TechFolio

근거 있는 선택과 **최적의 방식**을 고민하는
개발자 김예찬입니다.

Profile

안녕하세요! 저는 **근거 있는 선택**과 **최적의 방식**을 고민하는
웹 백엔드 개발자 김예찬입니다.

객체지향적이며 **누가 봐도 이해하기 쉬운 코드, 유지보수하기 편한 코드**가 좋은 코드라고 생각합니다.

단순 구현보다 **핵심원리를 이해하는 것**이 중요하다고 믿습니다.

기억에 의존하지 않고 기록하며 성장하기 위해 끊임없이 노력합니다.

함께의 힘을 믿으며 **커뮤니케이션도 개발의 일부**라고 생각합니다.

About Me

○ GitHub
https://github.com/macmorning0116

○ Tech Blog
https://constant1601.tistory.com/

○ Email
yechan1601.dev@gmail.com

○ Phone
010-4936-5651

Skills

Back-end

○ **Java** - SOLID 원칙을 준수하는 객체지향적인 코드 작성 가능
○ **Spring Boot** - MVC 기반 RESTful API 서버 구현 가능
○ **MySQL** - 쿼리 튜닝을 통해 성능개선 가능
○ **Redis** - 캐싱 및 락의 구현을 통해 성능개선 및 동시성 제어 가능
○ **MongoDB** - 기본적인 NoSQL 데이터 모델링 가능
○ **JUnit5, Mockito** - 정확한 목적을 가진 단위테스트 작성 가능

Infra

○ **Docker** - Docker Compose를 활용한 컨테이너 환경 구성 가능
○ **AWS** - EC2, S3, RDS, Elastic Beanstalk을 활용한 무중단 배포가능

Front-end

○ JavaScript, React
○ HTML, CSS

Tools

○ GitHub, Figma
○ Slack, Notion

Education

○ 명지대학교 — 컴퓨터 공학과
2022.03 ~ 2024.08

○ 학점은행제 — 경영학과
2020.03 ~ 2022.02

Experience

프로그래머스 데브코스 웹 백엔드과정 1기 전공반
2024.07 ~ 2024.12

○ Java, Spring등 백엔드 이론 및 실무 기술 학습
○ CS지식 학습(운영체제, DB, 네트워크와 정보보안 등)
○ SQL 문법과 RDB에 대한 학습
○ AWS를 활용한 배포 환경 구성 학습
○ 코틀린 마이그레이션
○ 3번의 실무형 프로젝트 진행

MEERKET

근처 중고물품 블라인드 경매 플랫폼

타겟 고객

주변 사람들과 중고물품을 블라인드 입찰로
사고 팔기 희망하는 모든 이용자들

제공 가치

중고물품 거래에 블라인드 입찰을 추가함으로써
단순한 중고거래에 새로운 재미를 제공

주요 기능

- 판매자는 물건의 사진, 정보, 최소 입찰가격, 거래장소를 지정 후 등록
- 위치를 기반으로 한 직거래 편리성 제공
- 판매자는 현재 최고 입찰가를 확인하고 경매 조기마감 가능
- 경매 종료 시 최고가 입찰자와 판매자 간의 채팅방 자동 생성
- 블라인드 경매 방식으로 실시간 경매 방식의 문제점 해소
- 물품에 대한 질문은 댓글을 통해 남김으로써 투명성 보장

기간 : 2024.11 ~ 2025.01
인원 : Back-end 3명, Front-end 3명, Designer 1명
업무 : Back-end
링크 : GitHub Link
배포 : Meerket 배포링크

