꾸준히 공부하는 개발자, 박상준

INTRODUCE

Java/SpringBoot 기반 웹·앱 서비스 개발을 3년간 경험한 비전공 출신 백엔드 개발자입니다.

전공자가 아님에도 개발에 시간을 많이 쏟아부어 내가 우러러 보는 개발자분들 처럼 되고 싶은 미생입니다.

사내에서는 코드 리뷰와 기술 토론에 적극적으로 참여하여 좋은 개발 문화를 형성하고 있으며, 개인 시간에는 외부 스터디를 통해 다양한 사람들과 기술 경험을 공유하고 학습하며 계속해서 발전하고 있습니다. 또한, 새로운 기술 동향을 파악하고 배운 지식을 사이드 프로젝트에 적용하는 것을 즐깁니다.

사용자를 중심으로 생각하며 서버 개발자의 영역에만 국한되지 않고 이해관계자들과 협업하여 비즈니스 문제를 해결하며 회사의 성장에 기여하는 것을 좋아합니다. 능동적이고 적극적인 의사소통 능력으로 비즈니스 발전에 참여하고 있습니다. 이러한 점을 바탕으로 "함께 일하고 싶은 사람"이 되기 위해 끊임없이 성장하고 노력하고 있습니다.

저는 아래 가치를 중요하게 생각합니다.

Active Learning

- 단순한 실패나 실수도 배움의 기회로 삼고
  혼자 잘하는게 아니라, 배운 걸 팀과 공유하는 것이 중요하다고 생각합니다.
- 저는 사내 게시판에 사내 js css 등에 대한 레디스 캐싱 최신화 (.css?v={version} 같은 캐싱갱신을 위한 매개변수) 방법이나 스테이지에서 ELK 색인 용량 초과시 대처방법 등에 대해 문서화를 통하여
  향후 다른 개발자들이 들어와도 해당 문서를 통해 헷갈리지 않도록 게시물을 계속적으로 작성하였습니다.
- 또한 블로그 운영을 통하여 내 지식에 다른 사람들이 도움을 받거나 내 생각을 정리하여 블로그에 올릴 수 있도록 함으로 해당 가치를 정말 소중하게 여기고 있음을 알 수 있습니다.

개인 블로그

- https://until.blog/@qkrtkdwns3410

깃허브

- https://github.com/qkrtkdwns3410

EDUCATION

건국대 글로컬 - 관세물류학과

2014.03 ~ 2022.02

- 학부 전공은 물류/무역 분야였지만, 재학 중 IT와 개발에 흥미를 느껴 진로를 전환하였습니다.

QUALIFICATIONS

Amazon Web Services (AWS) Solution Architect Associate - C03

- 2024.11

정보처리기사

- 2024.06

EXPERIENCE

2024.05 ~

아이템베이

- 아이템베이 사이트 유지보수
  - 레거시 프로젝트 구조를 분석하여 기존 로직과 충돌 없이 안전하게 설계하였으며, 회원 인증 및 마일리지 전송부터 계좌 직접 송금까지 **신규 기능을 안정적으로 개발·적용**하였습니다.
  - 메인 배너의 `구성 정보(노출 여부, 정렬 등)`를 Redis에 캐싱하여 DB 부하 없이 빠른 응답을 제공하고,
    트래픽이 집중되는 메인 페이지에서 조회 성능과 안정성을 증대하였습니다.
- 커머스 내부 프로젝트

