SGS – Smooth

Discord

Back – 박병찬
Back – 김희동
Front – 김민지
iOS – 김두리

Smooth하게 소개하겠습니다!

박병찬

Backend

김희동

Backend

김민지

Frontend

김두리

iOS

CONTENTS INDEX

CHAPTER.1

목표 설정

개인 목표와 팀 목표

박병찬 개인 목표

MSA가 뭐예요?
그냥 ~ 해봤어요

모놀리식 구조로만 개발 해 본 상태이다.
MSA 구조는 이론만 알고있다.

이해 없이 간단한 경험을 한 상태로, 여러가지 기술스택을 사용했다.

현재 상태

MSA 구조를
이해하고 설계하기!

MSA 구조의 아키텍처를 이해하고 분산된 서비스에서 각각의 api들이 원활히 작동하도록 설계할 수 있는 상태이다.

기술 스택에 작동 원리를 파악하며 정확한 이유를 가지고 사용할 수 있는 상태이다.

목표

꾸준함이 힘!
습관을 들이자

기존 기술 스택의 작동 원리를 제대로 파악하여 정리하는 습관을 들인다.

MSA 구조에 대한 책을 통해 전체적인 구조를 이해한다. (DDD Start!)

새로운 기술 스택을 접할 때마다 작동 원리부터 이해하는 습관을 들인다.

상세 계획

김희동 개인 목표

기능 구현이 우선
성능? 그런거 몰라요 ”

대용량 서비스를 고려해서 설계 및 개발해본 경험이 없다.

만들어야 하는 기능에 대해 고민 및 이유 없이 만드는 습관을 가졌다.

성능 측정 및 기능 개선에 대한 소홀함이 있다.

현재 상태

대용량 서비스 개발하기!
성능 측정 및 개선하기! ”

대용량 서비스를 고려한 설계 및 개발 능력을 갖춘 상태이다.

사용한 기술 및 방법에 대해 이유를 설명할 수 있는 상태이다.

성능 측정 및 기능을 개선하는 습관을 들인다.

목표

이해하고 적용하기!
기능 개발 후 성능 측정하기! ”

책을 통해 사전 배경지식을 이해한다.
대규모 시스템 설계 기초 – 1회독

서비스가 분산될 때를 고려한다. → MSA를 이해하고 적용한다.

기능을 개발한 이후에는 성능 측정 도구를 활용하여 성능을 측정한다. (측정 도구 : nGrinder)

상세 계획

김민지 개인 목표

소규모 프로젝트 경험이 부족해요

대용량 서비스 프로젝트에 참여한 경험이 없다.
소규모 프로젝트만 참여했다.

간단한 기능만 개발해봐서 코드 작성 경험이 많지 않은 상태이다.

현재 상태

클린 코드를 작성하는 개발자 되기!

대용량 서비스 프로젝트 개발 능력을 갖춘다.

1년 뒤에 봐도 빠르게 이해할 수 있는 클린코드 작성 능력을 갖춘다.

목표

코드 개선하기!
코드 리뷰하기!

1주일에 한 번 자가 검진하며 코드를 더 개선할 수 있는지 고민한다.

2주일에 1번 주기로 가지며, [박병찬] 님께 요청한다.

코드리뷰하며 목표가 잘 지켜지고 있는지 점검한다.

상세 계획

김두리 개인 목표

iOS의 깊이가 부족해요 비동기...? 반응형 ..?

단순하게 iOS 매커니즘을 이해하는 수준의 경험 만 있다.

프로젝트의 특성을 바탕으로 효과적인 설계란 무엇일까에 대한 고민의 수준에서 멈춰있다.

비동기 및 이벤트 프로그래밍에 대한 관심은 있으나, 반응형 프로그래밍으로 개발해보지 않았다.

현재 상태

유지보수 쉬운 설계하기! 비동기의 완벽 이해하기!

유지보수 하기 쉬운 설계에 대한 통찰력을 갖춘 상태이다.

RxSwift를 이해하고, 상황에 맞는 비동기 코드를 작성한다.

목표

점검 후 설계하기! 비동기 흐름 생각하기!

주요 기능 단위 개발 시 컴포넌트 다이어그램을 먼저 설계함으로서 적절한 설계인가에 대해 점검한다.

프로토콜 지향 프로그래밍을 통해 다양한 상황에서 유연하게 대처할 수 있는 코드를 작성한다.

새로운 기능 단위의 개발을 시작하기 전에, 비동기 흐름에 대해 먼저 구상한다.

상세 계획

그래서, 우리의 팀 목표는

1년 뒤에 봐도 부끄럽지 않은 코드를 작성하자!

다른 사람들이 이 프로젝트를 잡아도 금방 개발할 수 있을 만큼 쉽게 이해할 수 있는 코드이다.
이 코드를 작성하기 위해서는 해당 아키텍처의 전체적인 흐름을 읽을 수 있어야 한다.

모든 기술 스택에는 '왜?'를 생각하는 습관을 들이자!

왜 쓰는지 아는 것은 그 기술 스택이 어떻게 작동하는지 아는 상태이다.
그 기술 스택을 작동 원리부터 공부하여 제대로 파악하는 것이 중요하다.

팀 목표 관리 방안

클린 코드 작성

변수 또는 클래스 이름을 clear하고 직관적으로 작성한다.

프론트엔드와 백엔드에서 사용하는 naming에 차이가 없게끔 한다.

백엔드의 naming을 기준으로 프론트엔드가 맞춰 짓는다.

(FE,BE) 기능 설계시 미팅

포지션(프론트엔드/백엔드)의 기능 설계 시 미팅을 열어, 참석을 원칙으로 하며 상호간 포지션의 이해도를 높이도록 한다.

단, 미팅은 필수가 아닐 수 있고 문서로 대체가 가능하다.

기술스택에 대한 고민

새로운 기술 또는 오픈소스를 도입 시, 관련해서 적합한 선택이었는가에 대해서 고민하고 Wiki의 Footer 부분에 흔적을 남긴다.

팀원에게 직접 설명하는 시간을 가지어 의문을 갖는 태도를 기를 수 있도록 한다.

CHAPTER.1

주제 선정 배경

Discord

디스코드 선정 배경

선정 이유

조원 모두 다 평소에도 자주 사용하는 서비스로 애착이 깊으며, 이해도가 높다.

채팅 서버, 시그널링 서버, 미디어 서버, 상태 관리 서버, 알림 서버 등 MSA 관점에서 확장 시킬 수 있는 프로젝트로 적합하다.

음성, 영상 채팅의 주제에서 고려해야 할 기능들은 작동 원리를 제대로 파악하는 것이 중요하므로 기술 스택의 이해부터 시작하는 목표와 적합하다.

핵심 기능

커뮤니티 단위의 메신저 구조 설계

실시간 채팅 서비스

실시간 음성, 화상 채팅 서비스

CHAPTER.2

업무 분담

Server, Client

업무 분담

박병찬(BE)

API Gateway
채팅 서버
인증 서버
알림 서버

김희동(BE)

시그널링 서버
미디어 서버
커뮤니티 서버
채팅 룸 서버

김민지(FE)

UI/UX 설계
Websocket 연동
WebRTC 연동
FCM 연동

김두리(iOS)

UI/UX 설계
채팅 구현
Web RTC 연동
FCM 연동

CHAPTER.2

업무 분담 관리 계획

정기 회의, 데일리 스크럼

회의

프로젝트 목표 달성을 위해, 정기/데일리 회의를 통해 각자 업무를 확인한다.

정기 회의
매주 월요일

일주일간의 업무 계획을 공유한다.

단, 예상 가능한 범위 내에서 이야기를 진행하며,
고민사항 등은 간단히 언급 정도만 한다.

* 규칙

회의를 할 때마다 회의록을 작성하여 기록한다.
정기 회의는 팀원의 요청이 있을 시 언제든지 가질 수 있다.
정기 회의는 1시간을 넘기지 않는다.

데일리 스크럼
매일 14:00시

전날 업무에 대한 회고를 한다.

금일 업무 계획을 공유한다.

* 규칙

회의를 할 때마다 회의록을 작성하여 기록한다.
스크럼 회의는 10~20분을 가진다.