사용 기술

Back-end

- Java, Spring Boot
- MySQL, Redis, MongoDB
- Junit5, Mockito

Front-end

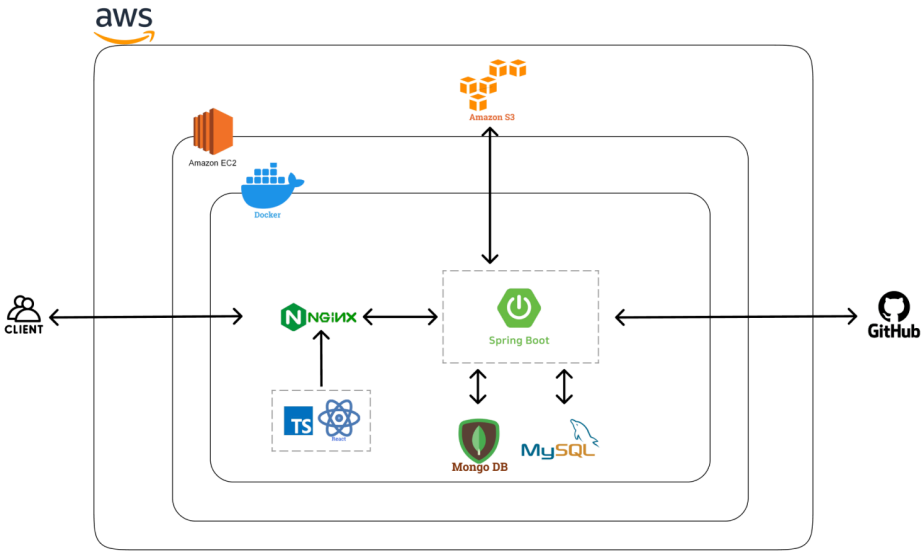
- TypeScript, React

Infra

- AWS, Docker

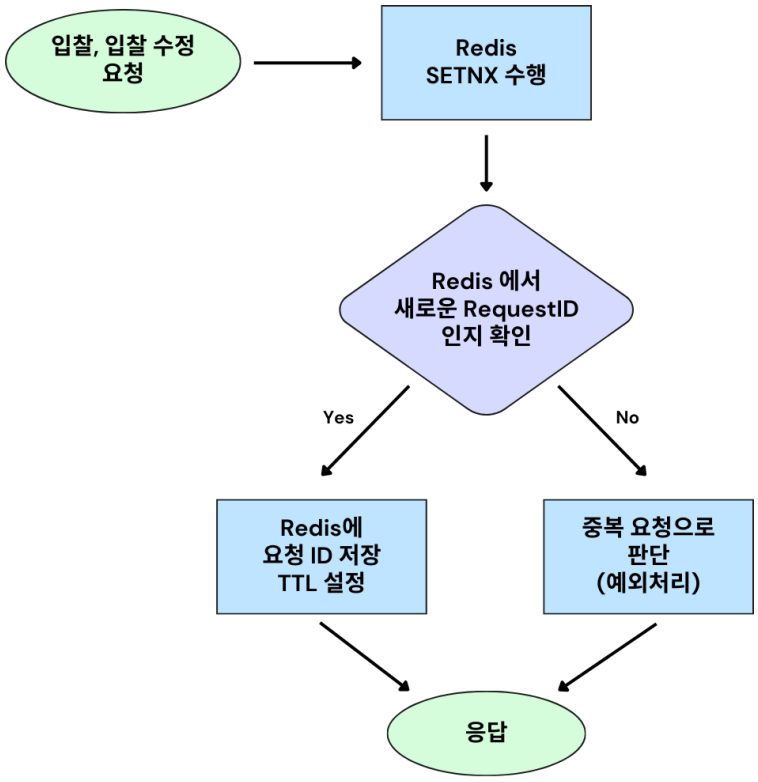
Tools

- GitHub, Slack
- Notion, Figma

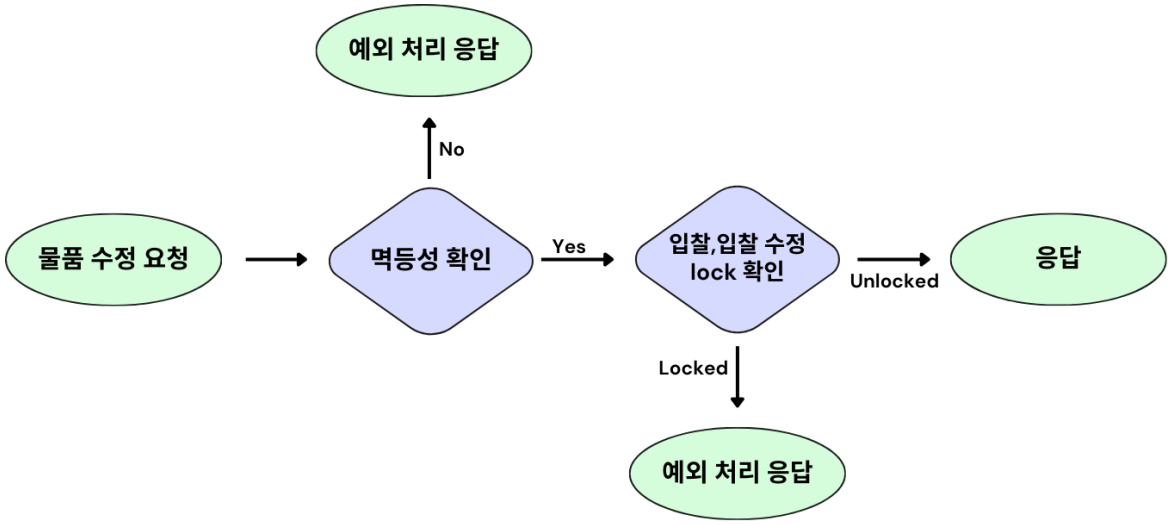


System Architecture

동시성과 멱등성을 보장하는 입찰 시스템 구축



입찰 및 입찰 수정 요청에 대한 멱등성 보장



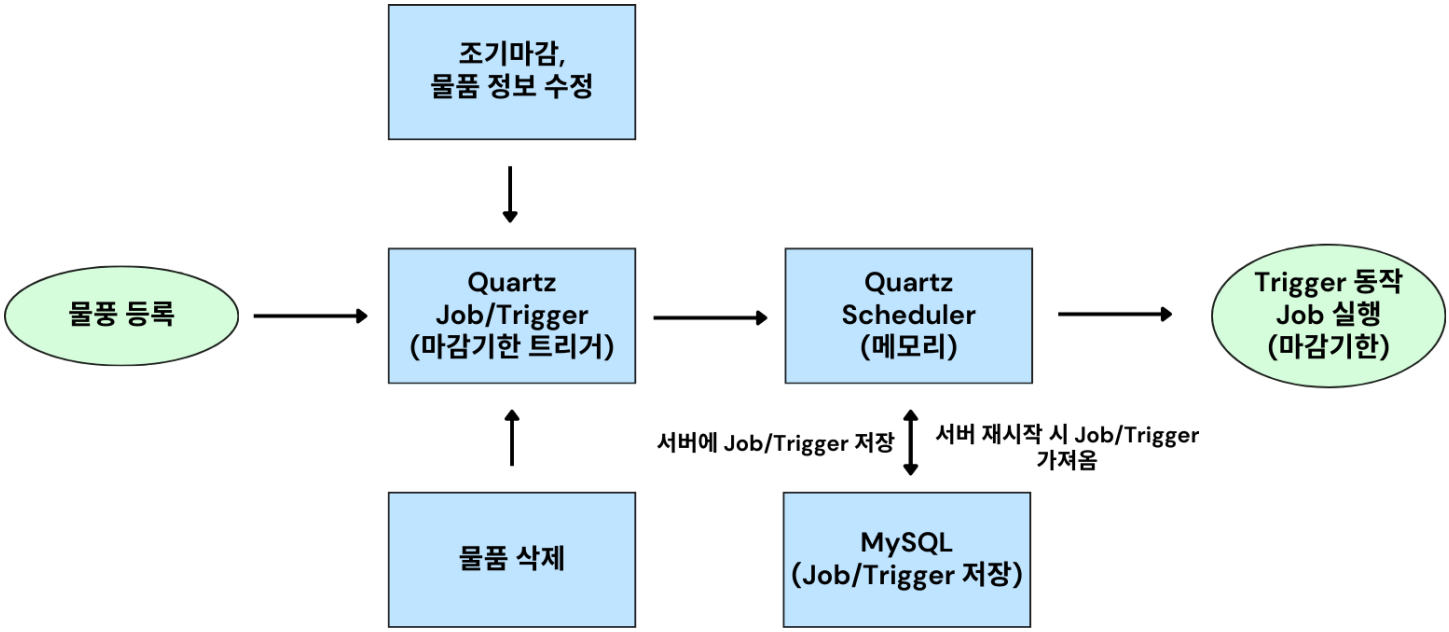
