



지역 광주

팀명 GuGu

이름 경주원, 김정인, 이서현, 이윤성



개인목표

이윤성

목표 : 재사용 로직을 작성해 효율적인 코드를 작성하는 개발자 되기

현재 상태	도착 상태
<ul style="list-style-type: none">구현에 시급해 중복되는 UI 컴포넌트를 작성하는 상태	<ul style="list-style-type: none">컴포넌트를 재사용 할 수 있도록 설계, 구현할 수 있는 상태
<ul style="list-style-type: none">api요청을 일일히 적음. 비슷한 메서드도 겹쳐 작성하는 상태	<ul style="list-style-type: none">custom hook을 사용해 api 요청 로직을 한번의 작성으로 손쉽게 사용가능한 상태
<ul style="list-style-type: none">비슷한 일을 하는 이벤트 핸들러를 컴포넌트 별로 작성	<ul style="list-style-type: none">custom hook을 이용해 동일한 작업을 하는 핸들러의 중복된 로직 감소

목표로 잡은 이유

- 중복된 코드를 항상 작성하므로 코드가 중복되어 중복되지 않을 때보다의 오버헤드가 소요됨
- 또한 코드가 필요 이상으로 길어지고, 중복되는 코드를 작성하는 무의미한 시간을 보내는 일이 많음

도착 상태가 되기 위한 계획

- custom hook을 사용하기 위해서는 리액트 라이프사이클과 useEffect, useState에 대해 완전히 이해해야 함
 - 남에게 설명할 수 있을 정도로 이해하기
- 어떻게 하면 재사용을 할 수 있는지 구글링해 공부하는 시간을 주 2시간씩 가짐
 - 공부한 내용은 Notion에 정리
- 기존에 진행했던 프로젝트에서 재사용이 가능한 부분을 고민해보는 연습
 - 이전 프로젝트에서 컴포넌트, 로직을 재사용할 수 있도록 짜는 연습을 프로젝트 개발과 병행
 - 1주마다 하루를 잡아 약 4시간정도 집중해 진행
 - 1시간은 UI 컴포넌트, 1시간은 api 요청 로직, 1시간은 재사용 로직에 관한 고민을 해볼 예정
 - 나머지 한시간은 스스로에게 질문을 남겨 해결하는 시간 가지기

개인목표

이서현

목표 : 모든 코드 한줄 한줄에 의도가 있는 개발자

현재 상태	도착 상태
<ul style="list-style-type: none">모든 비즈니스 로직에 공통 처리 미숙함 (에러 핸들링)	<ul style="list-style-type: none">데이터에 국한되지 않고 사진, 영상, 음악 등 다양한 데이터를 다루는 개발
<ul style="list-style-type: none">Filter, Interceptor 를 활용하여 커스텀 개발	<ul style="list-style-type: none">구현과 테스트 코드 작성의 나만의 루틴 만들고 체득하기 (나만의 리듬 만들기)
<ul style="list-style-type: none">Swagger 문서화 작성	<ul style="list-style-type: none">MSA 형태의 다중 서버 개발 경험 갖추기

목표로 잡은 이유

- 테스트 코드 작성을 통해 유지보수에 용이한 코드를 작성하기 위함
- 통일되지 못한 반환이나 예외처리 때문에 트러블 슈팅에 많은 시간이 소요됨
- 다양한 상황을 직면할 때마다, 깊은 고민 없이 해결하는 모습을 자주 확인함

목표를 향한 나의 계획

- 매 기능을 구현하기 전에 테스트 체크리스트 작성하기
- 메시지 큐 프레임워크를 이해하고 설명할 수 있게 학습하기 (~1 / 14일까지)
- 시스템 설계할 때 예외 처리 시스템도 같이 설계하기
 - (공통 예외 포맷 작성, 및 장애 전파 시스템 설계해보기)
- 기존에 진행했던 3개 프로젝트를 매주 하루를 선택해서 테스트 코드를 작성하고 리펙토링하기