CHAPTER.2

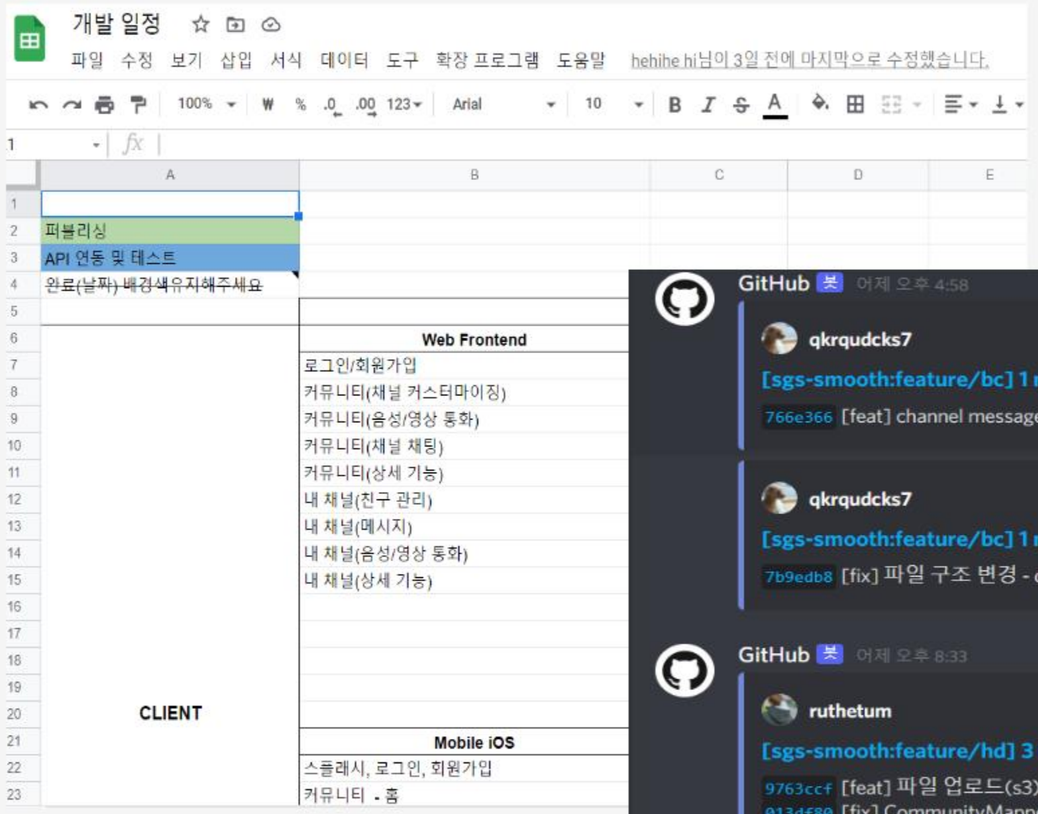
프로젝트 일정 관리

일정관리

일정 관리

Google SpreadSheets

개발 일정 단위를 파트별로 나누어 작성한다.

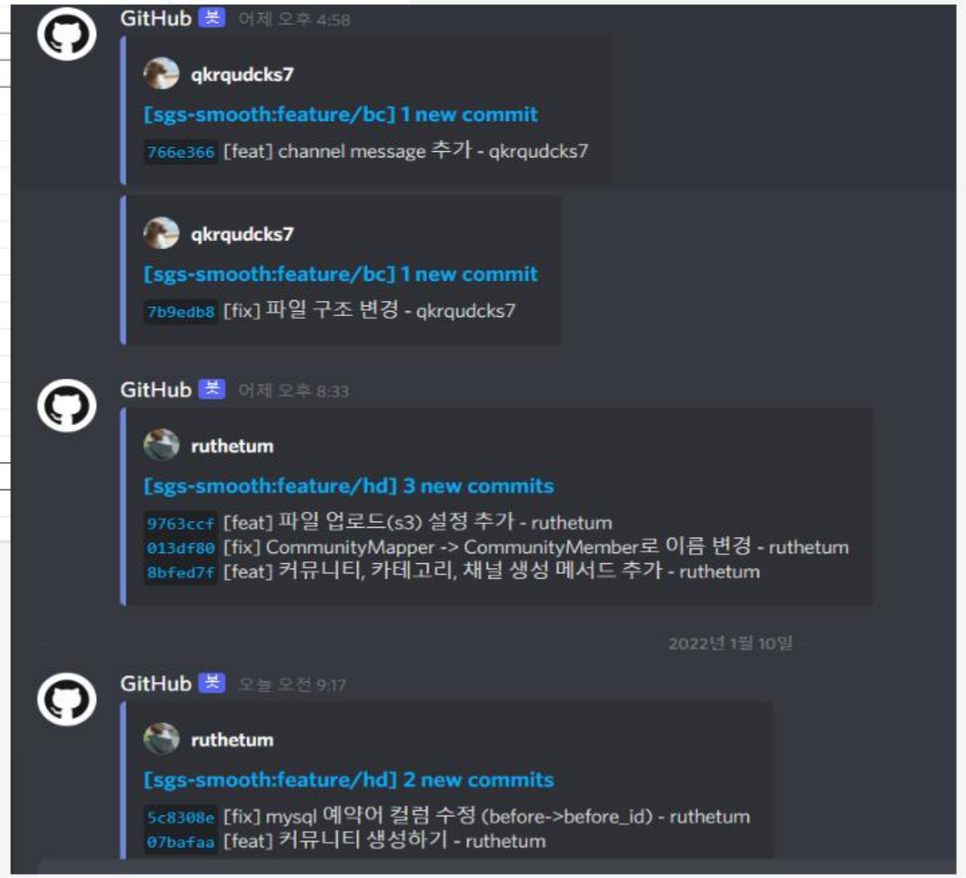


Github Wiki

개발 이슈를 위키에 작성하여 관리한다.

Discord + Github Webhook

디스코드에 웹훅을 연동하여 커밋 진행 상황을 확인한다.



CHAPTER.2

위험 관리

위험 요소, 관리 계획

위험 관리

위험은 **프로젝트 요구사항에 따른 위험**, **프로젝트 관리에 따른 위험**으로 나뉘며 프로젝트 진행 시 계속 변동된다.

프로젝트 요구사항에 따른 위험

* 위험 요소

프로젝트 요구사항에 따라 프로젝트 일정에 차질이 생길 수 있다.

* 관리 계획

프로젝트 요구사항에 따른 위험이 발생한다면, 즉시 회의를 소집해 프로젝트 일정을 조정하거나 프로젝트 요구사항을 조정한다.

이미 발생한 위험 요소는 정기 회의를 통해 어떻게 진행되고 있는지 공유한다.

프로젝트 관리에 따른 위험

* 위험 요소

Github의 다양한 기능을 사용하는데 있어서 미숙하여,
실제 업무와 프로젝트 관리툴(Github) 간의 차이가 있다.

* 관리 계획

Github 사용이 미숙한 팀원이 도움을 요청한 경우 다른 팀원이 사용을 돕는다.

Github 사용의 매뉴얼을 작성하여 공유한다.[소스코드 버전 관리 R&D]

---

* 위험 요소

일정 테스크 완료 시 프로젝트 관리 툴을 이용하여 다른 팀원들에게 테스트 완료가 되었음을
전파하는데 있어 어려움이 있다.

* 관리 계획

데일리 스크럼을 통해 완료한 일과 오늘 해야할 일을 시간 순으로 공유한다.

업무 중 태스크를 완료하였다면, 오프라인 업무 중일 경우에는 직접 이야기를 하고
온라인 업무 중일 경우 디스코드의 태스크-코드 채널을 통해 이야기한다.

소스코드 버전관리

Github를 통해 프로젝트 소스코드를 관리한다.

각 인원 별로 branch를 다르게 구성하여 develop branch로 push한다.

develop branch에서 main branch로는 조장이 확인하여 push한다.

소스코드 버전관리 R&D

박병찬, 김희동(BE) : 개인 브랜치에 push한다, 최종 병합은 상대가 approve 시 Merge한다.

김민지(FE) : 본인 확인 및 궁금한 사항 등 [박병찬] 팀원에게 PR 확인 요청한다. (approve 시 Merge)

김두리(iOS) : 본인 확인 및 Merge한다.

