TOGETHER

정소희
전우재
홍지표

목차

개인 목표

정소희

웹 프레임워크와 웹 서비스 구조의 사이 단계가 어떻게 동작하는지에 대한 이해를 팀 프로젝트를 통해서 이해하기.
네트워크 프로그래밍에 중점을 둔 개발을 하기.

홍지표

Java, Spring Boot의 동작 원리를 이해하고 근거있는 코드를 작성하기

영속성 컨텍스트에 대한 정확한 이해를 가지고 효율적인 쿼리 작성하기

MSA 구조의 서비스 개발하는 경험 해보기 단순한 CRUD 요청을 처리하는것외에도 web socket, webrtc 등을 활용한 새로운 기능 개발해보기

전우재

MSA 구조에 대해 공부하고 프로젝트에서 필요한 구성요소들을 Spring boot로 개발하기

MSA 구조에서 데이터 일관성 유지하도록 MQ 사용하기

프로그램 흐름의 이해하고 아키텍처 설계하기

팀 목표

- 새로운 아키텍처를 사용해보면서 설계 능력과 구현 능력이 갖춰진 개발자가 되자
  - 아키텍처를 정확하게 만들어서 자원을 효율적으로 사용할 수 있다.
- 기술을 왜 썼는지 알고 쓰는 개발자가 되자
  - 새로운 기술을 도입할 때 다른 방법들도 찾아보면서 비교하고 해당 내용 개발 일지에 작성한다.
- MSA 를 사용한 프로젝트 완성하자
- 시작하면 끝을 내는 개발자가 되자
- 구체적인 계획과 규칙으로 실제 현업에서 일하는 팀처럼 일하자

How to work - 정소희

[채팅 서버 구조 파악하기]

가상 면접 사례로 배우는 "대규모 시스템 설계 기초 배우기"를 읽고 조사한다. 이해한 내용을 바탕으로 DM 채널인 1:1 채팅 서버는 어떻게 구현할 것인지, 1:N 채팅 서버는 어떻게 구현할 것인지를 설계하여 본다.

[MSA 구조에 대해서 이해하기]

내가 설계한 채팅 서버 구조와 MSA구조가 어떻게 아우러질 수 있는지 확인하고 만약 설계가 어렵다면 재설계한다. 이와 같은 과정은 MSA 메인 설계를 담당하는 전우재님과 함께 협의하여 진행한다.

[사용 언어 후보 정하기]

Python의 프레임워크 중 채팅 서버의 특성에 제일 걸맞는 언어를 선택 해당 언어를 사용한다.

[데이터베이스 구조 설계]

DB는 RDBMS를 기본적으로 선택하되 Cache DB를 넣고 Cache DB의 교체 정책을 (LRU, LFU, FIFO)의 간단한 것에서 Clock Algorithm, Second Clock, Working Set 까지 사용하여 보고 각 성능을 모니터링하여 본다.

How to work - 홍지표

[MSA 관련된 기본적인 개념 습득 및 설계]

API GateWay 구성 방법, Component간의 통신 방법, Component를 나누는 기준 DB의 데이터 동기화 방법등 MSA를 설계하고 구현할때 어려움이 없도록 기본적인 개념들과 프로젝트에 어떤 툴들을 사용해야 할지 정도를 결정할 수 있을 기술적 이해하기, 이해한 것을 바탕으로 프로젝트에서 만들 서비스에 MSA를 적용시킬때 Component 나누는 단위, 툴 결정에 대한 명확한 근거를 제시하기

[좋은 코드 작성하기]

Java의 상속, Spring Boot의 DI를 활용하여 서비스 추가 기능에 대비해 SOLID원칙 준수하는 코드 작성하기

Java의 정석을 1회독을 하여 Java의 기능 적극 활용하여 코드 간결화 하기

PR시에 팀원에게 코드의 가독성, 효율성 등에 대한 피드백을 받고 적극 반영하기

[WebRTC 사용하기]

명확하게 WebRTC가 어떻게 동작하는지 이해를 하고 이를 기반으로 제작할 서비스에 있는 통화, 화상 회의 기능을 위해 WebRTC를 사용하기

[효율적인 Query 작성]

JPA에 대한 강의를 수강하여 JPA를 사용할 시에 영속성 컨텍스트의 원리를 파악하여 불필요한 추가적인 Query가 발생하지 않도록 한다.

