남은찬 신입

1999 (24세)

namhm23@naver.com 010-9922-2448

(16216) 경기 수원시 장안구 연무동

학력	전공	업무경험	희망연봉	포트폴리오
대학교(4년) 졸업예정...	컴퓨터공학	-	3,800~4,000만원	남은찬 포트... .pdf

백엔드/서버개발

안녕하세요. 밥 사주고 싶은 개발자 남은찬입니다.

* 끊임없이 배움을 추구하고 기술을 탐구하려 노력하고 있습니다.
* 팀 전체의 협업 분위기를 형성에 기여하는 긍정적이고 열려있는 태도로 함께합니다.
* 새로운 기술 탐구와 트러블 슈팅에 대해서 기록하며 공유하려고 노력합니다.
* `정확성', '속도', '가독성`의 교집합에 들어간 테스트 코드를 작성하려고 노력합니다.

저에 대해 더 자세히 알고싶다면?

나의 스킬

Java Spring Boot RDBMS MY-SQL JPA Querydsl Redis Swagger RestDocs GitHub Github Action Docker Amazon EC2 Amazon RDS Notion Slack

학력

대학교(4년) 졸업예정

경기대학교(수원)(4년제) 2018.03 ~ 2024.02 (졸업예정)

컴퓨터공학

학점 3.7/4.5

자격/어학/수상

정보처리기사 (최종합격) 2023.09

한국산업인력공단

SW 상상기업 장려상 2023.11

경기대학교

심화캡스톤경진대회 은상 2023.06

경기대학교

SW 상상기업 우수상 2022.11

경기대학교

경험/활동/교육

프로그래머스 : 클라우드 기반 백엔드 엔지니어링 5기 2023.09 ~ 2024.03

* Spring 백엔드 개발자 역량 - Spring, JPA, DB 등
* 클라우드 서비스 역량 - AWS, Docker, CI/CD 등
* 멘토 및 동료 개발자와 교류 - 코드리뷰와 프로젝트 협업 경험

InQ 2021.03 ~ 2024.02

경기대학교 컴퓨터공학부 개발동아리
현재 동아리내 백엔드 멘토로 활동중입니다

포트폴리오 및 기타문서

포트폴리오

남은찬 포트폴리오.pdf

자기소개서

자기소개

개발자는 더 나은 서비스를 제공하기 위해 고민하고, 이를 위해 새로운 기술을 학습하고 도전하는 도전 정신은 필수적인 역량이라고 생각합니다. 저는 항상 더 나은 서비스를 제공하기 위해 고민하고 이러한 도전 정신을 가지고 있습니다. 이에 대한 실천으로 이전에 수행한 'We Share WIsh Hair' 라는 프로젝트에서의 경험을 소개하겠습니다.
프로젝트엔 리뷰와 좋아요 기능이 있었습니다. 처음 구현에 리뷰의 좋아요 개수에 대해서 동시성 문제를 유발하지 않도록 좋아요 개수를 나타내는 변수를 리뷰에서 제거하고 좋아요 개수가 필요할 때 마다 좋아요와 조인하는 방식으로 구현했습니다. 하지만 이는 조회 성능에서 평균 7 TPS 이라는 심각한 문제를 유발했습니다. 그래서 이를 개선하고자 Reds를 활용하여 좋아요 개수 변수를 관리하며 동시성 문제없이 좋아요 개수 변수를 사용할 수 있었고, 조회 성능 또한 기존 7 TPS 에서 159 TPS 까지 끌어올릴 수 있었습니다.
기존 방법으로도 문제없이 동작하는 기능이었지만, 더 나은 서비스를 제공하기 위해 성능 테스트를 통해 개선점을 파악했고 개선하기 위해 Reds라는 새로운 기술을 학습하고 도전했습니다. 결과적으로 요구사항을 만족하며 더 좋은 성능의 서비스를 만들 수 있었습니다.

저는 앞선 소개 글에서 "밥 사주고 싶은 개발자"라고 저를 소개했습니다. 이런 소개를 한 데에는 제 협업 능력과 관련이 있습니다. 저는 팀 협업 분위기가 긍정적인 방향으로 흘러갈 수 있도록 주도적으로 나섭니다. 이전에 수행한 'We Share Wish Hair'에서는 팀장을 맡으며 이를 실천한 경험을 소개하겠습니다.
해당 프로젝트는 교내 팀 프로젝트로 팀원 중에 안면이 있는 팀원도 있었지만 아닌 팀원도 있었습니다. 효율적인 협업을 수행하기 위해 두 가지 방법을 실천했습니다. 주도적인 대화 시도와 주기적인 대면 모임이었습니다. 가볍게라도 주도적으로 대화를 시도하며 새로운 팀원과 대화를 끌어냈고, 대화에 대해서 거부감 없는 긍정적인 분위기를 조성할 수 있었습니다. 그리고 주기적인 대면 모임을 통해서 효과적으로 서로의 프로젝트 진행 현황을 파악하고 다음 작업에 대해서 계획할 수 있었고, 조직력을 더 단단하게 다질 수 있었습니다.
결과적으로 해당 프로젝트는 심화캡스톤경진대회에서 은상과 SW 상상기업 프로그램에서 장려상을 수상할 수 있었습니다.

