹 사이트가 구축되면서 '내가 해내고 있다'는 희열감이 있었고, 성취감이 있었지만, '이왕이면 끝까지 마무리 짓고 싶은데'라는 아쉬움이 있었습니다. 그러나 첫 솔에 배부를 수 없듯, 이후에 언젠가 참여할 혹은 언젠가 진행할 프로젝트를 통해 더더욱 성장하고자 하는 갈망이 생긴 것 같습니다.

지금도 GitHub에 PR을 올리며 노력하며 수고하고 계신 분들, junssong 님을 비롯해 1차까지만 마무리를 지으신 분들 모두 고생하셨고 함께 성장 할 수 있어 감사했습니다.



@junssong 송준상

Peer다웠던 Peer에 참여하며

왜 Peer 였나요?

23년 1월 Piscine을 할 때였습니다. 42카뎃에게 RUSH 평가를 받을 수 있는 기회가 있었는데, 42는 카뎃들이 직접 프로젝트를 만들고 그것을 지속 운영하고 있다는 사실을 알게 되었습니다. 그중에서 가장 흥미가 있었던 프로젝트가 42Peer 였습니다. 대학교에서도 컴퓨터 공학을 공부했지만, 코로나여서 비대면이기도 해서 팀, 동료들과 같이 코딩을 할 수 있는 시간이 많지 않았습니다. 하지만 42와 Peer는 스스로 스터디를 만들고 동료학습을 할 수 있도록 권장하는 일들을 하는 것이었습니다. 그때는 노션으로 스터디를 만들어 팀원을 모으는 것조차 대단하다고 생각했습니다.

저도 Peer 프로젝트 같은 것을 하고 싶다고 RUSH 평가 때 막연하게 생각하고 있었는데 본과정에 합격하게 되고 Peer 운영진을 뽑는다는 글을 보고 바로 신청하게 되었습니다. 운영진과 개발진을 같이 할 기회였고 운 좋게도 모두 할 수 있게 되었습니다.

왜 backend 였나요?

결론부터 얘기하면 처음에는 프론트보다 백을 공부하는 게 더 취업에 유리할 것이라는 생각이 있었습니다. 웹서버가 어떻게 동작하는지 알고 싶었고, 우리 눈에 보이지 않는 웹의 뒷면을 막연하게 배워보고 싶다는 생각이 많았습니다. 한국에서 백엔드로 취업하려면 당연하게 Java는 할 줄 알아야 한다고 생각했습니다. 지금 생각해 보면 굉장히 안일한 생각이었다고 생각합니다. 어떤 것을 하는 게 중요한 것이 아니라 얼마나 열심히 포기하지 않고 하는지가 중요한 것 같습니다.

spring에 대해서 전혀 알지 못했고 웹서버의 동작 방식에 대해서도 전혀 알지 못했지만, 처음 2개월 동안 백엔드 팀원들과의 스터디를 통해서 spring의 싱글턴 방식이나 mvc 패턴 등 동작하는 흐름에 대해서 알 수 있었습니다. 계속해서 스터디를 이끌어주신 jwee님께 이 자리로 다시 감사드립니다. 백엔드를 개발한다는 것 웹을 개발한다는 것에 부담감과 압박감이 심했는데 백엔드 팀원분들의 도움으로 일정 기간까지 포기하지 않고 개발할 수 있었습니다. 지금은 백엔드를 개발한다고 하면 spring이 아니어도 두려움은 없을 거 같습니다. 이것 하나만으로도 4개월 넘는 기간의 Peer 개발이 헛되지 않았다고 생각합니다.

12명이 넘는 팀원들과의 협업은 어땠나요?

Peer 개발을 하면서 처음부터 마지막까지 haryu님이 얘기하신 말이 있습니다. '개발자는 '소프트 스킬'

이 굉장히 중요하다.' 저는 이 말이 처음에 큰 의미가 있는지 몰랐습니다. 하지만 백엔드에서 3명의 팀원이 나가는 것을 직접 겪고 나니까 팀 프로젝트에서는 실력만큼 중요한 것이 소프트 스킬이라는 것을 알게 되었습니다.

spring과 Java 공부는 열심히 했지만, 소프트 스킬을 높이는 노력은 하지 못했던 거 같습니다. 나의 불만을 얘기할 줄 아는 것도 노력이 필요했고, 팀원들이 불만을 나에게 얘기할 수 있도록 하는 것은 더욱 노력이 많이 필요한 일이었습니다. 앞으로도 협업을 계속할 텐데 그럴 때마다 Peer를 개발했을 때가 생각날 거 같습니다. 가장 좋은 협업은 내가 못 한 부분과 부족한 부분을 인정하고 그 일을 빠르게 처리할 수 있는 상황으로 만드는 것으로 생각합니다. Peer 개발을 하면서 소프트 스킬의 중요성을 많이 배운 거 같습니다.