물품 수정과 입찰 요청에 대한 동시성 제어

문제 상황
입찰 시 요청이 중복되거나, 판매자의 물품 정보 수정과 구매자의 입찰 요청이 동시에 발생할 경우 낙찰자가 변경되거나 최소입찰가보다 낮은 금액으로 입찰되는 문제 발생

문제 해결
Redis의 SETNX와 TTL을 활용하여 멱등성을 보장하고, 락을 구현하여 동시성을 보장하는 입찰 시스템 구현

결과
서비스의 신뢰성을 높이고, 예외 상황에 안정적으로 대응할 수 있는 아키텍처 구현 동시성, 멱등성 제어 PR링크

서비스 장애 복구 및 입찰 마감 자동화 프로세스 구축



Quartz라이브러리를 활용한 장애복구 프로세스

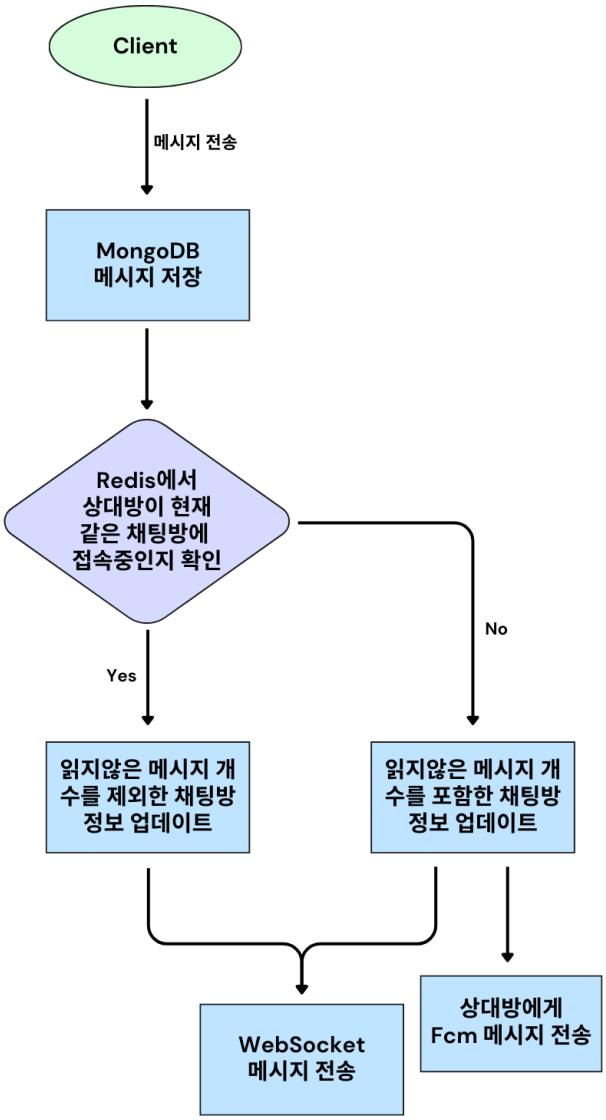
문제 상황
스프링의 기본 스케줄러는 서버 재실행시 경매 마감 트리거 복구 불가 및 경매 마감 시 자동화 프로세스 구축 불가능한 문제 발생

