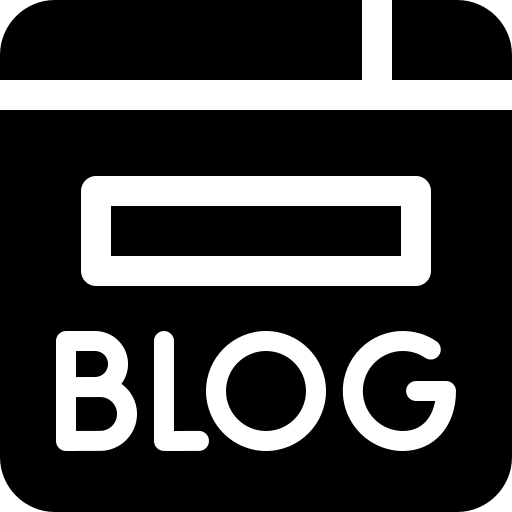
**김광현 Backend Engineer**



**010-5382-2137**



gwanghyeonkim01@gmail.com



[Github](https://github.com/gwanghyeon-k) [Blog](https://velog.io/@gwanghyeonkim/posts)

**SUMMARY**

**실제 기업과의 연계 프로젝트를 통해 실사용 유저들의 페인포인트를 분석하고 요구사항에 맞는 서비스를 설계하고 구현한 경험이 있습니다.**

**서비스 런칭을 진행해 트래픽을 받고 성능 개선 및 모니터링, 유지보수를 진행한 경험이 있습니다.**

**End-to-End 프로젝트** 개발 경험을 통해 기획부터 배포에 이르는 프로젝트의 과정을 이해하고 있습니다.

**Docker**를 이용하여 컨테이너 빌드하고, **AWS EC2**를 이용하여 클라우드 서버를 배포할 수 있습니다.

**PROJECT**

**무비부키** [Github](https://github.com/MOVIEBOOKIE/MOVIEBOOKIE_BE)

누구나 쉽게 영화관의 장소적 특성을 활용하여 일상을 특별한 순간으로 만들 수 있도록 돕는 영화관 대관 서비스

기간 : 2025.04 - 진행 중)

**Tech-Stack**

SpringBoot, MySQL, Github Actions, k6, Redis, NCP(Naver Cloud Platform)

**[락 기반 동시 접근 제어로 데이터 정합성 보장] -** [**blog**](https://velog.io/@gwanghyeonkim/%EB%9D%BD-%EA%B8%B0%EB%B0%98-%EB%8F%99%EC%8B%9C-%EC%A0%91%EA%B7%BC-%EC%A0%9C%EC%96%B4%EB%A1%9C-%EB%8D%B0%EC%9D%B4%ED%84%B0-%EC%A0%95%ED%95%A9%EC%84%B1-%EB%B3%B4%EC%9E%A5%ED%95%98%EA%B8%B0)

1. 여러 사용자가 동시에 이벤트를 신청할 때 최대 신청 가능 인원을 넘어가는 동시성 문제와, 카운트 증가 로직에서 데드락이 발생하여 화면에 보여지는 참여자 수와 실제 DB 레코드 간의 데이터 불일치가 발생
2. redis의 분산락을 이용하여 초기 구현햇지만 단일 인스턴스 서비스인 점을 고려하여 자바의 synchronized 도입해 스레드의 순차 진입을 보장하고, 비관적 락을 함께 사용해 DB 트랜잭션의 일관성을 확보
3. synchronized와 비관적 락 이용하여 최대 인원 초과를 방지하고, 서비스 로직 및 데이터 무결성을 안정적으로 보장
4. k6 200 가상유저로 30초 부하테스트 시 약 20000건의 요청 수에도 최대 인원 초과 신청 0%, 응답 지연을 300ms 이하로 단축하여 동시성 충돌과 데이터 불일치 문제를 해소

**[인증코드 인메모리 캐싱 최적화를 통한 응답속도 개선] -** [**blog**](https://velog.io/@gwanghyeonkim/%EC%9D%B8%EC%A6%9D%EC%BD%94%EB%93%9C-%EC%9D%B8%EB%A9%94%EB%AA%A8%EB%A6%AC-%EC%BA%90%EC%8B%B1-%EC%B5%9C%EC%A0%81%ED%99%94%EB%A5%BC-%ED%86%B5%ED%95%9C-%EC%9D%91%EB%8B%B5%EC%86%8D%EB%8F%84-%EA%B0%9C%EC%84%A0)

