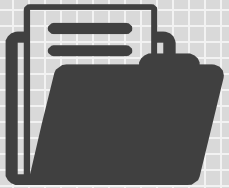
2022 SmileGate Winter :// Dev.Camp



Architecture

Team Ottogi

Architecture Review

FRONT END : 김현우 허다은
BACK END : 김수찬 박규현 백종인

Architecture

FE

BE

Q & A

Discord Clone Coding

Discord ?

- ✓ 음성, 채팅, 화상통화 등을 지원하는 인스턴트 메신저
- ✓ 대한민국에서는 주로 온라인 게임을 즐기는 사람들이 많이 이용하는 편이며, 게임용 메신저의 대명사.
  - ***실시간 채팅***
  - ***실시간 음성***
  - ***개별 서버 단위의 메신저***

주제 선정 이유

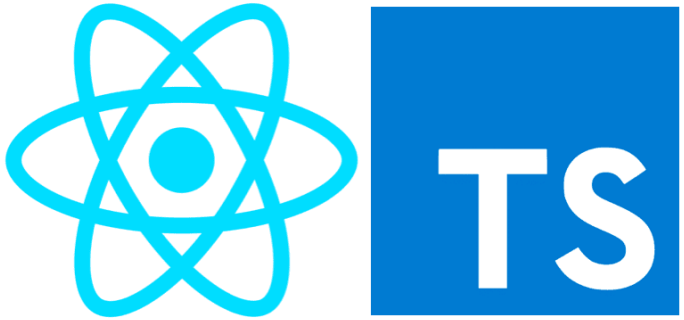
- 팀원들이 평소에도 대부분 사용해본 서비스로 이용 경험이 다수 있다.

- 채팅서버, 시그널링 서버, 알림 서버, 인증 서버, 상태관리 서버 등 기능에 있어 복합적으로 이루어져 있고, 다양한 기능의 구현을 통해 성장을 원하는 팀원들의 니즈에 적합하다.

- 채팅, 음성, 화상대화 등 각 기능을 구현 하는데 있어서 팀원들이 기존에 접해보지 못했던 아키텍처와 기술 스택들이기 때문에 도전 목표에 적합하다.

Front End

React TS



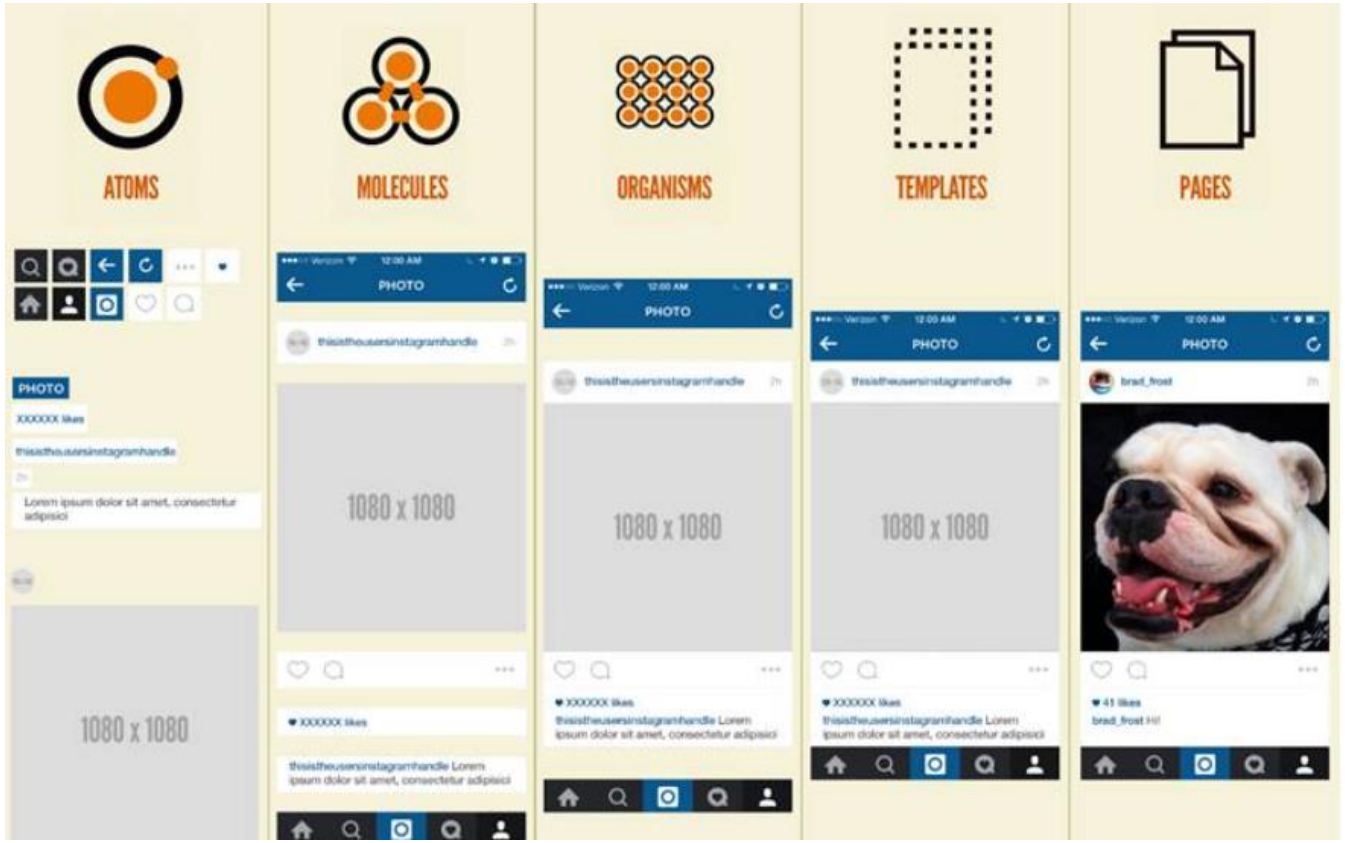
- 선택 이유
  - React + Ts로 많이 씀, 추천 많이 함
  - 프론트 맡은 두 명 모두 이 조합의 프로젝트 경험 X