문제 해결
Quartz라이브러리를 도입하여 Job/Trigger를 DB에 보관하고 서버 재실행시 복구 가능하도록 함
트리거를 통해 낙찰자 선정, 물품 상태 변경, 판매자와의 채팅방 생성, FCM 알림 발송 등 입찰 마감 자동화 프로세스 구축

결과
장애상황에서도 서비스의 안정성을 보장하는 아키텍처 구현

블로그 정리 링크

실시간 채팅 시스템 설계 및 구현

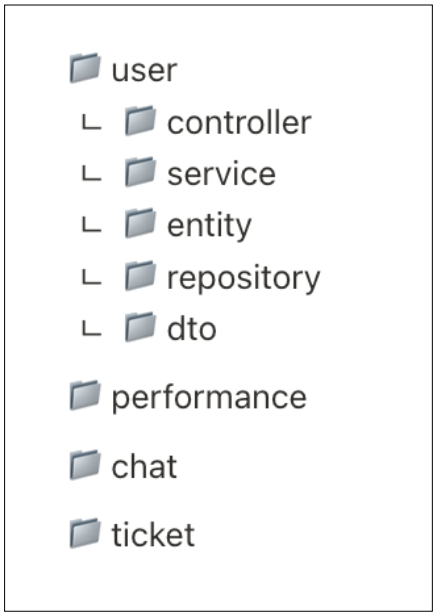


문제 상황
상대방의 상태에 따라 알림을 전송 하면서, 빠르고 안정적인 채팅 시스템 필요

문제 해결
STOMP 프로토콜을 기반으로 실시간 메시징을 구현하여, 이후 RabbitMQ, Kafka 등 외부 메시지 브로커와 손쉽게 연동 가능하도록 함
MongoDB를 통해 방대한 채팅 데이터를 효율적으로 저장 및 조회할 수 있도록 설계함
Redis를 활용하여 빠르게 채팅방 접속 상태를 관리할 수 있도록 함
FCM을 활용하여 클라이언트가 백그라운드 상태여도 알림을 보낼 수 있도록 함

결과
빠르고 안정적이며, 서비스 확장에 대응할 수 있는 채팅 시스템 구현

멀티 모듈 도입



기존 모놀리틱 단일 모듈

멀티 모듈 적용

문제 상황

확장에 따른 고민

1. 서비스 특성상 관리자가 필수 ⇒ 관리자 API 구현
2. 사용되지 않는 이미지 처리 등 대용량 데이터 배치 처리가 필요 ⇒ 배치 기능 구현
3. 시스템이 커지고 요구사항이 변경됨에 따라 담당하는 책임과 역할이 커지고 복잡해짐

문제 해결

멀티 모듈 아키텍처를 도입함

결과

중복 코드 발생을 줄임
모듈 별 최소한 의존하여 자유도/확장성 향상
관심사별 분리로 명확한 구조화
DDD, MSA 친화적인 설계 구현

WebSocket HTTPS Connection 이슈

문제 상황

```
ERROR 58986 --- [nio-8080-exec-6] o.s.w.s.s.s.DefaultHandshakeHandler
 : "Handshake failed due to invalid Upgrade header: null"
```

배포 서버에서 웹소켓 연결 실패하며, 위와 같은 에러 발생

원인 가정

1. 요청에 Upgrade헤더 미포함 (X)

Headers	Messages	Initiator	Timing
Connection:			upgrade
Upgrade:			websocket

- stomp.js 사용으로 자동으로 헤더에 포함

2. ELB 설정 문제로 웹소켓 헤더 유실 (X)
   - 배포 서버에서 ELB 사용하지 않음

3. EC2 Nginx 설정 문제로 웹소켓 헤더 유실 (O)
   - Nginx설정에서 헤더 설정 누락으로 인해 발생한 문제임을 확인

문제 해결

```
server {
  listen 443 ssl http2;
  server_name tfinder.store;

  # ssl 인증서 적용
  ssl_certificate /etc/letsencrypt/live/tfinder.store/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/tfinder.store/privkey.pem;

  # 파일 크기 제한 설정
  client_max_body_size 10M;

  location / {
    proxy_pass http://13.209.165.156:8080/;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
  }
}
```

