
Member

김종준



임채승



이하얀



정하영



Project Introduction

Discord 클론 코딩

websocket, WebRTC, MSA를 중심으로



Motivation

01

팀원 모두가 평소에
많이 사용하여
익숙하고 **이해도**가
높은 서비스

02

MSA 관점에서 확장
시킬 수 있는 여러
서버(채팅, 미디어)로
나뉘고 서버간 연동

03

데스크탑 기반

Goals - 임채승

Current

데스크탑 개발만 하고 이해하지 못하는 상태

After

데스크탑 개발에 나아가 사용자 편의를 생각할 수 있는 수준

Boilerplate에만 의지하는 수준

상황에 맞는 개발 환경을 세팅할 수 있는 상태

실시간 통신을 이해하지 못하고 있는 상태

실시간 통신이 어떻게 동작하는지 이해하고 설명 할 수 있는 상태

Goals - 임채승

1. SQLite의 강력함을 알고 싶음

SQLite를 사용하여 사용자에게 쾌적한(빠른) 사용경험을 주고 싶음.

가능하다면 평소 사용하던 localStorage와 비교하여 최소 10x이상의 빠른 속도로 로딩과 indexing 경험을 주고 싶음.

가능하다면 모든 데이터들을 table화 시켜서 관리하고 싶음.

2. 데스크탑 개발

electron을 통해 데스크탑 개발

electron내 IPC의 정확한 동작을 이해하고 사용자에게 보여줄 때 까지의 전체적인 프로세스를 알고 싶음.

공식 문서를 통해서 이해를 진행할 예정임.

Goals - 임채승

3. Vite를 통해서 빠른 개발 경험

boilterplate로만 이용하던 Vite를 현재 작업에 맞게 커스텀하여 사용하고 싶음.

공식 문서를 통해서 기준에 사용하던 설정이 무엇인지 알아보고, 이후 프로세스(ts파일, css파일, 컴파일 과정 등)에 맞는 설정을 찾아보고 적용할 것임.

4. websocket 연동

Node.js로 라이브러리 도움 없이 매직 패킷을 통해 통신할 수 있음.(현재)

실시간 통신을 하는 동안 나타나는 오류를 제어하고, heartbeat등 안정성을 높이고 싶음.

Discord의 websocket동작 방식을 파악할 것임.

5. WebRTC 연동

WebRTC를 프로젝트에 적용하고 싶음.

시간이 허용된다면, 쾌적한 사용자 경험을 위해서 빠른 통신을 위해 최대한 커스텀을 진행할 예정임.

How To Work - 임채승

1. 화면 구성 조사

화면이 어떤 Layout이 되어 있는가
모든 요소들이 어떻게 이루어져 있는가

2. Styled-components를 적극 활용

component와 page로 나누어 개발
component에는 styled-components를 활용하여 적극적인 재사용을 추구하기

3. electron으로 데스크탑 앱을 개발

SQLite와 websocket, WebRTC를 유기적으로 연결하는 방법을 찾아보기
사용자에게 쾌적한 사용환경을 전달하기 위해 노력하기
데이터의 구조가 헷갈린다면 유추하지 말고 주저없이 백엔드 팀에게 물어보기

Goals - 정하영

Current

대용량 서비스를 고려해서 설계 해본 경험이 없다.

After

대용량 서비스를 고려한 설계 능력을 갖춘 상태

왜 해당 기술을 사용할지 고민이나 이유 없이 무작정 도입했다.

기술 선택 이유 설명할 수 있는 상태

Goals - 정하영

대용량 서비스를 고려한 설계 능력을 갖춘 상태

1. 대용량 서비스를 설계하기 위해 책(대규모 시스템 설계 기초)을 통해 배경지식을 쌓고 프로젝트에 적용할 수 있는 부분을 정리하고 활용 계획을 수립한다.
2. 서비스가 분산될 때를 고려하여 MSA구조의 아키텍처를 이해하고 프로젝트에 적용한다.

Hystrix, Resilience4j, feign 등의 기술(spring cloud) 기술 활용 예정)을 활용하여 타 서버의 장애가 개발 서버로 전파되지 않도록 만들고 api들이 원활히 작동하도록 만든다.

강의(Spring Cloud로 개발하는 마이크로서비스 애플리케이션(MSA)) 완강하여 스프링 클라우드, 스프링 프레임워크 바탕 MSA 구조 아키텍처 이해하기

완강후 충분히 이해가 되지 않는다면 도메인 설계로 시작하는 마이크로서비스 개발 1회독

Goals - 정하영

대용량 서비스를 고려한 설계 능력을 갖춘 상태

3. 기능 구현 완료후 성능테스트(ngrinder & pinpoint 사용)를 진행하여 병목 지점을 확인 및 개선하여 목표로 하는 TPS 달성시키기. 일련의 과정은 문서화 하여 팀원과 공유한다.