* 최종 소스코드 관리자 : 김희동 (파일 디렉토리, branch 등 전체 관리)

Git flow 구조

* main (배포용)

* develop (개발용)

  - server
    - feature/bc
    - feature/hd

  - client

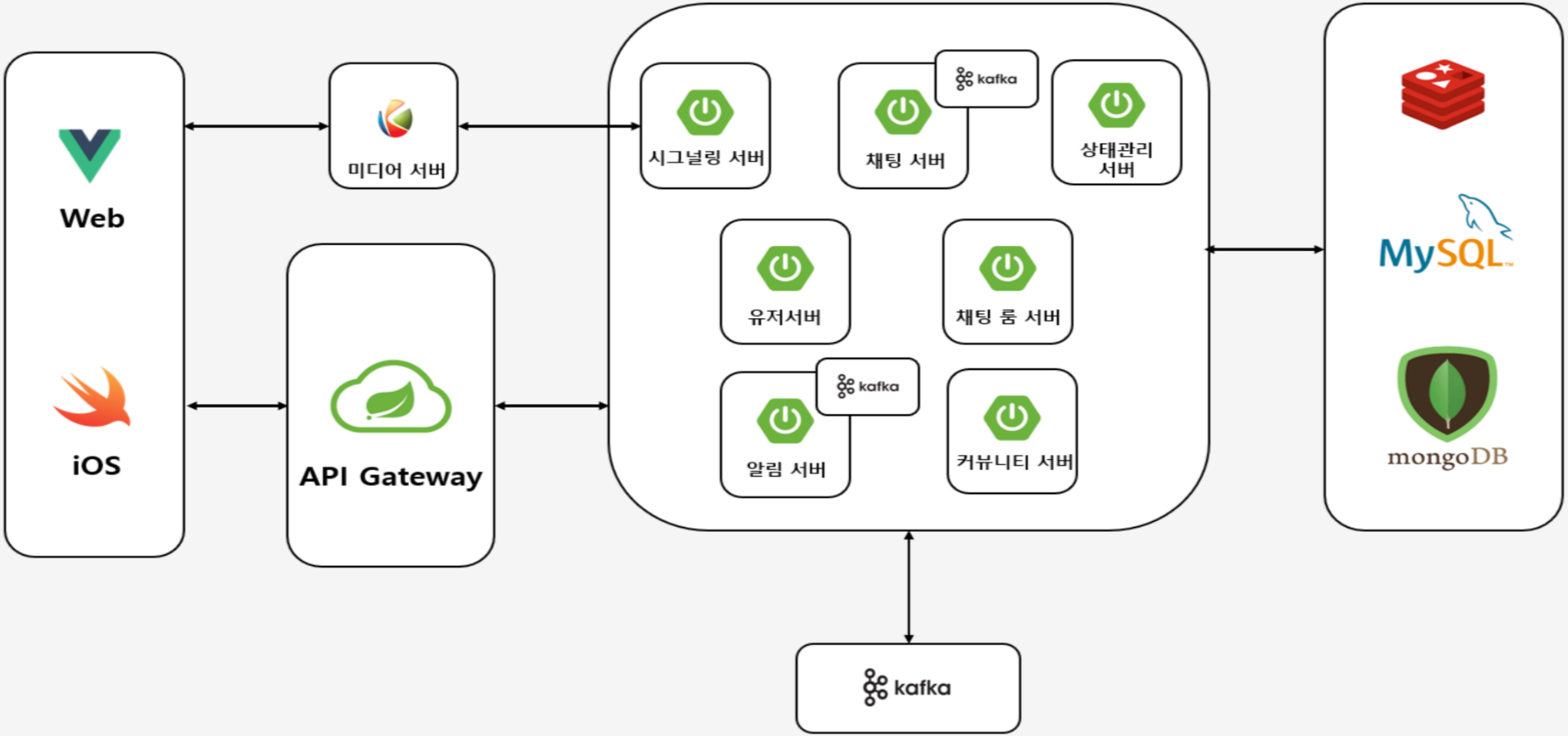
  - iOS

CHAPTER.3

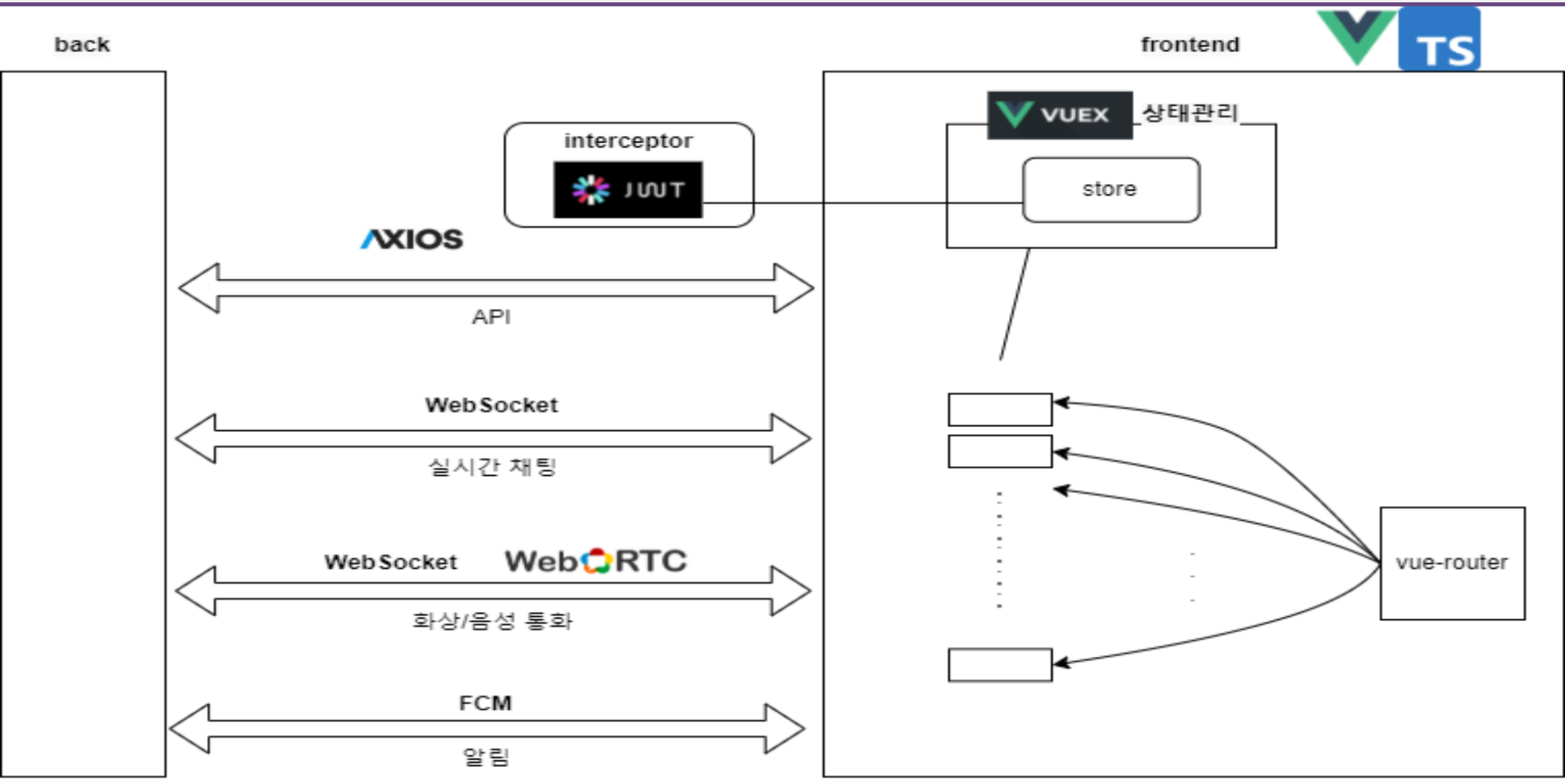
아키텍처 설계

전체 아키텍처(BE / FE / iOS)