Before

```
server {
  listen 443 ssl http2;
  server_name tfinder.store;

  # ssl 인증서 적용
  ssl_certificate /etc/letsencrypt/live/tfinder.store/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/tfinder.store/privkey.pem;

  # 파일 크기 제한 설정
  client_max_body_size 10M;

  location / {
    proxy_pass http://13.209.165.156:8080/;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
  }
}
```

After

EC2의 Nginx config 파일에 WebSocket 관련 proxy_set_header 추가

SOMUN

소규모 공연 예약 커뮤니티 플랫폼

타겟 고객

동아리, 버스킹 같은 소규모 공연을 개최하거나
관람하고자 희망하는 이용자들

제공 가치

학교 외부에 홍보하기 어려웠던 대학 동아리 및
규모가 작은 공연들의 정보를 제공하고, 공연에 관한
유저 간 커뮤니티 제공

주요 기능

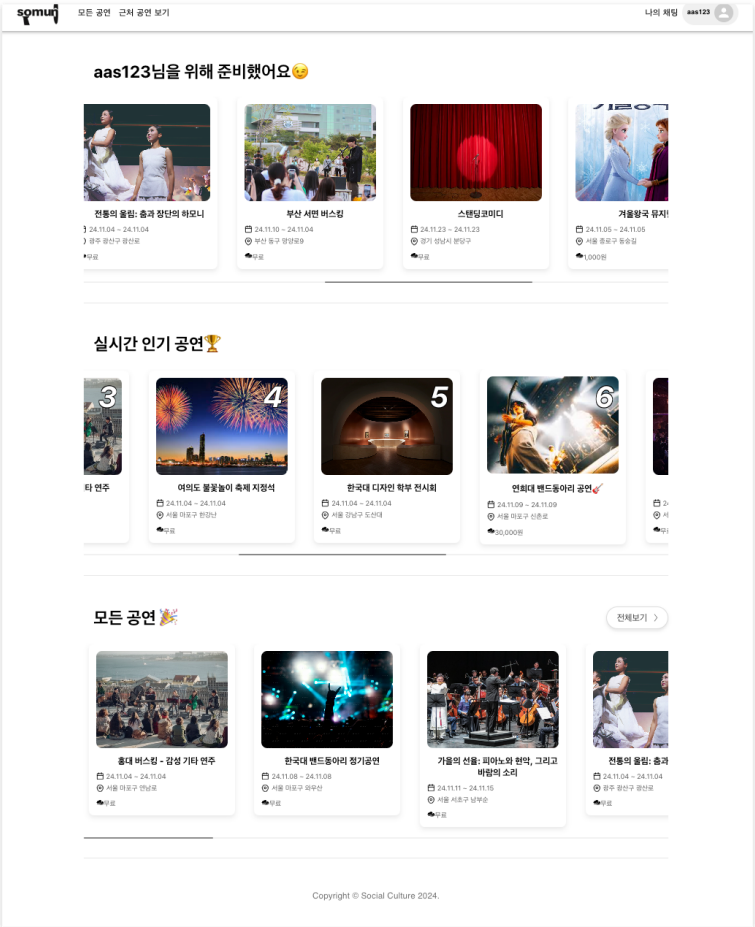
- 공연관리자 권한을 얻은 사용자는 공연 등록 가능
- 공연관리자 간 채팅을 지원하여, 관리자 간 커뮤니티 제공
- 사용자별 맞춤 추천 공연과 실시간 인기 공연 목록 제공
- 사용자의 위치를 기반으로 가까운 공연목록 제공
- 선착순 쿠폰을 통해 유료 티켓구매 시 할인 적용