기술 선택 이유 설명할 수 있는 상태

기술 선택 이유를 설명할 수 있다는 것은 기술의 동작 과정을 아는 상태다. 기술의 동작 과정을 파악하고 팀 상황(개발 일정, 다른 기술과의 호환성, 장단점 등)을 고려하여 최적의 기술을 선택한다. 이러한 과정은 문서화 하여 팀원과 공유한다.

How To Work - 정하영

1. MSA에 대한 배경 지식 학습

강의(Spring Cloud로 개발하는 마이크로 서비스 어플리케이션), 책(대규모 시스템 설계 기초)를 통해 MSA 아키텍처 설계에 필요한 배경지식을 학습하기 그리고 개인 노션에 문서화하여 이해한 내용 다시 정리하는 시간 가지기

2. 개발 일정에 따라 개발 진행

대용량 트래픽 처리를 염두해두고 전체 서버 아키텍처를 개략적으로 그려보기

처음부터 완벽하게 설계하는데 시간을 쏟지 말고 개략적으로 완성하여 개발하면서 부족한 점 개선해가기

기술을 선택할 때는 개발 기간, 장 단점, 등 여러 부분을 고려하여 적절한 기술을 따져 선택하기

기술 선택 과정은 문서화하여 팀원과 공유하여 피드백 받기

아키텍처는 팀원이나 캠프장님께 피드백을 받고 수정할 부분을 찾아 아키텍처를 완성하기

개발 진행

필요한 부분은 책 or 강의 or 구글링을 통해 학습하며 빠르게 적용할 수 있는 부분 위주로 학습하되 스프린트 기간동안 해야 할 일들을 고려하여 적절하게 학습 시간 분배하기

개발 중 문제 발생시 트러블 슈팅 과정을 블로그에 기록하여 프로젝트 기간을 단순히 경험만 하는 시간으로 보내지 말기

Goals - 이하얀

Current

웹 백엔드 개발자가 되기 위한 '나'의 기술 스택을 결정하지 못한 상태

After

기술 스택을 결정하고, 실제 웹 개발에 적용이 가능한 상태

웹 백엔드 개발을 위한 기초 이론만 학습하고, 서버를 개발하고 배포하여 서비스에 적용을 위한 지식과 실습 경험이 없는 상태

실습 경험을 쌓아 서버 개발 및 구축, 배포가 가능한 상태

팀 프로젝트를 위한 알림 서버와 상태 확인 서버의 동작 방식을 잘 모르고, 더 나아가 여러 사람을 통해 다양한 기술 스택으로 개발된 서비스 간 통신 방법을 모르는 상태

알림과 상태 확인 서버의 동작 방식 및 구축 방법에 대해 이해하고 활용할 수 있는 상태

Goals - 이하얀

1. 프레임 워크 사용으로 개발 시간 단축

기술 스택을 결정하기 위해 언어와 데이터베이스는 이미 활용 가능한 수준.

개발에 있어 효율적인 도구 선택하지 않아 시간이 오래 걸리면 이는 배포시간에 큰 영향을 미침.

효율적인 도구 선택하여 웹 개발에 모든 방면을 효율적으로 제어할 수 있는 기술 스택을 구축하고자 함.

2. 서버의 개발 및 배포 경험

서버를 구축하기 위한 개념적인 학습을 함.

리눅스 OS를 이용하여 APM서버를 구축하는 수준이며, 실제 개발 및 배포에 대한 경험은 전무함.

서버의 개발 및 배포를 경험하여 웹 서비스를 제공할 수 있는 상태가 되고 싶음.

3. MSA를 통한 규모있는 서비스 설계 경험

규모있는 서비스는 대부분 혼자하기 보다는 다양한 사람들과 팀을 이뤄 협업으로 구축하는 것이 대부분.

캠프를 통해 팀원들과 협업하여 서비스를 개발하기 위해 MSA 아키텍처를 학습, 사용하고자 함.