전체 아키텍처

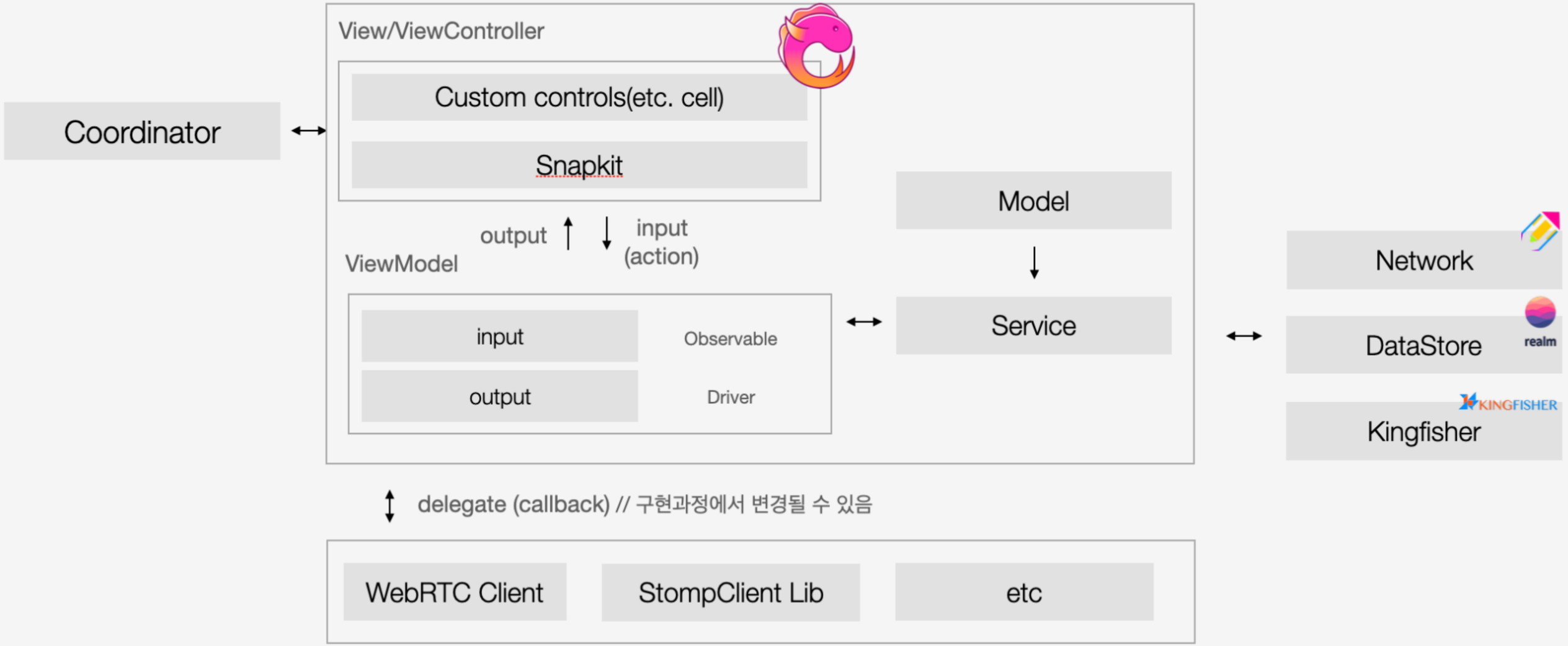

Web
iOS
미디어 서버
API Gateway
시그널링 서버
kafka
채팅 서버
상태관리 서버
유저서버
채팅 룸 서버
kafka
알림 서버
커뮤니티 서버
kafka
MySQL
mongoDB

FE 아키텍처


back
frontend
TS
VUEX
상태관리
interceptor
JWT
store
AXIOS
API
WebSocket
실시간 채팅
WebSocket
WebRTC
화상/음성 통화
FCM
알림
vue-router