1. 단일 인스턴스 서비스인 점을 고려하여 Redis 대신 JVM의 힙의 인증 코드를 3분간 저장/검증하도록 변경하여, 네트워크 왕복 오버헤드 제거
2. 동일 키에 대한 10초 이내 재요청을 방지하여 인증코드의 중복 발급 방지
3. 무분별한 재요청 시 메모리 캐시에 인증 코드 항목이 과도하게 쌓이는 것을 방지하기 위해, 메모리 스케줄러를 1분 주기로 설정하여 캐시에 만료된 코드와 중복 요청 차단 락 정보를 제거하도록 설계하여 메모리의 누수가 없도록 함
4. 부하테스트를 통해 평균 응답 시간을 1,030ms -> 6ms로 redis 대비 오버헤드 없는 메모리 접근으로 응답 속도를 크게 향상

**[알림 전송 요청 비동기 처리 및 전송 자동 재시도]**

1. 최대 500건 알림을 한번에 전송할 때 메인 실행 흐름이 블로킹되어 응답 지연과 응답 시간 초과가 자주 발생
2. 알림 전송 작업이 메인 쓰레드에서 동기적으로 실행되어 대량의 알림 전송 요청을 처리하는 동안 전체 흐름이 블로킹 됨
3. 비동기(@async) 실행 기능과, 알림 전송 전용 스레드풀(최대 200개, 대기 500개)을 구성하여 알림 전송을 별도의 스레드풀에서 비동기 실행, 전송 실패 시 지정된 예외 상황에 대해 최대 3회 자동 재전송(@Retryable)을 적용해 오류를 자체적으로 복구하도록 설계
4. 스레드풀의 여러 스레드가 동시에 알림 전송 처리를 할 때, 이벤트:사용자별 식별 키를 추가하여 첫 시도에만 전송을 실행하여 이후 요청은 건너뛰도록 하여 중복 전송 요청이 발생하지 않도록 구현하여 알림 응답 처리 시간 70% 단축 및 메시지 누락 오류 가능성 제거

**Next-Page**  [GitHub](https://github.com/2023-Winter-Bootcamp-Team-J/NextPage)

사용자에게 스토리를 입력받아, 그림을 생성해주고, 릴레이 소셜 형식으로 분기를 생성해주는 서비스

기간 : 2023.12 – 2024.07

**Tech-Stack**

Spring boot, PostgreSQL, Neo4j, Nginx, Docker, AWS(EC2, S3, Lambda), Github Actions

**[서버리스 이미지 처리 환경 구축] -** [**blog**](https://velog.io/@gwanghyeonkim/AWS-Lambda%EB%A1%9C-%EC%9D%B4%EB%AF%B8%EC%A7%80-%EB%A6%AC%EC%82%AC%EC%9D%B4%EC%A7%95-%EC%9E%90%EB%8F%99%ED%99%94-%EC%A0%81%EC%9A%A9)

1. AI로 생성되는 이미지의 크기가 3MB로, 한 페이지에 수십 장의 이미지를 동시에 로드할 경우 페이지 로딩 속도 저하 문제 발생
2. 사용자에 요청마다 페이지에 적합한 사이즈의 여러 개의 이미지로 리사이징 작업이 추가로 필요했기 때문에, 서버의 부담이 증가할 수 있다고 판단
3. 서버리스 방식인 AWS Lambda를 활용해 메인 서버와 이미지 처리 로직을 분리하여 서버 부담 감소, 이미지 크기를 3MB -> 200~400KB로 줄여 평균 로딩 속도 80~90% 향상

**ACTIVITIES**

2023.12 ~ 2024.02 Techeer Silicon Valley Hackaton 수료 (우수 프로젝트 선정)

2024.02 ~ 현재 Techeer 7기 - 실리콘밸리 기술 기반 스터디 테커 멤버

2024.05 ~ 2024.05 Techeer Ideaton Hackaton 참여

2024.06 ~ 2024.08 Techeer Silicon Valley Hackaton 수료 (우수 프로젝트 선정)

2024.09 ~ 2024.12 TU-VCC(창업동아리) 3기 활동

2024.09 ~ 2025.01 UMC TUK 7기 Server part 학회원 활동

2025.02 ~ 2025.03 한국방송광고진흥공사(kobaco) 기업 연계 솔루션 제공 프로젝트

2025.02 ~ 2025.06 KUSITMS 31기 Backend 학회원 활동 (우수 프로젝트 선정)

2025.09 ~ 현재 UMC TUK 9기 Server part 활동

**EDUCATION**

**한국공학대학교**

컴퓨터공학과 | 2020.03 - 2026.02(졸업예정)