백엔드 팀원과의 협업만큼 프론트 팀원들과의 협업 또한 만만하지 않았습니다. API를 설계할 때 dto와 변수명을 정했지만 제대로 확인하지 않거나 문서 업데이트가 되지 않아서 한 글자 차이로 오류가 나는 경우가 빈번했습니다. 회사에 다니면서는 이런 일이 적겠지만, 다음 협업에서는 정확하고 명확한 변수명 정리와 API 설계를 먼저 하는 게 중요하겠다고 생각했습니다.

Peer 개발에서 내세울 것이 있다면?

Peer의 백엔드 개발을 모듈별로 진행하는 원칙을 따라, 각 팀원은 맡은 부분을 직접 개발했습니다. 저는 로그인과 로그아웃 기능을 맡아 개발했는데, 이는 웹 서비스에서 필수적이면서도 중요한 기능 중 하나입니다. 로그인이 없는 웹 서비스는 거의 없다고 할 정도로, 로그인은 사용자 경험과 보안 측면에서 필수적인 기능입니다.

백엔드와 웹 개발도 처음이었던 상황에서 로그인을 구현하는 건 굉장히 어려운 작업이었습니다. 특히 JWT Token이라는 것을 활용하여 기존의 쿠키 방식의 로그인이 아닌 토큰 방식의 인증과 인가는 저에게 굉장히 낯선 개념이었습니다. 그럼에도 개발은 완료해야 했기에 RFC 문서라는 것도 처음 읽어보고 구글링 실력도 늘려가면서 Spring Security와 Redis를 활용하여 로그인과 로그아웃을 구현할 수 있었습니다. 개발이 늘어지는 상황이 발생하여 각자 맡은 모듈을 변경했어야 해서, OAuth2.0을 적용하여 42, 구글 로그인까지 구현하지 못한 것이 아쉬웠지만, 다음 웹 개발에서는 이 부분까지 구현해 보고 싶습니다.

Peer는 협업을 수월하게 하기 위한 서비스입니다. 그러므로 가장 중요한 로직이 Team과 관련된 부분이었는데 이 부분을 맡아서 개발했습니다. 특히 entity를 만들 때 ManyToMany(다대다) 관계의 entity를 사용할 때 실수했던 것을 고치면서 entity와 관계형 데이터베이스(MySQL)에 대해서 배울 수 있었습니다.

Peer를 개발하면서 느낀 점?

이 글을 쓰면서 내가 얼마나 좋은 기회를 잡았었고 좋은 팀원들과 함께했었는지 다시 생각하게 되었습니다. 요즘 개발이 재미가 없고 스스로 늘어진다고 생각했었는데 다시 5개월 전의 나를 생각하면서 글을 작성했습니다. 그때는 배우는 모든 것들이 다 처음이었고 신기했습니다. 아직도 모르는 게 많고 CI/CD는 건드려

보지도 못했습니다. 42본과정 과제를 하면서 계속 Peer 개발 때가 생각날 거 같습니다. 그러면서도 후회가 많이 되는 거 같습니다. 내가 좀 더 열심히 했으면 11월 안에 끝날 수 있지 않을까? 팀원들과의 소통도 많이 하지 않았던 거 같습니다. 후회로 끝나는 협업이 되지 않길 바랐지만, 스스로에서 아쉬운 점이 많은 것 같습니다. 하지만 배운 점이 정말 많았고 스스로 부족한 부분을 찾을 수 있었던 긴 시간이었습니다.

하류님 말대로 Peer만큼 좋은 팀원들 구하기 어려울 것 같습니다. 하지만 Peer가 완성되면 누구보다 많이 사용할 것 같습니다. 저는 혼자 공부하는 것보다 여럿이서 같이 공부하는 게 훨씬 능률이 좋다는 것을 깨닫는 좋은 시간이었습니다. 처음으로 돌아가면 조금 더 주도적으로 일을 진행해서 세부 기획 회의 때 놓친 세세한 부분을 빨리 잡아서 적용했다면 일이 더 수월하게 진행되었을 거 같습니다. 리더가 있고 기획을 해주는 사람들이 있다는 생각으로 코더가 된 것처럼 행동했던 게 아쉽습니다.