개인목표

경주원

목표 : 학습한 내용을 적용하고 기록하며 나만의 레퍼런스를 가진 개발자

현재 상태	도착 상태
<ul style="list-style-type: none">리액트를 사용할 때 성능 최적화를 고려하지 않고 개발함	<ul style="list-style-type: none">리액트의 성능 최적화를 고려해 개발한다
<ul style="list-style-type: none">개발 과정에서 생긴 이슈를 극복하며 얻은 지식을 기록하지 않아 휘발됨	<ul style="list-style-type: none">프로젝트를 진행하며 생긴 이슈와 해결 과정을 글로 기록해 나만의 레퍼런스를 만든다

목표로 잡은 이유

- 사용자에게 좋은 성능의 서비스를 제공하기 위해서는 단순 구현이 아니라 프로젝트 시작 단계부터 성능을 고려해 개발해야 함.
- 개발 과정에서 학습한 지식을 흡수하지 못해 반복적인 이슈에 부딪힘.

도착 상태가 되기 위한 계획

리액트 성능 최적화를 고려해 개발한다.

- 집중 개발 기간인 1월 25일 전까지 리액트 성능 최적화 방법을 학습한다: 리액트 렌더링, 메모이제이션 함수와 Hook, 최적화를 위한 상태관리와 컴포넌트
- 성능 최적화를 고려하며 개발한다
 - 이슈 단위로 분리된 PR을 보내기 전에 학습한 내용을 바탕으로 성능 최적화를 적용해 본다.
 - 크롬 개발자 도구의 React Developer Tools를 활용해, 이벤트 발생에 따른 컴포넌트 렌더링을 추적해 성능을 측정해 비교한다.
 - 성능 최적화 과정과 최적화 전후 성능을 기록한다.

여러 레퍼런스를 활용해 필요한 지식을 학습하고 나만의 글로 기록한다.

- 개발 과정에서 부딪히는 이슈를 해결할 때마다 노션에 해당 이슈와 참고한 레퍼런스, 해결 방법을 간결히 기록한다.
- 매주 월요일 2시간씩 개인 이슈 회고를 진행한다.
 - 이슈 회고는 이슈 유형, 참고한 레퍼런스, 해결 방법, 적용한 코드를 포함한다.
 - 이슈 회고 과정에서 레퍼런스를 다시 한 번 살펴보며, 더 좋은 해결 방법이 있는지 고민해본다.

개인목표

김정인

목표 : 코드를 작성할 때, 동기가 분명한 개발자 되기

현재 상태	도착 상태
<ul style="list-style-type: none">백엔드 개발 프레임워크인 Spring을 자유롭게 사용하지 못함	<ul style="list-style-type: none">Spring에서 제공하는 기능을 사용할 때, 각각의 역할과 사용 동기를 명확하게 설명하기
<ul style="list-style-type: none">프로젝트 개발 시, 단순 CRUD 구현에 그침. 성능 개선에 대한 고민 못함	<ul style="list-style-type: none">개발한 기능의 성능을 테스트할 수 있는 방법을 고민하고, 개선 방법을 고민함
<ul style="list-style-type: none">개발 시간이 많이 걸림. 특히 코드 작성에 시간이 많이 걸림	<ul style="list-style-type: none">하나의 알고리즘을 푸는 시간을 현재 기준 10% 단축 시키기

목표로 잡은 이유

- 성장을 하려면 기능을 구현하는 것에서 그치는 것이 아니라, 개선할 수 있게 충분히 고민하는 것이 필요하다고 생각함
- 코드를 작성하는데 시간을 단축한다면, 다양한 기능을 더 많이 개발할 수 있을 것이라 생각함

