

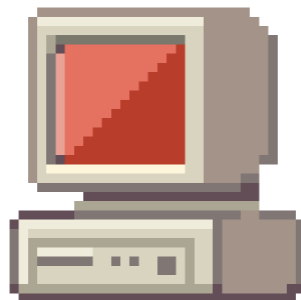
클라우드 게이밍 기술을 활용한 교육용 메타버스 플랫폼

금정산삼고라니 

팀원 소개



- 정현모
- 백엔드, 로깅 담당
- <https://github.com/gusah009>



- 이재욱
- Unity, DB 담당
- <https://github.com/shasuri>



- 전설
- Unity, 인프라, 프론트 담당
- <https://github.com/redundant4u>

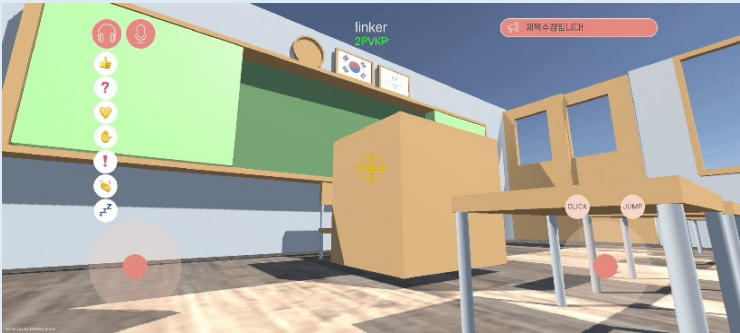
목차

1. 프로젝트 목표
2. 시연 영상
3. 전체 구조
4. 세부 구현
 - Unity, Signaling Server
 - Frontend
 - Backend
 - Logging

프로젝트 목표

[필요성]

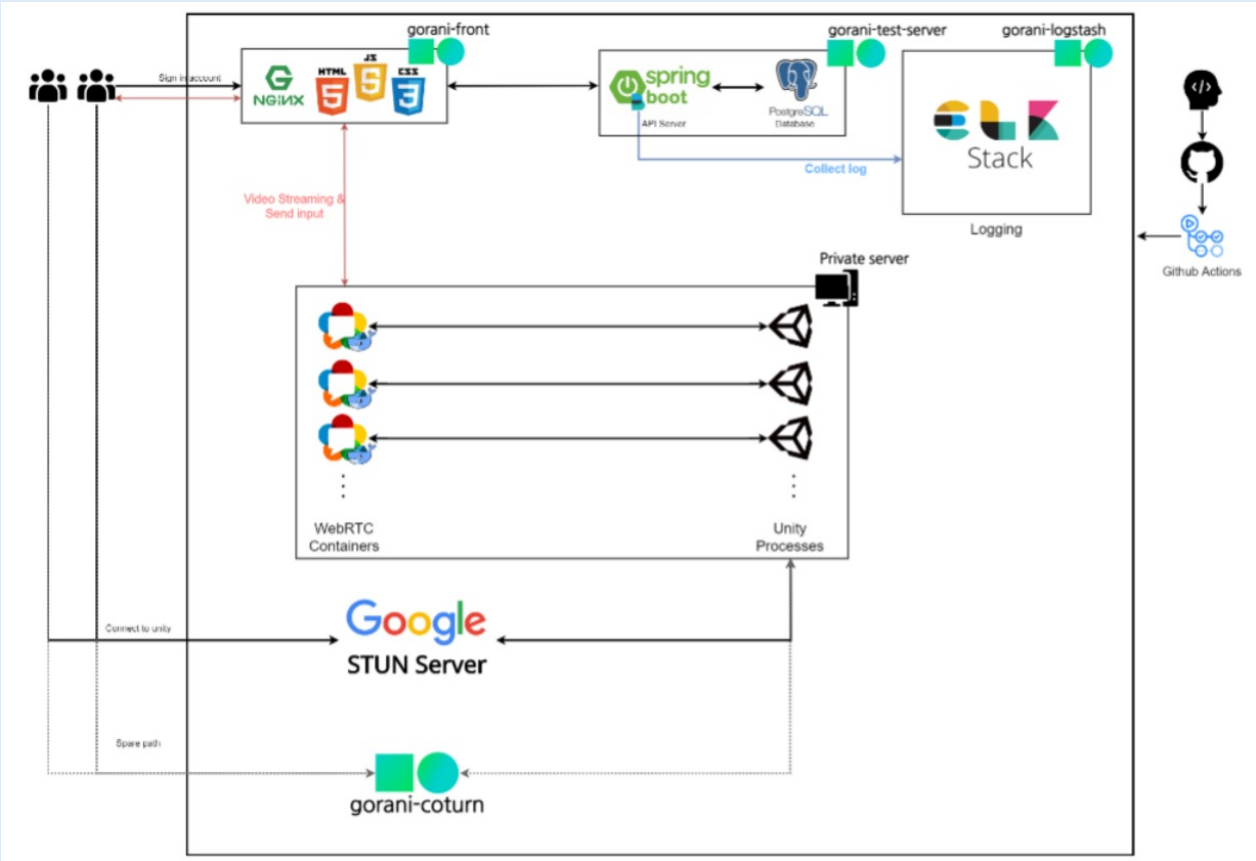
- 기존은 회의를 위한 플랫폼이 대다수임
- 학생끼리 친밀감을 형성하기에 부족함이 있음
- 정규교육과정을 위한 추가 편의기능 개발이 필요함



교실과 유사한 UI, 주변 학생과 상호 작용 등의 기능이 필요함

[클라우드 게이밍 기술을 활용한 메타버스 플랫폼]