좋은 기회로 Peer 웹 개발에 참여할 수 있었습니다. 저는 팀원분들에게서 많은 도움을 얻었고 백엔드 개발은 두려움 없이 할 수 있게 되는 성장을 하게 되었습니다. 운영진의 일도 개발진의 일도 다 끝내지 못하고 그만두는 것처럼 되어 아쉽지만, 내년에 또 기회가 되면 참여하고 싶습니다. 다들 고생하셨고 감사합니다. Peer 개발이 없었다면 또다시 처음 부딪히는 삽질의 연속이었을 것입니다. 이번 개발을 통해 웹 서버 과제와 트랜센더스까지 잘 달려갈 자신이 있습니다. 앞으로도 새로운 기술과 프로젝트에 도전하며 성장해 나가겠습니다.



@jwee 위치혜

개발 후기

익숙하지 않은 길을 걷는 동안

저는 나름 긴 시간동안 쉬지않고 걸어왔던 교육자로서의 길이 개발자라는 새로운 길과 교차하는 지점에서 나는 42seoul을 만났습니다. 다른 사람보다 조금 늦은 삶의 지점에서 만난 42seoul이지만 마치 내가 처음 만나는 교육인것처럼 그렇게 열정을 다해 보기로 다짐했었어요.

첫 과제부터 1년 하고도 3개월이라는 시간동안 만난 과제 하나하나에 즐거움을 느끼고 공을 들여가며 기초부터 차근히 개발자로서 역량을 쌓고 있는 중이었습니다. 그러던 중 한가지 의문이 생겼고 그 의문은 곧 저를 Peer라는 프로젝트로 이끌었습니다.

나를 키운건 8할이 의심이요

앞서 이야기 한 의문은 사실은 제가 picine을 할 때부터 가져왔습니다. 제가 Picine에서 느낀 가장 강렬한 것은 바로 무력감입니다. 흔히 말하는 완전한 '노베이스'로 참여하면서 picine에서 주어지는 4번의 팀 프로젝트에서 나 스스로 아무것도 도움이 되지 못한다는 느낌에 압도되었던 기억이 아직도 생생합니다. 그 강렬한 경험은 42seoul학습을 진행하면서도 나 자신에게 "나는 무엇을 할 수 있는 사람인가?"라는 질문을 끊임 없이 던져 왔습니다.

본론으로 돌아와서 42seoul에서 과제 하나하나에 힘을 쏟으며 학습해나가는 과정을 정말 멋지고 즐거운 일이었지만 학습을 진행하면 할 수록 제가 아무것도 모른다는 사실을 동시에 확인하는 과정이기도 했습니다. 그래서 학습을 진행하면 할 수록 무엇인가 만들어보고 싶다. 곧, 프로젝트를 하고 싶다는 마음으로 연결되었습니다.

여러분 제 동료가 되어주실래요?

프로젝트라는 말은 두려운 단어입니다. 특히 판을 벌려놓고 수습할 일을 생각하면 더 그렇습니다. 그래서 개발자로서 판을 벌리고 사람을 모으고 수습하고 결국에는 값진 것을 만들어내는 사이클을 겪어보고 싶어서 Peer에 문을 두드렸습니다. 면접도 보고 여러 우여곡절이 있었지만 Peer에서 저의 첫번째 역할을 미갈루라는 팀의 리더였습니다. Peer는 웹서비스 개발의 경험자와 비경험자로 구성된 팀이었기 때문에 초반에는 학습하는 팀과 프로젝트 전반에 대한 설계하는 팀으로 나누어져 진행되었습니다. 학습하는 팀이었던 미갈루팀의 미션은 1개월동안 빠르게 Java와 spring boot를 익히고 파이널 미션을 통해 성장을 증명하는 것이었습니다.

초보자라는 딱지를 붙인, 그리고 팀프로젝트를 거의 경험해보지 못한 팀원들이 모인 미갈루팀은 저를 포함해서 개인적인 불안감이 많이 느껴졌습니다. 그래서 팀활동 초반에는 우리가 팀이라는 것을 유난히 더 강조했던 것 같습니다.