React TS

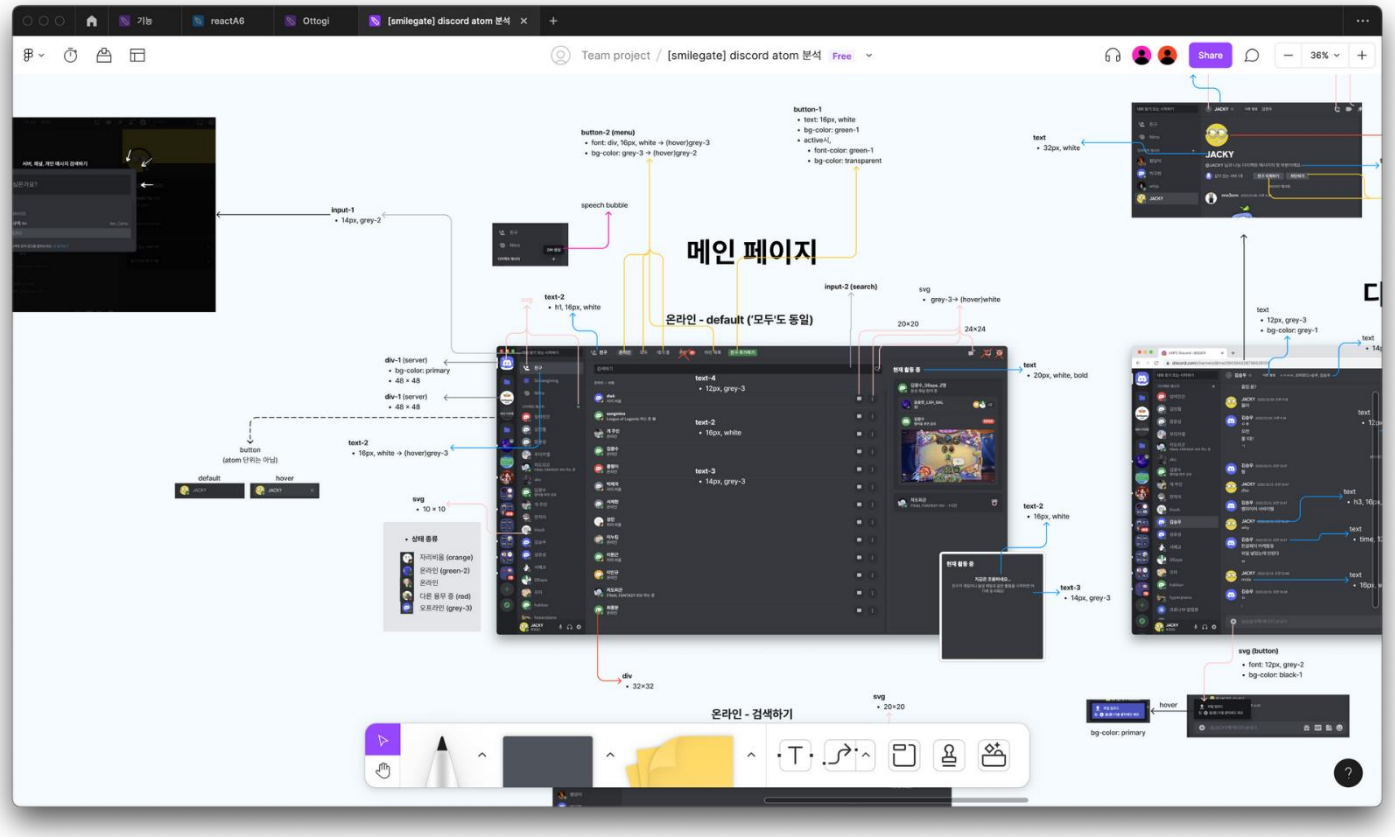
- Typescript의 장점
  - 버그 예방
  - 더 나은 개발자 경험과 코드 퀄리티 향상
  - 크로스 브라우징 문제 해결
  - 실시간으로 에러를 잡아준다.
  - 규모가 커져도 안전하다

Atomic Design Pattern

Atomic Design Pattern



Atomic Design Pattern : Discord 분석



Atomic Design Pattern

1안)

/components
- atoms
  - A
  - B
- molecules
  - A
  - B
- organisms
  - A
  - B
- templates
  - A
  - B

Atomic → Group

VS

2안)