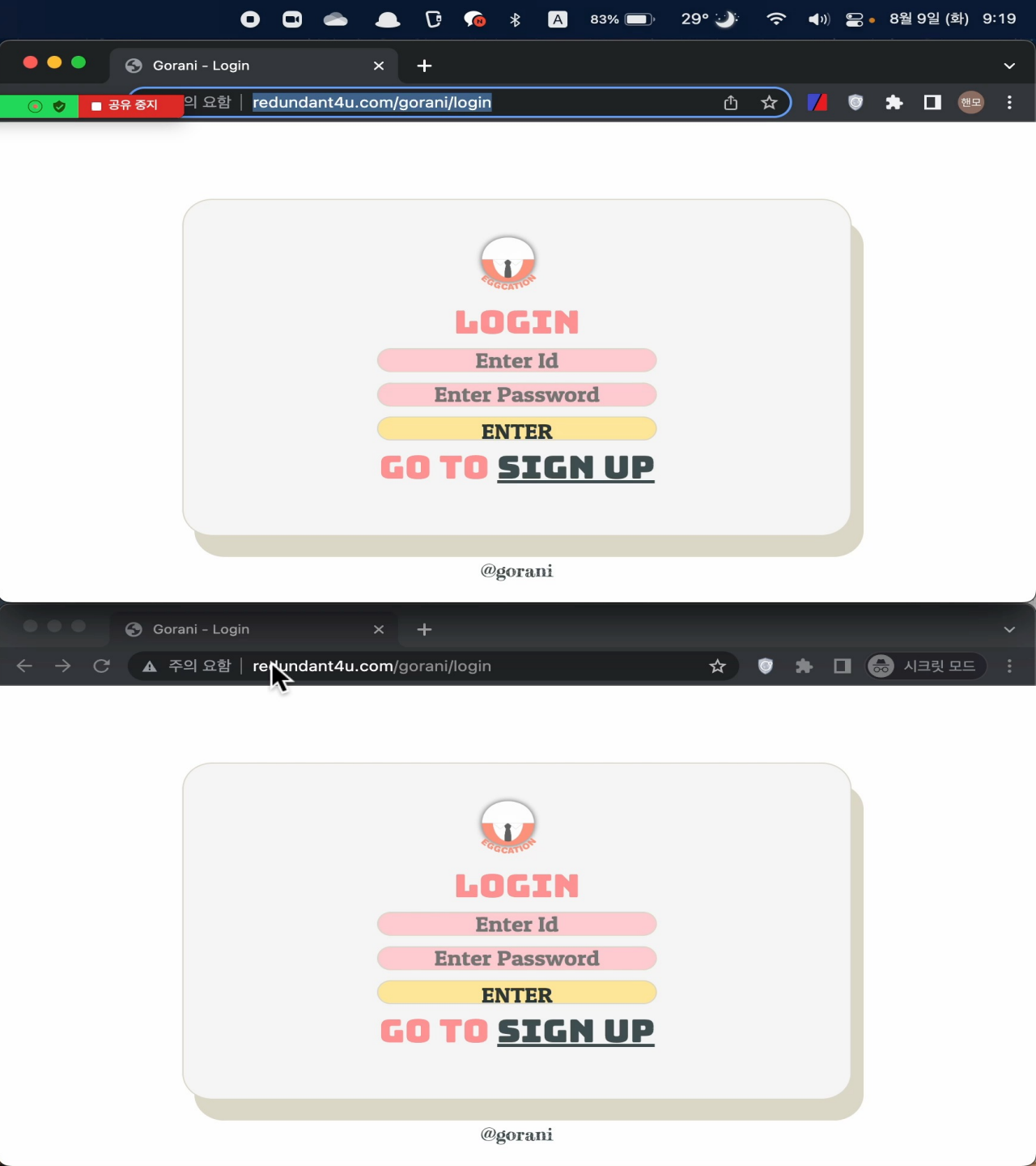
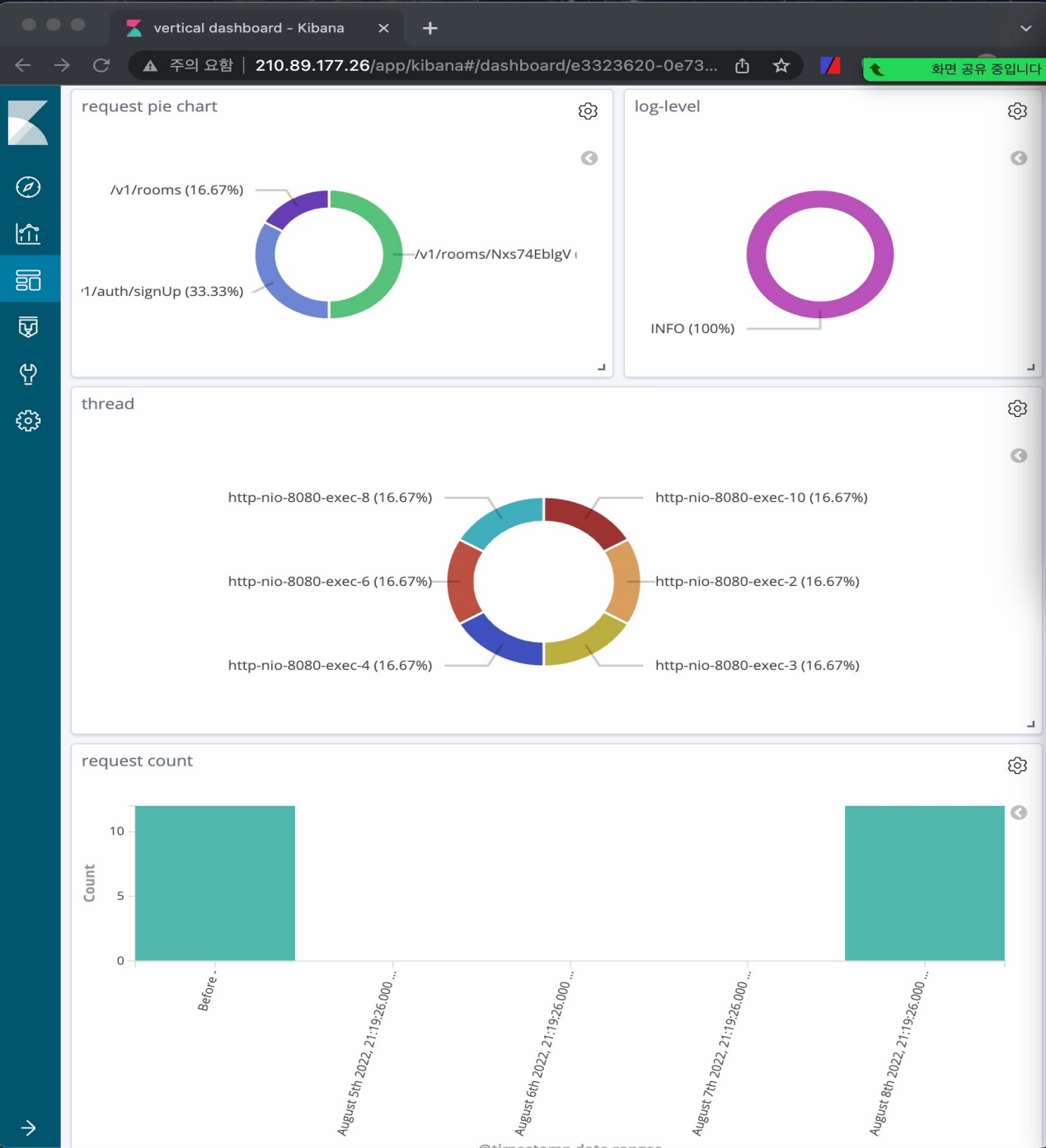
- Render Streaming 기술=> 하드웨어 사양에 따른 제약 없애기
- 여러 교실을 동시에 운영할 수 있는 안정적인 인프라 구축