How to work - 전우재

[MSA 아키텍처 구성하기]

MSA에 대해 전체적인 구조를 강의로 공부하고 정리하며 설명할 수 있도록 이해하고 활용 사례를 공부하기. 컴포넌트를 나눌 때 어떤 서비스를 묶어서 설계할지 정하고 해당 서비스가 어떻게 통신할지 설계하기. 서비스에서 데이터가 어떻게 교환되며 일관성을 유지할 것인지 확실하게 정의하기. 설계한 내용이 이상이 없는지 효율적인지 팀원들과 점검하기. 최종적으로 아키텍처 리뷰를 받아 수정하기.

[다양한 기술 연습하기]

기본적으로 사용하는 도구들의 근거를 확실히 이해하고 사용하기.

MSA 구조에서 토큰 기반 인증. 인증 과정과 권한 결정하기. 토큰 캐싱으로 refresh token 만들기.

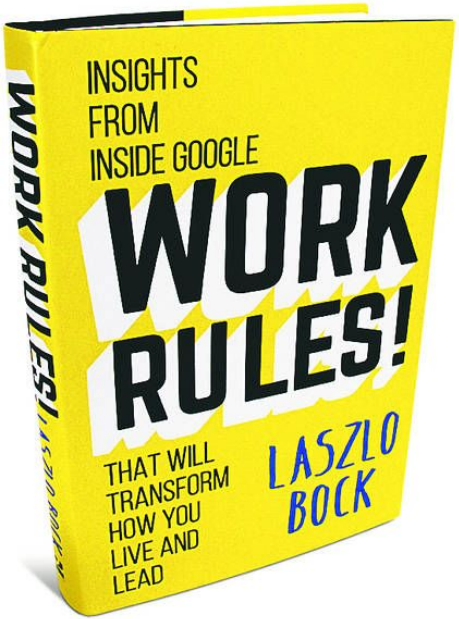
[의사소통 잘하기]

서비스를 만들기 전에 기능을 확실히 정의하며 각 서비스 폴더에 기능을 작성하기. 만약 변경사항이 있으면 팀원이 알기 쉽게 문서화. 구현하면서 기능을 정의할 때 애매한 것이 있다면 slack이나 github discussion 등을 이용하기. 코드 PR전 단위 테스트를 하고 contributor들의 확인을 받은 후 merge를 수행하기. trouble shooting은 문서로 작성해두며 반복하지 않도록 공유하기.

주제 설정 - 디스코드

- Discord는 사용자의 접근성이 뛰어나기 때문에 누구나 사용하기 쉽다는 장점을 가지고 있습니다. 따라서 프로토타입을 만들었을 때 사용자의 피드백이 쉽습니다.
- 각 기능이 명확하게 (음성 채팅, 채팅, 기본적인 CRUD)가 나뉘어져 있어 MSA구조를 목표로 하는 저희의 팀 목표에 적절합니다.
- 자주 사용해본 플랫폼이어서 구조가 눈에 익어 아키텍쳐 설계와 기능정의가 간편합니다.
- 여러 사람이 동시에 채팅하는 메신저이니만큼 비동기통신과 웹소켓에 대한 이해를 하기에 적절합니다. 또한 이것은 여러 프레임워크가 경쟁하는 기능이기 때문에 기술 도입 관리, 선정 문제를 해결하기에 적합합니다.

Work Flow



Daily Meeting

- 매일 오후 8시에 10분간 진행
- 오늘 하루 진행할 일을 간략하게 보고하여 서로 진행상황을 공유한다
- 이슈가 발생여부 체크하고 이슈 발생시 다음 회의 주제로 지정
- 팀원간의 유대감 증진 및 프로젝트의 긴장감 유지를 위해 시행

Weekly Meeting

- 매주 토요일 오후 1시에 시작
- 주말인 만큼 넉넉한 시간을 두고 주간에 발생한 다양한 이슈에 대한 토의
- 한 주간 진행된 개발 진척 정도 확인
- 개인 개발 계획이 잘 준수되고 있는지 확인

