YamYam

현명은, 구자현, 박동진, 조현민

목차

NoPOKER 소개

게임 컨셉

인디언 포커를 기반으로 한 난투형 액션 게임

게임 규칙

- 인디언 포커를 진행한다
- 숫자가 높은 사람이 좋은 무기를 가질 수 있다.
- 인디언포커가 끝난 후, 플레이어간 전투를 한다.
- 이 라운드에서 모든 상대방을 쓰러뜨린 경우 베팅된 금액을 모두 가져간다.
- 파산하지 않은 최후의 1인만 남는 경우 게임이 종료된다.

기능목록

인증 서버 구현
회원가입/로그인화면 구현

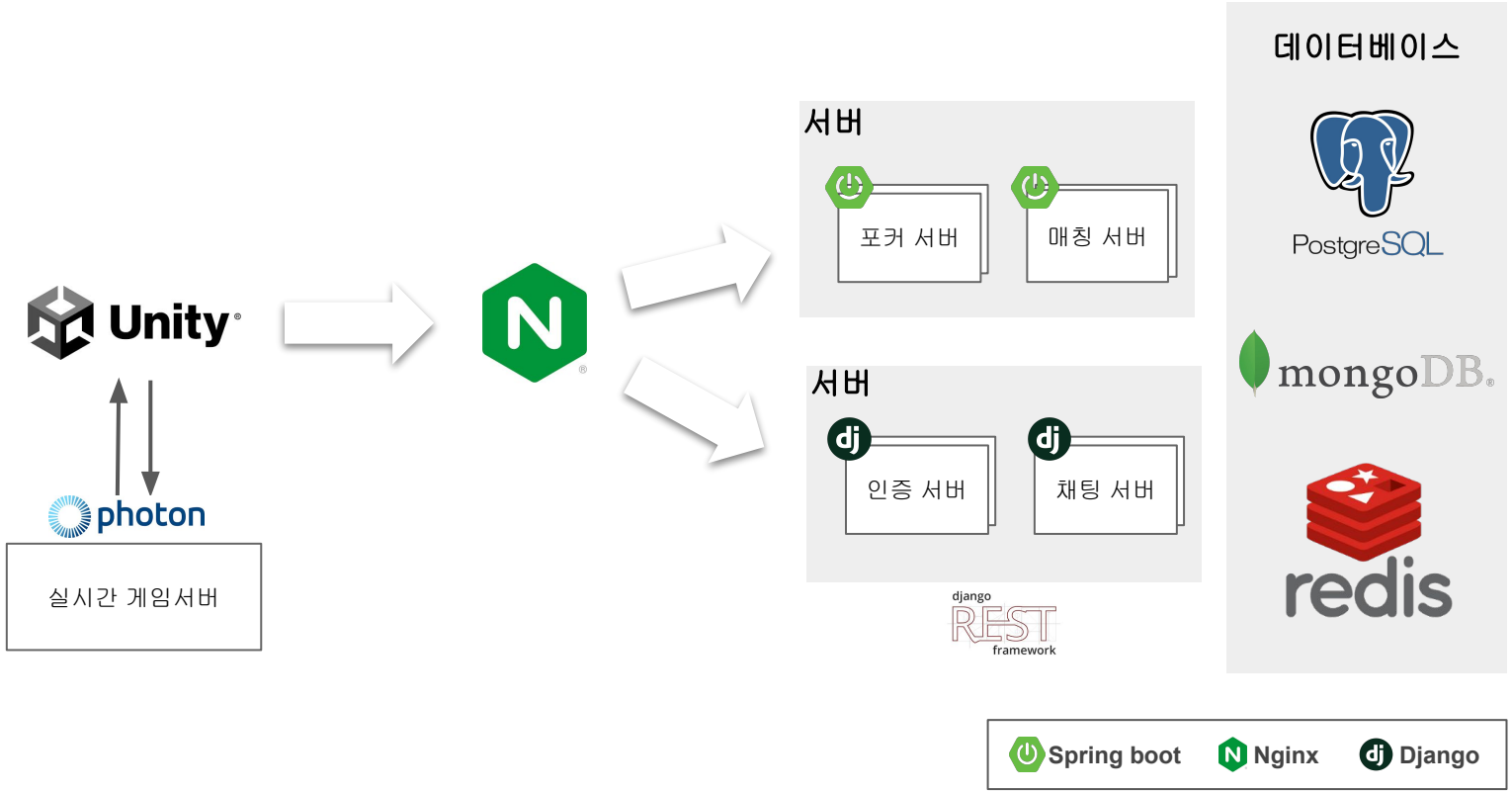
채팅 서버 구현
채팅 UI 구현

로비 서버 구현
로비 구현

게임 서버 구현
2D 포커게임 구현

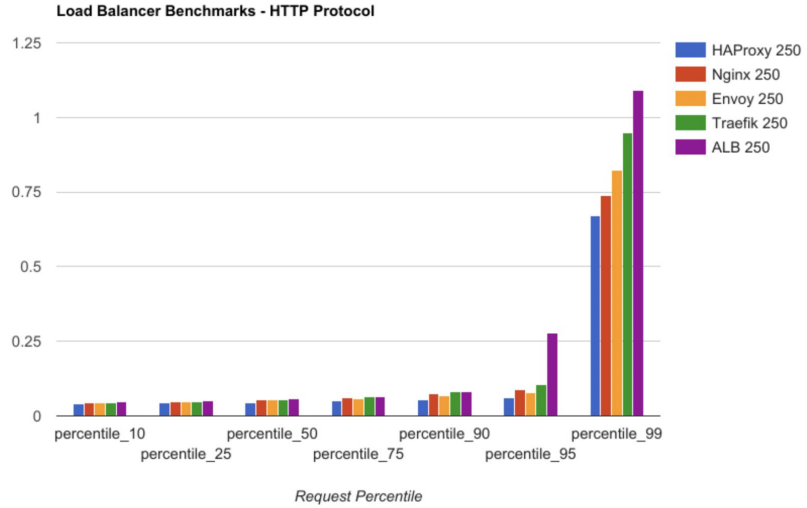
실시간 게임 서버 구현
3D 액션게임 구현

아키텍처


Unity
photon
실시간 게임서버
서버
포커 서버
매칭 서버
서버
인증 서버
채팅 서버
django
REST
framework
데이터베이스
PostgreSQL
mongoDB.
redis
Spring boot
Nginx
Django

아키텍처

로드 밸런서



장점 : 구현(

단점 : Healt

ck 무료

는 상대적으로 떨어지는

https://www.loggly.com/blog/benchmarking-5-popular-load-balancers-nginx-haproxy-envoy-traefik-and-alb/

아키텍처

API Gateway

- 인증 API를 호출하면 이후의 API호출에 필요한 토큰을 돌려 받음
- 이후에 사용자가 이 토큰을 가지고 원하는 API를 호출하면 API Gateway는 Access KEY는 검사하지 않고 이 토큰이 맞는지만 체크한 후 해당하는 서비스를 호출
- 원하는 서비스를 호출하는 방식은 사용자가 호출할 서비스와 내부에서 개발된 엔드포인트를 테이블 형태로 가지고 있다가 사용자가 호출한 서비스를 비교해서 내부의 엔드포인트로 바꿔서 호출하고 그 리턴 값을 사용자에게 돌려주는 방식사용

아키텍처

API Gateway

테이블로 관리
외부로 알려지는 서비스 URL
내부 URL

필요에 따라 내부 URL을 바꿔서 응답 가능

내부 URL은 HTTPS를 지원하지 않더라도
외부에는 HTTPS형태로 노출가능

아키텍처

API Gateway

다양한 내부의 서비스에 대한 관리 편리성과
공통으로 개발해야할 부분을 API Gateway가 해결 가능

Nginx를 활용해 API Gateway 구현

아키텍처

ERD


회원
user
식별자 id Domain Type
아이디 username Domain Type
비밀번호 password Domain Type
닉네임 nickname Domain Type
이메일 email Domain Type
상태 status Domain Type
승 victory Domain Type
패 loose Domain Type
게임
game
식별자 id Domain Type
게임방_pk room_id Domain Type
회원_pk user_id Domain Type
상태 status Domain Type
칩 chip Domain Type
턴 turn Domain Type
시간 created_at Domain Type
채팅
chat
식별자 id Domain Type
회원_pk user_id Domain Type
내용 content Domain Type
시간 created_at Domain Type

아키텍처 관련 질문사항

회원정보

아이디	비밀번호	이메일	닉네임	승리개수	패배개수	관리자권한?