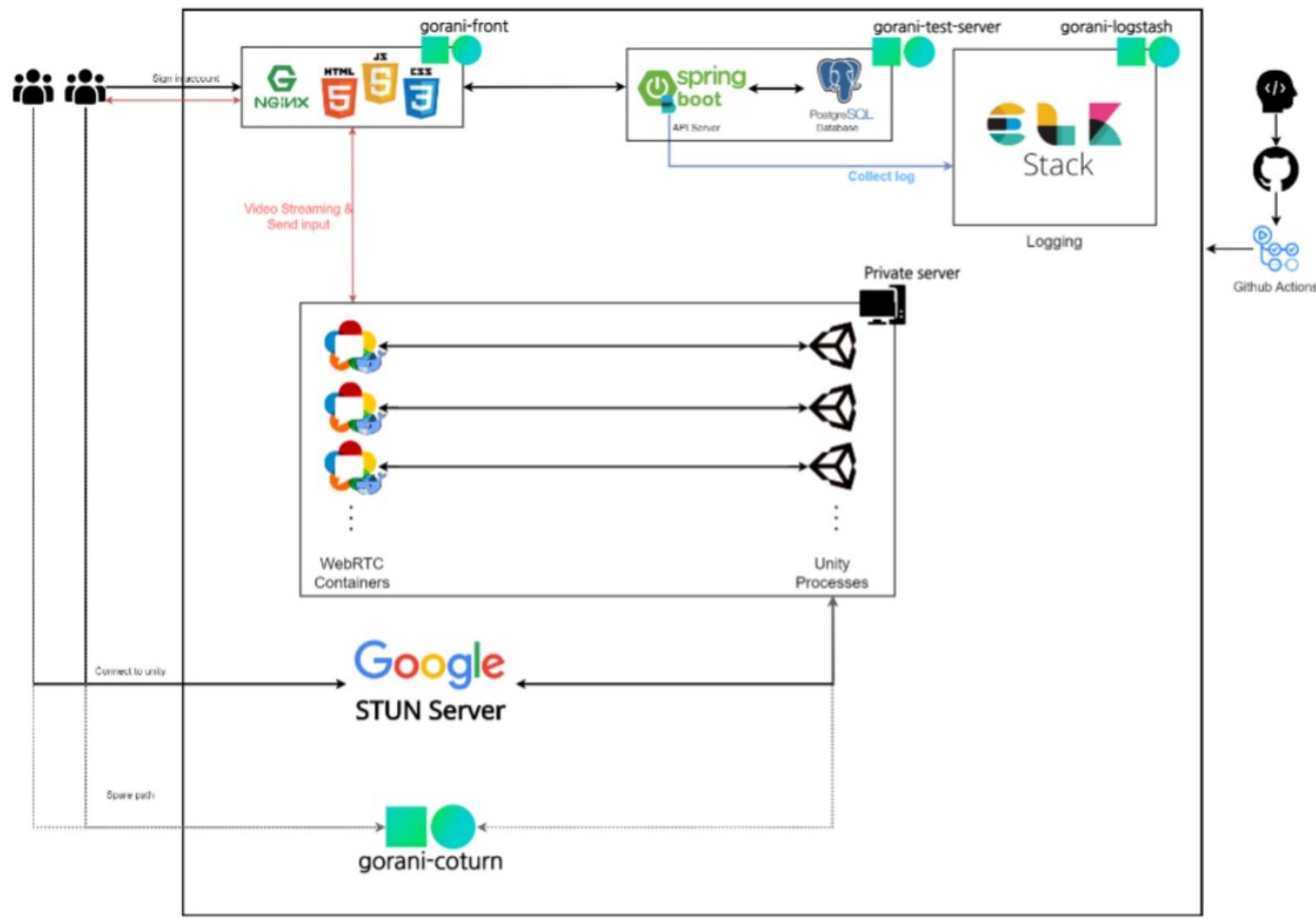
프로젝트 목표

프로젝트 난이도와 진행 상황을 고려하여 기본적인 기능 위주로 목표를 축소함.

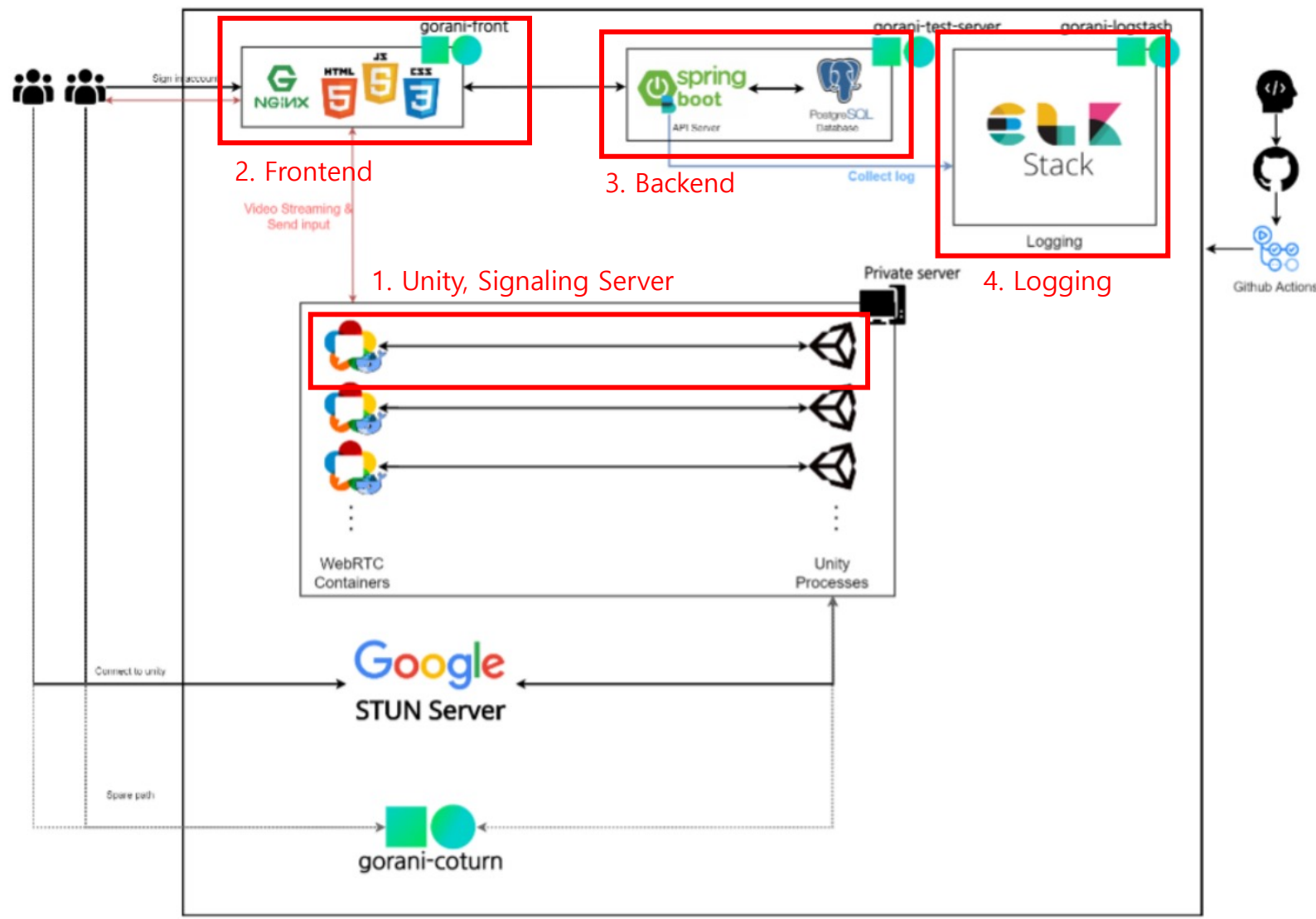
- ✓ 회원가입, 로그인 등 서비스 이용을 위한 기본적인 기능
- ✓ 교실 생성, 교실 입장 등의 교실에 연결되는 과정
- ✓ 클라우드에서 랜더링되는 교실 환경
- ✓ 사용자의 입력을 통해 캐릭터 제어