도착 상태가 되기 위한 계획

- 개발 범위를 정하고, 구현하는데 걸리는 시간을 측정하여 현재 상황을 확인한다.
- 사용하고 있는 IDE의 단축키를 외우고, 매일 알고리즘 문제 풀이를 하여 코드를 작성한다.
- Spring을 사용하는 이유, 생태계 등장 배경, 사용하고 있는 기능 등장 사용 이유에 대해 정리한다.
 - 스프링 핵심 원리, 스프링 웹 개발 핵심 기술 1편, 스프링 입문을 위한 자바 객체 지향의 원리와 이해를 통해 학습한다.
 - A라는 기능을 다른 사람이 구현한 방식을 참고하여 구현하고, 이후 직접 해본다.
- 구현하는 기능 중 하나를 선택해서 성능 개선을 할 수 있는 방법에 무엇이 있는지 조사하고, 이를 적용한다.
 - 본인이 맡은 채널과 서버, 유저들이 관리하는 부분을 담당하니, 다른 프로젝트들은 어떻게 하는지, 조사하고 팀의 문제 상황과 비교한 후 적용할 수 있는 방법을 찾음
 - 프로젝트 일정에 개발한 기능을 고민하고, 개선하는 시간 추가
 - 현재 방식이 어떤 문제를 초래할 수 있을지 고려함
 - 구현한 기능에 대해 팀원들과 어떤 문제를 일으킬 수 있을지 충분히 토의하는 시간을 갖는다.
 - 구현한 기능에 대해 팀원들과 논의한 내용을 잘 정리하여 멘토님께 적극적으로 질문한다.
 - 프로젝트 진행 과정에서 고민한 것들을 개발 일지에 기록한다.

팀 목표

다른 사람이 의도를 파악할 수 있는 코드 작성

- **Clean code**를 고민해보며 구현하기
 - 코딩 컨벤션을 정의한 후 준수한다. 컨벤션 준수를 확인하기 위해 **Lint**를 설정한다.
 - 각자 개발한 코드를 같은 분야의 팀원이 코드 리뷰를 하고 피드백을 남긴다.



클린 코드란 원하는 로직을 빠르게 찾을 수 있는 코드, 모든 팀원이 이해하기 쉽도록 작성된 코드라고 정의할 수 있습니다.

- 이유가 있는 코드를 작성하는 개발자로 성장하기
 - 한 주마다 이유가 있는 코드를 **체화하기 위한 목표**를 세우고, 주간 회의에서 **개인 목표 회고** 진행
 - 각자 세운 목표를 달성 했는가? 달성 했다면 어떻게 달성 했는가?
 - 만약 달성하지 못했다면 왜 달성하지 못했는가? (어느 부분이 부족 했는가?)
 - **다음 주 목표** 수립하기
 - 서로간의 피드백
 - 전체 목표를 얼마나 달성 했는지 논의 하기

프로젝트 역할 분담

이윤성(팀장)

- 스마일게이트 원터 캠프 운영진과 소통 담당
- 팀 분위기 조성 및 리마인더 관리

이서현 (DevOps 담당)

- 프로젝트 이슈 관리
- DevOps 컨벤션 및 PR, 이슈 컨벤션 추적
- 협업 툴 관리(JIRA, Slack, Trello, ZenHub 등)
- Dependencies 변경할 때마다 문서 담당자에게 전달

김정인 (문서 담당자)

- 회의록 작성
- 회의록 및 노션 규격화
- Notion, README 관리

경주원 (행동대장)

- 회의 주제 공고
- 회의 길라잡이: 회의 주제에 벗어나지 않게 조정
- 팀원들의 업무 진행 파악 및 독려

컨벤션

이슈 컨벤션(issue convention)

- [FE / BE 선택] 이슈 제목
- 내용은 ISSUE_TEMPLATE에 맞춰 작성 (DevOps 가 추후 작성 후 공지 예정)

컨벤션

커밋 컨벤션(commit convention)