iOS 아키텍처

iOS Architecture

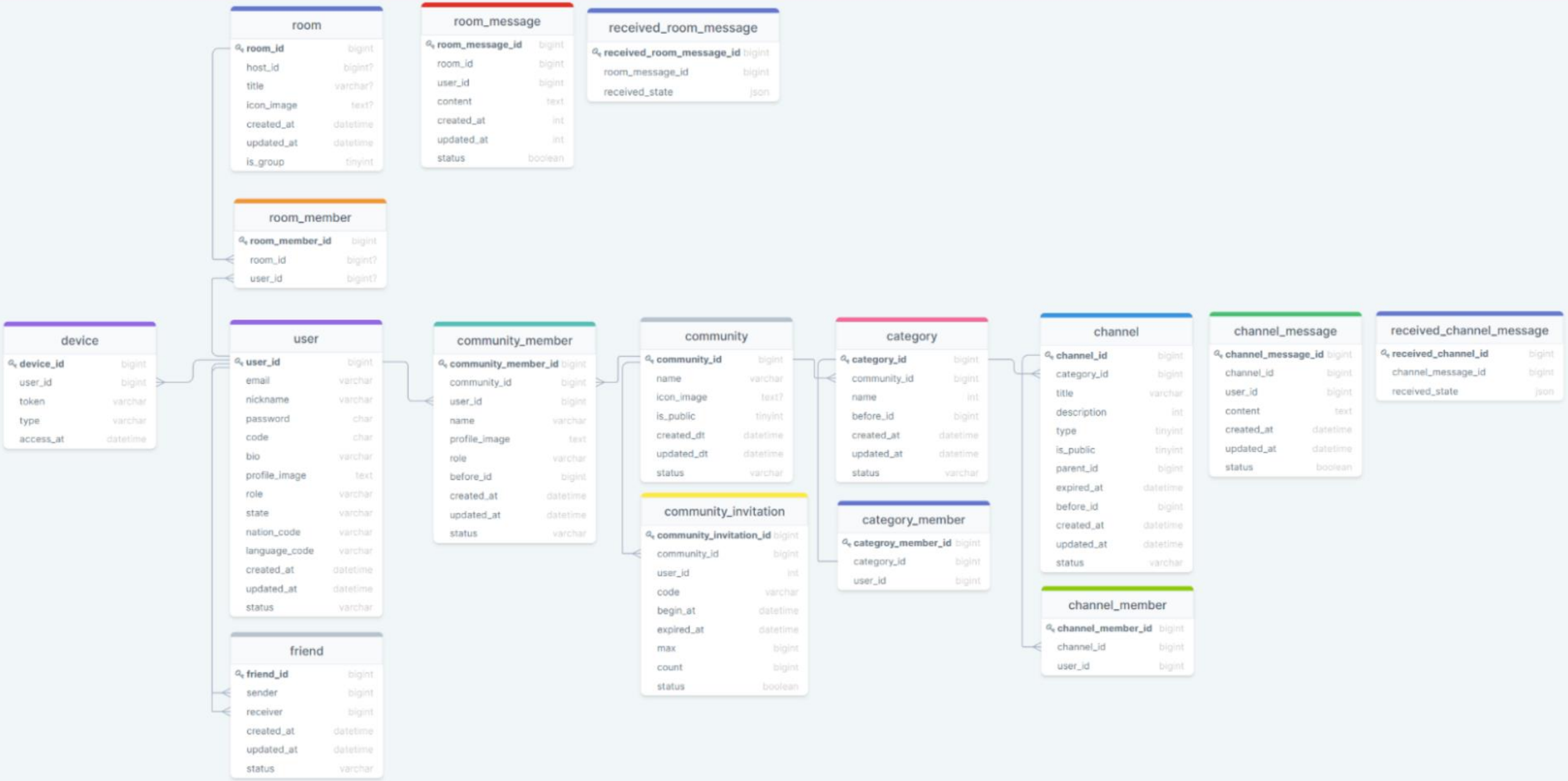


CHAPTER.3

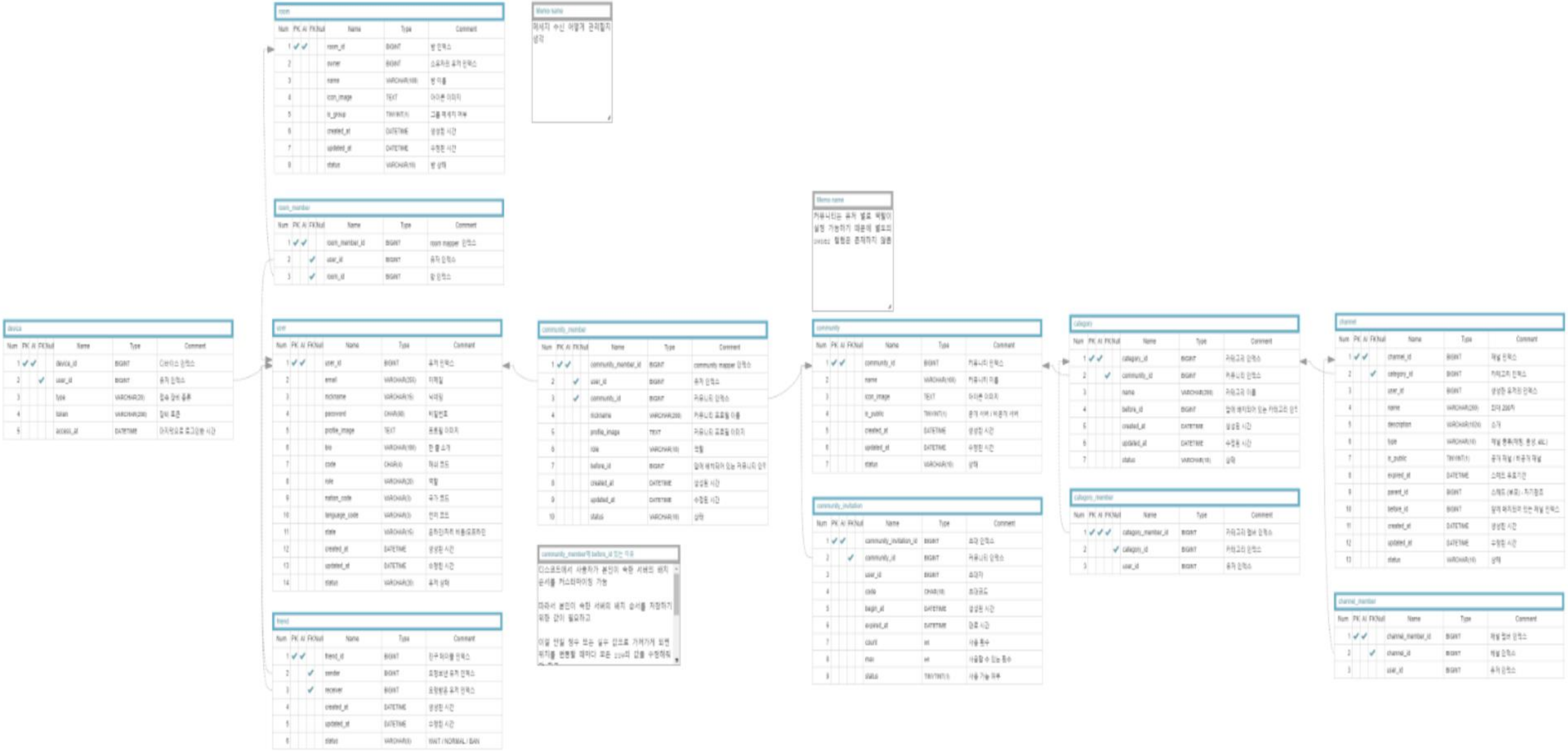
DB 스키마

전체 아키텍처

DB 스키마


room
room_id bigint
host_id bigint?
title varchar?
icon_image text?
created_at datetime
updated_at datetime
is_group tinyint
room_message
room_message_id bigint
room_id bigint
user_id bigint
content text
created_at int
updated_at int
status boolean
received_room_message
received_room_message_id bigint
room_message_id bigint
received_state json
room_member
room_member_id bigint
room_id bigint?
user_id bigint?
device
device_id bigint
user_id bigint
token varchar
type varchar
access_at datetime
user
user_id bigint
email varchar
nickname varchar
password char
code char
bio varchar
profile_image text
role varchar
state varchar
nation_code varchar
language_code varchar
created_at datetime
updated_at datetime
status varchar
community_member
community_member_id bigint
community_id bigint
user_id bigint
name varchar
profile_image text
role varchar
before_id bigint
created_at datetime
updated_at datetime
status varchar
community
community_id bigint
name varchar
icon_image text?
is_public tinyint
created_dt datetime
updated_dt datetime
status varchar
community_invitation
community_invitation_id bigint
community_id bigint
user_id int
code varchar
begin_at datetime
expired_at datetime
max bigint
count bigint
status boolean
category
category_id bigint
community_id bigint
name int
before_id bigint
created_at datetime
updated_at datetime
status varchar
category_member
categroy_member_id bigint
category_id bigint
user_id bigint
channel
channel_id bigint
category_id bigint
title varchar
description int
type tinyint
is_public tinyint
parent_id bigint
expired_at datetime
before_id bigint
created_at datetime
updated_at datetime
status varchar
channel_member
channel_member_id bigint
channel_id bigint
user_id bigint
channel_message
channel_message_id bigint
channel_id bigint
user_id bigint
content text
created_at datetime
updated_at datetime
status boolean
received_channel_message
received_channel_id bigint
channel_message_id bigint
received_state json
friend
friend_id bigint
sender bigint
receiver bigint
created_at datetime
updated_at datetime
status varchar

DB 스키마



CHAPTER.3

Backend 진행 상황

서버 개발 진행 상황

서버 개발 우선 순위

유저 서버, API Gateway

대부분 api 호출에 토큰 값이 필요하다.

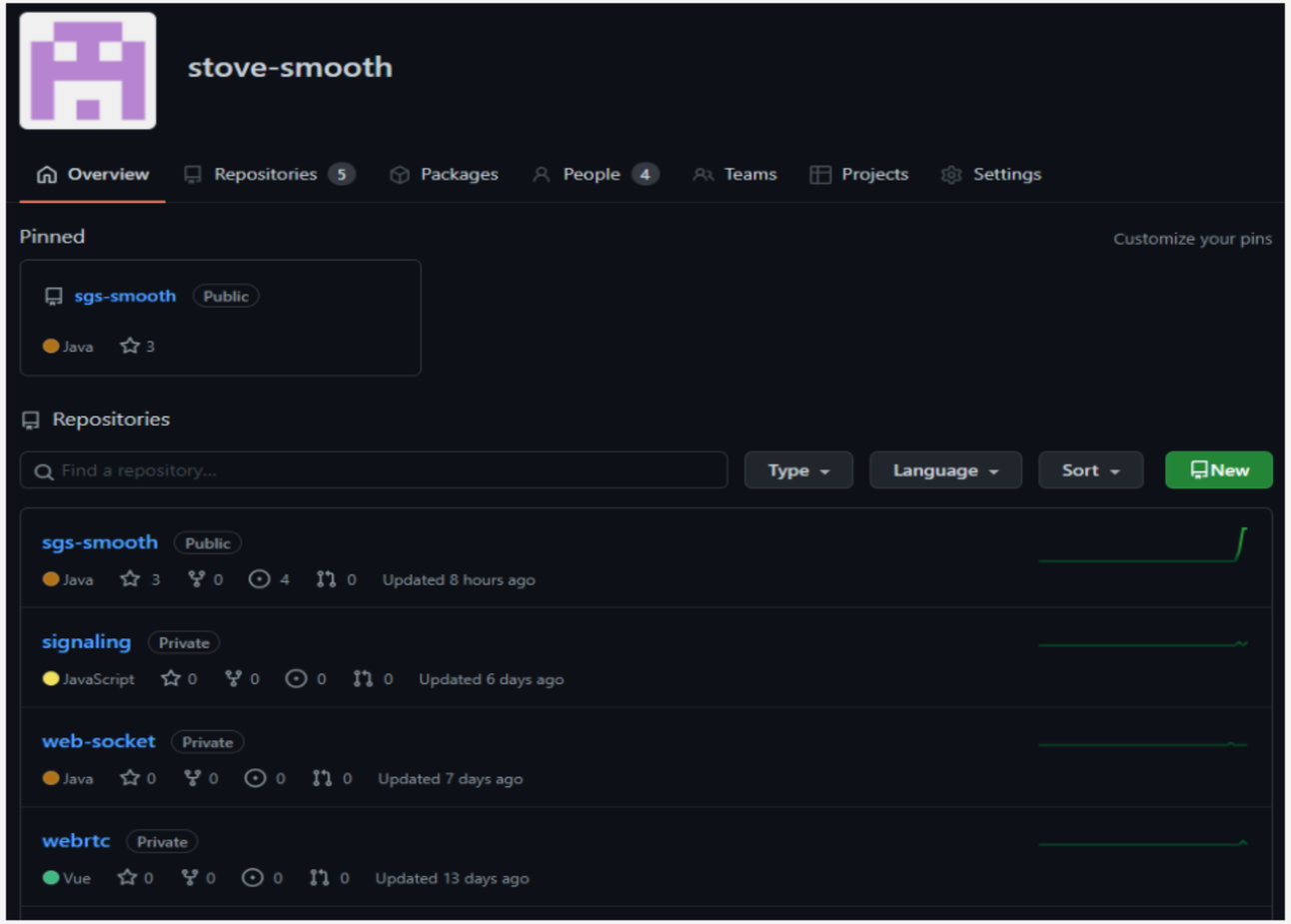
라우팅한 서버들의 토큰 필터 작업이 필요하다.