/components
- A
  - atoms
  - molecules
  - organisms
  - templates
- B
  - atoms
  - molecules
  - organisms
  - templates

Group → Atomic

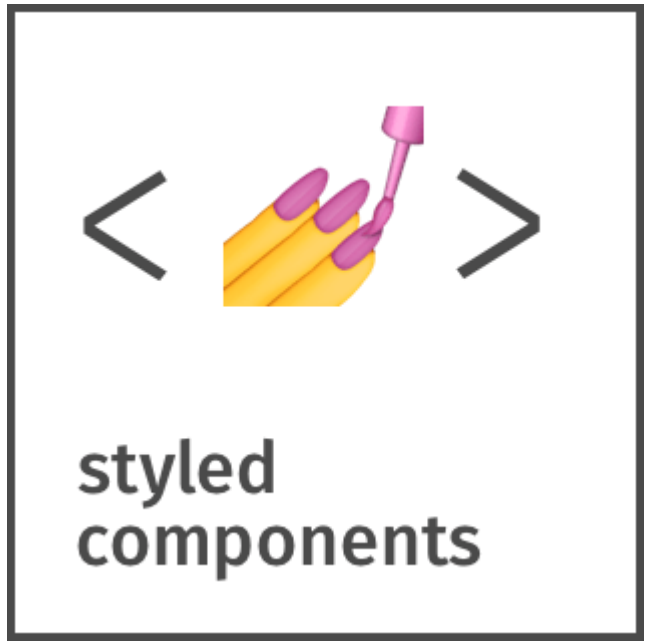
Storybook

UI를 컴포넌트 단위로 테스팅 가능한 툴

Storybook

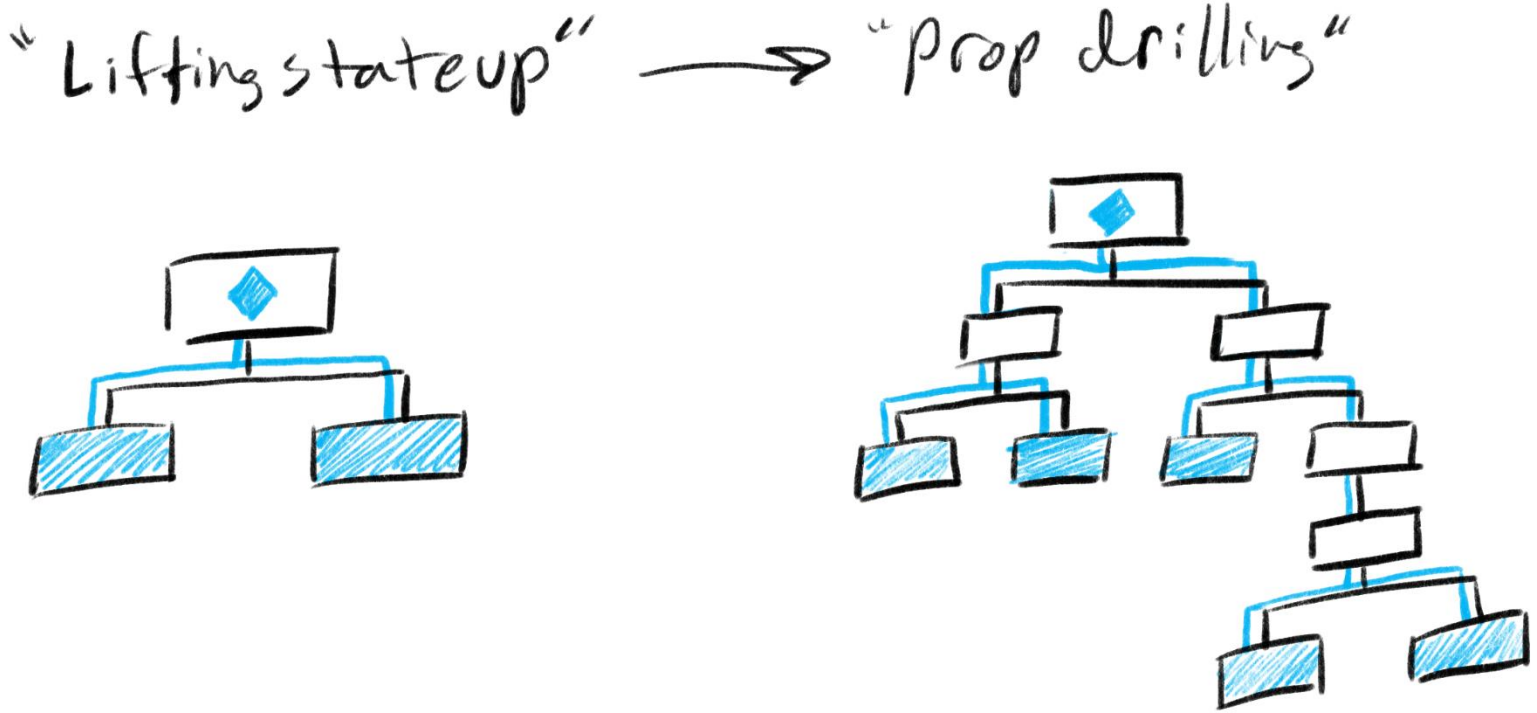
- 컴포넌트 단위 별 작업 + 테스트 가능
- UI 자동 문서화 + MDX 작성을 통한 커스텀 가능
- storybook 배포(Chromatic), 쉬운 배포&쉬운 공유