전체 구조



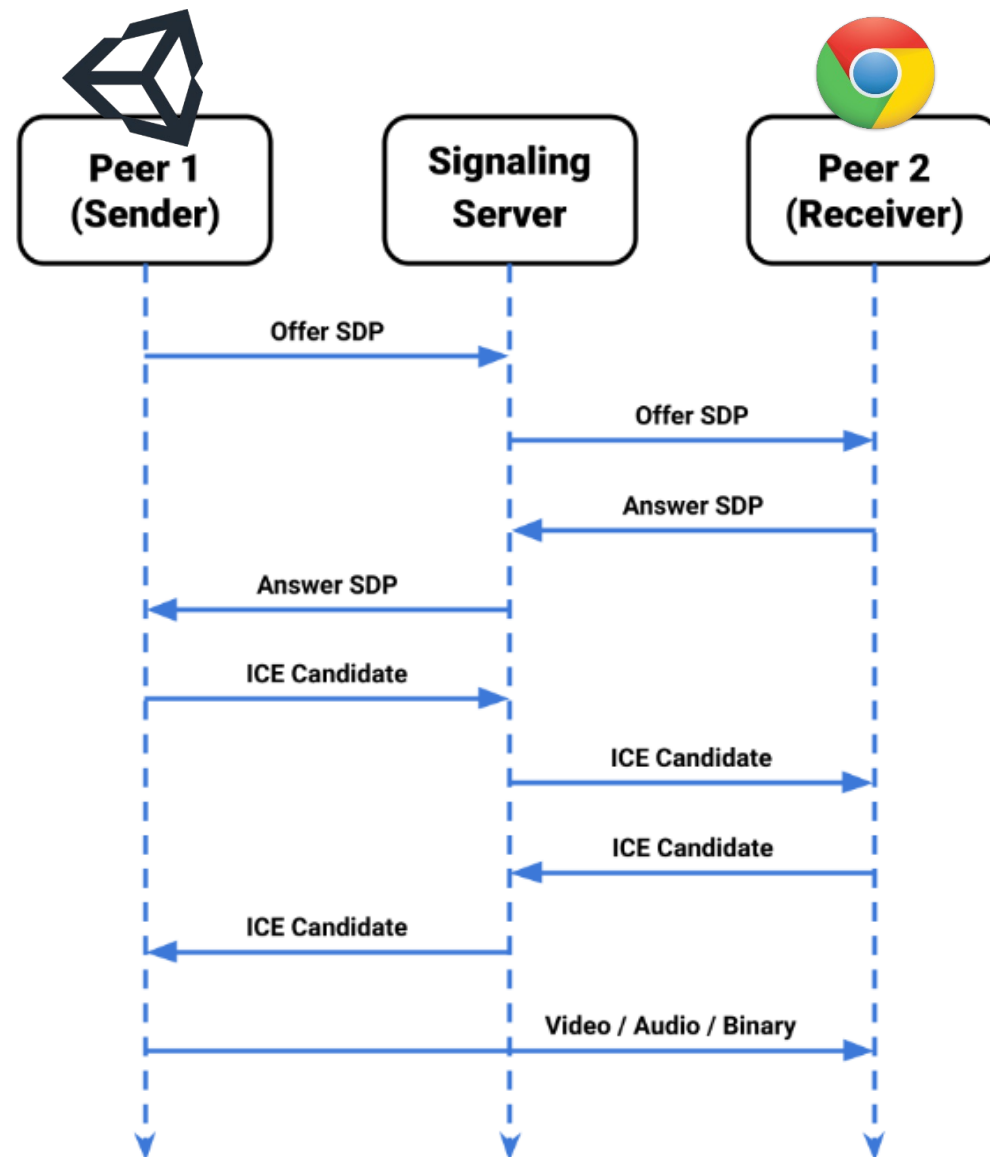
전체 구조



Render Streaming이란

Unity에서 랜더링된 화면을 네트워크를 통해 전송하여
사용자 브라우저에 전송하는 기술

- WebRTC 통신을 사용
- Signaling Server를 통한 connection이 맺어진 이후에는
Sender, Receiver간에 p2p 통신이 진행
- Signaling server는 STUN, TURN을 통해 가져온
Sender, Receiver의 ip, port 주소를 가지고 클라이언트 사이를
연결함



Unity, Signaling Server

[Sender]

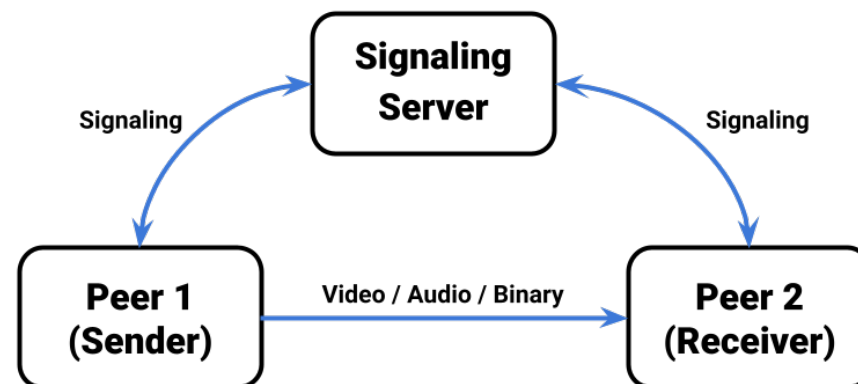
- Render Streaming 패키지 API를 활용하여 broadcasting, 입력에 따른 제어 구현
- 기본적인 교실 환경, 사용자 캐릭터 구현
- https://github.com/9orani/unity_server

[Signaling Server]

- Typescript를 사용하여 SDP, ICE 프로토콜 구현
- <https://github.com/9orani/webrtc>

[Receiver]