하지만 그 과정에서 조심스러움이라는 이름으로 어쩌면 팀원들을 꼼꼼하게 살피고 충분히 소통하지 못해 발생한 문제들을 수습하기 위해 많은 시간을 투자해야 하기도 했습니다. 그리고 그 과정에서 팀원 한 명을 떠나보내는 아픔을 겪었습니다. 이는 우리 모두에게 어려운 순간이었고, 팀으로서의 연대감과 책임감을 더욱 깊이 느끼게 해주었습니다. 그러나 이 경험은 우리 팀에게 중요한 교훈을 남겼습니다. 모든 팀원이 각자의 역할을 수행하는 동안, 상호 의존성과 팀워크의 가치가 얼마나 중요한지를 깨달았고, 이후에 제가 훈들리는 상황에서도 팀은 저를 다시 세워주었으니까요. 초반에 혹여나 깨질까 정말 조심스럽게 손을 내밀었던 동료들이 이제는 든든하게 성장하여 팀의 필요한 역할들을 다 해내는 모습은 우리의 결과물 이외에도 제가 자랑스럽게 생각할 수 있는 일부이지 않을까 생각합니다.

기획보다는 개발?

Peer 프로젝트를 통해서 배운것, 또는 남는 것이 무엇인지 곰곰히 생각해보면 결국은 기획과 실행이 아닐까 싶습니다. 처음에 팀에 들어갈때까지만 해도 저는 "개발"을 하고 싶다는 막연한 생각이었습니다. 결국 이 생각은 개발자로서 아직 갖추어지지 않은 어리석은 생각이었다는 것을 곧 깨닫고 말았습니다. 기획이 없이는 제대로된 코드 한줄을 쓸 수 없었습니다. 그래서 Peer의 리더진들은 긴 시간 조금은 코통스럽게 기획을 해나갔습니다. 때로는 의견이 맞지 않아 버튼 하나를 놓고 아주 오래 시간동안 싸우다 시피 논의를 진행해야 했고 때로는 생각지도 못한 새로운 아이디어에 흥분하면서 우리의 서비스를 하나하나 기획해 나갔습니다. 이 과정을 글로 다 설명할 수는 없지만 솔직히 저는 이 과정에 완전히 지쳐버려서 기획보다는 개발을 하고 싶다고 공공연하게 말하기도 했습니다.

그렇게 본격적인 개발 단계에 돌입했을 때 기획의 중요성을 절실히 깨달을 수 있었습니다. 기획 단계에서의 논의와 토론, 심지어 갈등까지도 모두가 의미 있는 과정이었습니다. 이러한 과정을 거치지 않고서는, 우리가 만들고자 하는 서비스의 본질을 제대로 이해하고, 효과적인 솔루션을 개발할 수 없었을 것입니다.

개발에 몰두하기 시작하면서, 각 기능과 요소가 어떻게 서로 연결되고, 전체 시스템에 어떤 영향을 미치는

지를 이해하기 시작했습니다. 기획 단계에서 논의된 내용들이 실제 코드로 구현되는 과정을 보면, 모든 세부 사항이 얼마나 중요한지를 깨달았습니다. 그러한 세부사항들이 모여서 전체적인 사용자 경험과 시스템의 안정성을 결정했습니다.

Peer 프로젝트를 통해, 기획과 실행의 밸런스가 얼마나 중요한지를 배웠습니다. 처음에는 단순히 코드를 작성하는 것이 개발자의 전부라고 생각했었습니다. 하지만 이제는 아이디어를 현실로 만들어내는 과정 자체가 개발의 중요한 부분임을 이해합니다. 기획 단계에서의 깊은 사고와 논의는, 실제 개발 과정에서 효율성과 명확한 목표를 제공하고 팀원들이 제대로 된 협업을 할 수 있도록 만듭니다.

이 경험은 단순한 기술적인 성장을 넘어, 전략적 사고와 팀워크의 가치를 깊이 있게 이해할 수 있는 기회가 되었습니다. 앞으로 성숙한 개발자가 되고, 복잡한 문제를 해결하는 데 있어서 보다 깊이 있는 접근을 할 수 있을 것입니다.

그래서 나는 뭘 할 수 있는데?

이 여정의 마지막에 서서 앞서의 의문을 다시 한번 자문해봅니다, "그래서 나는 뭘 할 수 있는데?"