게임정보 2D → 3D

방번호	현재 참여하고 있는 유저	유저의 카드 숫자	게임 종결여부

게임정보 (JSON) 3D → 2D

방번호	승자 닉네임

질문사항

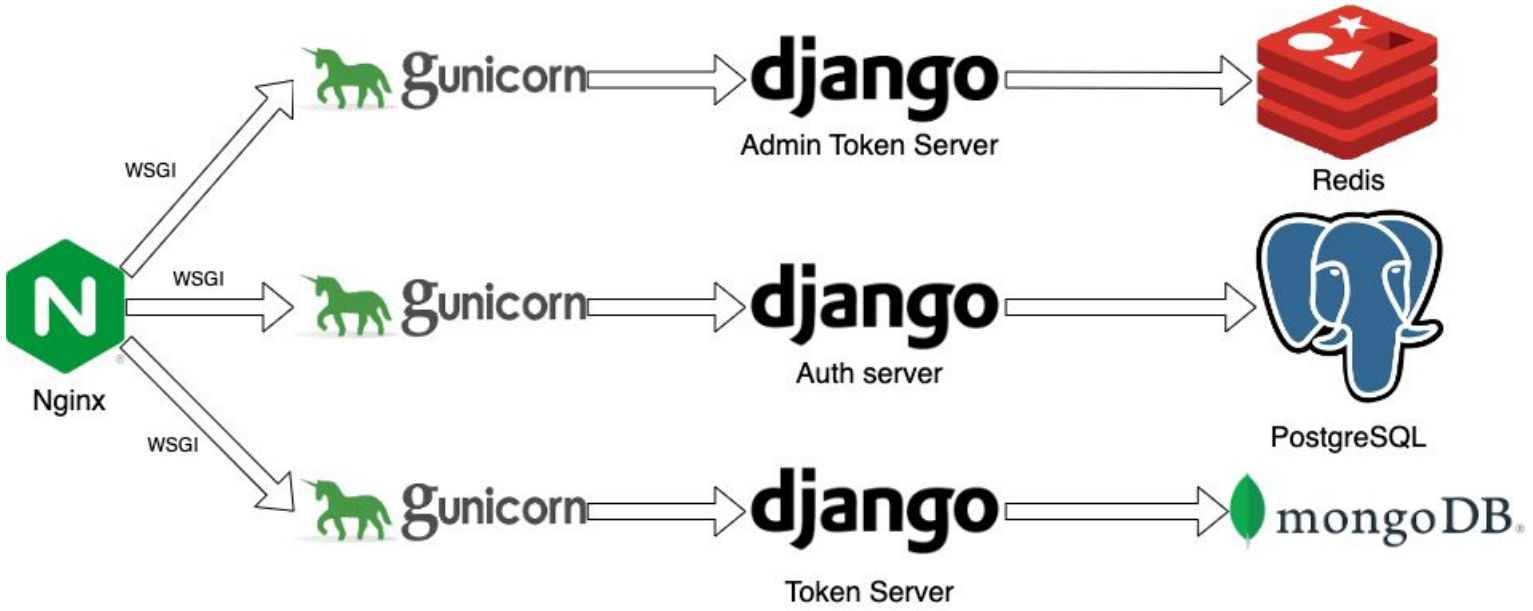
회원과 관리자를 합칠 것인가? (유저 테이블이 권한 테이블을 참조)
- YES : 겹치는 정보가 많다. (관리가 쉬움)

게임 중에 방을 나갔을 때, 5분 동안 게임에 참가못하는 페널티를 주고 싶음
- DB attribute(참가 허용 Yes/Null)로 관리?
- 아니면 Redis(cache)에서 금지된 플레이어 ID 관리?

- 현재 참여하고있는 유저를 DB에? Cache에?

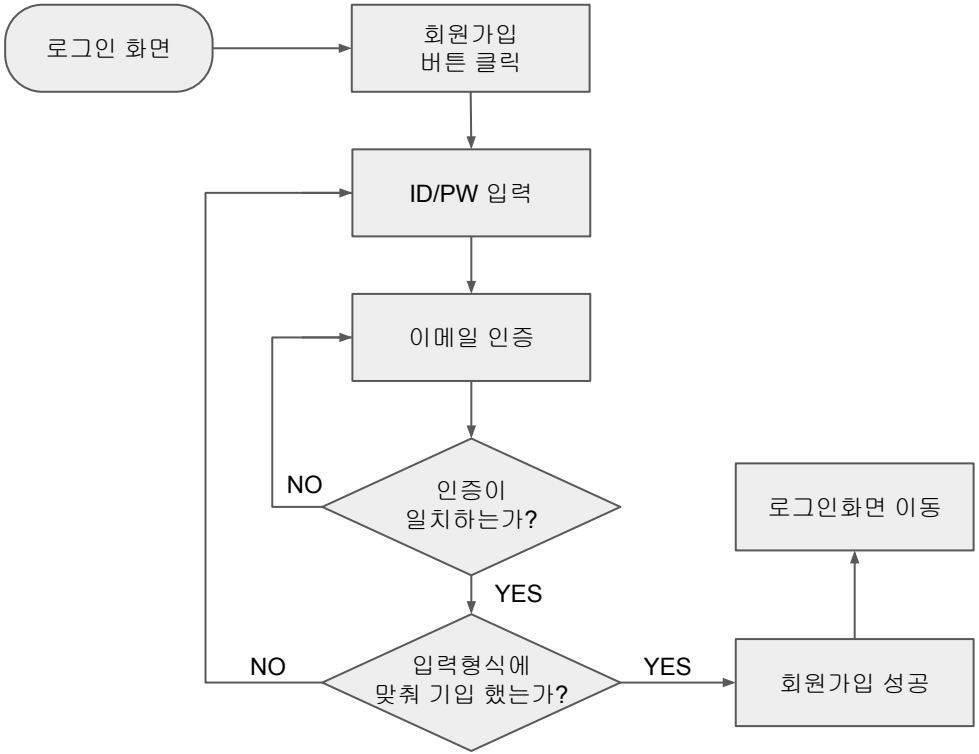
인증 서버

인증 아키텍처

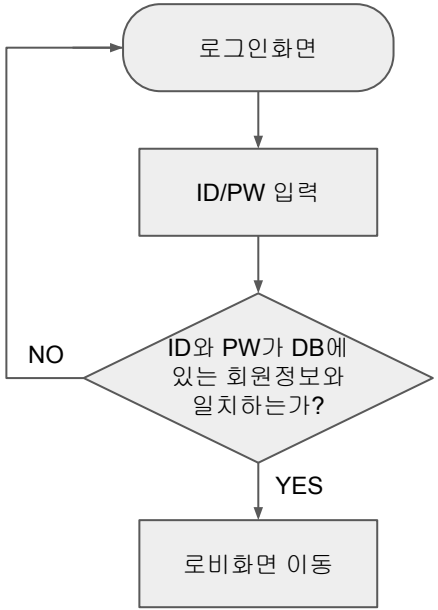


질문 : 관리자 API, 메일 전송 API를 다루는 API Server를 따로 만들어야 할까요?

인증(회원가입 & 로그인) - 흐름도

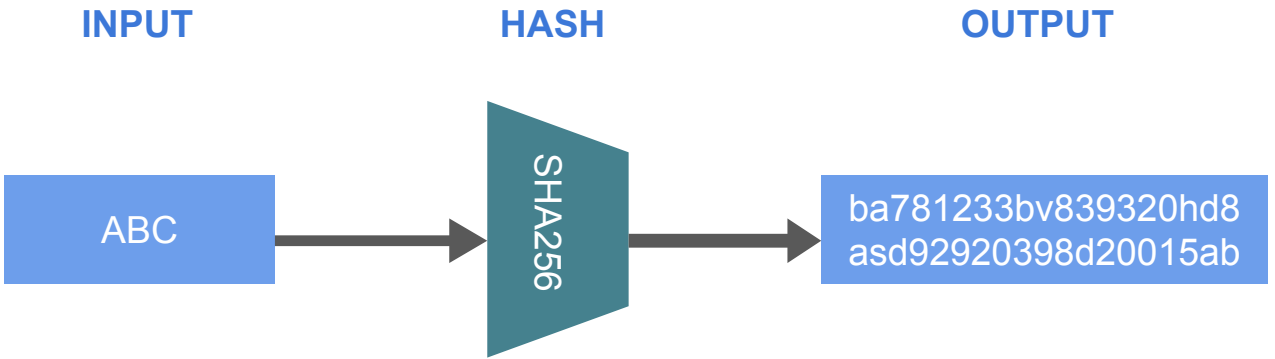


회원가입



로그인

인증(회원가입) - Auth Server



- 회원 가입 시 이메일로 인증 코드 발송(SMTP 프로토콜)
- password를 SHA256 방식으로 암호화하여 사용자정보를 RDB에 저장(PostgreSQL)

인증(로그인) - Auth Server

로그인 시, Auth Server에서 Access Token, Refresh Token 발급 (JWT 기반)

이때, Refresh Token은 DB에 저장

Access Token의 유효 기간은 짧게, Refresh Token의 유효 기간은 길게

인증(로그인) - Auth Server

클라이언트는 httpOnly cookie에 Token저장	Refresh Token 저장하는 DB
스크립트 기반 공격을 방어	Read performance가 중요하므로 NoSQL 사용(MongoDB)