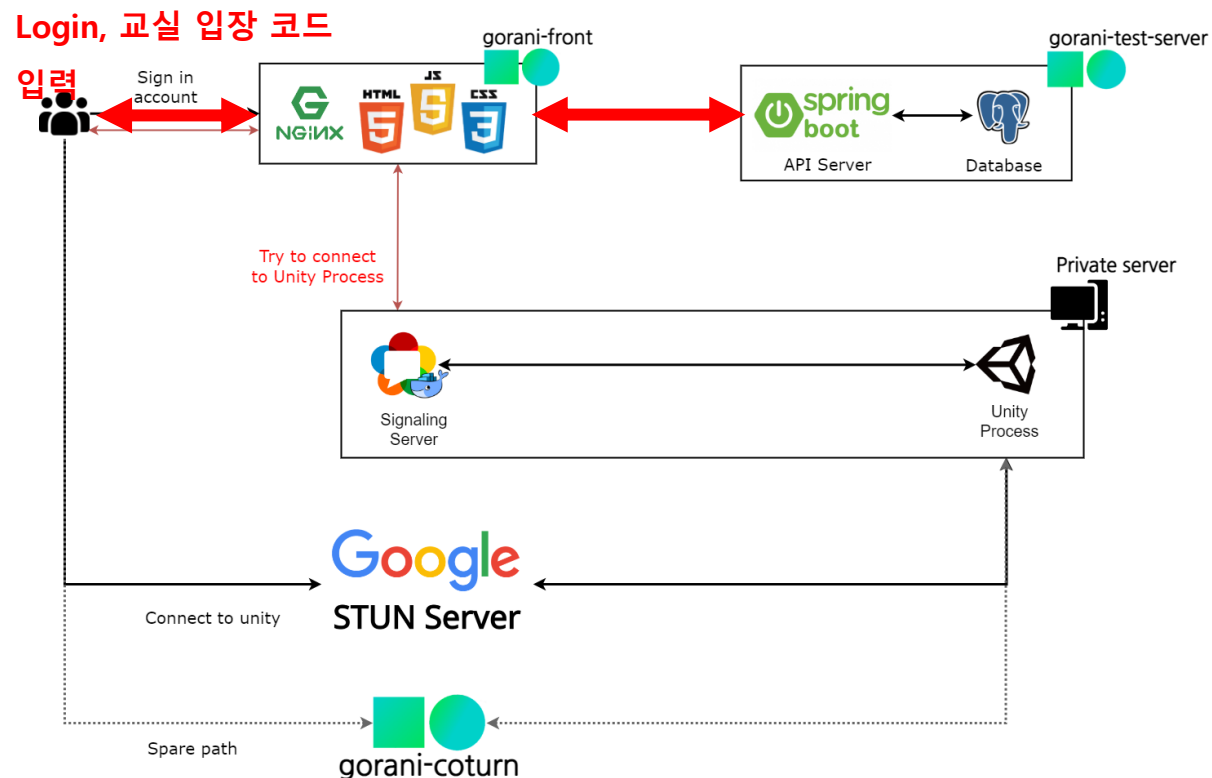
- Javascript를 사용
- 브라우저에서 제공되는 WebRTC API를 활용
- Unity video를 받고 사용자 입력을 전달하는 부분을 구현
- <https://github.com/9orani/front>



Unity, Signaling Server

[교실 입장과정]

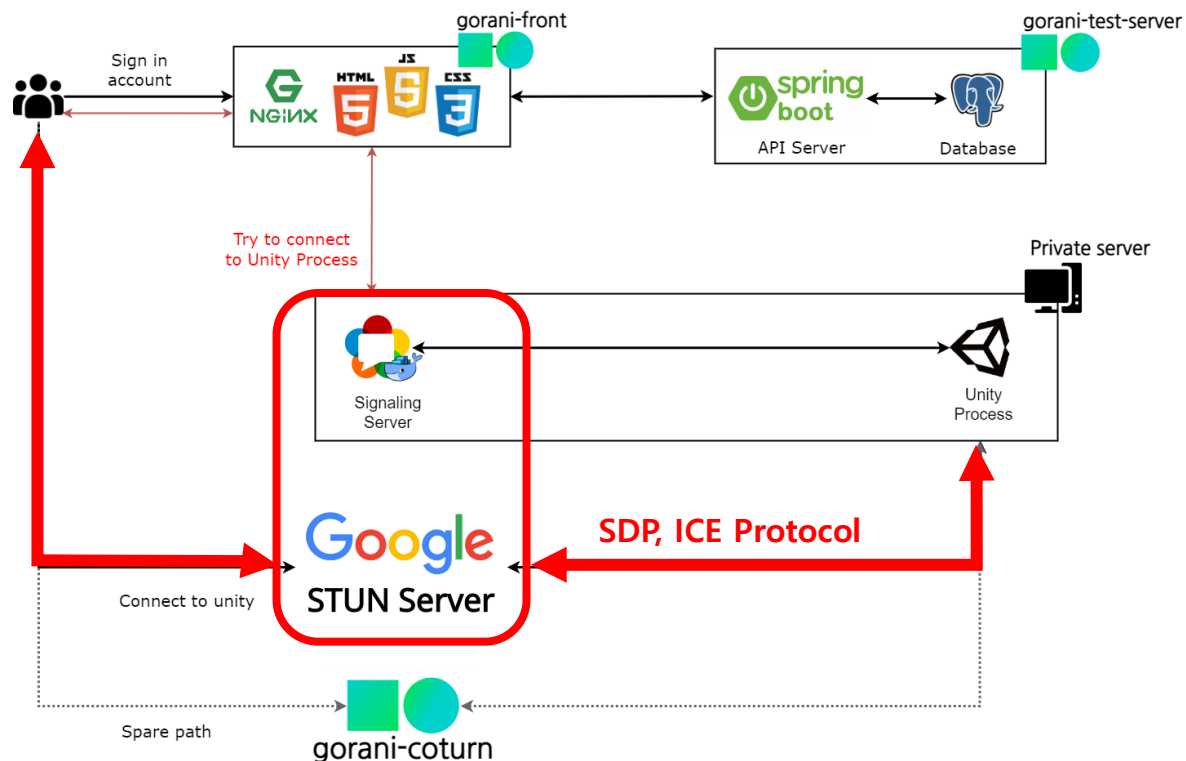
1. 로그인하여 교실 입장 페이지에 접근
2. 교실 입장 코드를 이용하여
교실에 매칭되는 signaling server주소를 획득
3. signaling server를 통해서
google STUN 서버를 활용한 p2p connection 시도
4. p2p connection 실패 시,
gorani-coturn 서버를 통한 relay connection



Unity, Signaling Server

[교실 입장과정]