Picine에서 무력감을 느끼던 저는 내가 가진 기술의 한계에 부딪쳐 좌절감을 겪었습니다. 말 그대로 주제와 관련한 아이디어도 코드 한줄을 써내는 것도 불가능한 상황이었습니다. 협업을 잘하고 소통을 잘하는 능력이 기술적 무능함 앞에서 아무 일도 하지 못하는 경험이었죠.

그래서 Peer 프로젝트 초반에는 다른 무엇보다 내가 이 프로젝트를 해나가는데에 대한 기술을 익히는 것에 초점을 맞췄습니다. Java라는 언어와 프레임워크에 대한 이해, 데이터베이스에 대한 설계 능력과 구현 능력 등을 얻었습니다. 그리고 이런 기술적, 지식적 획득으로 저는 제가 프로젝트를 순조롭게 해 나갈 수 있을 거라고 믿었던 것 같습니다.

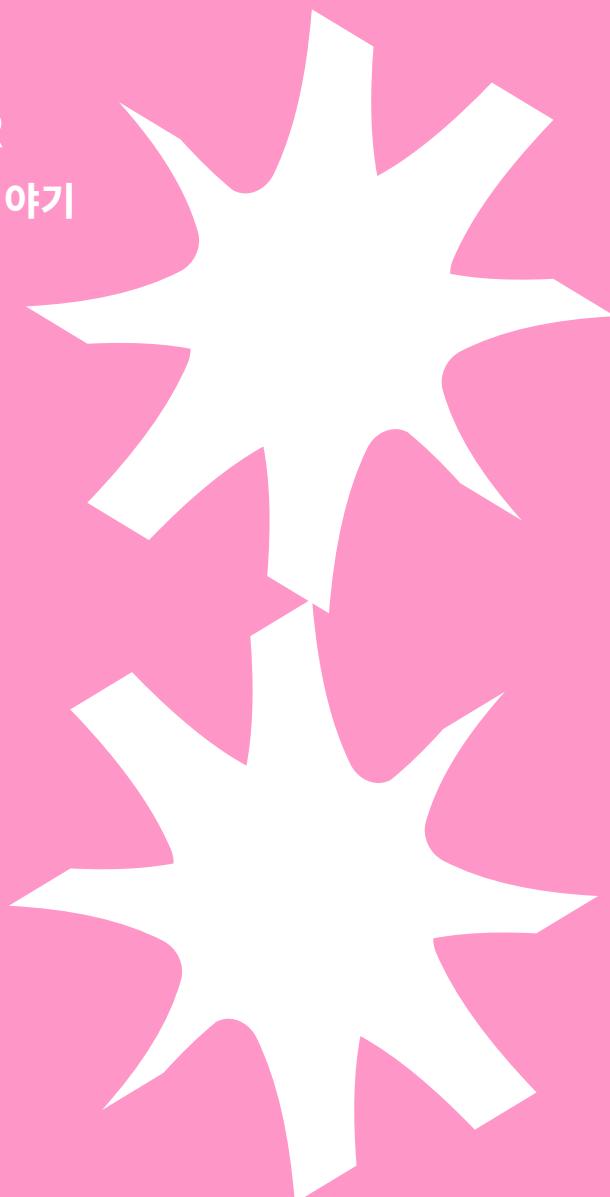
그러나 결국 기술적 성장은 개인으로서 갖춰야 할 가장 "기본"적인 것이었고 팀프로젝트는 단순히 기술적인 성장 그 이상을 요구하는 것이었습니다. 제가 경험한 Peer프로젝트에서는 기술적 문제 해결뿐만 아니라 프로젝트의 방향을 설정하고, 가치에 맞게 서비스를 디자인하며, 기획과 실행 사이의 균형을 이해하고, 함께 일하는 다양한 사람들의 욕구와 상황을 조율하고 결국 결과물로 세상과 소통해야 한다는 점 등 정말 다양한 측면에서 개발자로서의 역량에 대해 배울 수 있었습니다.

이 총체적이고 종합적인 여정은 내가 개발자로서 무엇을 할 수 있는지 찾아가는 여정이었던 동시에 앞으로의 여정을 조금 더 즐겁게 갈 수 있도록 해주는 발판이 되기를 간절히 바래봅니다.