시그널링, 미디어 서버

시그널링, 미디어 서버에 대한 이해가 필요하다.

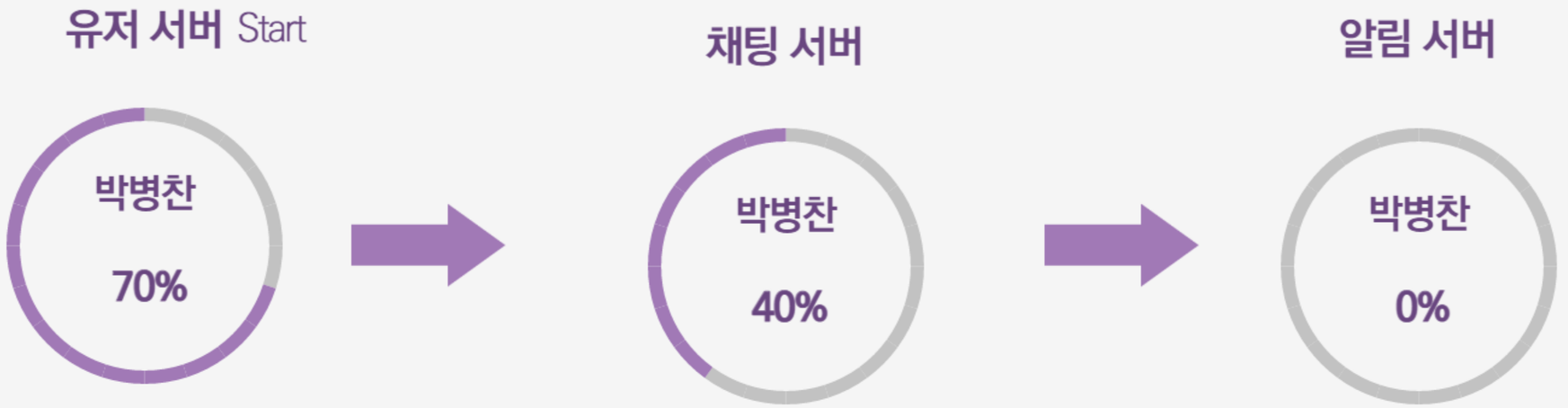
디스코드의 핵심기능 이므로 우선으로 처리한다.

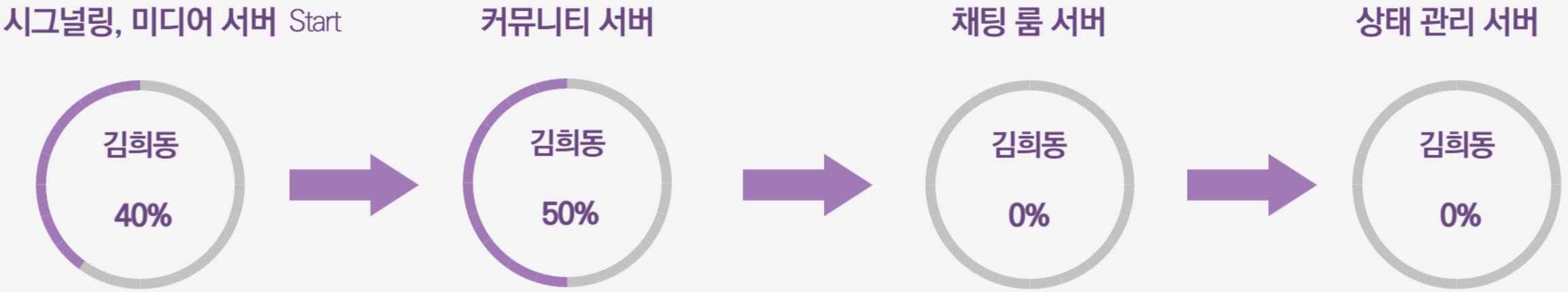
시그널링, 미디어 서버 연결 테스트를 진행했다.



* 서버 개발자들의 각각의 서버를 맡아 개발하는 방식으로 진행한다.

BE 진행 상황


유저 서버 Start
박병찬
70%
채팅 서버
박병찬
40%
알림 서버
박병찬
0%


시그널링, 미디어 서버 Start
김희동
40%
커뮤니티 서버
김희동
50%
채팅 룸 서버
김희동
0%
상태 관리 서버
김희동
0%

API 명세서

	Index	Method	URI	Description	명세서	로컬 개발 여부	개발서버 반영 여부	실서버 반영 여부	iOS 적용 여부
Auth Service	1	GET	/auth-server/auth/info	내 정보	Y	Y	Y		
	2	POST	/auth-server/sign-up	회원가입	Y	Y	Y		
	3	POST	/auth-server/sign-in	로그인	Y	Y	Y		
	4	POST	/auth-server/send-mail	인증 이메일 보내기	Y	Y	Y		
	5	GET	/auth-server/check-email/{key}	이메일 인증번호 확인하기	Y	Y	Y		
	6	PATCH	/auth-server/auth/role	role 바꾸기	Y	Y	Y		
	7	POST	/auth-server/refresh	accessToken 재발급	Y	Y	Y		
	8	POST	/auth-server/auth/find-id-list	id값들을 바탕으로 account 정보들을 반환	Y	Y	Y		
	9	POST	/auth-server/auth/friend	친구요청	Y	Y	Y		
	10	GET	/auth-server/auth/friend	친구 요청 리스트, 내가 보낸 친구요청 리스트	Y	Y	Y		
	11	PATCH	/auth-server/auth/friend	친구요청 수락	Y	Y	Y		
	12	DELETE	/auth-server/auth/friend	친구요청 거절, 친구 삭제	Y	Y	Y		
	13	POST	/auth-server/auth/profile	프로필 사진 업로드	Y	Y	Y		
	14	PATCH	/auth-server/auth/profile	프로필 자기소개 수정	Y	Y			
	15								
Community Service	16	GET	/community-server/community/{communityId}/{channelId}	메인 - 사용자의 커뮤니티 정보 조회하기					
	17	POST	/community-server/community	커뮤니티 생성하기					
	18	PATCH	/community-server/community/name	커뮤니티 이름 수정하기					
	19	PATCH	/community-server/community/icon	커뮤니티 아이콘 이미지 수정하기					
	20	POST	/community-server/community/invitation	초대장 만들기					
	21	GET	/community-server/community/{communityId}/invitation	초대장 조회하기					
	22	GET	/community-server/community/{communityId}/member	커뮤니티 멤버 조회하기					
	23	POST	/community-server/community/member	초대장으로 커뮤니티 들어오기					
	24	DELETE	/community-server/community/member?id=1	멤버 추방하기					
	25	POST	/community-server/community/member/ban	멤버 차단하기					

API 명세서

/auth-server/auth/info	내 정보						
	Method	**GET**					
	Header	**Name**	**Type**	**Mandatory**	**Example**	**Default**	**Description**
		AUTHORIZATIC	String	Y	eyJhbGciOiJIUz		액세스 토큰
	Body	**Name**	**Type**	**Mandatory**	**Example**	**Default**	**Description**
	Path Variable	**Name**	**Type**	**Mandatory**	**Example**	**Default**	**Description**
	Query String	**Name**	**Type**	**Mandatory**	**Example**	**Default**	**Description**
	Response Parameters	**Name**	**Type**	**Mandatory**	**Example**	**Default**	**Description**
		isSuccess	boolean	Y	TRUE		성공 유무
		code	String	Y	1000		코드
		message	String	Y	요청에 성공했습니다.		메세지
		result	list	Y			
		result.email	String	Y	pak7380@naver.com		이메일
		result.name	String	Y	박병찬		이름
		result.code	String	Y	5653		디코 이름 옆 번호
	Response Sample						

```
{
  "isSuccess": true,
  "code": 1000,
  "message": "요청에 성공하였습니다.",
  "result": {
    "email": "pak7380@naver.com",
    "name": "박병찬",
```