    - 관리자와 사용자 간 실시간 리소스 동기화의 강결합 문제 해결을 위해 RabbitMQ 기반 Pub/Sub 모델 도입
      Spring AMQP 연동 용이성과 Kafka 대비 간단한 설정으로 선택하여 모듈 간 독립성과 유지보수성을 확보함.
    - OpenFeign 사용 시 재시도 설정 미비로 인한 외부 API 호출 실패 문제를 인식하고, Retryer 및 ErrorDecoder 설정을 통해 503 오류 대응 및 호출 안정성 확보를 구현했습니다.
      이를 통해 일시적 장애 상황에서도 성공률을 높여 UX를 개선할 수 있었습니다.
- 신한카드 연동 내부 광고 플랫폼 관련 프로젝트 개발
    - 광고 시청 후 포인트 적립 시 발생한 Lost Update 문제를 해결하기 위해, `JPA의 비관적 락(Pessimistic Lock)`을 적용하여 동시 요청에 의한 중복 적립을 방지했습니다.
      데이터 일관성과 충돌 빈도를 고려해 재시도 없는 안정적인 적립 처리를 구현.
- **Skill Keywords**
    - `Java` `Spring Boot` `JPA` `MySQL` `Redis`

2022.08 ~ 2024.01 (주)이파트

- 사내 솔루션 Spring Boot 마이그레이션 처리
    - 레거시 하드코딩된 소셜 로그인 로직을 Spring Security OAuth2 기반으로 통합하여 유지보수성과 확장성을 크게 개선했습니다.
      인증/인가 처리를 일원화해 신규 플랫폼 추가와 사용자 매핑 로직을 효율적으로 구현했습니다.
- 이메일 전송 기능 개선
    - 기존 동기 방식의 메일 발송을 @Async 기반으로 비동기 처리하여 사용자 응답 속도를 개선하고, REQUIRES_NEW 트랜잭션 분리 및 프록시 구조 수정으로 메일 발송과 로그 저장의 안정성까지 확보했습니다.
- **Skill Keywords**
    - `Java` `SpringBoot` `JPA` `QueryDsl` `MySQL`

ETC

2024.01 ~ 2024.09 청소년기반 인기투표 사이드 프로젝트 진행

- 결론적으로는 프로젝트원들의 개인 사정으로 프로젝트 중단되었습니다.
- 신입 개발자분들에게 제가 아는 지식을 알려드리고, 서로 스터디도 진행할 수 있는 좋은 기회였습니다.

2025.05 ~ 퇴근 후 프로젝트 - 케이크 주문제작 사이트

- 케이크 주문제작을 위한 플랫폼을 기획하고, 사용자·판매자·관리자용 3개의 독립된 서비스로 개발 예정
- 공통 기능(인증, 공통 예외 처리, 로깅, 응답 포맷 등)을 별도의 **공통 모듈로 구성**하여, 각 플랫폼 서비스에서 모듈을 의존성으로 가져가도록 설계중입니다.
- 프로젝트 구조를 멀티 모듈 형태로 구성하여 **재사용성과 유지보수성**을 강화
- 총 개발 기간은 약 7개월 이상 예상

상세 경력기술서, 박상준

EXPERIENCE

2024.05 ~

아이템베이

- 스포츠카드 중개 사이트 서비스 개발
- 신한카드 포인트 연동 외부 프로젝트 개발
- 이벤트 및 쿠폰 발행 로직 개선
- Redis 캐싱 도입으로 메인 배너 조회 속도 개선
- 신규 프로젝트 테스트 코드 도입
- 레거시 청산 및 기존 비즈니스 유지보수
- **Skill Keywords**
  - Java Spring Boot JPA QueryDsl MySQL MSSQL Vue.js JSP

2022.09 ~ 2024.01

(주) 이파트

- CMS 개발
- 기존 개발 환경 개선
- **Skill Keywords**
  - Java Spring Boot JPA JPQL QueryDsl MyBatis MySQL Jsp

PROJECT

스포츠카드 거래 플랫폼 개발 (개발자 전원)

