12조

LIVE CHAT SERVICE

WebChat

실시간 채팅의

대규모 트래픽 처리

LIVE CHAT SERVICE

INDEX

01 프로젝트 소개

02 시스템 아키텍처 및 기술적 의사 결정

03 트러블 슈팅 / 성능 개

04 프로젝트 회고

LIVE CHAT SERVICE

01

프로젝트 소개

❶ 프로젝트 기획 배경

❷ 프로젝트 시연영상

01 프로젝트 기획 배경

여러 메신저앱 먹통

카톡, 새해 첫날부터 2시간여 먹통…“시스템 오류”
디스코드, API 문제로 두 시간 반가량 먹통

실시간 채팅의 대규모 트래픽 처리

현대 사회에 필수불가결한 메신저의 대용량 데이터 발생과 이로 인한 부하를 견뎌내기 위해 필요한 대책을 직접 리서치 및 적용

데이터 무결성 보장

전자지갑을 구현하여 동시성 제어를 통해 데이터 무결성을 보장을 챌린징 포인트로 구상

02

시연영상

채팅


회비관리 메신저 서비스
Sign Up
Login
현재 User
1
2
3
4
5
6
7
F
Channel 생성
Channel 참여
참여중 Channel 조회
공개 Channel 검색
참여 중 Channel Navig
1
2
3
4
5
F
개인 Wallet 개설
개인 Wallet 조회
개인 Wallet 내역
개인 Wallet 예약 내역
개인 Wallet 입금
1
2
3
F
Messenger-Service-Log
LoginModal 실행
{"email":"user1","password":"1234"}
LoginModal 결과
없는 이메일 입니다.
SignUpModal 실행
{"email":"user","username":"강영준","password":"1234"}
SignUpModal 결과
이미 가입된 이메일입니다.
SignUpModal 실행
{"email":"user1","username":"강영준","password":"1234"}
SignUpModal 결과
{"email":"user1","username":"강영준"}
LoginModal 실행
{"email":"user1","password":"1234"}
LoginModal 결과
{"email":"user1","username":"강영준"}
ChannelCreateModal 실행
{"channelName":"커피","channelDescription":"커피 토론장","channelPassword":"1234"}
ChannelCreateModal 결과
{"channelId":44,"channelCreateUsername":"강영준","channelName":"커피","channelDescription":"커피 토론장","isPrivate":false}
ChannelNavigateModal 실행
{"channelId":"44"}
Footer