- 제목 : 명령조로 작성
 - 끝에 . 을 붙이지 않음
 - 본문은 한 줄 띄어서 작성
 - 끝에 연관된 이슈 번호 붙임(issue Tracker)

- feat : 새로운 기능 추가, 기존의 기능을 요구 사항에 맞추어 수정
- fix : 기능에 대한 버그 수정
- build : 빌드 관련 수정
- chore : 패키지 매니저 수정, 그 외 기타 수정 ex) .gitignore
- docs : 문서(주석) 수정
- style : 코드 스타일, 포맷팅에 대한 수정
- refactor : 기능의 변화가 아닌 코드 리팩터링 ex) 변수 이름 변경
- release : 버전 릴리즈
- merge : 병합

컨벤션

PR 컨벤션(PR convention)

- 제목: [Feat] 핵심적인 부분만 간략하게 명령조로 작성
 - 맨 끝에 . 을 붙이지 않음
 - PR 제목은 300자가 넘지 않도록 주의
- 내용: 개조식으로 작성
- 라벨: FE, BE, FEAT, REFACTOR, LAYOUT, ERROR

깃허브 마일스톤 컨벤션(Github Milestone convention)

- 이슈를 각각의 마일스톤으로 그룹화해서 사용
- 컨벤션은 이슈 컨벤션을 따른다

컨벤션

디렉토리 명명 (directory naming)

- 숫자, 언더스코어(_)로 시작 불가
- 첫 글자는 소문자로 시작
- 단어와 숫자를 조합하는 경우 언더스코어(_) 생략

파일 명명 (file naming)

- 숫자, 언더스코어(_)로 시작 불가
- 첫 글자는 대문자로 시작
- 단어와 숫자를 조합하는 경우 언더스코어(_) 생략

컨벤션

코딩 컨벤션(Coding convention)

- Front-end : Airbnb JavaScript Style Guide를 따른다.
- Back-end : 객체지향 생활체조 원칙을 따른다.

Ground Rule

1. 항상 그날 회의가 끝나면 다음 회의록을 미리 공란으로 만들어 둔다.

- 그날 회의 전까지 각자 회의 주제가 생각났으면, 회의 주제란에 작성한다.
- 이 담당은 문서관리자가 진행한다.

2. 모든 회의는 문서화하며, 회의 성격에 맞는 태그를 작성한다.

3. 회의 때는 항상 존칭을 사용한다.

- ~~님이라는 존칭을 붙이고, ~요체나, ~다체를 사용한다.
- 별칭과 원래 이름을 혼용하여 취향껏 사용한다.

4. 프로젝트 진행에 필요한 역할을 두고, 각각의 역할을 성실히 수행한다.

- 해당 역할은 역할 분담 참고

5. 각각의 역할 진행 중, 업무량이 많으면 분배를 건의할 수 있다.

Ground Rule

6. 회의 시작은 일주일 근황으로 어떤 공부를 했는지 서로 이야기 하는 시간을 가진다.

- 일어서서 하자.

7. 하고 싶은 사람이 있다면, 먼저 슬랙을 통해 발표할 내용을 공지한다.

- 발표 내용은 개발, IT 분야에 한해서 준비한다.
- 발표 내용, 발표 예상 소요 시간을 간략히 슬랙 gugu 채널에 기재한다.
- 회의 시작 전에, 발표자는 발표를 한다.
- 최대 5분의 질의 응답을 가진다.
- 발표자가 있으면 협의에 의해 6번은 생략한다.

Ground Rule

8. 새로운 기술을 도입할 때 제안자는 그 기술의 필요성과 효용을 팀원들에게 납득 시켜야 한다.

9. 틀린 제안이라고 생각이 들어도 비난하지 말자.