Styled-components



- Styled-components를 통해 CSS-in-JS 방식 사용
  - Props를 활용한 조건부 스타일링 가능
  - 짧은 길이의 유니크한 클래스를 통해 코드 경량화
  - CSS의 컴포넌트화로 스타일시트의 파일 유지보수 필요 X(모듈성)

상태관리 라이브러리 : 필요한 이유



상태관리 라이브러리

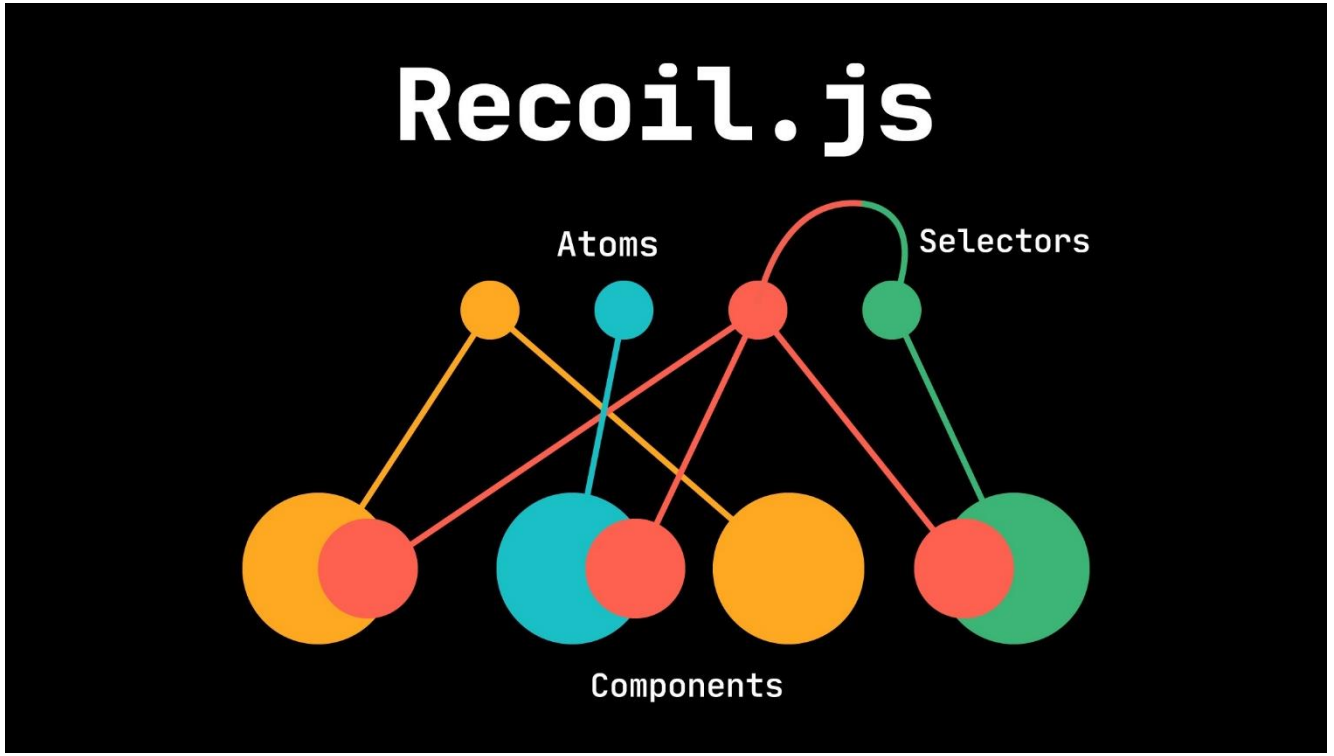
상태관리 라이브러리 : 고려할 요소들

사용해보지 않은
라이브러리 경험

비동기 상태 관리
용이 여부

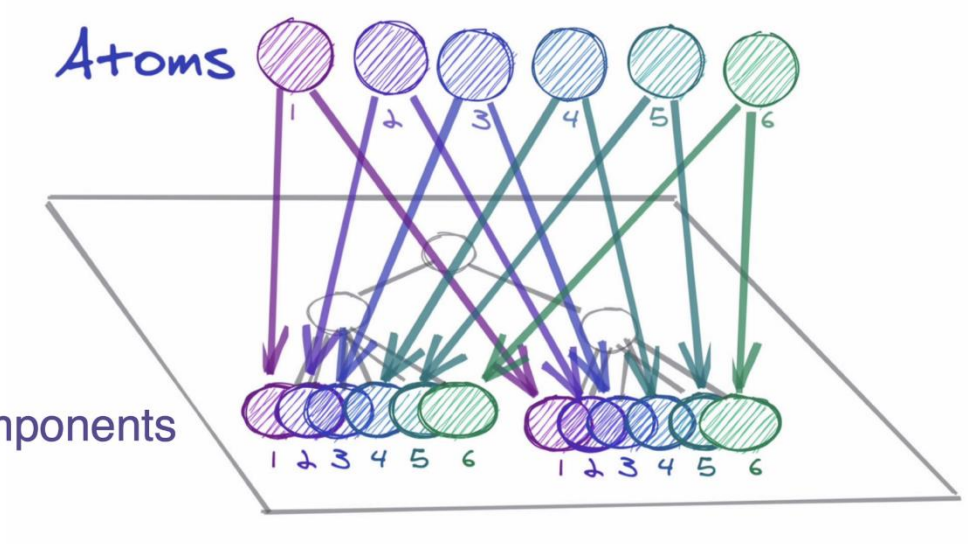
Atomic Design Pat
tern 과의 연결성

상태관리 라이브러리



상태관리 라이브러리