인증(토큰 유효 확인 API) - Token Server

- 토큰이 유효한지 확인하는 API 호출 시, Token Server에서 Access Token과 Refresh Token 확인

Access Token이 유효	HTTP 200 정상 응답
Access Token이 유효x **Refresh Token이 유효** **Refresh Token이 DB에 존재**	Access Token, Refresh Token을 재발급 (새로운 Refresh Token은 DB에 저장, 기존 Refresh Token은 DB에서 삭제 = **Refresh Token Rotation**)

Refresh Token Rotation

- Refresh Token은 Access Token을 한번만 재발급해줄 수 있음 (일회용)
- Refresh Token 탈취 시, 리스크를 줄일 수 있음

인증(토큰 유효 확인 API) - Token Server

- 토큰이 유효한지 확인하는 API 호출 시, Token Server에서 Access Token과 Refresh Token 확인

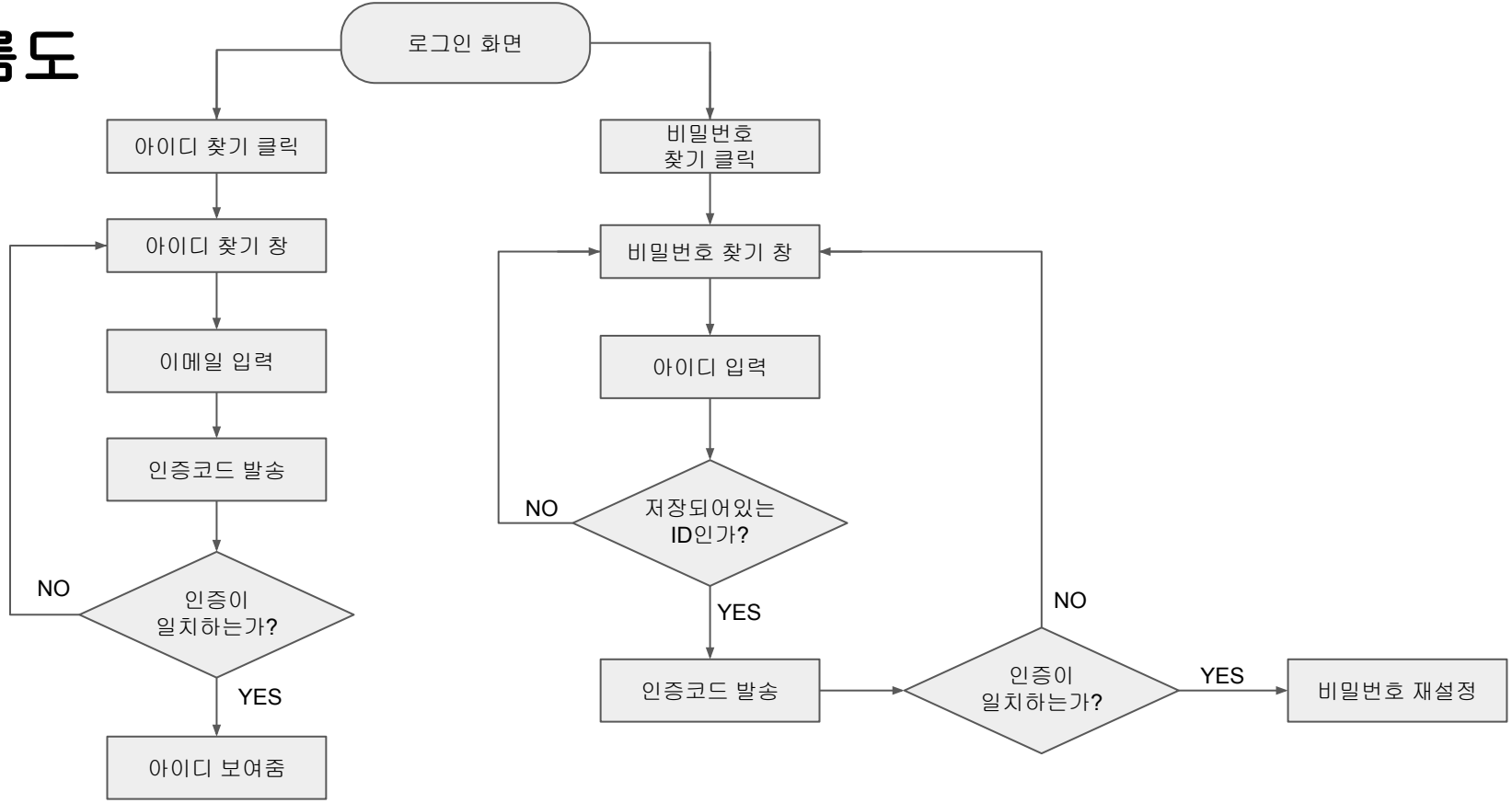
  - **Access Token이 유효x**
    Refresh Token이 유효
    Refresh Token이 DB에 존재x

    HTTP 401 ERROR 응답

  - **Access Token이 유효x**
    Refresh Token이 유효x

    HTTP 401 ERROR 응답

흐름도

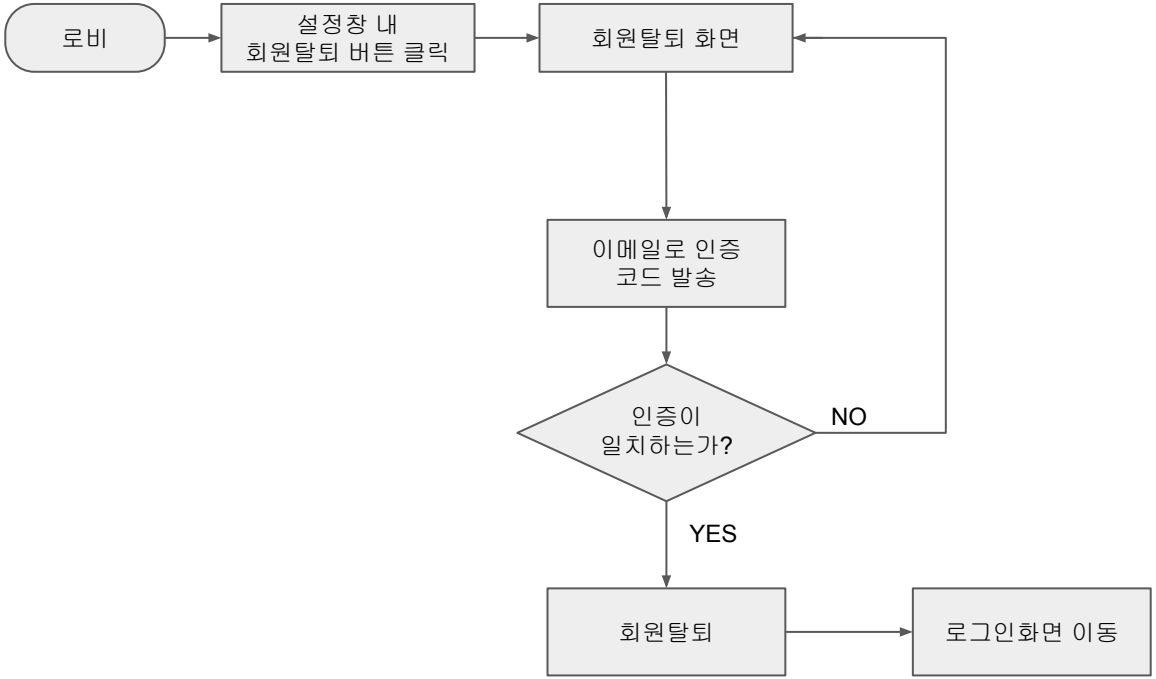


아이디/비밀번호 찾기

인증(아이디 찾기, 비밀번호 찾기) - Auth Server

- 이메일로 인증 코드 발송(SMTP 프로토콜)
- 아이디 찾기 : 이메일과 연동된 아이디 RETURN
- 비밀번호 찾기 : 비밀번호를 재설정 할 수 있음

인증(회원 탈퇴) - 흐름도


로비
설정창 내
회원탈퇴 버튼 클릭
회원탈퇴 화면
이메일로 인증
코드 발송
인증이
일치하는가?
NO
YES
회원탈퇴
로그인화면 이동

인증(회원 탈퇴) - Token Server

- 이메일로 인증 코드 발송(SMTP 프로토콜)
- 사용자가 갖고 있는 Refresh Token을 DB에서 삭제
- DB에서 사용자 정보 삭제
- Access Token 유효기간은 짧기 때문에 리스크 최소화 가능