- 틀린 제안이라고 생각이 들면, 그 사람이 그렇게 생각한 이유를 들어보자.
- 설령 이유가 잘못되더라도, 논리적으로 생각해보자.
- 틀린 제안에 피드백을 주더라도, 서로 슬퍼하지 말자.
- 나에 대한 공격으로 받아들이지 말자.
- 다양한 의견은 나올수록 좋다. (맨땅에 헤딩)

10. 회의는 대면이 기본원칙이고, 특수한 경우가 발생할 경우, 팀원들의 동의를 구해 회의 방식을 변경한다. (당일 불가)

- 회의 시간도 마찬가지다.

Ground Rule

11. 감정적으로 서운한 부분이 있다면 담아두지 말고 바로 소통하자.

- 남을 이해하는 마음을 기르고 포용하려고 노력하는 마음가짐을 가지자.

12. 그라운드 룰 개선 제안이나, 역할 분담 개선 제안은 팀원들과 협의를 통해 개정 한다.

13. 개발 일지는 이서현 → 이윤성 → 경주원 → 김정인 순서로 돌아가며 작성 후, 일요일에 제출한다.

14. 매일 포커스 타임을 가지자.

1. 매일 팀원 일정을 공유하기 어려우니, 와카타임이라는 프로젝트 러닝타임을 측정하는 어플리케이션을 이용한다.
2. 매일 일일 스프린트를 Slack으로 공유한다. #gu-gu-일일-스프린트 채널에 공유한다.

추후 개발 계획

김정인 (BE)

01/04 ~ 01/10

- Discord 기능 면세 검토, 구체화
- 스프링 기능 학습

01/11 ~ 01/17

- 서버 관리 기능 개발
- 채널 관리 기능 개발

01/18 ~ 01/25

- 유저 관리 개발
- 유저 역할 설정 개발
- DB 성능 최적화를 위한 학습

01/26 ~ 02/01

- 이모지 개발

02/02 ~ 02/14

- 부족한 기능 보완

02/15 ~ 02/20

- 리팩터링

추후 개발 계획

이서현 (BE)

01/04 ~ 01/10

- Kafka 관련 스트림 자료 조사
- Discord 기능 명세 검토, 구체화

01/11 ~ 01/17

- CI / CD 스크립트 작성 및 개발
- MSA 서버 개발

01/18 ~ 01/25

- 영상 통화 기능 개발
- 음성 통화 기능 개발

01/26 ~ 02/01

- 채팅 서버 개발
- 시그널링 서버 개발

02/02 ~ 02/14

- 리팩토링
- 쿼리 최적화
- 오류 전파 방지 서버 개발

02/15 ~ 02/20

- 부족한 기능 보완

추후 개발 계획

경주원 (FE)

01/04 ~ 01/10

- WebRTC, Socket IO 등 사전 기술 조사 기간
- Discord 기능 명세 검토, 구체화
- 성능 최적화를 위한 리액트 렌더링 학습

01/26 ~ 02/01

- 이모지 개발

02/02 ~ 02/14

- 부족한 기능 보완

01/11 ~ 01/17

- 유저 관리 개발
- 유저 역할 설정 개발
- 성능 최적화를 위한 메모이제이션 함수와 흑
학습

02/15 ~ 02/20

- 리팩터링

01/18 ~ 01/25

- 음성통화 개발
- 채팅 개발
- 화면 공유 개발
- 성능 최적화를 위한 상태관리와 컴포넌트 학습

추후 개발 계획

이윤성 (FE)

01/04 ~ 01/10

- WebRTC, Socket IO 등 사전 기술 조사 기간
- Discord 기능 명세 검토, 구체화

01/26 ~ 02/01

- 이모지 삽입 기능 개발
- 알림 기능 개발

01/11 ~ 01/17

- 채널 관리 기능 개발
- 사용자 설정 부분 개발
- 서버 설정 부분 개발

02/02 ~ 02/14

- 부족한 기능 보완

01/18 ~ 01/25

- 영상통화 기능 개발
- 채팅 기능 개발

02/15 ~ 02/20

- 리팩터링