02

시연영상

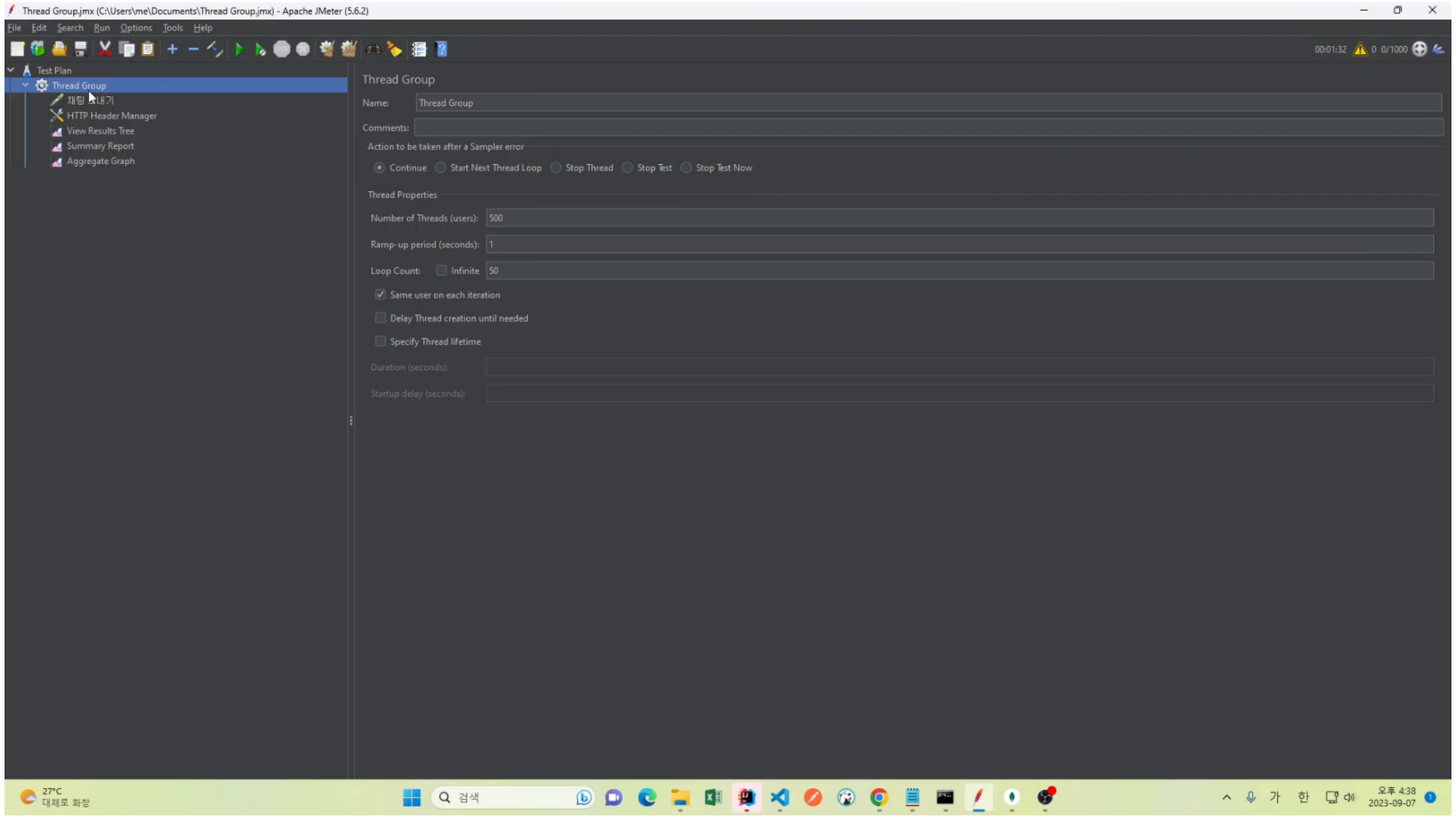
전자지갑


회비관리 메신저 서비스
개인 Wallet 개설
개인 Wallet 조회
개인 Wallet 내역
개인 Wallet 예약 내역
개인 Wallet 입금
1
2
3
F
그룹 Wallet 개설
그룹 Wallet 전체조회 by ChannelId
그룹 Wallet 참여
참여한 그룹 Wallet 조회
그룹 Wallet 조회
그룹 Wallet 내역
그룹 Wallet 예약 내역
1
2
3
4
송금
예약
현재 User 실행
현재 User 결과
{"email":"user","username":"강영준"}
PersonalWalletModal 실행
{"password":"1234"}
PersonalWalletModal 결과
{"walletId":9,"money":0,"username":"강영준"}
PersonalWalletCreateMoneyModal 실행
{"amount":"10000000"}
PersonalWalletCreateMoneyModal 결과
{"walletId":9,"money":10000000,"username":"강영준"}
ChannelCreateModal 실행
{"channelName":"커피","channelDescription":"커피 마시는 사람들","channelPassword":"1234"}
ChannelCreateModal 결과
{"channelId":10,"channelCreateUsername":"강영준","channelName":"커피","channelDescription":"커피 마시는 사람들","isPrivate":false}
GroupWalletCreateModal 실행
{"channelId":"10","walletName":"커피","description":"커피값 모으기","password":"1234"}
GroupWalletCreateModal 결과
{"groupWalletId":10,"money":0,"walletName":"커피","description":"커피값 모으기","channel":10}
GroupWalletGetModal 실행
{"groupWalletId":"10"}
GroupWalletGetModal 결과
{"groupWalletId":10,"money":0,"walletName":"커피","description":"커피값 모으기","channel":10}
Footer