기간 : 2024.09 ~ 2024.11

인원 : Back-end 5명

업무 : Back-end 및 Front-end

링크 : GitHub Link



사용 기술

Back-end

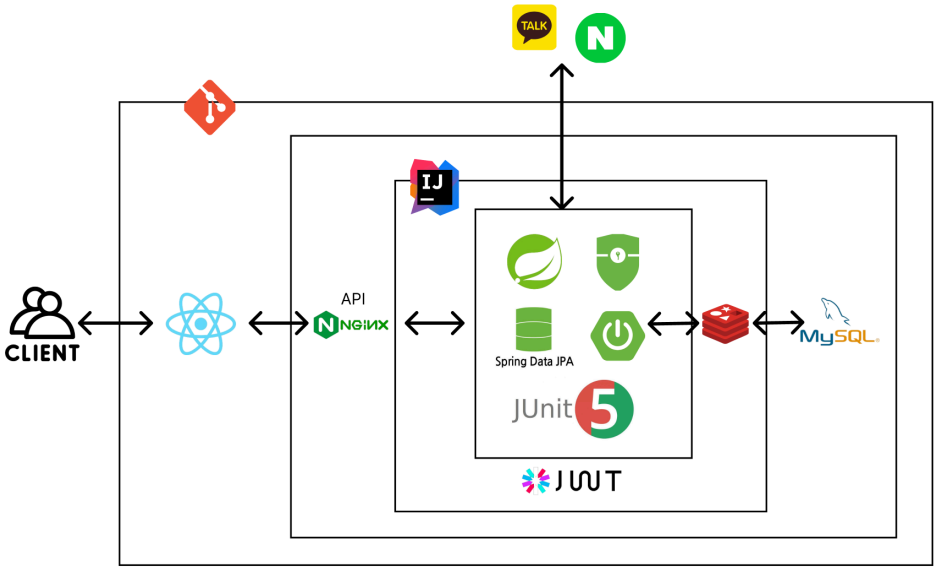
- Java, Spring Boot
- MySQL, Redis

Front-end

- JavaScript, React

Tools

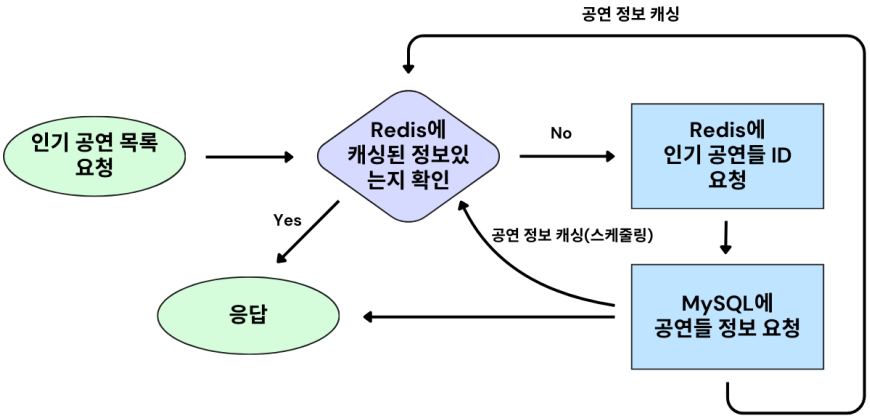
- GitHub, Slack
- Notion, Figma

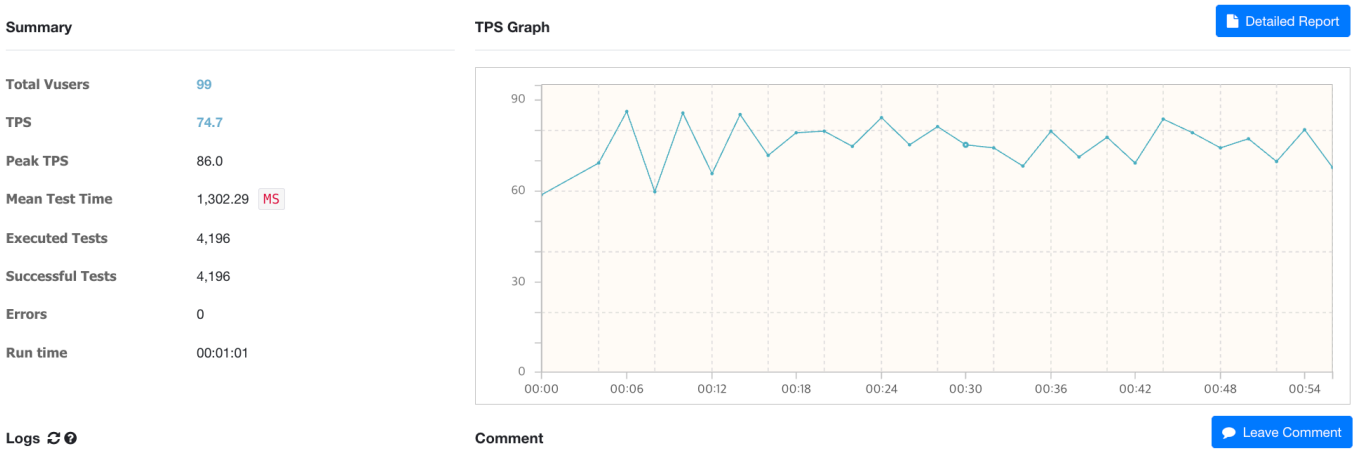


System Architecture

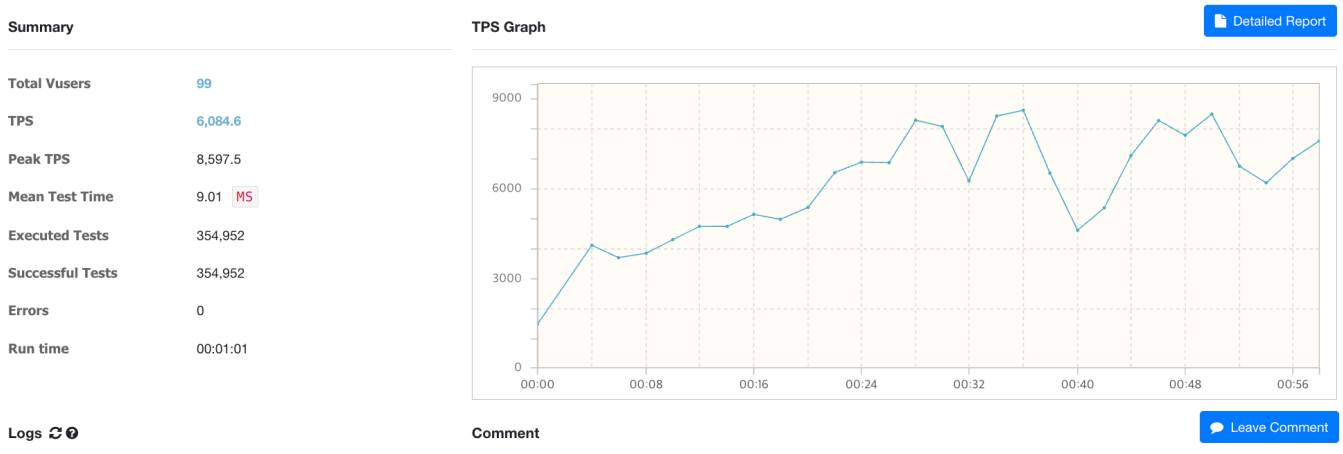
실시간 인기 공연 조회 구현

캐싱과 스케줄러를 활용한 성능 개선 및 부하 감소 (부하 테스트 진행)





개선 전 결과



개선 후 결과

문제 상황