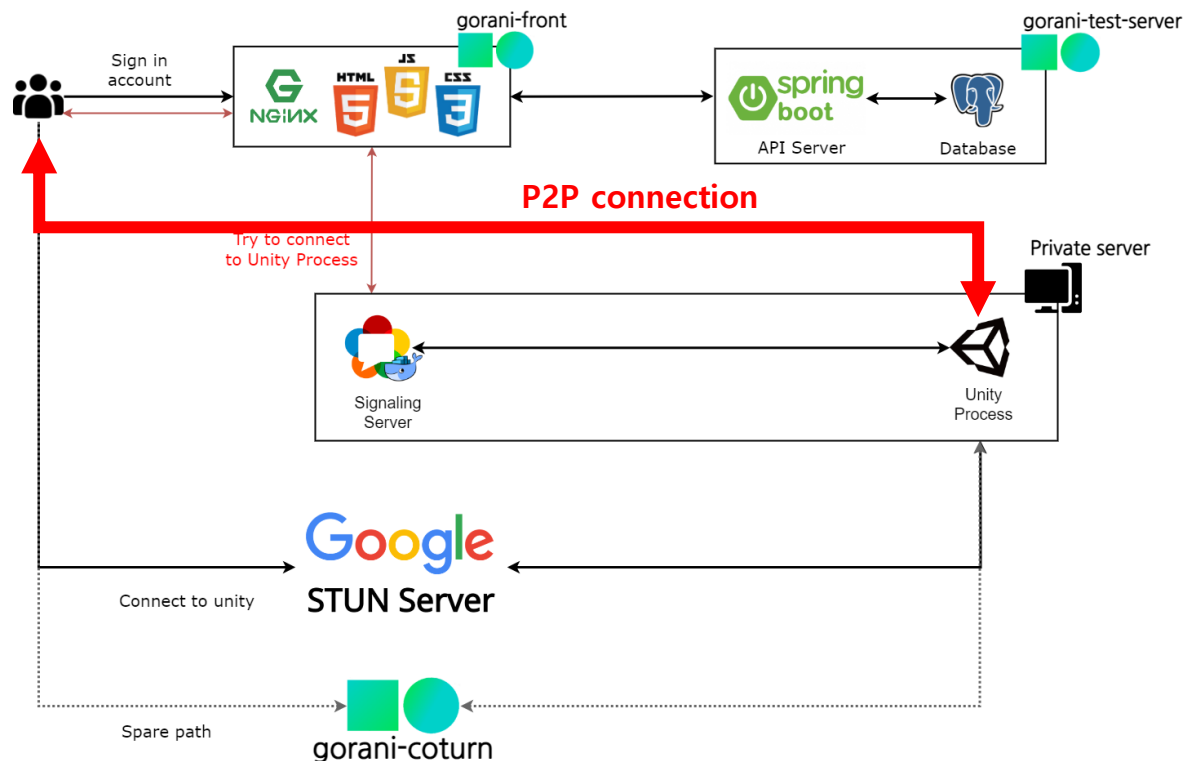
1. 로그인하여 교실 입장 페이지에 접근
2. 교실 입장 코드를 이용하여
교실에 매칭되는 signaling server주소를 획득
3. signaling server를 통해서
google STUN 서버를 활용한 p2p connection 시도
4. p2p connection 실패 시,
gorani-coturn 서버를 통한 relay connection



Unity, Signaling Server

[교실 입장과정]

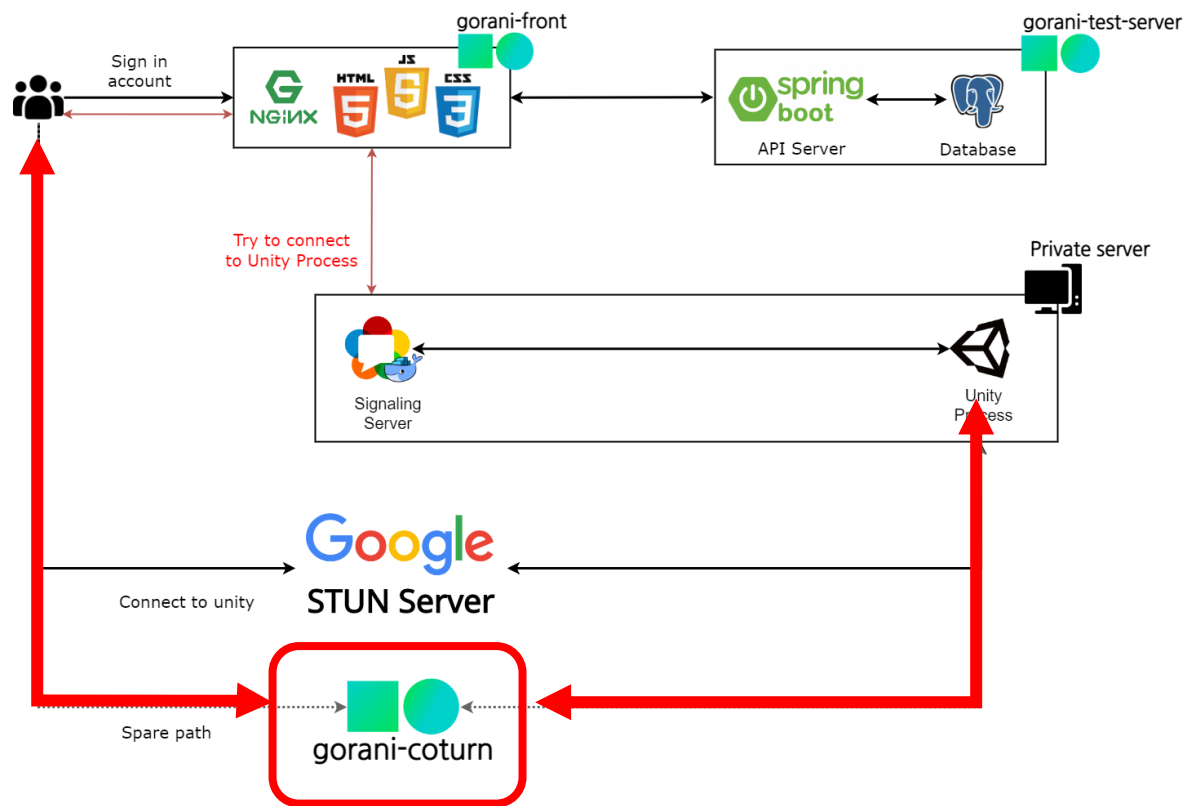
1. 로그인하여 교실 입장 페이지에 접근
2. 교실 입장 코드를 이용하여
교실에 매칭되는 signaling server주소를 획득
3. signaling server를 통해서
google STUN 서버를 활용한 p2p connection 시도
4. p2p connection 실패 시,
gorani-coturn 서버를 통한 relay connection



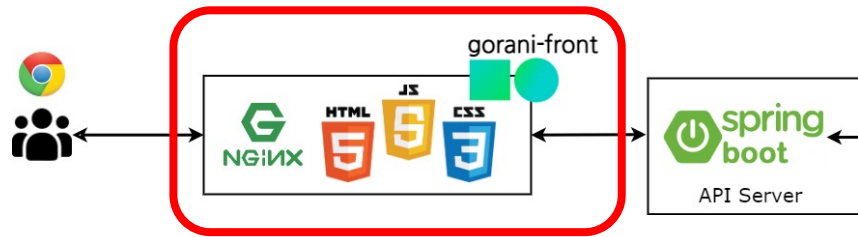
Unity, Signaling Server

[교실 입장과정]

1. 로그인하여 교실 입장 페이지에 접근
2. 교실 입장 코드를 이용하여
교실에 매칭되는 signaling server주소를 획득
3. signaling server를 통해서
google STUN 서버를 활용한 p2p connection 시도
4. p2p connection 실패 시,
gorani-coturn 서버를 통한 relay connection



Frontend




- HTML, Javascript, CSS를 활용하여 구현함
- NCP instance 위에서 nginx를 통하여 구동됨



[로그인 화면]

Frontend



REGISTER

ID

Enter Id

PW

Enter Password

Username

Enter User

email

Enter Email

SIGN UP

go back to [LOGIN](#)

@gorani

[회원가입 화면]