02

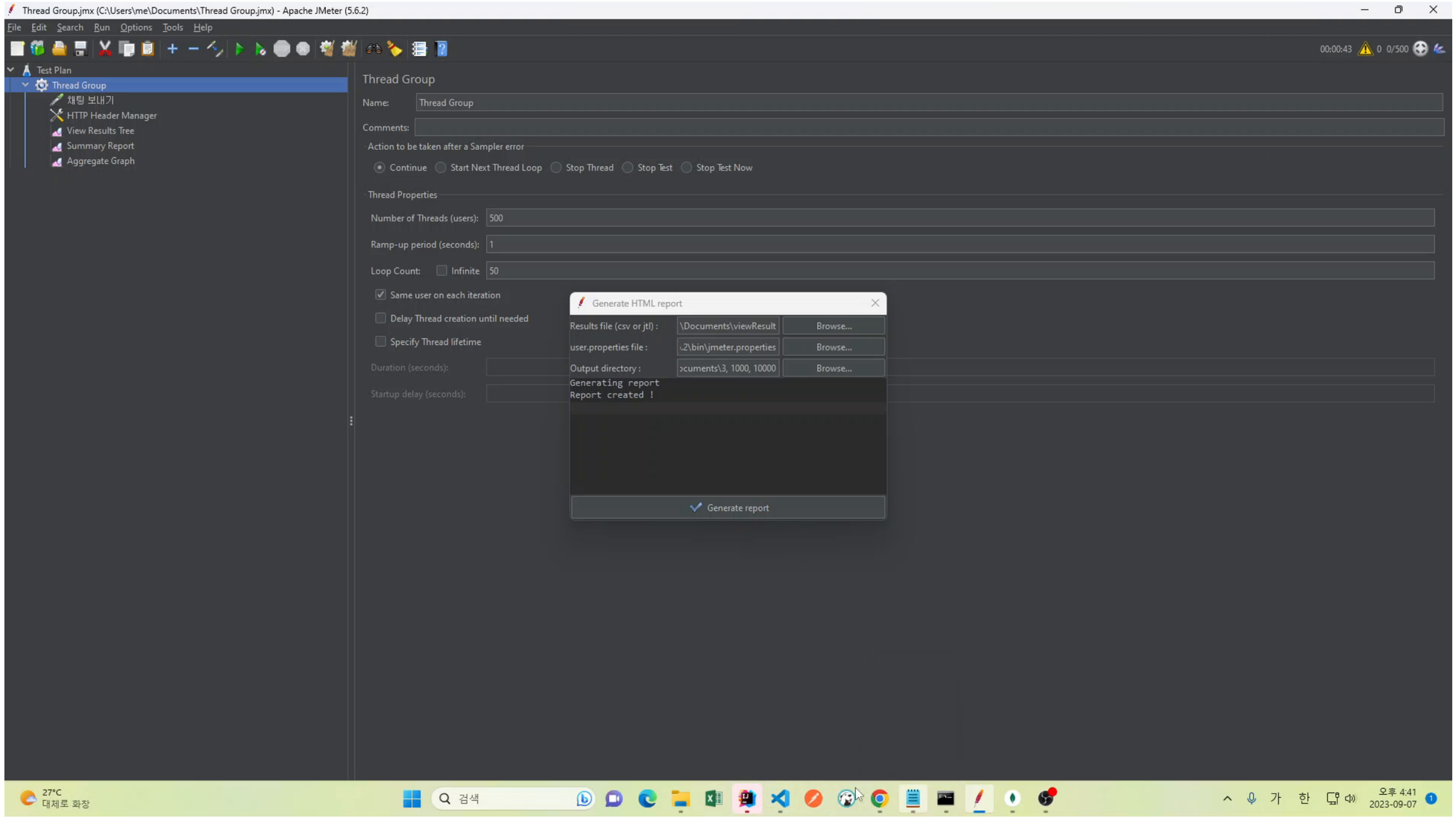
시연영상

jmeter 시나리오 테스트



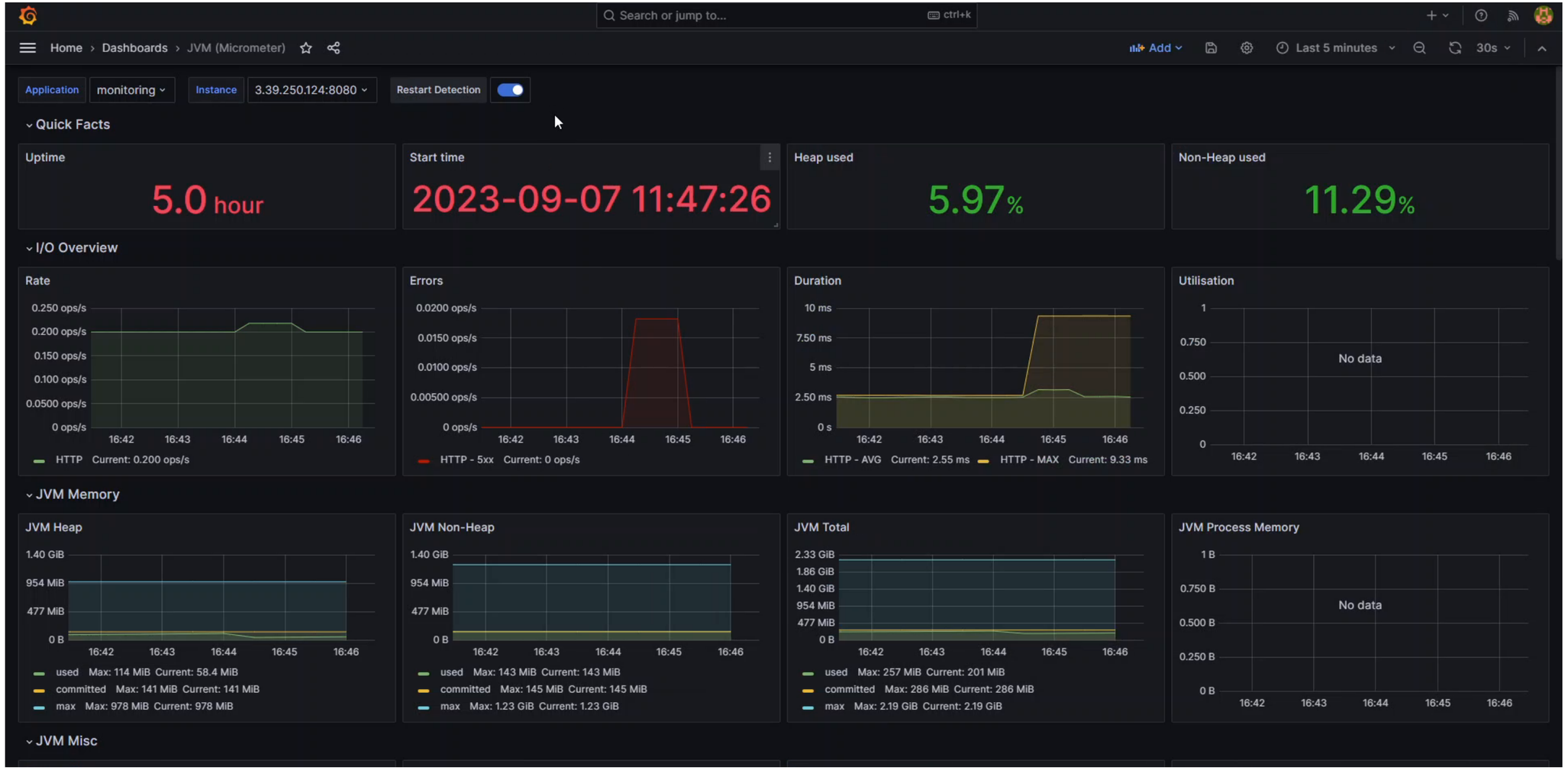
02 시연영상

jmeter 리포트



02 시연영상

Grafana



LIVE CHAT SERVICE

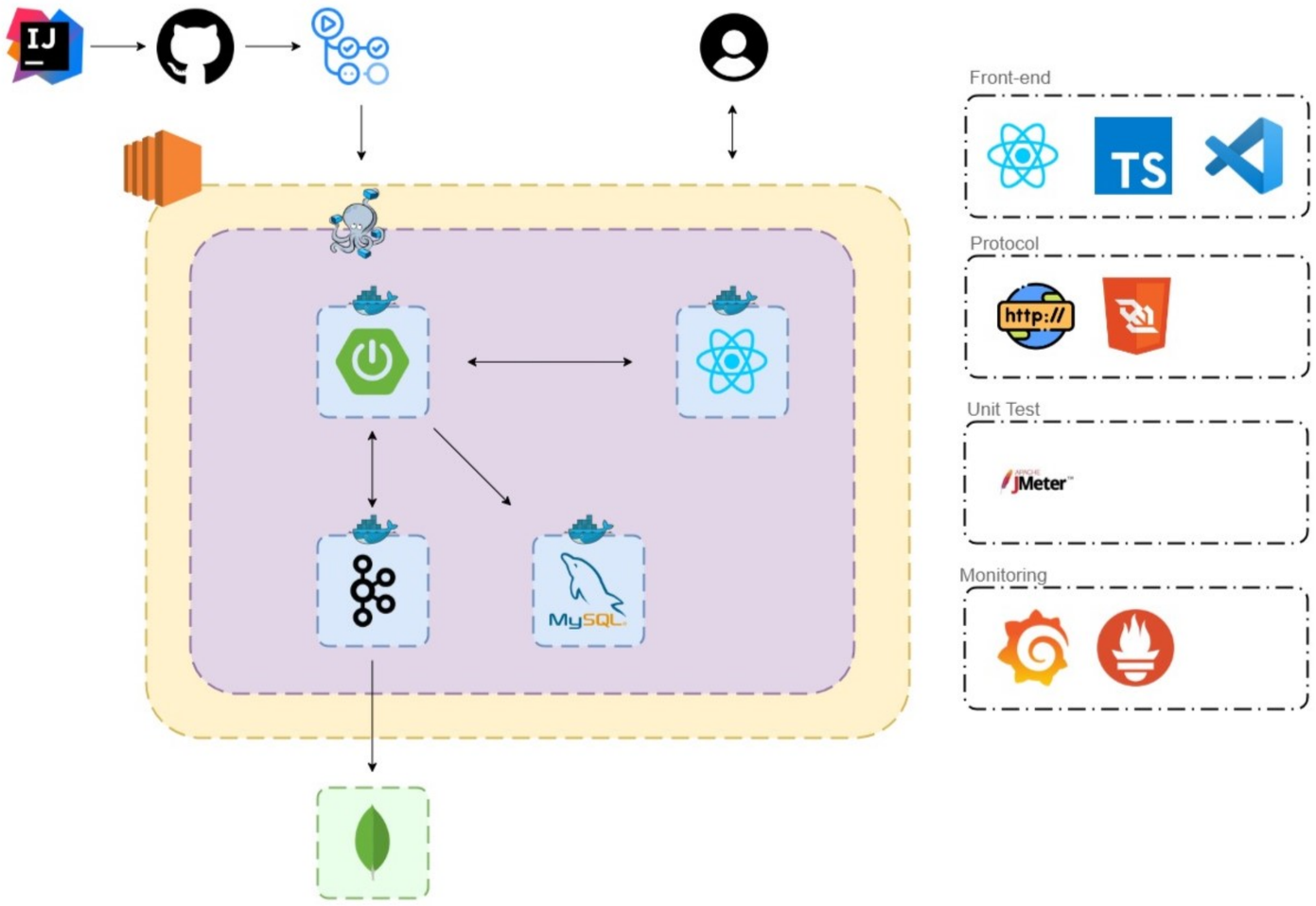
02

시스템 아키텍처 및 기술적 의사 결정

❶ 시스템 아키텍처

❷ 기술적 의사 결정

01 시스템 아키텍처



Front-end

Protocol

Unit Test

Monitoring

02 기술적 의사 결정

실시간 채팅, 부하 테스트

Web socket, Stomp

- 서버가 클라이언트에게 비동기 메시지를 보낼 때 가장 널리 사용하는 기술
- 양방향 메시지 전송까지 가능

Jmeter

- jmeter는 다양한 플러그인과 확장기능을 제공,
  많은 사용자들에게 알려진 도구로서 문제 해결이나 도움을 받기 용이
- 소켓 통신 테스트를 위해 jmeter 선택