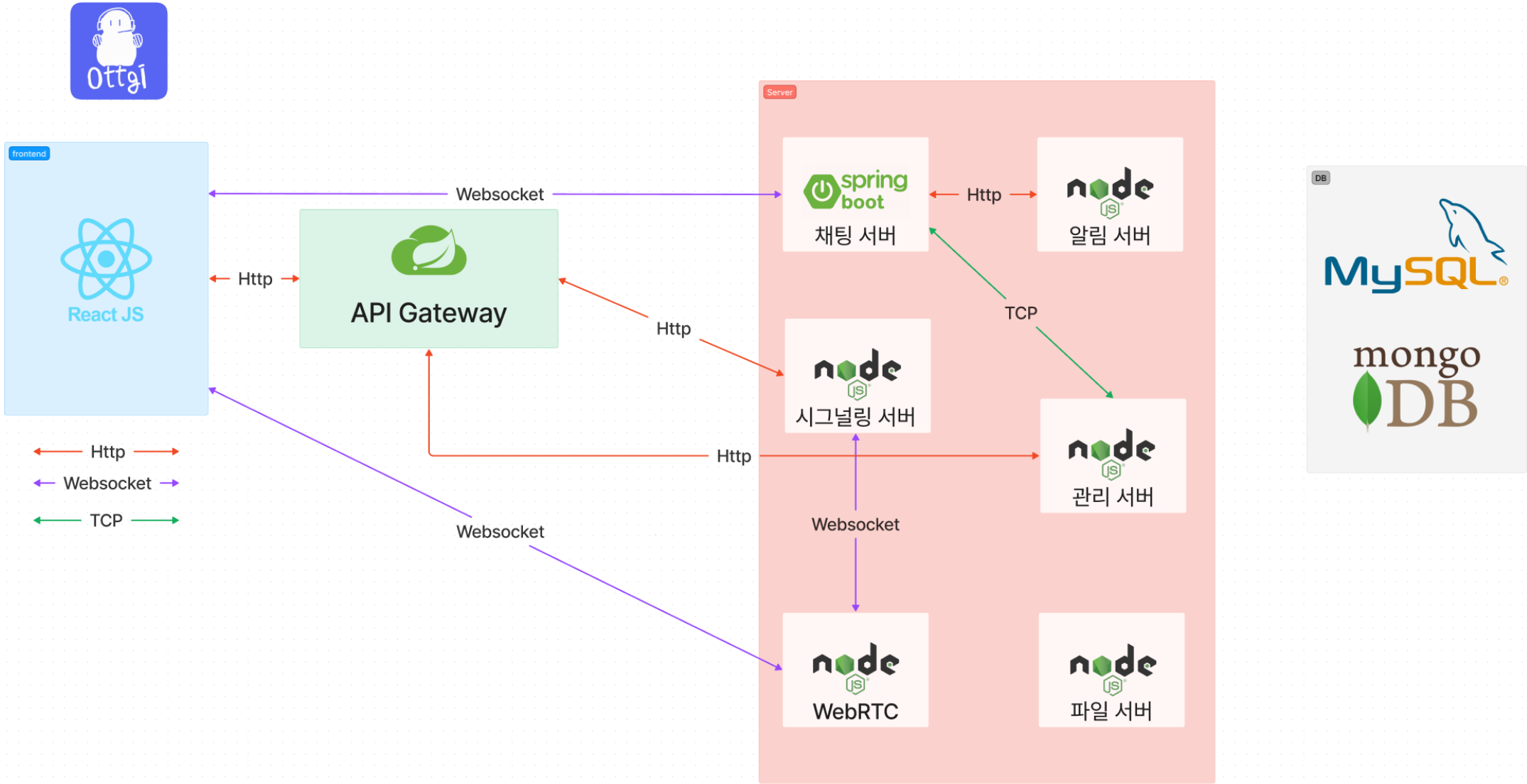
recoil



- 비동기 문제를 깔끔하게 처리할 수 있다.
  - 캐싱을 통한 빠른 비동기 데이터 처리
- 단순하다(atom, selector가 전부)
- 상태를 분산적으로 둘 수 있어 코드 스플리팅 가능

Back End

BE


Ottgi
frontend
React JS
Websocket
Http
API Gateway
Server
spring boot
채팅 서버
Http
node
알림 서버
TCP
Http
node
시그널링 서버
Http
node
관리 서버
Websocket
Websocket
node
WebRTC
node
파일 서버
DB
MySQL
mongo
DB
Http
Websocket
TCP