Hello, test

LOGOUT

CREATE ROOM

ROOM NAME

Enter Roomname

MAX CAPACITY

30

create

@gorani

ENTER ROOM

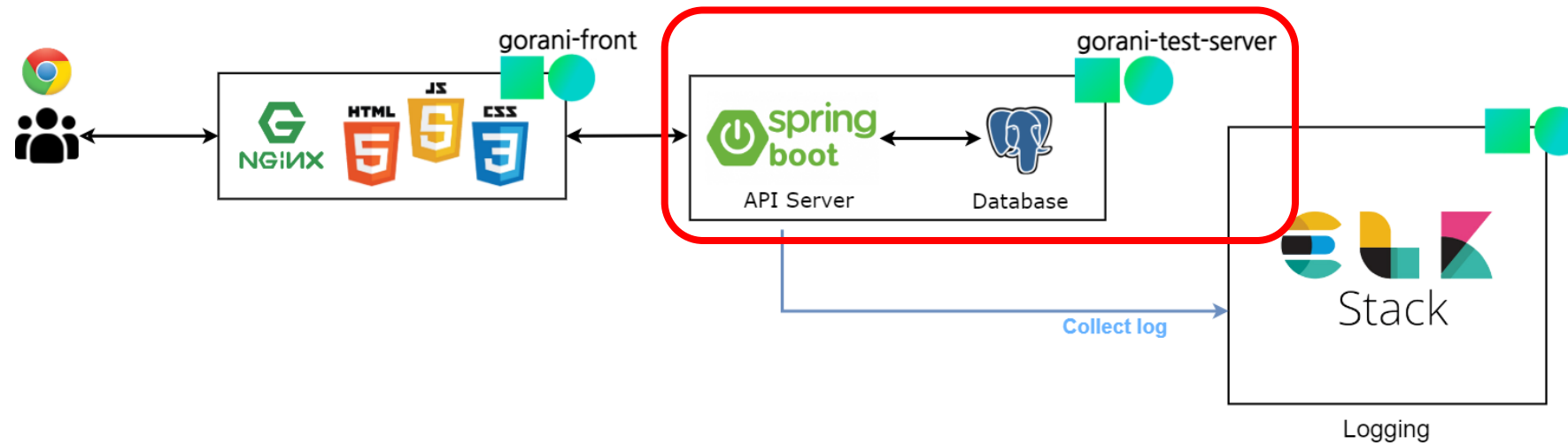
ENTER CODE

entercode

enter

[로비 화면]

Backend



- Backend는 java Spring boot를 사용하여 구현했으며, DB는 PostgreSQL 사용함
- NCP instance 위에서 동작함

인증 / 인가

- Spring Security
- JWT

DB 연동

- Spring Data JPA

API 문서화

- Spring Rest Docs

회원가입

요청

Request

```
POST /v1/auth/signUp HTTP/1.1
Content-Type: application/json;charset=UTF-8
Content-Length: 133
Host: gorani.com
```

```
{
  "loginId" : "testLoginId1",
  "password" : "testPassword",
  "emailAddress" : "testEmailAddress",
  "username" : "testUsername"
}
```

Request Fields

Parameter	Type	Neccesary
loginId	String	true
emailAddress	String	true
password	String	true
username	String	true