실시간 인기 공연 조회 시 매 요청마다 메인 DB를 조회하여 불필요한 쿼리가 반복 실행됨. 이로 인해 DB 부하 증가 및 응답 속도 저하

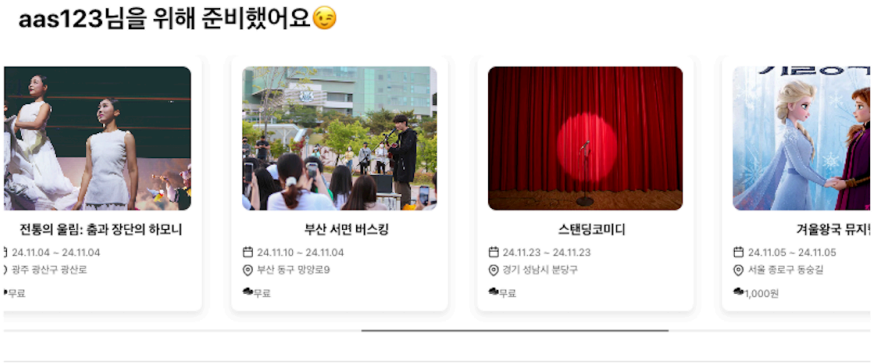
해결 과정

Redis에 데이터를 캐싱하여 불필요한 DB 조회를 줄이고, 스케줄러를 활용하여 주기적으로 정보를 업데이트하여 최신 데이터 보장

성능 개선 결과

nGrinder 부하 테스트에서 TPS : 74.7 → 6084.6, Mean Test Time : 1302ms → 9ms 로 성능 개선

사용자 맞춤 추천 공연 조회 구현
SQL 쿼리 튜닝을 통한 성능 개선 (65% 성능개선)



문제 상황
사용자 맞춤 공연 조회의 기존 쿼리는 JOIN된 전체 데이터를 대상으로 GROUP BY를 수행하여 불필요한 데이터도 연산에 포함되어 쿼리 실행 속도가 저하됨

해결 과정
서브쿼리를 사용하여 GROUP BY 연산을 미리 수행한 후, 불필요한 JOIN을 제거하고 필요한 데이터만 필터링하여 JOIN을 수행함으로써 쿼리 실행 속도 향상