제 4장

DESIGN

DESIGNER
디자이너들의 이야기



@Bori 이보람

132

@Yachae 양채윤

133

@Joah 조하연

134



@BORI



디자인 후기(UX/UI)

인생 첫 사이드프로젝트를 시작하며, 설렘과 긴장을 안고 피어에 합류한 게 엊그제 같은데, 드디어 그 대장정을 마무리하는 순간이 왔네요.

프로젝트 중간에는 막막했던 순간들이 생각보다 많이 있었습니다. 아이디어 자체로는 너무 색다르지만 사용자가 어떻게 받아들일지 고민해야 되던 순간들이 있었어요. 또한 그 아이디어가 개발을 통해 실현되기 위해서는 현실과 타협해야 하는 부분들이 있었답니다. 이런 고민이 가장 많았던 곳이 어디냐고 묻는다면, 팀폐이지였습니다. 아이디에이션 단계에서는 사용자들에게 흥미를 줄 수 있는 재미있는 요소들이 정말 많았는데, 최종 디자인에서는 빠진 것들이 많아서 아쉬움이 남습니다. 하지만 사용자들을 생각했을 때 당장의 재미 요소를 모두 포함하는 것보다는 팀페이지라는 목적 자체, 즉 팀원들의 작업과 소통에 용이한 페이지,라는 것에 집중하는 것이 옳다는 결정을 했습니다. 언젠가 피어 사용자들이 늘어나고 더 많은 기능이 사용자에게 필요한 순간이 왔을 때, 함께 고민했던 다양한 기능들을 선보일 수 있기를 기대해 봅니다.

서비스를 마무리 지으며, 다시 한번 사용자와의 소통의 중요성을 느꼈습니다. 팀원이자 동시에 사용자인 우리 피어의 개발진분들과 많은 고민을 하고 함께 해결해나가며 어느 순간 진정한 한 팀이 되는 것을 느꼈습니다. 동료와 소통을 중요하게 생각하는 피어 팀원들 덕분에 어려운 순간들도 극복할 수 있었던 것 같아요. 이 프로젝트가 끝나더라도, 늘 피어 사용자에게 귀를 기울이고 꾸준한 관심을 갖는 디자이너로 남고 싶습니다.

늘 긍정적인 응원과 격려로 힘을 준 피어임원진, 그리고 보이지 않는 곳에서 묵묵히 노력해주신 모든 피어 팀원분들, 누구보다 부족한 저를 끌고가느라 고생한 열정의 아이콘, 야채님께 감사의 말씀을 전합니다!



@Yachae

디자인 후기(UX/UI)

와. 이곳에 글을 쓰는 날이 오긴 오네요. 감회가 새롭습니다. 보리님 어깨너머로 눈인사 드렸던 때가 바로 어제 같은데 말이죠.ㅎㅎ 어느덧 피어 프로젝트가 막바지에 이르렀네요. 우선 인사 먼저 드릴래요. 모두 정말 정말 고생 많으셨습니다! 42서울에 대한 제 첫인상을 떠올려보면 “재미있다” 였어요. 동료 학습이라는 독특하면서도 의미 있는 방식으로 성장해나가는 개발 커뮤니티라니..! 그리고 하나의 프로젝트에서 이렇게 많은 개발자들과 함께 할 수 있는 기회가 앞으로 과연 있을까요? 이런 기회를 제안해 주신 보리님과 42서울 개발자분들께 감사드려요. 늦은 시간까지 스타벅스에 앉아서 컴포넌트 요소 하나하나에 열띤 논쟁(?)을 펼치기도 했고, 출석체크를 하면 아기고래가 성장하는 느낌으로 게임스러운 요소를 넣어보면 어떨까 무한한 아이디에이션을 펼치기도 했었죠. 비록 여러 가지 제한 속에서 포기하거나 더 나아가지 못하고 만족해야 하는 부분들도 있었지만, 서비스 론칭은 곧 또 새로운 시작이기도 하잖아요? 서비스가 실제로 운영되면서 성장해나갈 부분일 테니 아쉬움보다는 기대가 큩니다. 실과 바늘이 서로에게 꼭 필요한 존재이듯, 제 실과 바늘이 되어주는 디자인 메이트 보람님, 이미 서로의 실과 바늘이 되어주고 있는 멋진 본보기를 가진 42서울 식구들, 개발백서가 없다면 누가 우리의 과정을 알아줄까요? 개발백서를 이끌어주시는 조하님까지 모두에게 감사한 마음을 전합니다. 우리 피어 흥해라!!!