02 기술적 의사 결정

메세지 큐, 모니터링

Kafka

- 비동기적으로 DB에 저장할 수 있고 정합성을 보장
- Producer와 Consumer를 가지고 대용량의 데이터를 처리하는데 강점을 가지는 kafka 선택

Grafana, Prometheus

- 지표를 시각화 하는데 특화

02 기술적 의사 결정

데이터베이스

MySQL

- 데이터의 원자성이 보장되는 관계형데이터베이스 중 익숙한 MySQL로 구현

MongoDB

- 수정할 일이 없고 속도가 중요한 채팅에서 NoSQL이 강점을 가진다 생각하여 NoSQL중 비용에 강점을 가지는 MongoDB로 구현

02 기술적 의사 결정

배포 환경 구성, CI/CD

Docker compose

- Compose 파일에 정의된 서비스들을 한 번에 실행할 수 있으며, 단일 호스트와 비교적 간단한 애플리케이션인 경우 충분하다고 판단

Github Action

- 클라우드에서 동작하므로 어떤 설치도 필요 없고 모든 GitHub 이벤트에 대해 GitHub Actions를 제공하고 있음
- GitHub에 push, PR 이벤트가 발생할 때 자동 테스트, 배포가 쉽게 이루어지기 때문에 개발에 몰두할 수 있음