인증(관리자) - Admin Token Server

- 관리자는 DB에 직접 등록 (회원가입 없음)
- 관리자는 웹 페이지에서 로그인 가능
- 관리자는 사용자 관리 API를 사용할 수 있음

  사용자 조회 | 사용자 삭제 | 사용자 수정

- 관리자는 채팅 로그를 확인하는 API를 사용할 수 있음
- 관리자 로그인 트래픽은
  몰리지 않을 것이므로 인증은 세션 활용

  세션을 Redis에 저장 | 관리자가 API를 사용할 때, Redis에 세션있는지 확인

채팅 서버

채팅 아키텍처

- python의 asyncio, socket, socketserver 모듈만으로 서버를 구현하는 것은 어려움이 있음

  → **ASGI의 application() 객체를 직접 구현하는 것은 어렵다고 판단**

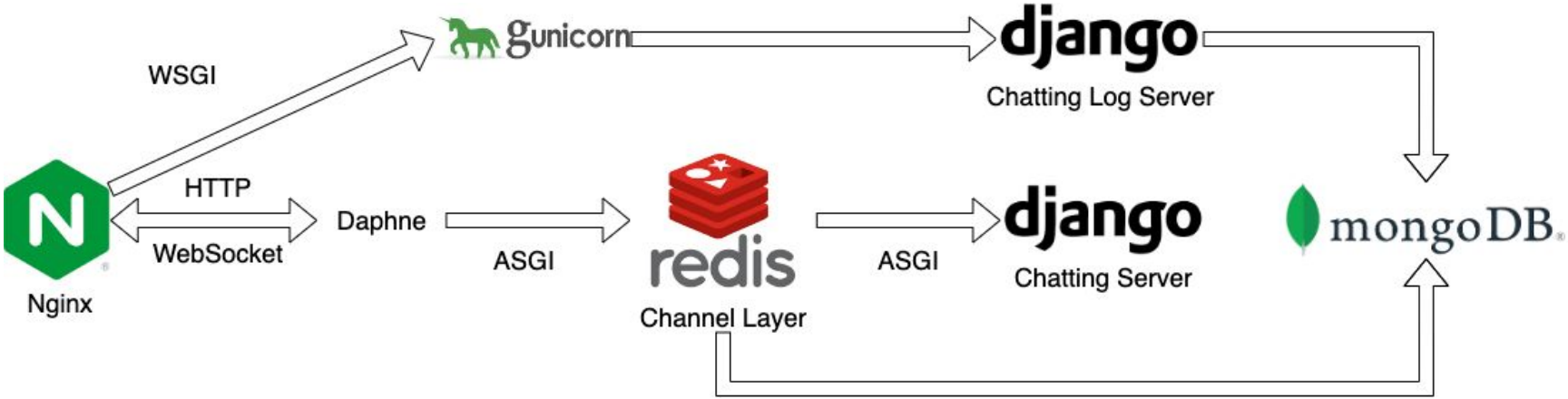
- ASGI를 지원하는 여러 framework 존재 (django, FastAPI, tornado 등등..)

  → **그 중 익숙한 django를 선택**

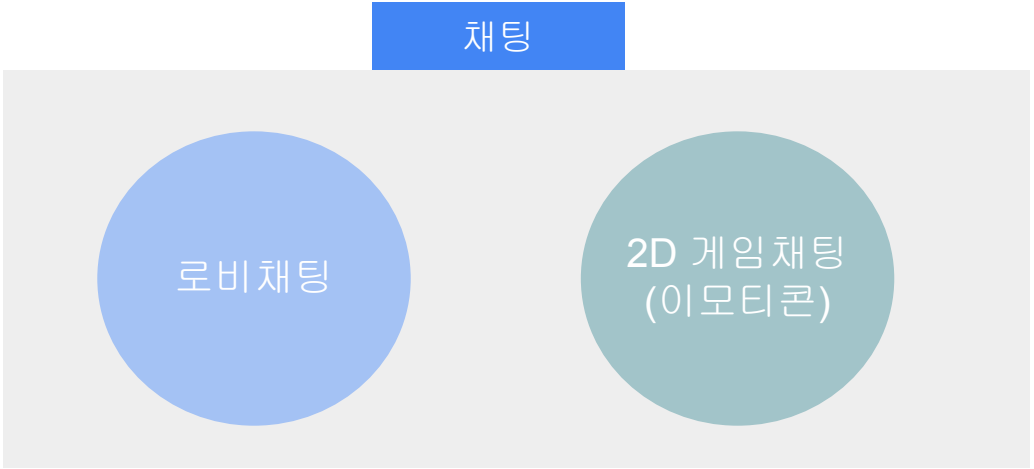
- ASGI를 사용하는 것은 처음이기 때문에 라이브러리의 도움을 받기로 결정

  → **channels 선택**

채팅 아키텍처


gunicorn
WSGI
django
Chatting Log Server
HTTP
WebSocket
Daphne
ASGI
redis
Channel Layer
ASGI
django
Chatting Server
mongoDB.
Nginx

채팅



추가할 기능?

- 관리자가 로비 채팅방에 공지할 수 있음
- 관리자가 게임방에 공지할 수 있음
- 귓속말 기능 (1:1 채팅)

채팅(로비)

- 접속한 유저들이 채팅을 확인하고 입력할 수 있음
- 채팅 로그를 DB에 저장 (빅데이터이므로 NoSQL)

채팅(2D게임)

- 각 게임에 참가하는 유저들은 서로 이모티콘을 통한 채팅이 가능
- 카드 숫자를 알면 안되기 때문에 일반적인 채팅은 불가능
- 로그(이모티콘 내용)를 남길 필요 없음

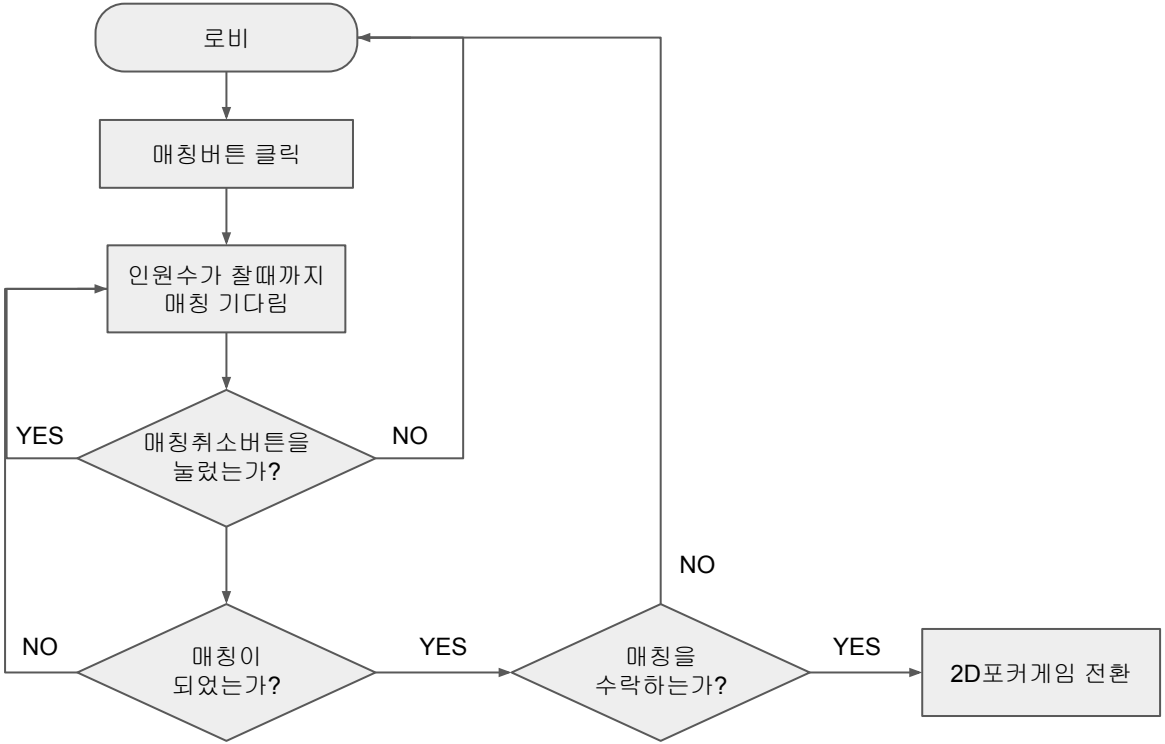
채팅(로그)

- 관리자는 채팅 로그 목록 API 호출 가능

질문 : 다른 DB를 접근하는 API들은 서로 다른 API Server에 만들어야 할까요?
e.g) 채팅 로그에 접근하는 API는 mongoDB 접근,
사용자 정보에 접근하는 API는 PostgreSQL 접근

로비 서버

흐름도


로비
매칭버튼 클릭
인원수가 찰때까지
매칭 기다림
YES
매칭취소버튼을
눌렀는가?
NO
NO
매칭이
되었는가?
YES
매칭을
수락하는가?
NO
YES
2D포커게임 전환