이달

Response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Length: 391
```

```
{
  "success" : true,
  "code" : 0,
  "msg" : "성공하였습니다.",
  "data" : {
    "loginId" : "testLoginId1",
    "username" : "testUsername",
    "emailAddress" : "testEmailAddress",
    "id" : 90,
    "token" : "Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI5MCiSnJvbGVzIjpbJlJTEVfVWVNFVjdlLCpYXQ0Y2E2"
  }
}
```

Response Fields

Path	Type	Description
success	Boolean	성공: true 실패: false
code	Number	성공 시 0을 반환
msg	String	성공: 성공하였습니다 실패: 에러 메시지 반환
data.id	Number	유저 고유 ID
data.loginId	String	로그인 아이디
data.emailAddress	String	이메일 주소
data.username	String	닉네임
data.token	String	JWT 토큰

[API 설계 및 문서화 예시]

Backend

[Backend 사용 기술]

인증 / 인가

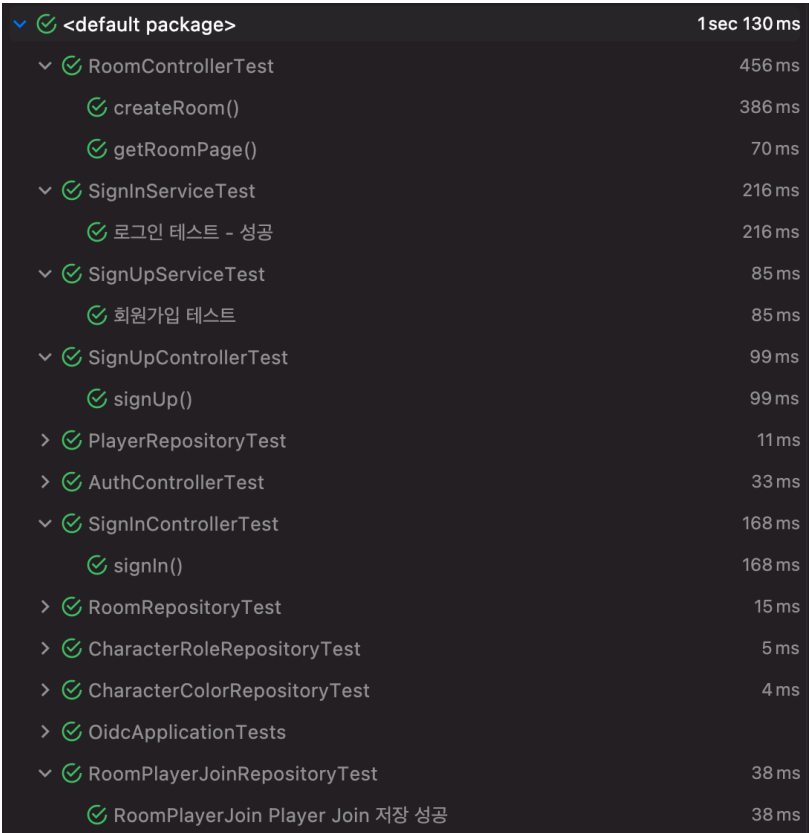
- Spring Security
- JWT

DB 연동

- Spring Data JPA

API 문서화

- Spring Rest Docs

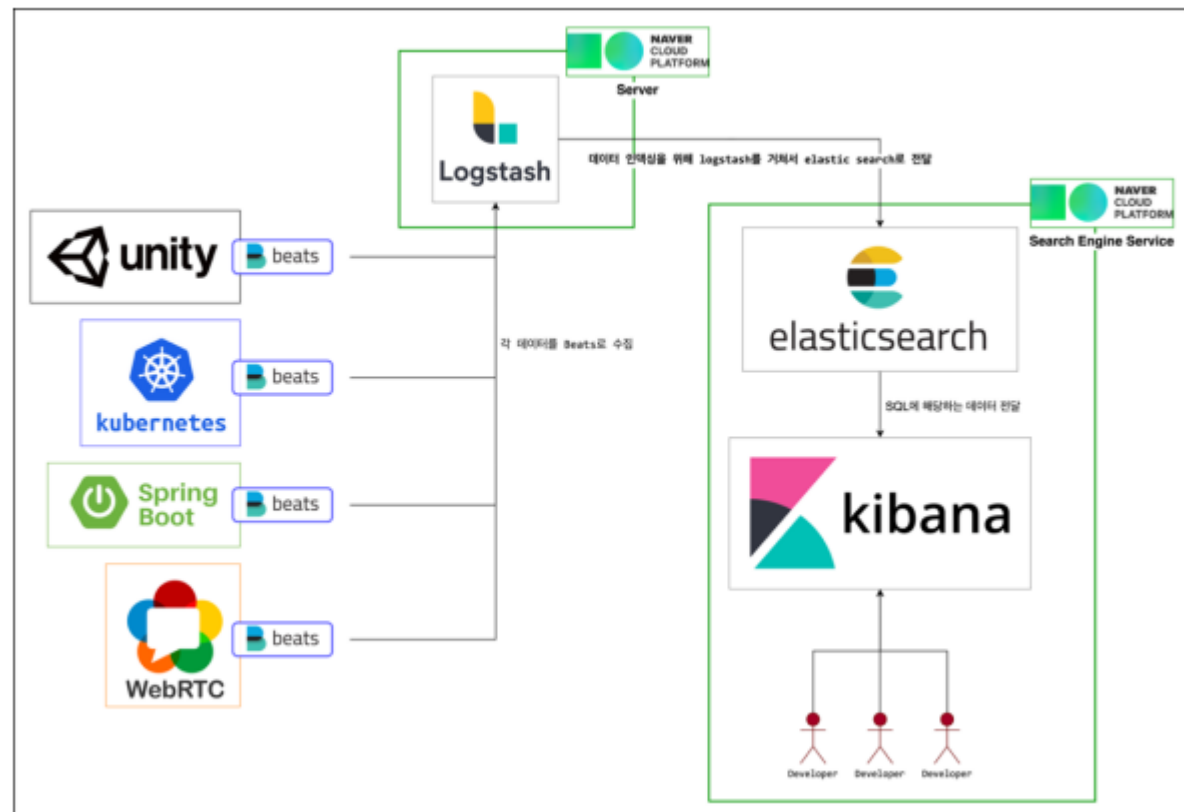


<default package>	1 sec 130 ms
RoomControllerTest	456 ms
createRoom()	386 ms
getRoomPage()	70 ms
SignInServiceTest	216 ms
로그인 테스트 - 성공	216 ms
SignUpServiceTest	85 ms
회원가입 테스트	85 ms
SignUpControllerTest	99 ms
signUp()	99 ms
PlayerRepositoryTest	11 ms
AuthControllerTest	33 ms
SignInControllerTest	168 ms
signIn()	168 ms
RoomRepositoryTest	15 ms
CharacterRoleRepositoryTest	5 ms
CharacterColorRepositoryTest	4 ms
OidcApplicationTests	
RoomPlayerJoinRepositoryTest	38 ms
RoomPlayerJoin Player Join 저장 성공	38 ms

[API 기반 테스트 코드 작성]

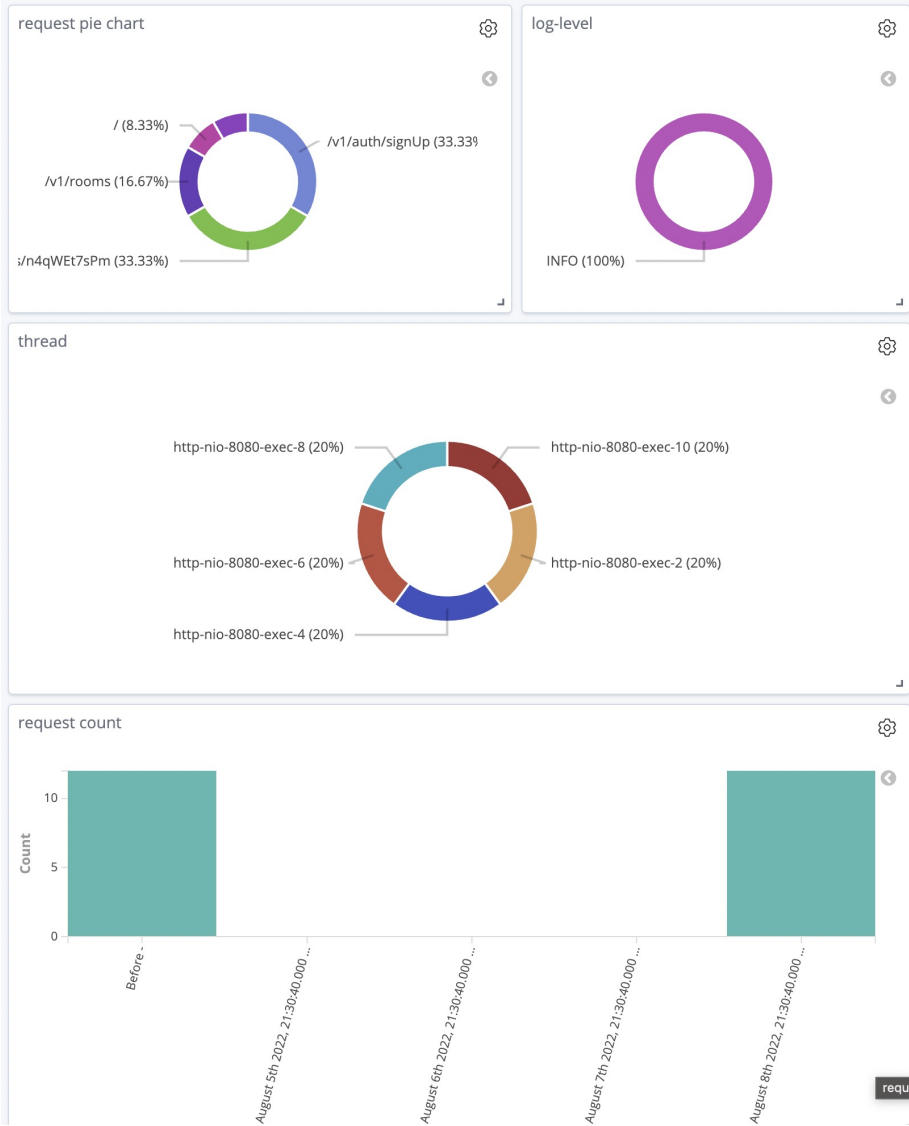
Logging

- 가벼운 어플리케이션을 위해 logstash 대신 filebeat를 사용함
- 로그 데이터는 logstash 서버로 보내 인덱싱이 진행됨
- Log Stash는 NCP instance 위에서 동작하며 EK는 NCP의 search engine service를 사용함



Logging

- 가벼운 어플리케이션을 위해 logstash 대신 filebeat를 사용함
- 로그 데이터는 logstash 서버로 보내 인덱싱이 진행됨
- Log Stash는 NCP instance 위에서 동작하며 EK는 NCP의 search engine service를 사용함



[log 수집 화면]