LIVE CHAT SERVICE

03

성능 개선 / 트러블 슈팅

❶ 테스트 조건

❷ 성능 개선

❸ 트러블 슈팅

01 테스트 조건

테스트 시나리오

하나의 채팅방에 채팅 과부화

+

요청하는 데이터 수

유저 수 : 100 ~ 2000 명
보내는 메세지 수 : 50, 100개

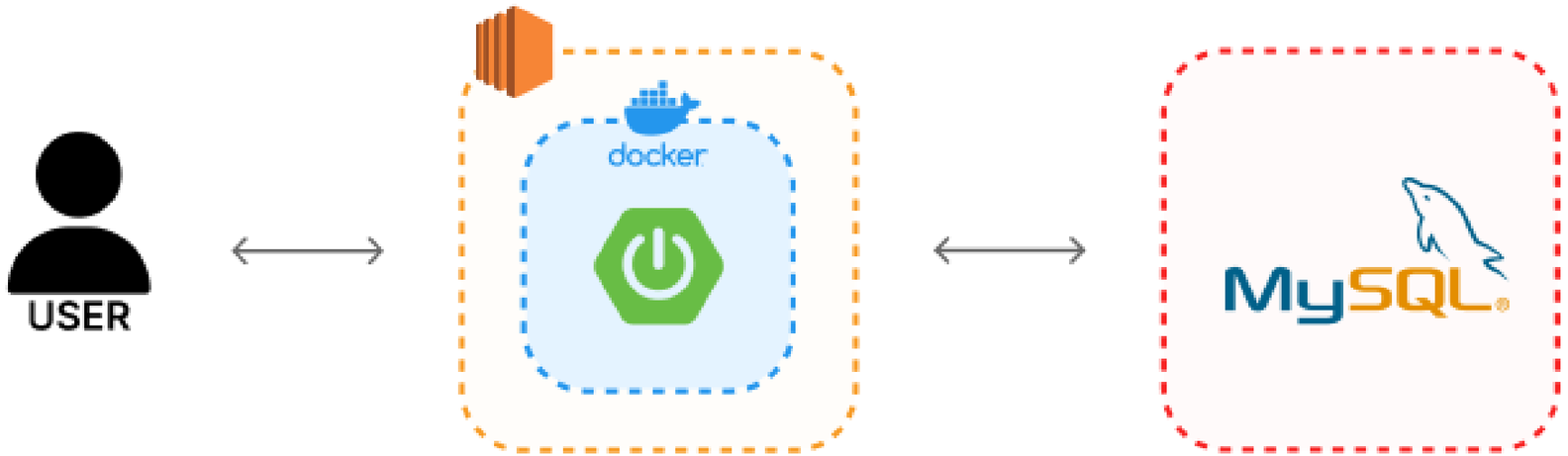
+

목표값

2000ms 이내의 응답 속도
1% 미만의 에러율

02 성능개선

version0.1



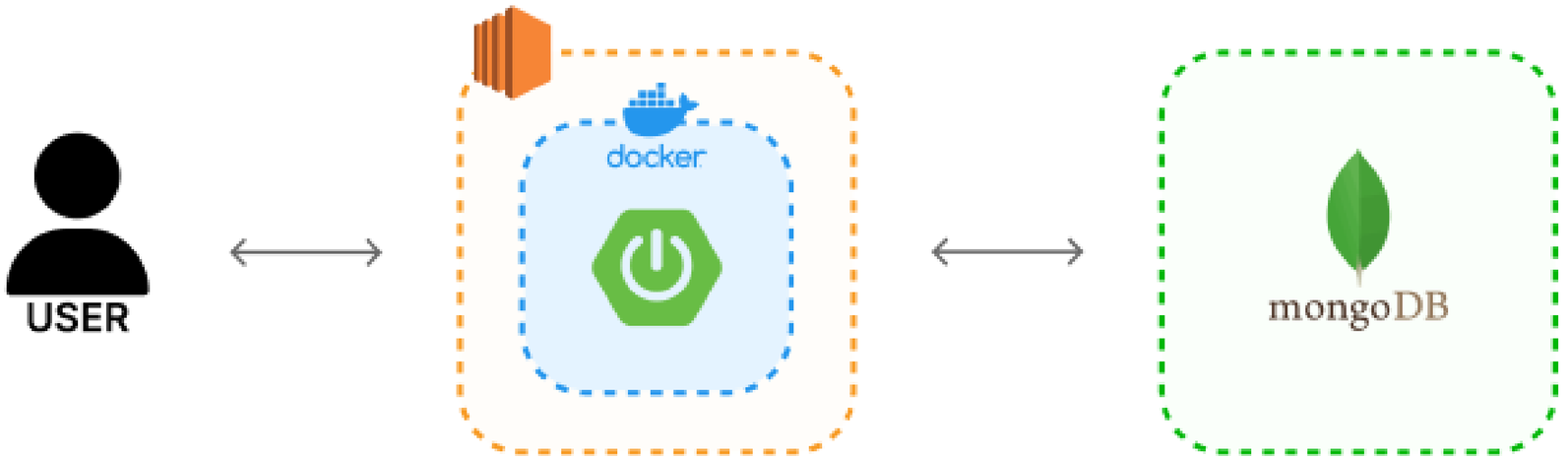
02 성능개선

version0.1

	표본수	평균	최소값	최대값	오류	KO
1초 100명 50번	5000	543.51	24	5951	0.00%	0
1초 100명 100번	10000	458.86	21	7363	0.00%	0
1초 200명 50번	10000	909.87	21	9524	0.00%	0
1초 200명 100번	20000	911.38	14	9889	0.00%	0
1초 500명 50번	25000	2322.3	24	13277	0.00%	0
1초 500명 100번	50000	2359.84	20	12726	0.00%	0
1초 1000명 50번	50000	4842.09	22	16345	0.00%	0
1초 1000명 100번	100000	4549.88	22	18060	0.00%	0

02 성능개선

version0.2



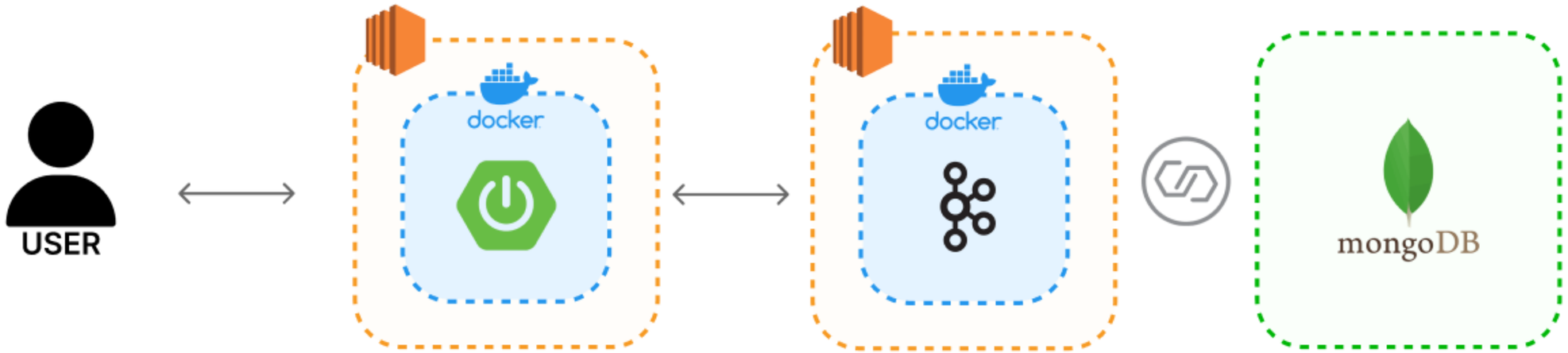
02 성능개선

version0.2

	표본수	평균	최소값	최대값	오류	KO
1초 100명 50번	5000	249.21	10	1886	0.00%	0
1초 100명 100번	10000	237.92	11	1472	0.00%	0
1초 200명 50번	10000	437.51	10	2468	0.00%	0
1초 200명 100번	20000	404	9	2138	0.00%	0
1초 500명 50번	25000	1137.82	15	4241	0.00%	0
1초 500명 100번	50000	1253.5	13	5045	0.00%	0
1초 1000명 50번	50000	2677.19	14	7287	0.00%	0
1초 1000명 100번	100000	2293.47	2	6352	1.45%	1447
1초 2000명 50번	100000	3821.39	3	75649	58.64%	58643