매칭

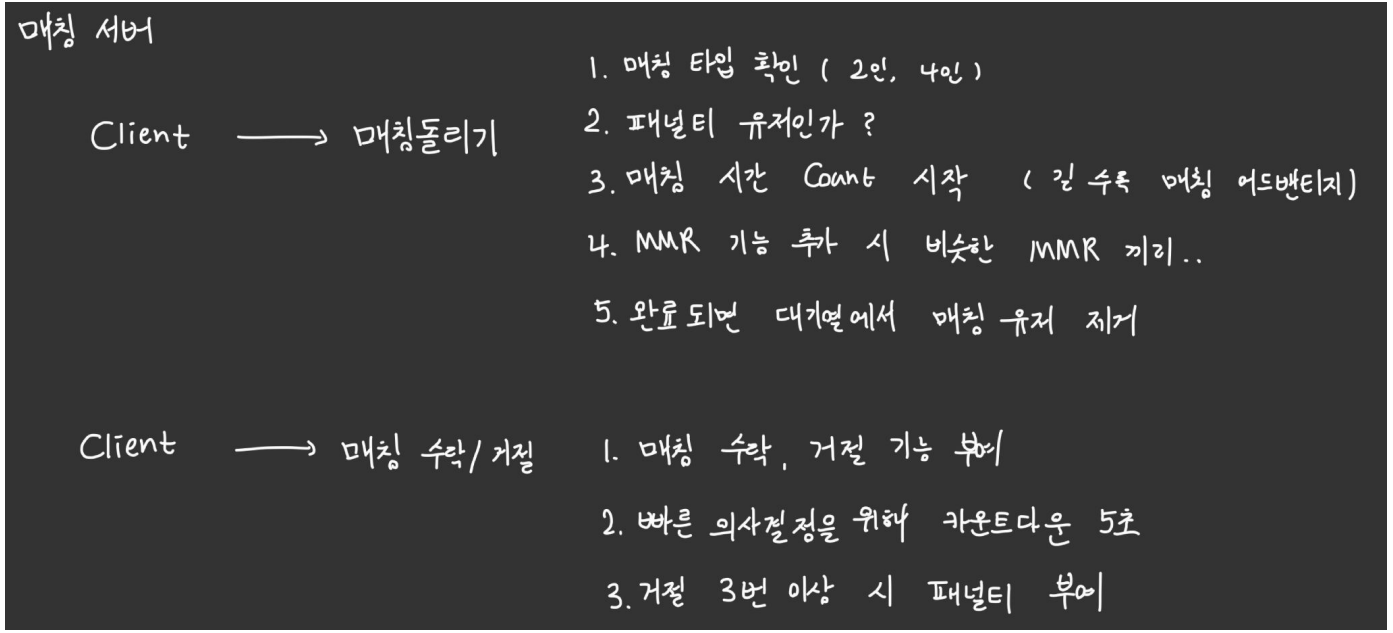
로비 서버

역할 : 매칭 대기 장소 + 매칭 서버 역할

- 게임 매칭 (매칭, 매칭 취소)
- 매칭 수락 및 거절
- 게임으로 접속

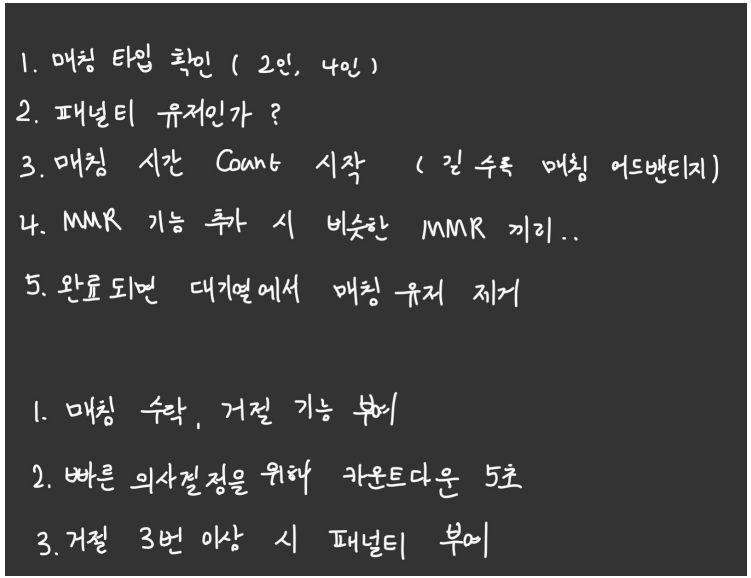
로비 서버

유스케이스 설계



로비 서버

도메인 설계



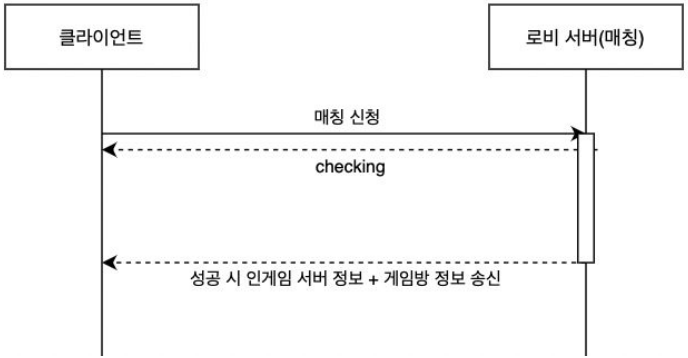
Matching

- Matching_Type
- Matching_Start_Time
- User (unique_Id, 패널티 정보)
- Reject_Count

로비 서버

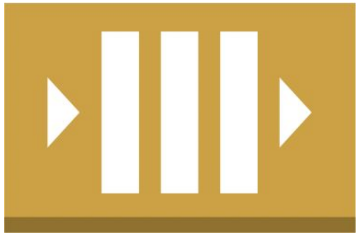
매칭 대기열

대기열은 어떻게 관리해야 하나?



우리가 바라는 대기열이 해줄 일

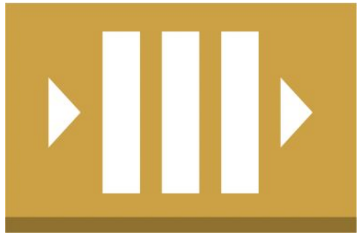
- 매칭 순서 부여 (선착순 매칭)
- 서버가 소화할 수 있는 만큼 대기열 처리



Queue!

로비 서버

매칭 대기열 -



BlockingQueue

- Blocking을 통해 Thread Safe을 보장할 수 있다.
- session 저장이므로 scale-out 환경에서는 사용 어려움.

- 해결 가능!
- 대신 데이터 정합성 문제 주의

로비 서버

redis - sorted set

요청한 순서대로 정리 → **ZADD**

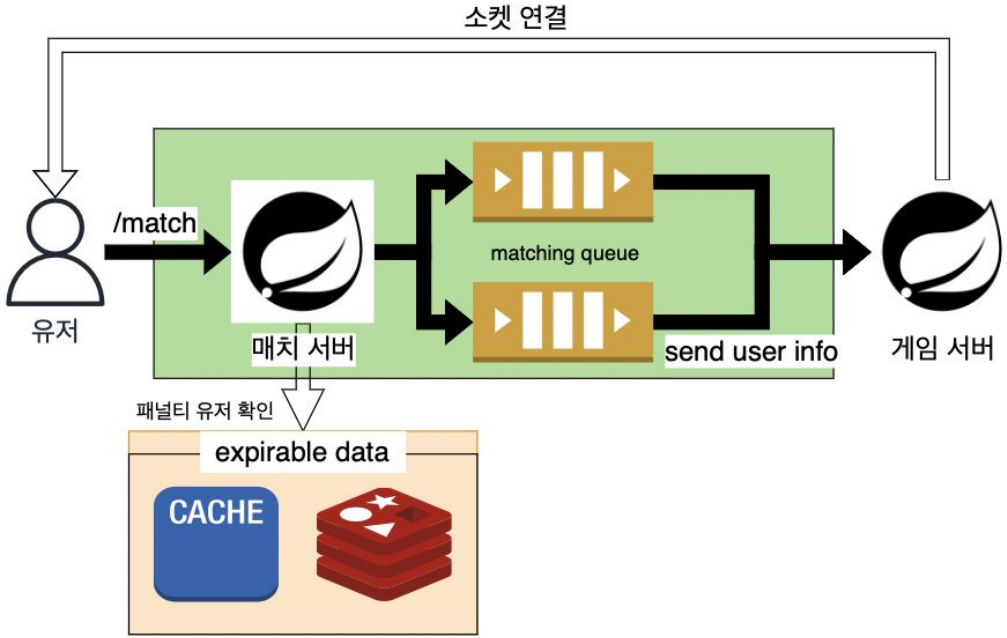
일정한 수만큼 처리 가능 → **ZRANGE**

단…

- KEYS 명령어 사용 시 풀스캔 발생! O(n)
- sorted set은 양이 많아질수록 성능 저하가 뚜렷함.