Servers 김수찬

1. WebRTC

오디오, 비디오, 화면을 Streaming 할 수 있도록 기기의 권한을 받아오도록 작업(TURN) IP주소, 포트, 네트워크 데이터 받아 와야함

- MediaStream 사용자의 카메라 마이크 input 기기의 Stream에 접근

- RTCPeerConnection 암호화, 대역폭 관리, 오디오 비디오 연결

- RTCDataChannel : 2번으로 연결 후, DataChannel 연결을 받아옴

2. 시그널링 서버

서버 - 클라이언트 간의 연결을 SFU(Selective Forwarding Unit) 모듈을 이용하여 각 Peer 간 연결을 할당

- 클라이언트와 서버 간 연결 확인

- 외부 클라이언트의 IP주소 및 포트 확인

- ICE를 통한 외부 클라이언트의 기기 권한 접근

Servers

박규현

1. 알림 서버

- 알림을 보내는 목적
- FCM 사용 예정
  - Why? : 여러 플랫폼 사용 가능

2. 관리 서버

- 중간 지점에서 데이터를 관리
- 가공하여 프론트에 전달할 목적
- MongoDB, Redis 사용 예정

3. 파일 서버

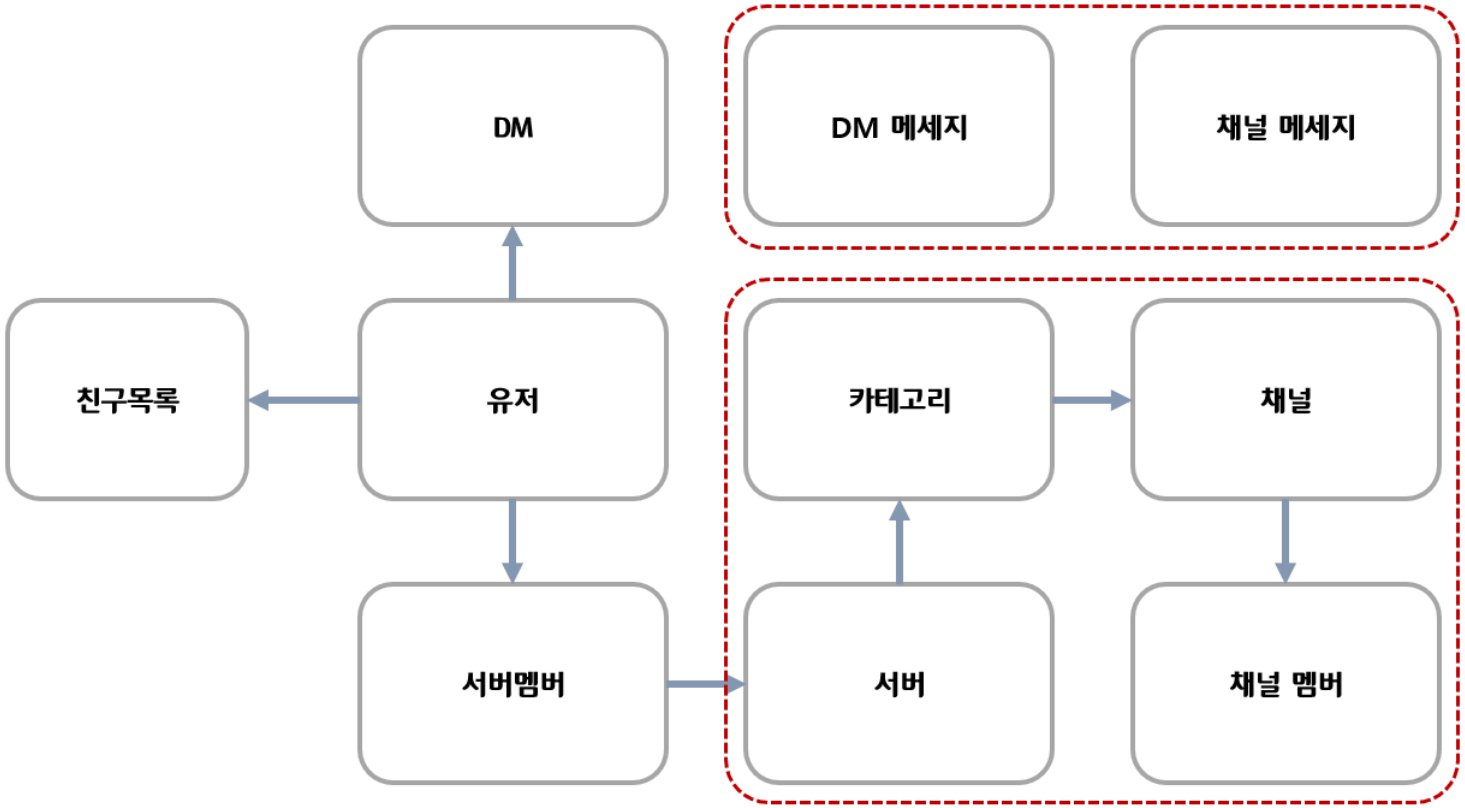
- 파일(pdf, 이미지, 동영상)등을 관리 할 목적
- Multer 모듈을 사용할 예정