02 성능개선

version0.3

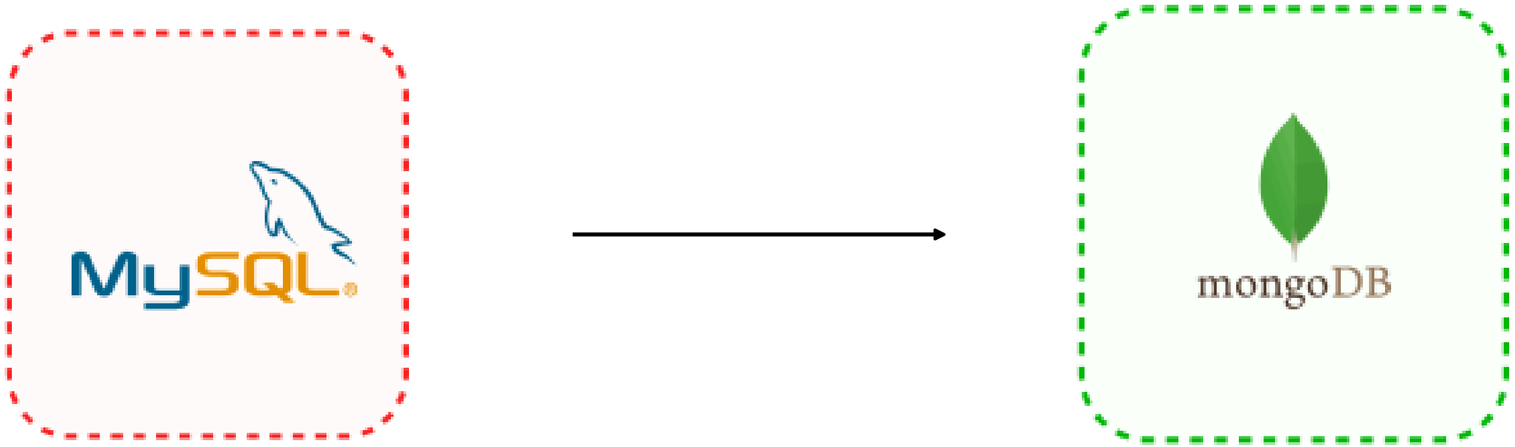


02 성능개선

version0.3

	표본수	평균	최소값	최대값	오류	KO
1초 100명 50번	5000	225.37	9	1182	0.00%	0
1초 100명 100번	10000	245.13	8	2143	0.00%	0
1초 200명 50번	10000	462.46	9	2143	0.00%	0
1초 200명 100번	20000	416	8	2376	0.00%	0
1초 500명 50번	25000	1114.47	11	4167	0.00%	0
1초 500명 100번	50000	988.53	10	3342	0.00%	0
1초 1000명 50번	50000	2052.25	6	4807	0.40%	198
1초 1000명 100번	100000	2069.8	9	5210	0.23%	232
1초 2000명 50번	100000	3914.68	3	13639	8.13%	8135

02 성능개선

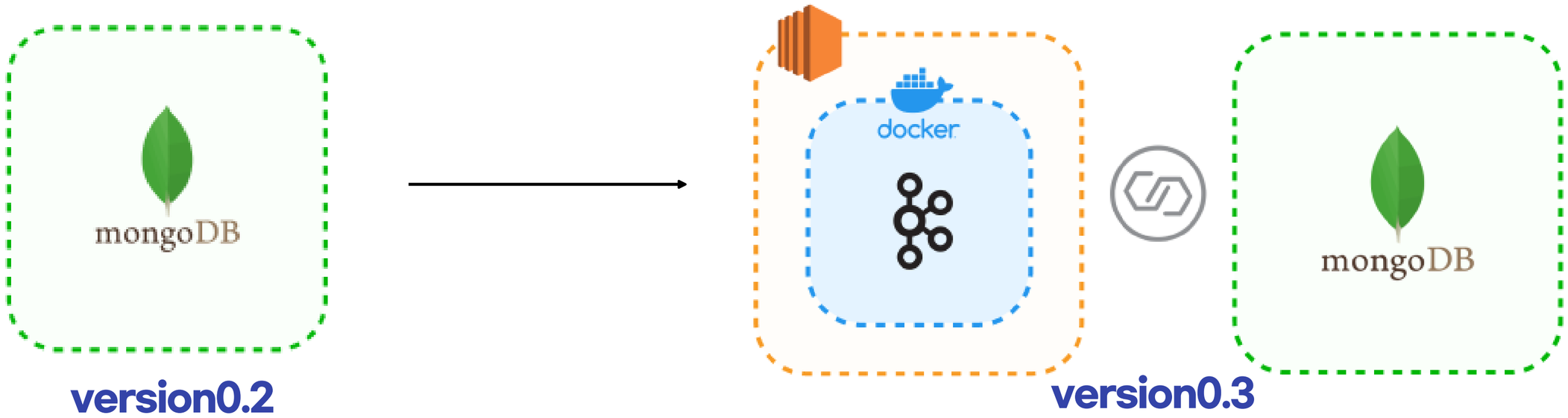


version0.1 → **version0.2**

- 평균응답속도가 전체 테스트에서 두 배 이상 씩 개선됨
- version0.1에서는 500명의 유저까지 목표치에 근사한 응답 결과를 보여줬으나 version0.2에서는 1000명의 유저까지 목표치에 근사한 응답 결과를 보여줌