/auth-server/auth/find-id-list	id값들로 유저정보 반환						
	Method	**POST**					
	Header	**Name**	**Type**	**Mandatory**	**Example**	**Default**	**Description**
		AUTHORIZATIC	String	Y	eyJhbGcDwada		accessToken
	Body	**Name**	**Type**	**Mandatory**	**Example**	**Default**	**Description**
		-	list	Y	[2,3,4,5,6,7,8,9]		id값 리스트
	Path Variable	**Name**	**Type**	**Mandatory**	**Example**	**Default**	**Description**
	Query String	**Name**	**Type**	**Mandatory**	**Example**	**Default**	**Description**
	Response Parameters	**Name**	**Type**	**Mandatory**	**Example**	**Default**	**Description**
		isSuccess	boolean	Y	TRUE		성공 유무
		code	String	Y	1000		코드
		message	String	Y	요청에 성공했습니다.		메세지
		result	list	Y			
		result.id	Long	Y	2		pk id 값
		result.name	String	Y	박병찬		이름
		result.code	String	Y	5653		디코 이름 옆 번호
		result.email	String	Y	pak7380@naver.com		이메일
		result.accessTok	String	Y	eyJhbGciOiJIUz		access Token
		result.refreshTok	String	Y	eyJhbGDWQSE		refresh Token
	Response Sample						

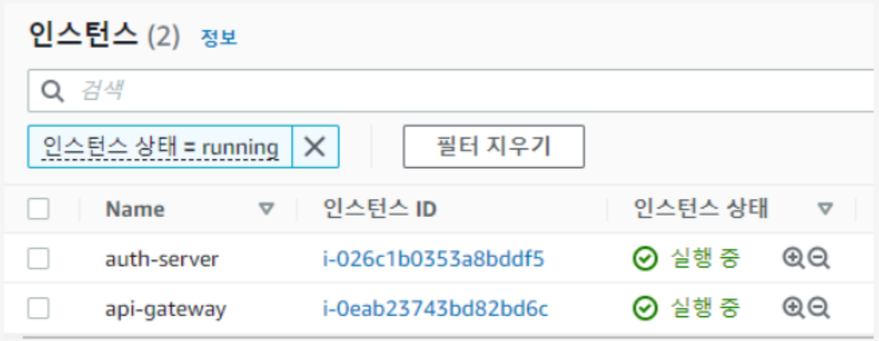
Request Sample

[2,3,4,5,6,7,8,9]

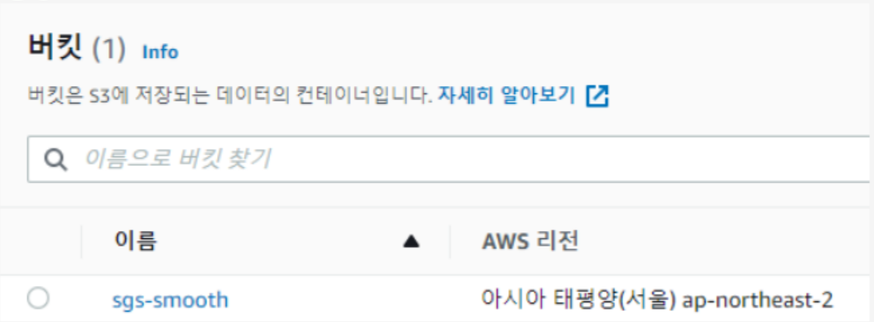
```
{
  "?": {
```

서버 관리

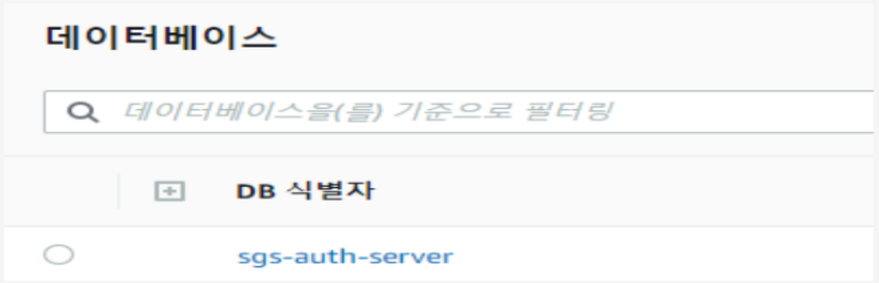
EC2



S3



RDS



개발 된 서버들은 EC2와 RDS를 활용하여 배포를 진행한다.

팀원들이 Postman으로 바로 확인할 수 있거나,

해당 api들을 바로 적용할 수 있도록 관리한다.

DB를 팀원 모두가 관리할 수 있도록 한다.