Servers 백종인

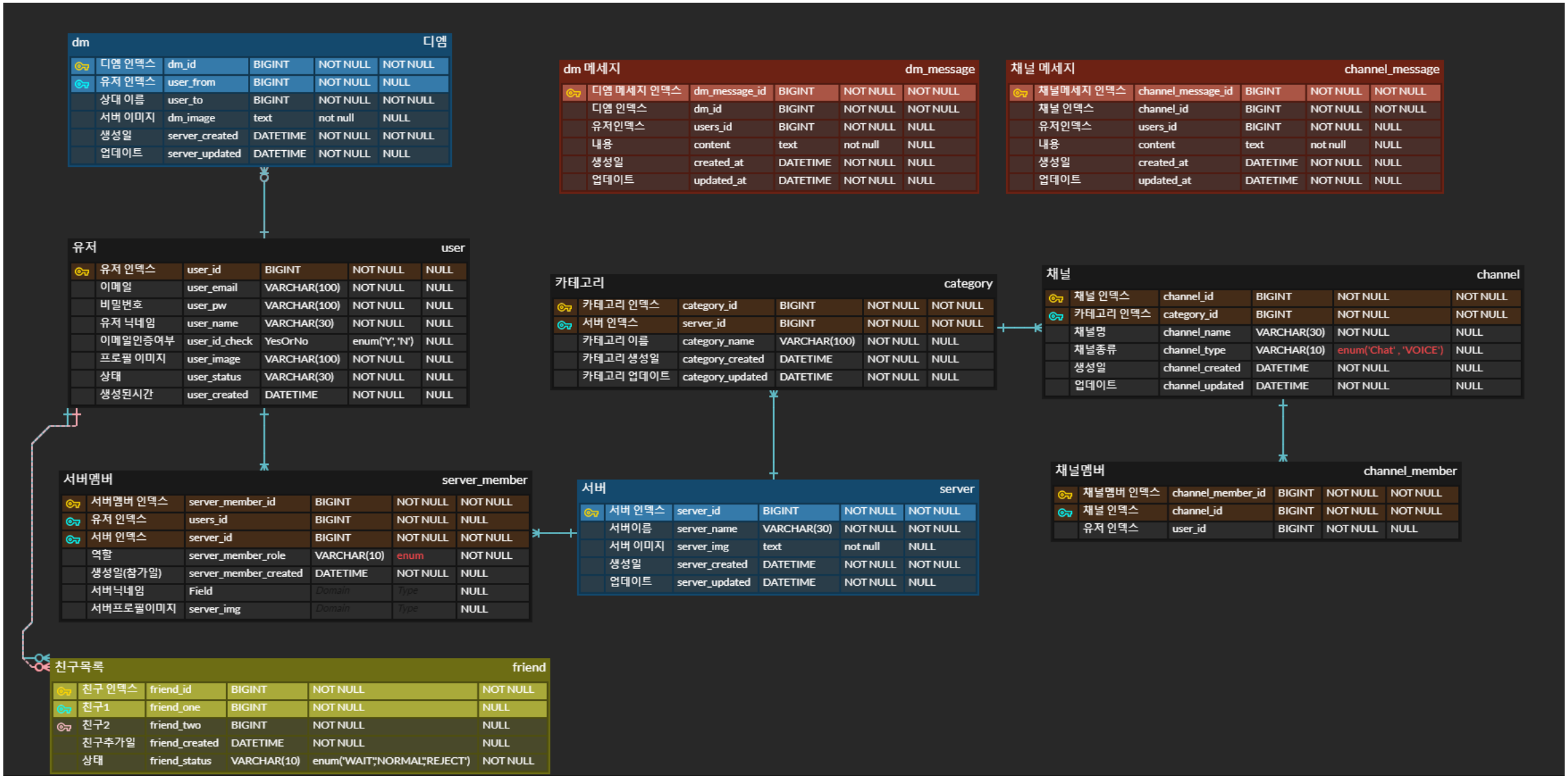
1. **API Gateway (Spring Cloud)**
   - Spring Cloud 사용
   - 각 msa서버로 연결 해주는데 사용
   - 서비스로 보내기 전, 공통 인증 (필터)에 사용

2. **채팅 서버 (Spring Boot)**
   - 채팅 메시지 발행해주는 서버
   - Spring STOMP 사용
   - Pub / sub 방식으로 구성

DB (MySQL)