- 1초 1000명 100번 테스트에서 version0.2에서 목표치를 넘어서는 오류가 발생함

02 성능개선



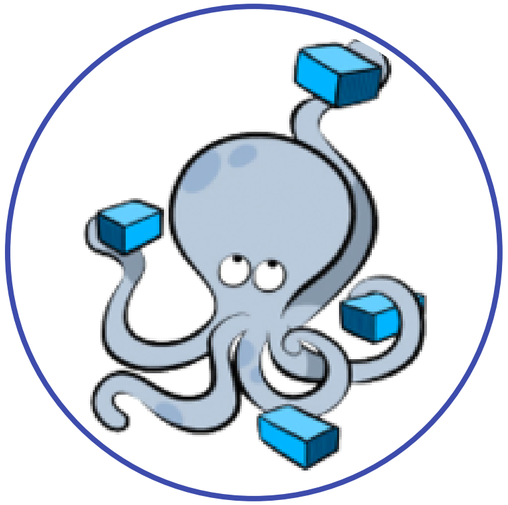
- 평균응답속도가 전체 테스트에서 근소하게 개선되나 유의미한 차이를 보이지 못함
- 1초 1000명 100번 테스트에서 발생한 오류가 목표치 미만으로 개선됨
- 1초 2000명 100번 테스트에서 오류가 크게 개선되었으나 목표치를 달성하지 못함

- version0.3의 1초 1000명 50번 테스트에서 version0.2에는 없었던 목표치 미만의 오류가 발생함

03

트러블슈팅

프론트엔드와 백엔드 연결 정상화



Docker compose

- 프론트엔드와 백엔드 모두 Docker Compose로 정상적으로 실행되는 것을 확인했으나 작성한 api들이 기능을 하지 못함
- Local환경에서 각 각 실행해서 테스트 했을 때는 정상적으로 작동함
- Docker host name을 인식 못해서 Frontend와 Backend 사이의 통신이 안되는 것으로 예상함
- Backend의 Docker host name 대신 배포 주소 사용하여 통신

03 트러블슈팅

Jmeter Stomp test

Jmeter

- Jmeter로 부하테스트 진행 중 Http프로토콜은 정상작동하는데 Stomp 프로토콜이 작동하지 않는 문제가 발생함
- 채널생성, Subscribe, Message 전송, Publish순서로 진행하였으나 작동하지 않음
- Http프로토콜을 이용하여 간접적으로 테스트 진행

03 트러블슈팅

Microservice Architecture

MSA

- 프로젝트 초기 계획에서는 Microservices Architecture (MSA) 를 목표로 설정하였으나, 관련 자료를 찾아본 뒤 MSA를 도입하기에는 시간이 부족하고 프로젝트에 오버스펙이라 판단하였음
- 이후 Modular Monolith로 대체하고자 하였으나, MSA와 같은 이유로 포기하였고 Monolith 아키텍처를 유지한 상황

04 프로젝트 회고

- 너무 아키텍처에 집중한 개발이었다.
- 채팅의 부하테스트에 프론트가 없어도 진행이 가능한데, 채팅도 부하테스트도 처음이어서, 프론트 개발에 시간을 많이 사용함.
- 프론트엔드 개발이 진행되는 동안, 백엔드에서는 이렇다할 테스트 결과 없이, 예상만으로 여러 아키텍처를 가져와서 version을 올려서 서로의 담당 부분에 이해가 떨어짐
- 처음 구상에서 동시성 제어를 도전해보고싶어서 만든 Wallet(인터넷뱅킹)의 구상과 구현에 많은 시간을 소모했으나, 주요한 대용량 트래픽 처리도 제대로 하지 못하여 결국 챌린징하지 못함
- 결과적으로 아키텍처 공부와 프론트개발에 많은 시간을 투자하였으나 테스트가 동반되지 않아서 결과가 없고, 프로젝트가 진행이 안되서 팀원들의 의지가 떨어지고 참여도가 저조해졌다.
- 남은 인원으로 최대한 프로젝트를 마무리지으려 했으나 담당하지 않은 부분에서 시간이 많이 소요되고 시간이 촉박하여 그마저도 제대로 하지 못함
- 프론트 없이 채팅을 apic plugin으로 연결 확인을 하고 부하테스트를 진행한 뒤, 근거에 맞춰서 아키텍처를 공부하여 도입하고 차근차근 version을 올렸더라면 좀 더 나은 결과가 있었을거라 예상됨

남은 인원들로 서버에서의 백엔드와 프론트간의 연결 문제를 정상화 하였으며, jmeter를 통한 테스트를 진행하는 등, 추가적인 구현보다는 채팅 기능 및 현재까지의 작업을 마무리 짓는형식으로 하였다.

LIVE CHAT SERVICEVICE

GitHub	https://github.com/innovationCamp/messenger-service
Notion	https://www.notion.so/12-c52babc42d65432987138e272a57a978