How To Work - 이하얀

1. 기술 스택 학습 및 적용

Spring 프로젝트의 핵심 기술을 강의와 교재 발췌 형식으로 빠르게 학습하기

학습 및 실습 내용 + 리뷰: 개인 Notion에 문서화해 정리하기

2. MSA 아키텍쳐 학습 및 적용

Spring Cloud로 개발하는 마이크로서비스 애플리케이션(MSA) 강의를 통해 학습하기

섹션 단위로 개인 Notion에 문서화하기

3. 알림, 상태 확인 서버에 대한 조사 및 서버 구축 & 타 서버 통신

[조사: 디스코드 클론코딩→](#)

Discord 기술블로그, 공식문서, Velog, Github 등다양한 예시들을 통해 이해하고, 개인 Notion에 문서화하기

How To Work - 이하얀

3. 알림, 상태 확인 서버에 대한 조사 및 서버 구축 & 타 서버 통신

서버 구축 방식 아키텍쳐 설계 → API 문서 작성 → API 서버 구현

비동기적 방식으로 진행, 각 단계별로 팀원 또는 캠프 관계자를 통해 피드백 예정

문제 발생 시 해결 과정은 개인 Notion에 기록하며 당일에 해결 → 미해결 시 코어 타임을 활용해 팀원에게 도움 요청하기

타 서버 통신

학습한 MSA 설계 방식 적용 →

전체 서버의 아키텍쳐, 설계 과정 등의 모든 과정을 특정 서버의 특성을 염두하여 진행하기(추후 더 자세하게 고민)

성능 테스트 →

시간, 구조 등이 기준치(추후 설정)에 부합하는지 여부를 체크하고 변경이 필요할 경우

반드시 팀원 피드백 진행 후 적용하기(코어 타임 or 그 외 시간)

4. 학습 시간에 너무 많은 시간을 허비하지 않을 것

개발에 필요한 수준으로 분량을 조절하여 과하게 학습만 하는 시간 지양하기

내가 가지고 있는 스택을 활용하고, 과하게 새로운 기술 학습 및 도입은 지양하기

Goals - 김종준

Current

예제를 통한 단순 인증과 라우팅만을 수행하는
API Gateway 구현 경험

After

예제 혹은 기본 설정보다 많은 양의 동시 접속을
감당할 수 있는 API Gateway 구현

WebRTC 활용 경험 없음

음성과 영상 미디어 서버를 따로 두어
음성과 영상 데이터가 분리된 화상 통화 구현

Goals - 김종준

1. API Gateway

마이크로 서비스들의 요청이 API Gateway를 거쳐 전달되므로 많은 양의 동시 접속을 감당할 수 있어야 한다고 생각함
스레드 크기 조절 및 인스턴스 추가와 같은 방법을 통해 기본 설정보다 10프로의 성능 개선을 목표로 함
인증의 경우 Redis를 활용한 캐싱을 통해 기존 로직 보다 10프로의 성능 개선을 목표

2. WebRTC

음성과 영상을 분리하여야 디스코드 화상 통화를 구현할 수 있다고 생각함
음성은 SFU 방식의 미디어 서버를 영상은 MCU 방식의 미디어 서버를 구축하여 음성과 영상의 데이터를 분리 시킬 예정
이를 통해 MCU 방식의 서버를 통해 음성, 영상 미디어 서버 모두를 구현 하였을 때보다 10프로 정도의 성능향상을 목표로 함

How - 김종준

공통

1. 책이나 강의를 통해 기본 지식을 익힌다.
2. 간단한 예제를 통해 습득한 지식을 활용해본다.
이를 통해 부족한 부분과 공부해야할 부분을 파악한다.
3. 공식 문서 및 멘토님들을 통해 부족한 부분을 보충한다.
4. 프로젝트를 진행한다.

API Gateway

1. IT 기업의 MSA로 전환 사례 영상을 찾아보며 지식 축적
2. Zuul 기반 MSA 예제를 통해 지식 활용
3. 공식 문서를 통한 지식 보충
4. 책을 통한 지식 보충
5. 프로젝트 진행

WebRTC

1. stove-smooth 팀의 Discord clone coding 문서
2. 예제를 통한 지식활용
3. 공식 문서를 통한 지식 보충
4. 프로젝트 진행

How To Work - 김종준

1. 개발 계획

1. 필요 기능 조사
2. 기본 예제 코드 구현을 통한 기본 기능 구현
3. 기본 기능 코드를 바탕으로 필요 기능 구현

2. 개발 규칙

1. 목표한 개발 일정을 지킨다
2. 우선 순위를 먼저 정한 후 개발을 진행한다.
3. 기본 구현의 경우 본 개발 기간이 아닌 공부 기간에 완료한다.
4. 필요 기능 구현시 해결하기 어려운 부분이 있으면 하루이상 고민하지 말고 도움을 요청한다.