CHAPTER.3

Frontend, iOS 진행 상황

전체 아키텍처

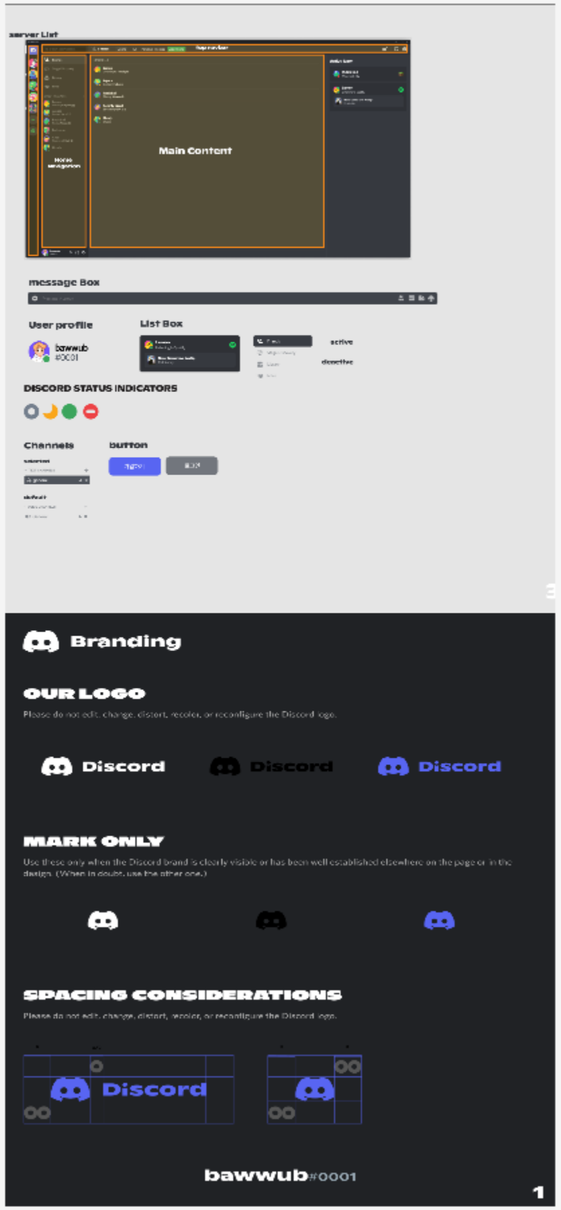
FE / iOS 기능 우선순위

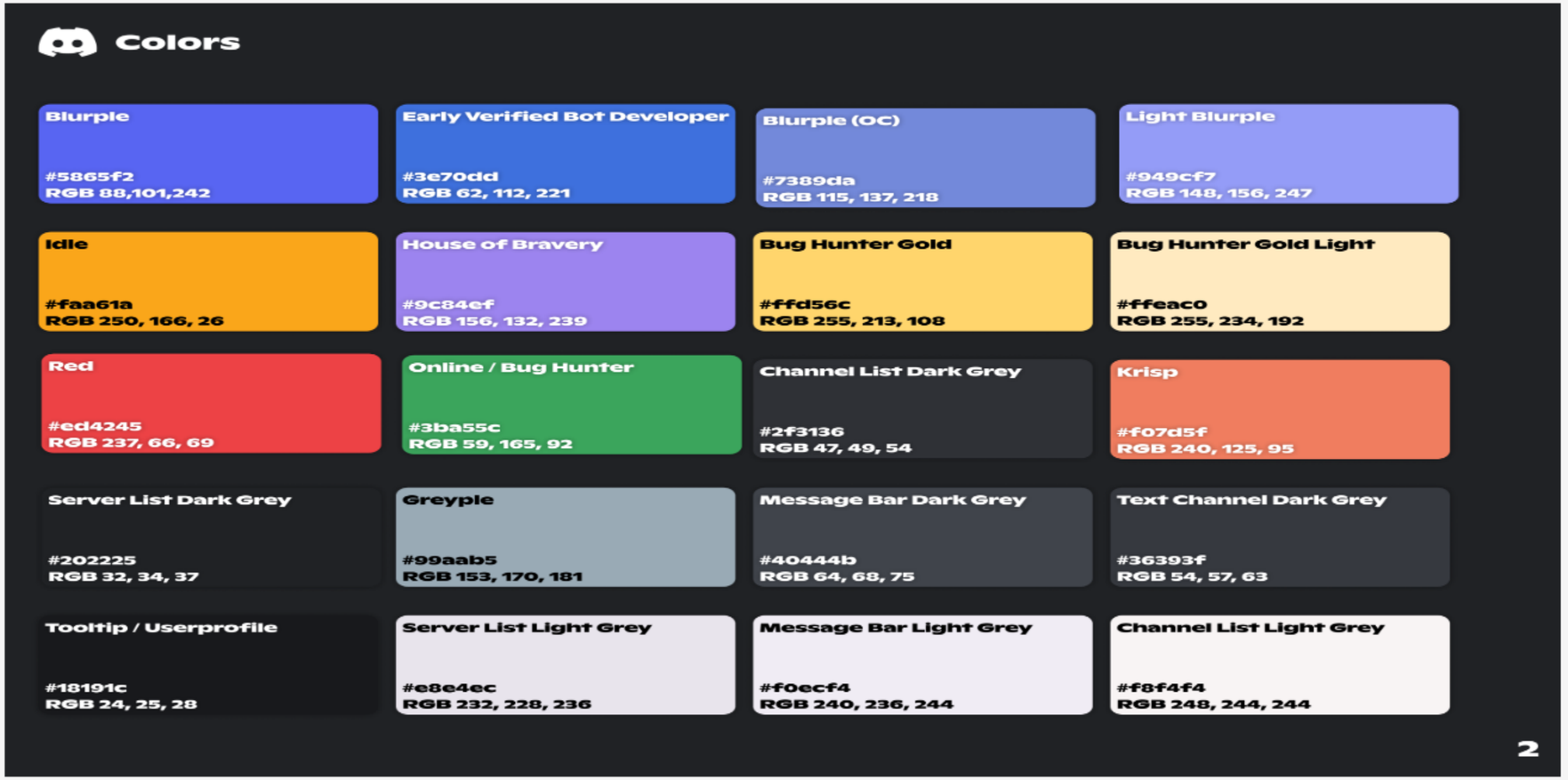
1. 유저(회원가입, 로그인)
2. 채널(WebRTC- 음성, 영상)
3. 채팅(Web socket)
4. 친구

FE / iOS 개발 흐름

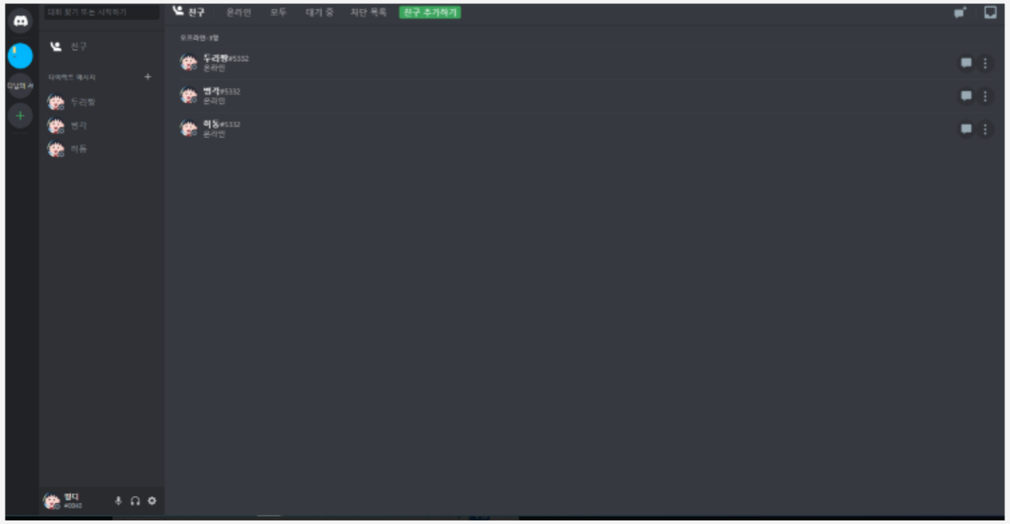
1. 기능의 필수 기술 또는 라이브러리 기반을 마련한다.
   - 기초 지식이 필요한 경우 학습한다. (필요한 경우 데모 제작)
   - 코드 베이스로 개발한다. (의존성 추가)
2. UI 개발
   - 데이터 바인딩이 꼭 필요한 경우 API Mocking 하여 진행한다.
3. api 연동

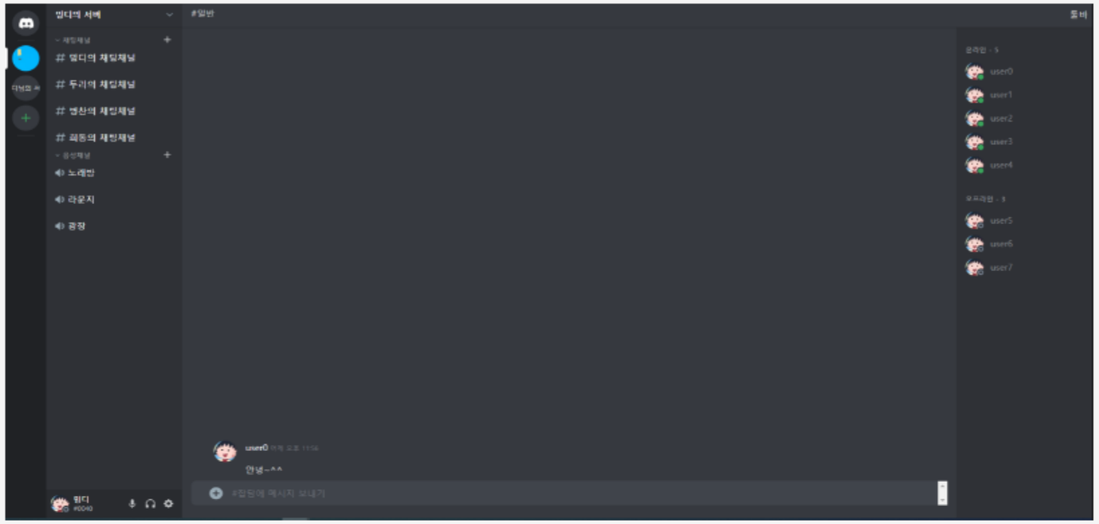
Front / iOS 진행 상황

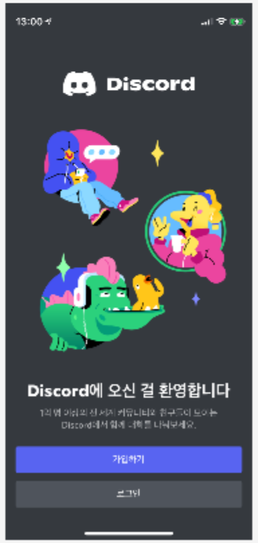


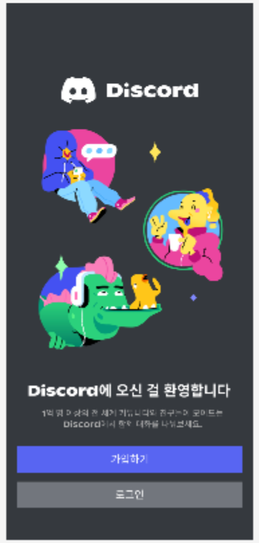


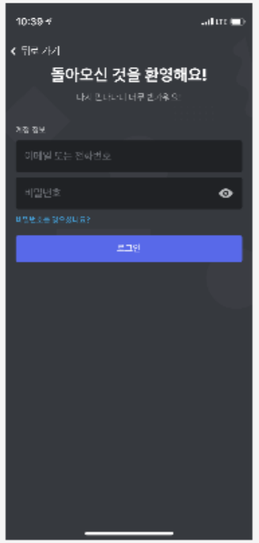
Front / iOS 진행 상황











감사합니다

Smooth PMP 문서 링크