학습과 성장에 있어서 공유는 본인의 지식을 단단하게 다지고 피드백을 통해 성장에 도움을 준다고 생각합니다. 평소에도 블로그를 통해 기록을 남기며 지식을 공유하는 활동을 해왔으며 현재 'Backend Explorer'라는 스터디 그룹을 만들어 그룹 내에서 블로그 포스팅 스터디를 주최하여 주기적인 블로깅과 지식 공유 및 피드백 활동을 이어가고 있습니다. 현재 지식에 안주하지 않고 성장해 나가기 위해 활동을 이어가고 있습니다.

직무 관련 경험 및 역량

제가 수행한 'Inter Ticket' 프로젝트에서 겪은 경험을 통해 직무 역량을 소개해보겠습니다.

[Redis 를 활용한 티켓 예매 동시성 제어 및 성능 개선]
티켓 예매는 동시성 이슈를 유발하기 때문에 기존에는 분산락을 사용한 방법으로 구현돼 있었습니다. 하지만 티켓 예매 비즈니스 특성상 많은 트래픽이 몰릴 것이라고 생각해서 대용량 트래픽 환경 속에서도 문제없이 동작하는 서비스를 만들고자 티켓 예매 기능을 개선하고자 했습니다.
먼저 비즈니스 특성을 분석했습니다. 티켓 예매 특성상 한번 예매된 좌석은 다른 사용자가 예매할 수 없다는 특성과 Redis 의 싱글 스레드로 작업을 처리한다는 특성을 이용해서 예매된 좌석의 정보를 Redis 에 저장하여 낙관적락을 구현하는 전략을 생각해 냈습니다. 핵심 로직 수행 전 Redis Key 체크와 핵심 로직 수행 후 Redis Key 체크 & Redis Key 세팅을 수행하며 Redis Key 체크 시 이미 Key 가 존재하면 실패시키는 방식으로 낙관적락을 구현했습니다. Key 체크를 앞뒤로 수행하며 검증과 예매에 사용되는 불필요한 쿼리를 막을 수 있었고, DB 업데이트에 실패하여 롤백 되는 경우에 선점한 Key 로 인해 발생하는 문제를 방지했습니다.
실제로 적용 후 100번의 동시적 요청에서 동시성 이슈가 발생하지 않으며 Select 쿼리를 200번에서 20번으로 줄일 수 있었고, 별도로 진행한 성능 테스트에서 평균 644 TPS 에서 4,259 TPS 까지 성능 향상을 기록했습니다.

[배치 서버의 failover 환경 구축]
프로젝트에 '예매 대기' 기능과 '랭킹 조회' 기능을 구현하며 스케줄러 작업이 필요해서 별도의 배치 모듈을 구현하여 배치 서버 환경을 구성했습니다. 당장 서버가 장애가 발생한 상황이 아니지만 언제 닥칠지 모르는 문제 상황에 대해서 장애 대응을 갖춘 서버를 구축하기 위해서 분산 배치 서버 환경을 구성하고자 했습니다. 하지만 배치 서버는 스케줄러 작업 때문에 분산 환경에서 작업이 중복적으로 실행되어 문제가 됩니다. 이를 해결하기 위해 한 개의 서버만 작동할 수 있도록 리더 일렉션 기술을 찾아보던 중 ShedLock 기술을 접했습니다. 앞서 찾아보던 리더 일렉션의 개념은 아니지만 스케줄러가 분산 환경에서 동기화된 작업을 수행할 수 있도록 지원했습니다. 그래서 ShedLock 기술을 학습하고 적용하여 스케줄러 작업을 동기화 시키며 배치 서버의 failover 환경을 구축할 수 있었습니다.

[MySQL 인덱스를 활용한 랭킹 조회 쿼리 튜닝]
랭킹 조회 기능은 캐시를 사용하며 한 시간에 한 번씩 배치 작업을 통해 DB 에서 데이터를 조회해 캐시를 업데이트하는 방식으로 구현됐습니다. DB 에서 데이터를 조회하는 과정에서 공연 (카테고리 개수 x 집계기간 개수) 만큼 쿼리가 발생합니다. 그래서 랭킹 조회 쿼리가 비효율적이라면 한 시간마다 DB 에 큰 부하를 줄 것이라고 생각해서 쿼리를 튜닝하기로 했습니다.
먼저 where 조건문과 join 조건문에 사용되는 칼럼들에 대한 인덱스를 만들었습니다. 하지만 inner join 에서 MySQL 의 옵티마이저가 조인 드라이브 테이블 순서를 변경하는 이슈로 만든 인덱스가 적용되지 않았고, straight join 명령어를 추가해서 조인 드라이브 테이블 순서를 명시하여 이를 해결했습니다. 결과적으로 5.64초가 걸리는 쿼리를 3.25초 까지 단축할 수 있었습니다.
여기에 안주하지 않고 추가적인 튜닝에 도전했습니다. 이번엔 조인 드리븐 테이블에 사용되는 인덱스를 커버링 인덱스로 활용했습니다. 기존 인덱스에 하나의 칼럼만 추가하면 커버링 인덱스로 활용할 수 있기 때문에 새로운 복합 칼럼 인덱스를 구성하여 최종적으로 1.02초 까지 단축할 수 있었습니다.

취업우대사항

병역 : 군필 (육군/병장)

2019.04 ~ 2020.11

만기전역