로비 서버

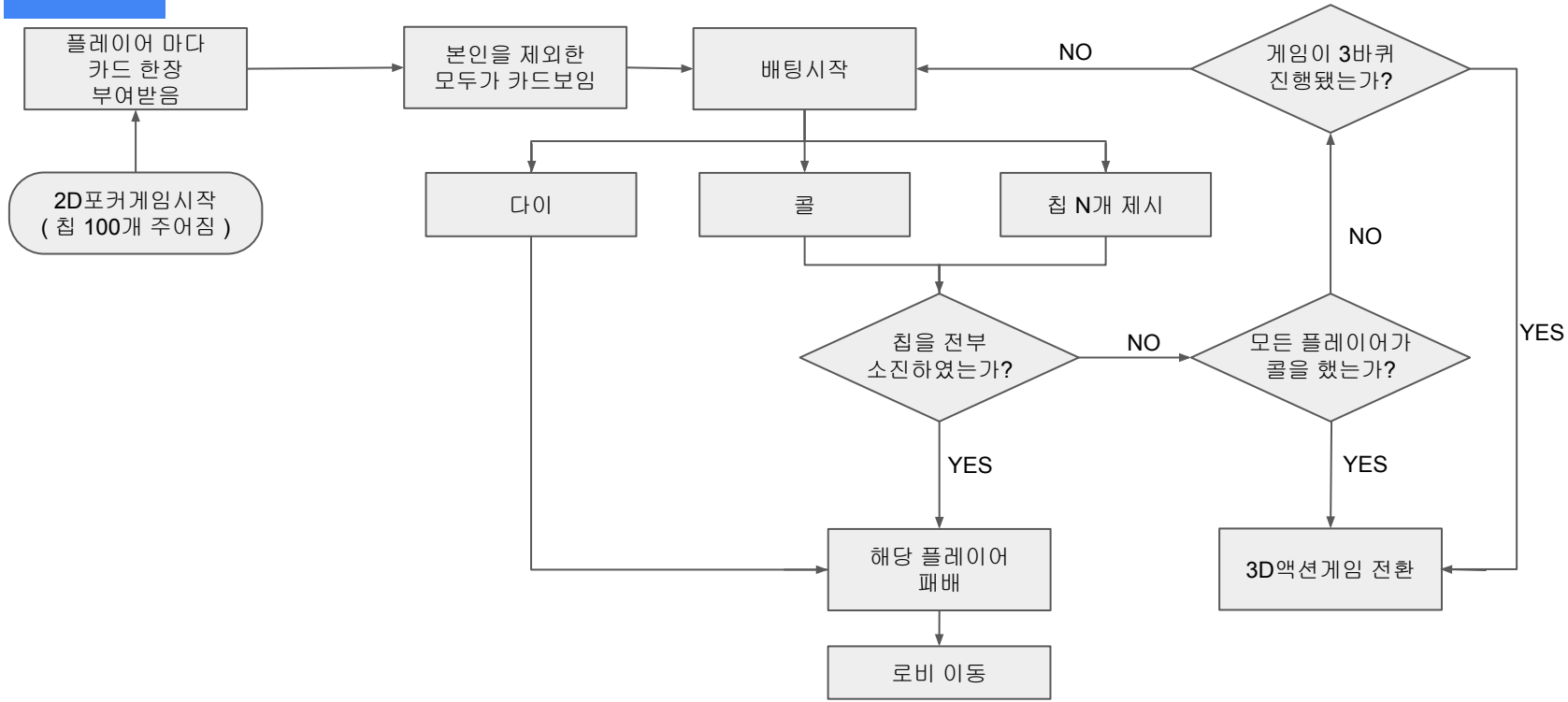
매칭 아키텍처


소켓 연결
/match
유저
매치 서버
matching queue
send user info
게임 서버
패널티 유저 확인
expirable data
CACHE

2D 포커 게임 서버

포커게임

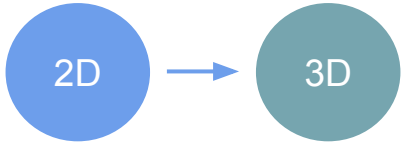
흐름도


플레이어 마다
카드 한장
부여받음
2D포커게임시작
(칩 100개 주어짐)
본인을 제외한
모두가 카드보임
배팅시작
다이
콜
칩 N개 제시
칩을 전부
소진하였는가?
NO
YES
모든 플레이어가
콜을 했는가?
NO
YES
게임이 3바퀴
진행됐는가?
NO
YES
해당 플레이어
패배
로비 이동
3D액션게임 전환

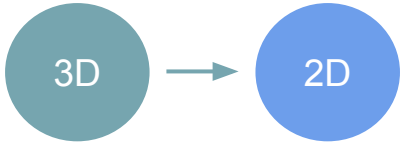
게임 서버

역할

첫번째 게임(포커) 진행



두번째 게임(난투) 넘어갈때
데이터 넘겨줌



두 번째 게임(난투) 끝날때
결과 데이터 넘겨받아 검증 후 칩 배분

게임 서버

유스케이스 설계

게임진행

- 카드 받기
- 자신의 턴이 되면 배팅
- 다른 사람들의 배팅 정보 조회
- 포커 단계가 끝나면 난투 단계를 진행할 서버로 데이터 전송 [유저 정보, 무기 정보]
- 난투 단계(3D게임)가 끝나면 승자 정보를 받아 칩 배분

게임 서버

유저간 통신

- 실시간이면 좋다.
- 하지만 각자의 턴이 명확하게 존재하므로 각자의 턴이 끝날때만 데이터를 모두에게 전달해줘도 됨
  (굳이 지속적으로 데이터를 받아오지 않아도 된다.)

WebSocket

Http Polling

SSE

게임 서버

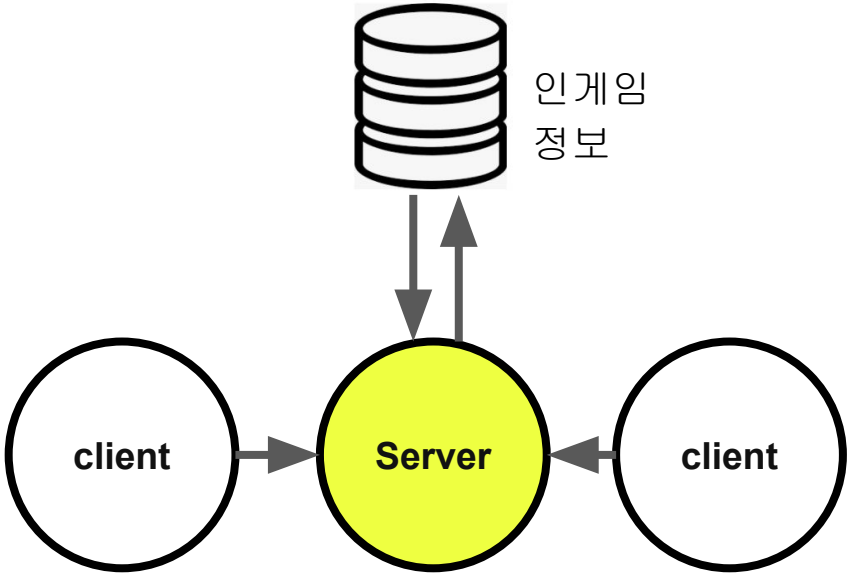
게임 데이터 저장

Room_ID : …
User_ID : …
Chip : …

2D -> 3D 전환 시 필요
(고정 세션 X 시)

[2023.01.01T11:30:00]
… user가 …chip 획득

history 용으로도 기록
필요

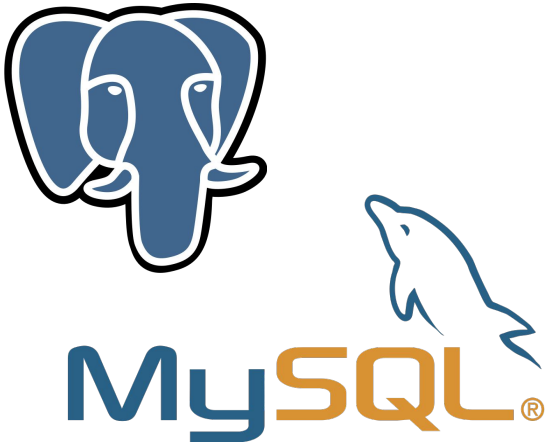


위변조 cross check

게임 서버

게임 데이터 저장

데이터 저장은 어떻게?