Communication

- 사소한 의견은 카카오톡에서 채팅으로 소통하기
- Slack에서는 격식을 갖추어 메신저 룰대로 채팅하기
- 모든 회의는 Notion에 회의록을 통해 기록으로 남기기
- Meeting은 여러가지 이슈에 대해 얘기해야 함으로 Discord에서 음성으로 진행
- 중요 사항을 논의할 시 대면으로 회의 진행

Work Flow - Ground Rules

- 이슈가 발생했을시 팀원에게 최대한 빠르게 공유한다.
- 다른 팀원이 이 이슈를 해결 가능할시 적극적으로 역할을 분담한다.
- 회의록을 팀원이 돌아가면서 꾸준히 작성함으로써 진행상황을 모두 명확히 인지한다.
- 회의때 최대한 자유롭게 논의한다.

위기 관리

case 1. 새로 공부해야 할 어려운 기술을 마주하게 되었을 때

-> 팀원들이 전부 시간을 할애하여 스터디를 진행한다.

case 2. 진행 중 나가야할 상황이 생길 시

-> 대체자를 구하고 나가는 것으로 한다.

case 3. 예상 못한 문제가 생길 시(지병, 스케쥴)

-> 최대한 빨리 문제가 생기자마자 팀원들에게 알린다.

case 4. 사람들 간 의사소통 중 갈등이 생겼을 때

-> 시원하게 해결한다.

개발 계획 - 팀

- **착수 단계**
  - 팀의 목표 수립.
  - 문제 인식 및 리스크 요소 확인.
  - 스케줄 작성
- **계획 단계**
  - 프로젝트에 필요한 기술 정의
  - 작업 순서 및 관계 정의
  - 업무 분담
  - 전체 아키텍처 설계 후 피드백
- **실행 단계**
  - 그라운드 룰 준수
  - 개인 작업 후 매일 회의 때 보고
  - 정한 데드라인에 도달할 때마다 통합 테스트 진행
- **모니터링 단계**
  - 통합 테스트 과정에서 꼭 단체로 문제가 없는지 확인
  - 주간 회의 때 만들지 못한 기능이 있다면 해당 기능을 우회하거나 다른 방법으로 구현.
  - nGrinder를 사용하여 모니터링 후 성능 개선 → 얼마나 개선할지 정해야함
- **종료 단계**
  - 프로젝트 산출물 작성
  - 계획 대비 부족한 사항 확인
  - 프로젝트 완료

개발 계획 - 정소희

아키텍쳐 설계 완성 (~1/9)

개발 환경 확정

데이터베이스 설계 확정

MSA 구조에 맞추어 설계 모듈화

채팅 API (~ 1/20)

채팅 서버 구현

채팅 캐시 관리 알고리즘 시험 단계

테스트

채팅 검색 API (~ 1/25)

채팅 검색 알고리즘 개발 및 구축

테스트 및 기능 추가(~ 2/10)

개발 계획 - 홍지표

MSA에 대한 조사(~ 1/9)

MSA에 구조와 원리에 대해 분석

MSA를 사용하기 위해 필요한 툴 조사

MSA 구조 설계(~ 1/15)

조사한 내용을 토대로 MSA 프로젝트 설계

컴포넌트 분리와 관리를 위한 툴 설정

음성 통화 기능 구현(~ 1/20)

유저의 음성 채널 생성 및 참여 구현

화상 회의 기능 구현(~ 1/27)

음성 채널 참여 유저들의 카메라 켜기 기능 구현

음성 채널 참여 유저들의 화면 공유 기능 구현

기능 테스트 및 추가 기능 구현(~2/10)

개발 계획 - 전우재

MSA 구조 구현 (~ 1/15)

eureka 서버에 서비스 등록

서비스간 통신환경 구성

아키텍처 설계대로 조정

기본 화면 만들 서비스 생성

user api 구현 (~ 1/ 29)

jwt로 글로벌 필터 인증

redis로 리프레시 토큰 저장

유저 crud

이메일 인증

데이터 일관성 유지

관리자 기능

추가 기능 구현 (~ 2/10)

결제 기능

모니터링 기능

END