성능 개선 결과
공연 데이터 7만 건 기준 평균 쿼리 실행 속도 200ms → 70ms 로 65% 성능 개선

EmbeddedRedis를 활용한 개발 환경 구성

문제 상황
일부 팀원이 로컬에 Redis가 설치되어 있지 않아 프로젝트 실행이 어려운 상황 발생

해결 과정
Embedded Redis를 도입하여 별도의 외부 환경 구성 없이 네이티브 환경에서 곧바로 실행할 수 있도록 개발 환경 구성

결과
별도의 환경 설정 없이 프로젝트 실행 가능하여 개발 효율성 증가

관련하여 블로그에 정리한 내용
블로그 링크

OAuth 2.0, JWT토큰을 활용한 소셜 로그인

문제 상황
기존에는 회원가입을 통한 이메일 로그인 방식만 지원하여 사용자 접근성 저하

문제 해결
소셜로그인을 통해 간편하게 가입하고, OAuth2.0, JWT 토큰을 활용하여 보안성을 고려한 소셜로그인 구현

결과
로그인 과정이 간소화되어 사용자의 접근성 향상

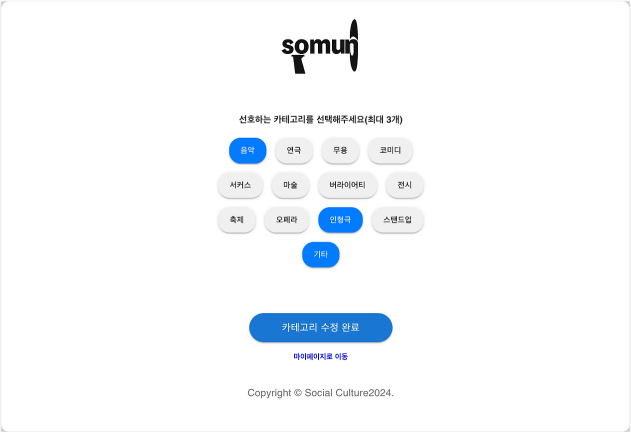
JavaMailSender를 활용한 SMTP 기반 이메일 인증

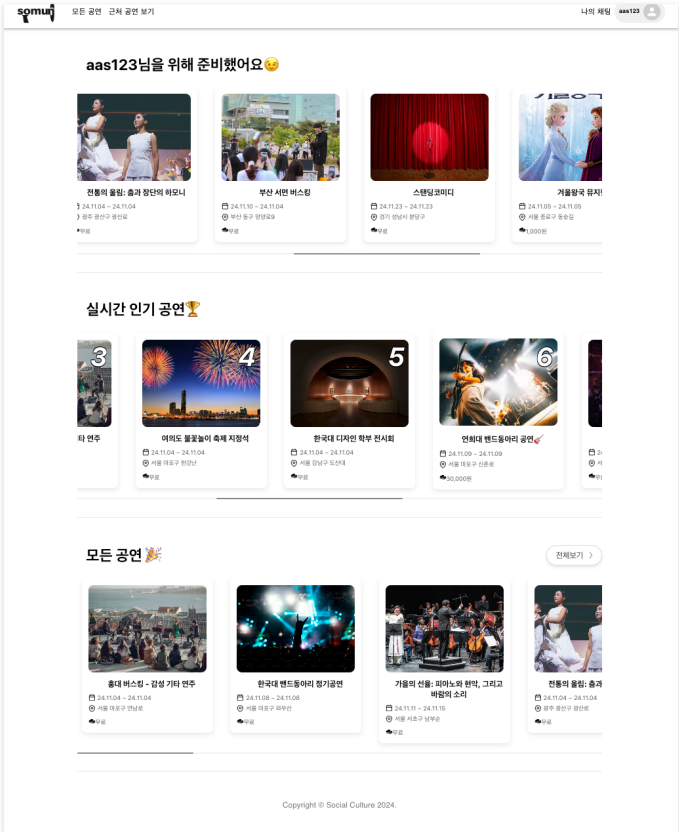
문제 상황
공연 관리자 권한 확인을 위해 사용자 인증이 필요

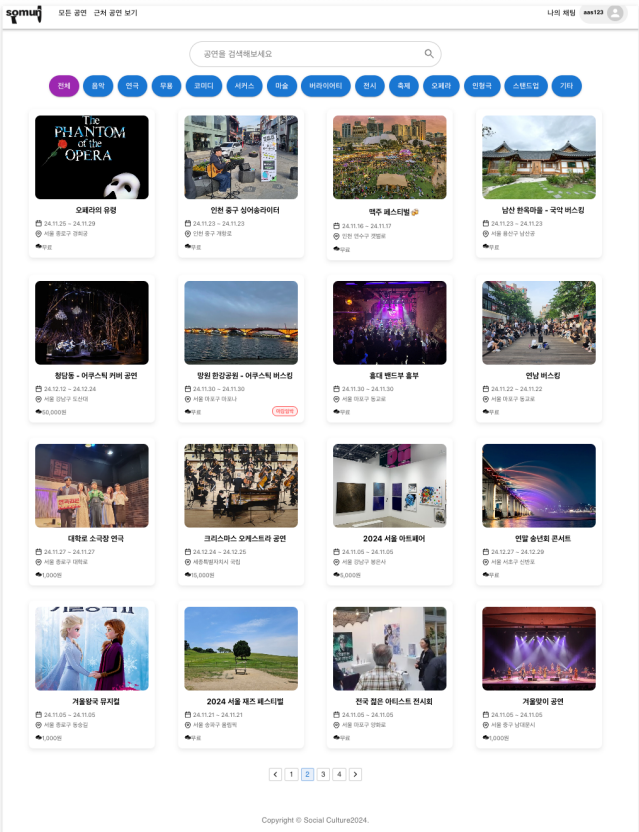
해결 과정
JavaMailSender와 Thymeleaf를 활용하여 SMTP 기반의 이메일 인증을 구현

결과
공연 관리자 권한 확인을 빠르고 쉽게 처리할 수 있게 됨

React를 활용한 다양한 웹 화면 구현







구현 내용

메인 홈을 포함하여 포트폴리오 상의 모든 화면을 React를 활용하여 구현
Axios Interceptor 적용하여 API 연동 최적화

Name : 김예찬
Email : yechan1601.dev@gmail.com
Phone : 010-4936-5651

읽어주셔서 감사합니다 :)