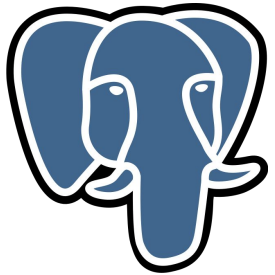
게임 서버

게임 데이터 저장(1)

PK	room_id	user_id	chip	status	created_at
…	1	1		normal	10:00:00

PK	room_id	user_id	chip	status	created_at
…	1	2		normal	10:00:00

PK	room_id	user_id	chip	status	created_at
…	1	1		loose	10:05:00



게임 서버

게임 데이터 저장(2)

예상 포맷1

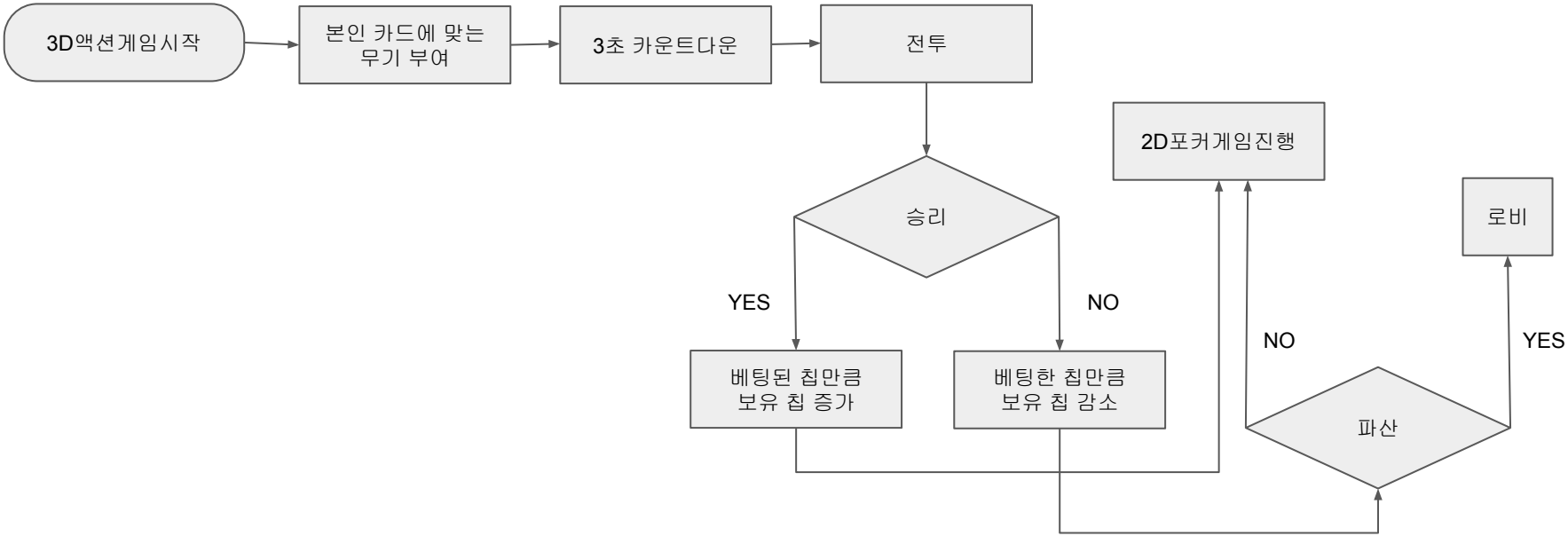
```
room_id : …
turn: 1
users : [
        user : { user_id: …, chip: … },
        user : { user_id: …, chip: … },
        user : { user_id: …, chip: … },
        user : { user_id: …, chip: … }
]
timeStamp: …
```

예상 포맷2

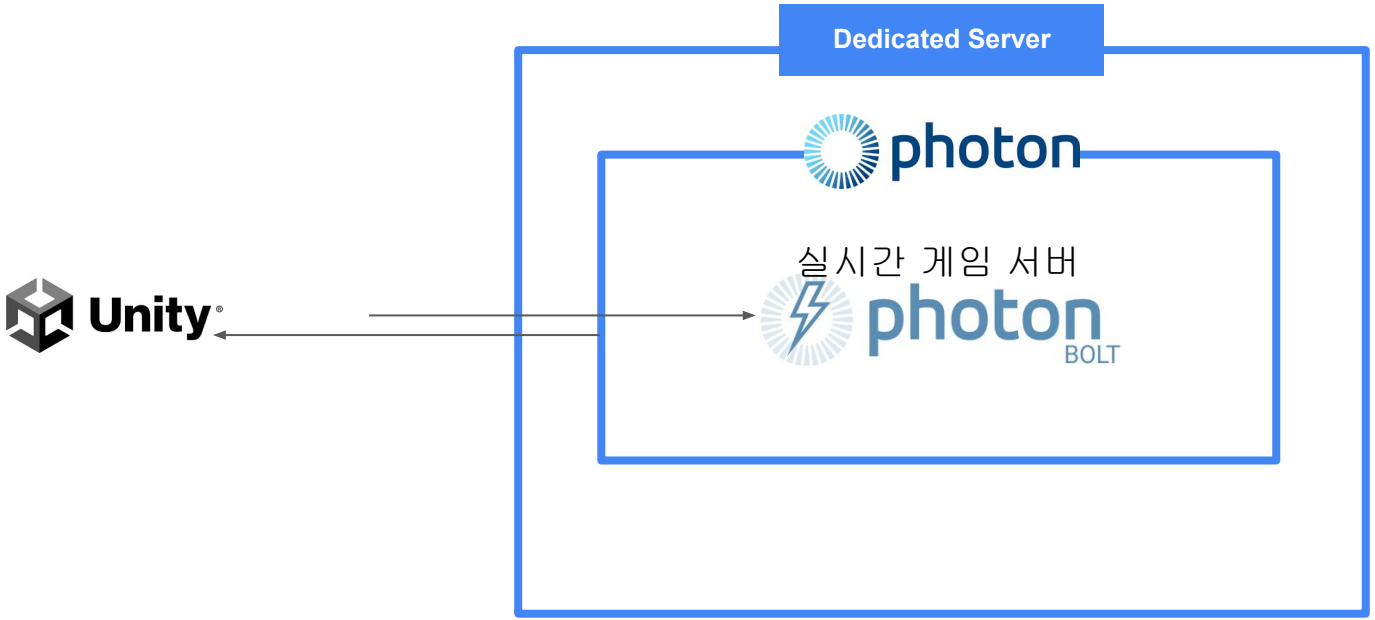
```
room_id : …
logs: [
        {
                users : [
                        user : { user_id: …, chip: … },
                        user : { user_id: …, chip: … },
                        user : { user_id: …, chip: … },
                        user : { user_id: …, chip: … }
                ]
                timeStamp: …
        }, …
],
recent_data: {
        users : [
                user : { user_id: …, chip: … },
                user : { user_id: …, chip: … },
                user : { user_id: …, chip: … },
                user : { user_id: …, chip: … }
        ]
        timeStamp: …
}
```

3D 액션 게임 서버

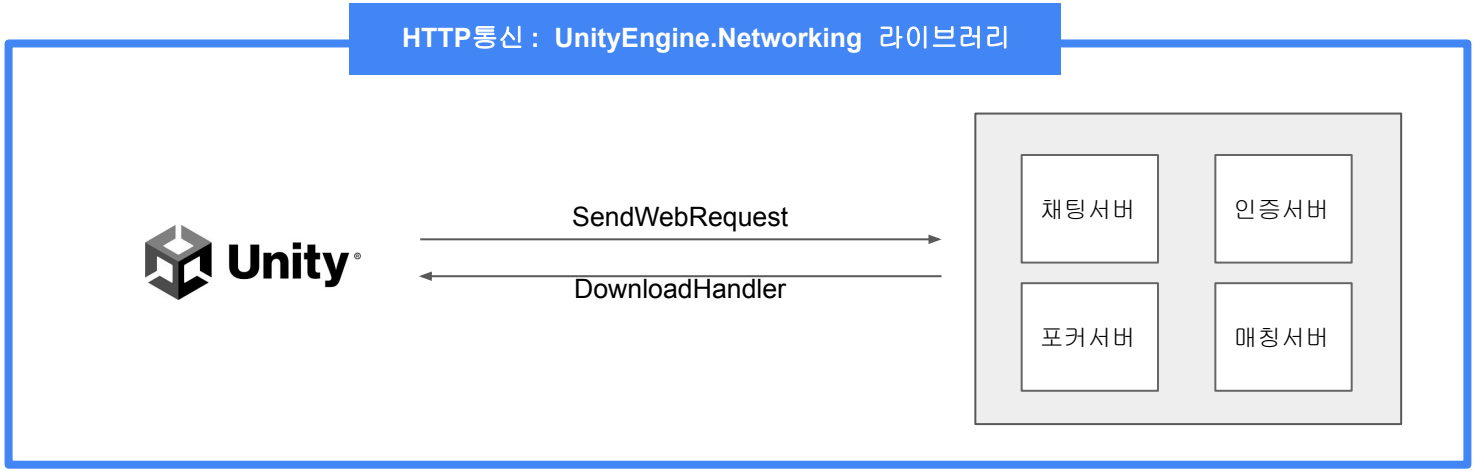
흐름도 (3D액션 시스템)