Planning

임채승

Discord의 websocket 동작방식 파악 ~1/10
WebRTC 작동방식 파악 ~ 1/15
electron의 동작 방식 ~ 1/16
Discord화면 구성 분석(figma) ~ 1/18
화면에 맞추어 구성 ~ 1/19
개발 ~ 2/18

정하영

대규모 시스템 설계 기초, 강의 완강, MSA아키텍처
설계 ~ 1.12
인증 서버 ~ 1.19
채널 서버 ~ 1.26
채팅 서버 ~ 2.9
성능 테스트 및 개선 ~ 2.18

김종준

Spring cloud 관련 공부 ~ 1/5
유저, API Gateway 구현 ~ 1/12
유저, API Gateway 테스트 및 개선 ~ 1/19
WebRTC관련 공부 ~ 1/19
화상 통화 구현 ~2/2
화상 통화 성능 테스트 및 개선~ 2/18

이하얀

Spring5 학습 및 Spring Cloud를 이용한 MSA 아키텍처
학습 및 설계 ~ 1.26
알림 서버 및 상태 확인 구축 방법 조사 ~ 1.26
알림 서버 개발 ~ 2.2
상태 확인 서버 개발 ~ 2.9
성능 테스트 및 개선 ~ 2.18

Goals - Team

01

새로운 기술을 도입시
팀 리뷰와 문서화 과정
거치기

02

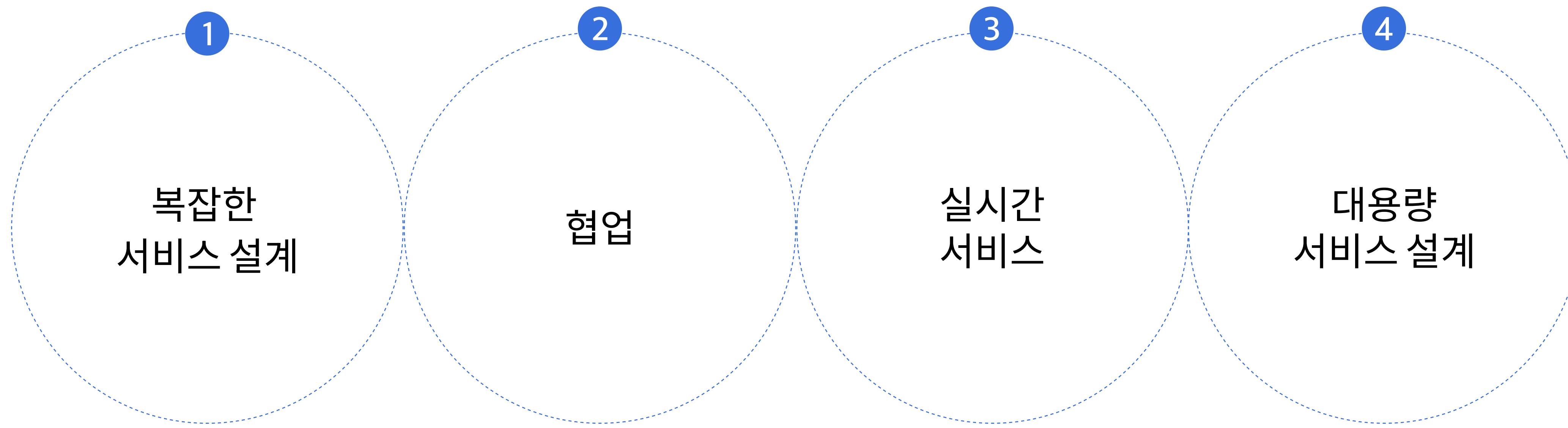
중복된 코드 지양

03

MSA 아키텍쳐
이해

Benefit

프로젝트를 통해 얻을 수 있는 것



Cooperative Work

FrontEnd - Backend

API - Swagger or Postman

Backend

코드 올리는 방법

개인 브랜치에서 PR → Approve → Rebase

브랜치 전략

JIRA에서 ID값을 가지고 와서 관리

Manageing Schedule

코어타임 월- 목 14:00 - 18:00, 금 14:00 - 17:00

반드시 참여해야 하는 시간, 불참시 적어도 전날 스크럼 시간에 팀원에게 공유

스크럼 월- 금 14:00, 최대 10분

전날 자신이 하지 못 한일, 하지 못한일이 있다면 왜 그렇다고 생각하는지, 오늘은 무엇을 할 것인지 등을 공유하는 시간

정기회의 목 14:00 - 16:00

목요일마다 회의를 통해서 백로그 관리, 저번주 스크린트 달성을 확인하여 이번주 스프린트 작업 설정
추가로 논의가 필요한 부분을 이야기 하는 시간, 필요하다면 코어 타임을 활용하여 추가 회의 진행

Risk Management

다음 문제를 아래와 같은 방법으로 해결

프로젝트 요구사항

문제

프로젝트 요구사항에 따라 프로젝트 일정에 차질이 생길 수 있다.

해결 방안

스크럼 시간, 회의 시간을 통하여 공유하기

프로젝트 관리

문제

Github의 다양한 기능을 사용하는데 있어 프로젝트 충돌 또는 로그 손실이 일어날 수 있다.

해결 방안

내부적으로 해결이 불가능할 시 주변에 도움을 요청한다.

Thank you