@Joah



디자인 후기(개발백서 편집)

이번에 개발백서 디자인 및 편집을 맡은 Joah입니다.

처음에 haryu님 소개를 받아서 프로젝트와 개발백서 편집에 대한 이야기를 들은게 엊그제 같은데 벌써 편집이 마무리가 되어가고, 서비스 론칭이 얼마 남지 않았다는게 실감이 나지 않습니다. 편집을 하면서 더욱 실감이 났지만, 기획, 개발, 디자인 모든 파트에서 열심히 최선을 다해주신 42 peer, 디자이너분들께 정말 수고 많으셨다는 이야기 드리고 싶습니다.

지금 생각해보면 이렇게 큰 프로젝트에 편집 디자인으로 참여했다는 것은 저에게 달고도 쓴 경험이었던 것 같습니다. 큰 프로젝트의 일원이 되었다는 설렘은 잠깐, 1차 시안을 열심히 잡는 도중 레퍼런스 미팅을 급히 끼워넣기도 했었고, 생각보다 많은 분량에 죄송하게도 약속된 시간을 지키지 못하기도 했습니다. 재밌지만 어려웠던... 정말 디자인 경험치가 고통을 통해 올라가는 느낌이랄까요...

특히 가장 어려웠던 건 메인표지 디자인이었습니다. 초반엔 제 디자인적(?) 욕심을 많이 부렸습니다. 하지만 레퍼런스 미팅을 진행하면서 운영진분들께 피어와 맡은 파트의 역할을 들었을 때 생각이 많이 바뀌었습니다. 가장 중요한 것은 표지 디자인을 통해 피어의 가치와, 분위기가 얼마나 잘 전달되느냐 라는 것을 뒤늦게 깨닫게 되었던 것 같습니다. 이번 개발백서를 통해 피어분들의 수고와, 가치가 보는 이들에게 잘 전달되었으면 하는 바랍니다.

디자인을 진행하며 부족한 모습들이 많았지만 다들 기다려주셔서 감사했습니다. 그리고 함께 참여할 수 있어서 영광이었습니다! 편집을 진행하면서 여러분들의 열정과 노력들이 그대로 느껴져 신기하고 감사했습니다! 더 많은 사용자들께 사랑받는 피어 서비스가 되길 응원하겠습니다! 다들 수고 많으셨습니다!

THANK YOU

FOR YOUR CONTRIBUTION

피어 총 책임자	jujeon 전준성	개발 총 책임자	haryu 류한솔
프론트엔드 리더	hoslim 임호성	프론트엔드 리더	hyna 나 현
데브옵스 매니저	hyeongki 김형찬	백엔드 리더 및 팀원	jwee 위지혜

프론트 엔드 개발을 진심으로 감사드립니다!

프론트엔드 팀원	hyeokim2 김현지	프론트엔드 팀원	woorikim 김우림
프론트엔드 팀원	hyunjung 정현섭	프론트엔드 팀원	jeyoon 윤정연
(전) 프론트엔드 팀원	hujeong 정희호		

백엔드 개발을 진심으로 감사드립니다!

전 백엔드 리더	yonghlee 이용훈	백 엔드 팀원	junssong 송준상
백 엔드 팀원	juhyelee 이주현	백 엔드 팀원	wochae 채우석
(전) 백엔드 팀원	zekim 김정준	(전) 백엔드 팀원	san 안소현

디자인 작업을 해주셔서 진심으로 감사드립니다!

디자이너	Bori 이보람	디자이너	Yachae 양채윤
디자이너	Joah 조하연		

개발 조언으로 참여해주심에 진심으로 감사드립니다!

Datus 우광명 이사

여러분의 모든 마음이, 노력이 피어를 만들어 냈습니다.

FRONT, BACK, DESIGN

피어 개발 백서

발행 | 2024년 2월 5일

발행 기획 | 피어 개발자 리더단

구성 | 류한솔

편집 | 조하연

디자인 | 조하연

www.peer-study.co.kr / 42peer@gmail.com

본 콘텐츠의 저작권은 42peer에게 있습니다. 무단으로 복제하거나 공유 및 배포할 경우 법적 조치의 대상이 될 수 있습니다. 개인적인 용도로만 사용 바랍니다.