3D액션게임시작
본인 카드에 맞는 무기 부여
3초 카운트다운
전투
승리
YES
NO
베팅된 칩만큼 보유 칩 증가
베팅한 칩만큼 보유 칩 감소
2D포커게임진행
NO
파산
YES
로비

실시간 게임서버



클라이언트와 백엔드연결



업무분담 및 개발계획

현명은

- 로비
- 2D 포커게임
- 로그인/회원가입

~ 1.22 : **화면설계 및 구현**
1. 회원가입/ 로그인 화면 구현
2. 매칭화면 구현
3. 2D포커게임 화면구현

~ 1.30 : **클라이언트 백엔드 연결**
1. 회원가입/로그인 백엔드 연결
2. 매칭 백엔드 연결

~ 2.7 : **게임기능 구현 및 연결작업**
1. 2D게임 구현
2. 2D게임 백엔드 연결
3. 2D게임 3D게임 전환

~ 2월중순 : **성능개선, 프로파일링 및 개선작업**

조현민

- 실시간 서버
- 3D 액션게임

~ 1.18 : **레벨 디자인**
1. Lo-Fi Prototype
2. 유니티 적용

~ 1.30 : **화면설계 및 구현**
1. 3D게임 화면구현
2. 3D게임 기능구현

~ 2.13 : **게임기능 구현 및 실시간 서버 연결**
1. 실시간 서버 연결
2. 2D게임 백엔드 연결

~ 2월중순 : **성능개선, 프로파일링 및 개선작업**

업무분담 및 개발계획

박동진

- 인증 서버
- 채팅 서버

~ 1.27 : **인증 서버 구현**
- -로그인, 회원가입
- -아이디 찾기, 비밀번호 찾기
- -이메일 인증
- -회원 탈퇴
- -관리자 로그인
- -관리자의 유저 관리 API
- -관리자의 채팅 로그 API

~ 2.10 : **채팅 서버 구현**
- -채팅 목록 API
- -채팅 입력 API

~ 2월중순 : **성능개선, 프로파일링 및 개선작업**

구자현

- 로비 서버
- 2D게임 서버

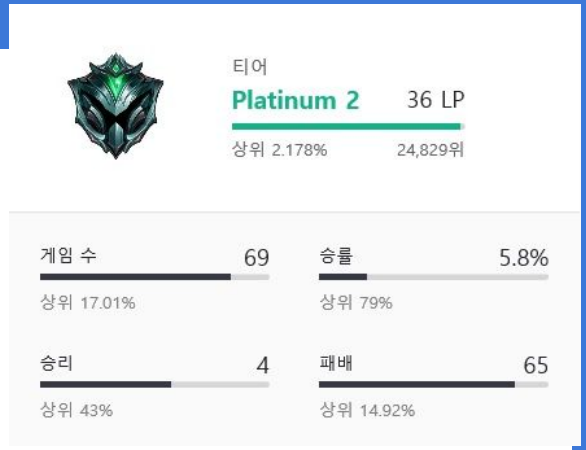
~ 1.27 : **로비 api 구현**

~ 2.7 : **2D 게임 서버 구현**

~ 2월중순 : **성능개선, 프로파일링 및 개선작업**

End Of Document

도메인설계

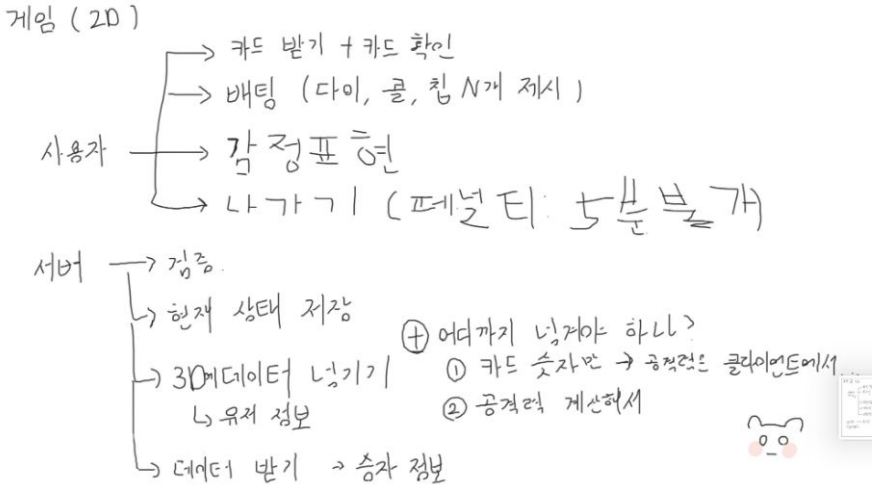


게임(2D -> 3D)
방번호, 현재참여하고있는 유저, 유저의 카드 숫자, 게임 종결 여부

게임(3D -> 2D)
방번호,승자여부 JSON형태로 보내기

3D-> 2D 서버에서 남은 돈 판단 -> 로비
몇 등(패배) 띄우고 로비?, 최후의 1인(승리) 띄우고 로비?
유저 테이블 수정(전적) API 요청

- 현재 참여하고있는 유저를 DB에? Cache에?
- 3D 게임 로딩 시간을 줄이기 위해 Read Performance가 뛰어난 NoSQL 선택 (2D 게임이 진행될 동안 Update, Delete를 실행하면 되므로 느려도 됨)
- 대신 3D 게임이 처음 시작 될 때, Create이 느린 것은 리스크, 하지만 Read가 훨씬 많기 때문에 결과적으로 이득
- Schema 없이 NoSQL 사용해서 2D -> 3D 데이터 넘겨줌



[아키텍처] 문서 작성 안내

윈터데브캠프 Empty

Add a property

조 Add a comment...

[아키텍처] 문서작성 안내

아키텍처 리뷰는 PPT 형태로 만들면 됩니다.
보통 PMP문서에서 아키텍처, 설계 부분을 추가해서 중간 발표 자료가 되고,
그것이 계속 발전해서 최종 발표 문서가 됩니다.
물론 편집 방향은 좀 달라져야 합니다.
PMP는 목표 설정이 중심이기 때문에 그 항목이 초반에 있지만,
프로젝트 발표는 "프로젝트 설명" 중심이 되어야 합니다.
목표는 뒷부분로 옮겨야하죠.
여러분이 프로젝트 발표를 보는 입장이라고 생각해보시면,
여러분의 팀의 개인적인/팀 전체의 목표나 생각등은 궁금하지 않을겁니다.
"뭘 만들었지?" 가 궁금하겠죠.
그래서 최종 문서 형태로 갈수록...
프로젝트 소개, 아키텍처 소개, 기술적인 특징, 개별 담당 부분,
그 후에 팀의 목표했던 것, 그리고 회고 같은 것이 들어가는 구조가 좋습니다.
아키텍처 리뷰에서는 여러분 개인이나 팀의 목표는 보지 않습니다.
이미 PMP에서 충분히 보여주시고, 피드백 받으셨으니까요.
주제에 따른 기능, 전체 구조, 기술 스택, 업무 분담, DB 구조 정도를 기반으로
앞으로 개발할 모든 설계와 계획에 얘기하는 시간입니다.
그에 맞춰서 자료를 준비하시면 됩니다.
검사하는 시간이 아니라, 개발 방향성을 확정하는 시간이라고 생각하시면 됩니다.

[아키텍처] 설계 리뷰 안내

아키텍처 리뷰는 전체 (주로 백엔드가 되겠죠?), 프론트엔드(+모바일) 구조와 주요 부분 Flow, 그리고 오픈 소스를 사용할 경우 어떤 것을 어떤 용도로 쓸지, DB 스키마/ERD, 클래스 설계 등 설계와 관련된 모든 내용에 대해 준비된 것까지 같이 얘기하는 시간입니다.
팀 프로젝트에서 최고로 중요한 부분이라고 보시면 됩니다. 여기서 설계는 물론이고 업무 분담이나 그런 것까지 모두 얘기를 해보게 됩니다. 일단은 각 개인의 구현에 대한 얘기까지도 나눌 수 있습니다만, 너무 깊이 들어가거나 (혹은 오래 얘기해야 할 내용이면 별도로 멘토링을 진행하게 됩니다.
자료를 만들어서 보여주면서 진행해야 합니다. 한 명이 다 설명해도 되고, 각자 자기 분야에 대해 설명해도 됩니다. 질문할 게 있으면 미리 준비해 오는 것도 좋습니다. 최근 팀 미팅을 해보면 확실히 예전에 비해 질문이 늘어나고 있습니다.
다만, 단순 기술 관련된 부분은 별도 질문으로 하는게 좋습니다.
ex> Python 에서 UWSGI 로 연결했는데 Celely 가 작동이 안됩니다.
질문을 최대한 "팀 프로젝트 주제" 에 어울리게 해주세요.