BE


dm 디엠
디엠 인덱스 dm_id BIGINT NOT NULL NOT NULL
유저 인덱스 user_from BIGINT NOT NULL NULL
상대 이름 user_to BIGINT NOT NULL NOT NULL
서버 이미지 dm_image text not null NULL
생성일 server_created DATETIME NOT NULL NOT NULL
업데이트 server_updated DATETIME NOT NULL NULL
dm 메세지 dm_message
디엠 메세지 인덱스 dm_message_id BIGINT NOT NULL NOT NULL
디엠 인덱스 dm_id BIGINT NOT NULL NOT NULL
유저인덱스 users_id BIGINT NOT NULL NULL
내용 content text not null NULL
생성일 created_at DATETIME NOT NULL NULL
업데이트 updated_at DATETIME NOT NULL NULL
채널 메세지 channel_message
채널메세지 인덱스 channel_message_id BIGINT NOT NULL NOT NULL
채널 인덱스 channel_id BIGINT NOT NULL NOT NULL
유저인덱스 users_id BIGINT NOT NULL NULL
내용 content text not null NULL
생성일 created_at DATETIME NOT NULL NULL
업데이트 updated_at DATETIME NOT NULL NULL
유저 user
유저 인덱스 user_id BIGINT NOT NULL NULL
이메일 user_email VARCHAR(100) NOT NULL NULL
비밀번호 user_pw VARCHAR(100) NOT NULL NULL
유저 닉네임 user_name VARCHAR(30) NOT NULL NULL
이메일인증여부 user_id_check YesOrNo enum('Y','N') NULL
프로필 이미지 user_image VARCHAR(100) NOT NULL NULL
상태 user_status VARCHAR(30) NOT NULL NULL
생성된시간 user_created DATETIME NOT NULL NULL
카테고리 category
카테고리 인덱스 category_id BIGINT NOT NULL NOT NULL
서버 인덱스 server_id BIGINT NOT NULL NOT NULL
카테고리 이름 category_name VARCHAR(100) NOT NULL NULL
카테고리 생성일 category_created DATETIME NOT NULL NULL
카테고리 업데이트 category_updated DATETIME NOT NULL NULL
채널 channel
채널 인덱스 channel_id BIGINT NOT NULL NOT NULL
카테고리 인덱스 category_id BIGINT NOT NULL NOT NULL
채널명 channel_name VARCHAR(30) NOT NULL NULL
채널종류 channel_type VARCHAR(10) enum('Chat' , 'VOICE') NULL
생성일 channel_created DATETIME NOT NULL NULL
업데이트 channel_updated DATETIME NOT NULL NULL
서버멤버 server_member
서버멤버 인덱스 server_member_id BIGINT NOT NULL NOT NULL
유저 인덱스 users_id BIGINT NOT NULL NULL
서버 인덱스 server_id BIGINT NOT NULL NOT NULL
역할 server_member_role VARCHAR(10) enum NOT NULL
생성일(참가일) server_member_created DATETIME NOT NULL NULL
서버닉네임 Field Domain Type NULL
서버프로필이미지 server_img Domain Type NULL
서버 server
서버 인덱스 server_id BIGINT NOT NULL NOT NULL
서버이름 server_name VARCHAR(30) NOT NULL NOT NULL
서버 이미지 server_img text not null NULL
생성일 server_created DATETIME NOT NULL NOT NULL
업데이트 server_updated DATETIME NOT NULL NULL
채널멤버 channel_member
채널멤버 인덱스 channel_member_id BIGINT NOT NULL NOT NULL
채널 인덱스 channel_id BIGINT NOT NULL NOT NULL
유저 인덱스 user_id BIGINT NOT NULL NULL
친구목록 friend
친구 인덱스 friend_id BIGINT NOT NULL NOT NULL
친구1 friend_one BIGINT NOT NULL NULL
친구2 friend_two BIGINT NOT NULL NULL
친구추가일 friend_created DATETIME NOT NULL NULL
상태 friend_status VARCHAR(10) enum('WAIT','NORMAL','REJECT') NOT NULL

Q & A

Q & A (BE)

- ✓ **알림 서버를 넣어야 할까요?**
  (알림을 단독으로 다루는 서버를 구현 해야 하는건지?
  채팅서버나, 프론트에서 다룰 수 있을 거 같고, 뭐가 효율적이고 맞는지 잘 모르겠습니다 😥)

- ✓ **제가 생각한 관리 서버가 (데이터를 가공, 관리하는) 필요한 것이 맞을까요?**
  팀의 결론과 저의 결론은 필요할 것 같다..?! 로 나왔는데, 의견을 듣고 싶습니다! 😢

- ✓ **관리 서버를 구현하면 따로 DB를 사용해야 할까요?**
  Smooth Team 을 참고해보니 따로 DB를 사용하여 여쭤봅니다!

- ✓ **알림 서버가 필요하다면 업데이트된 메시지를 다루는 DB를 따로 만들어야 할까요?**

- ✓ **msa간 서버 통신을 어떤 방식으로 하면 좋을까요?**
  Tcp 통신방식도 있던데 이는 어떤식으로 통신하는 건지 잘 모르겠습니다!

Q & A (BE)

- ✓ **Streaming 할 때 NO SQL을 사용해야 할까요?**
- ✓ **채팅 메시지는 어떤 DB에 저장하는 게 유리한가요?**
- ✓ **채팅 메시지를 삭제하면 실제 DB 에서는 삭제 안하는 게 맞을까요?**
  상태로 삭제했다는 flag 를 두고 저장 해두는 게 맞는지
- ✓ **채팅 DB에 어떤 정보들을 저장하면 좋을까요?**
  사용자마다 불러오는 내역이 다를텐데, 어떤식으로 구성하면 좋을지 잘 모르겠습니다!

Q & A (FE)

- ✓ Storybook에서 제공하는 테스트만 사용해도 충분할지?
- ✓ OT때 언급해 주셨던 JEST 정도까지 사용을 해야 할지?
- ✓ Redux를 사용하는 것이 좋을까요?
- ✓ 상태관리 라이브러리를 통해서 props drilling issue는 해결 되는 게 맞을까요?
- ✓ Atomic Design Pattern 적용이 괜찮을지?
  적용할 때 5단계로 나누는 기준이 애매하다는 문제는 어떻게 푸는게 좋을지