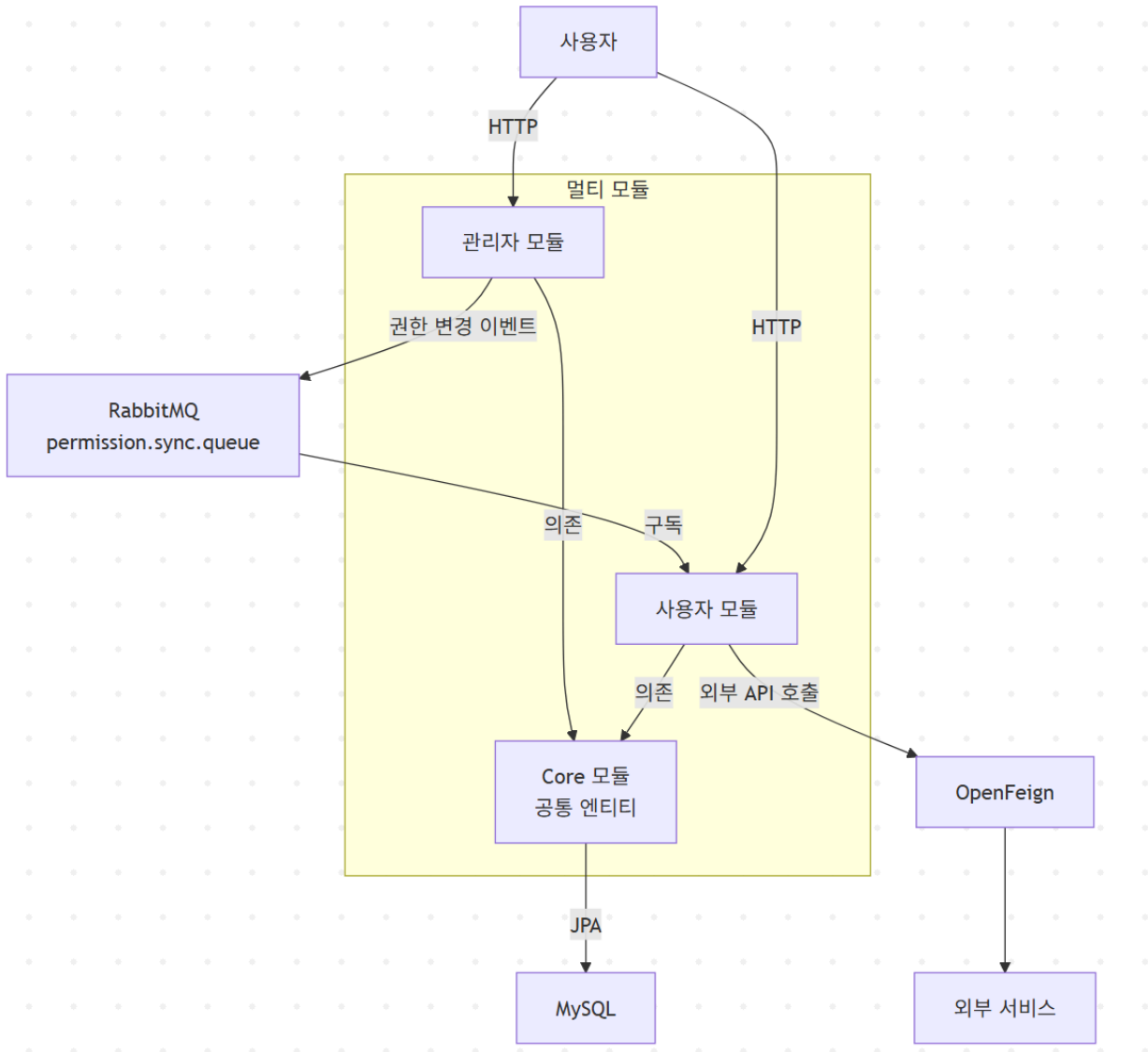
2024.12 ~ 진행 중

Spring Boot 2.7 Java 17 RabbitMQ OpenFeign MySQL

프로젝트 목적

스포츠카드 중개 사이트 개발을 통해 사용자들이 스포츠카드를 거래하고, 관련 정보를 공유하며, 실시간으로 상호작용할 수 있는 플랫폼을 제공하는 프로젝트입니다.

인프라 아키텍처 설계



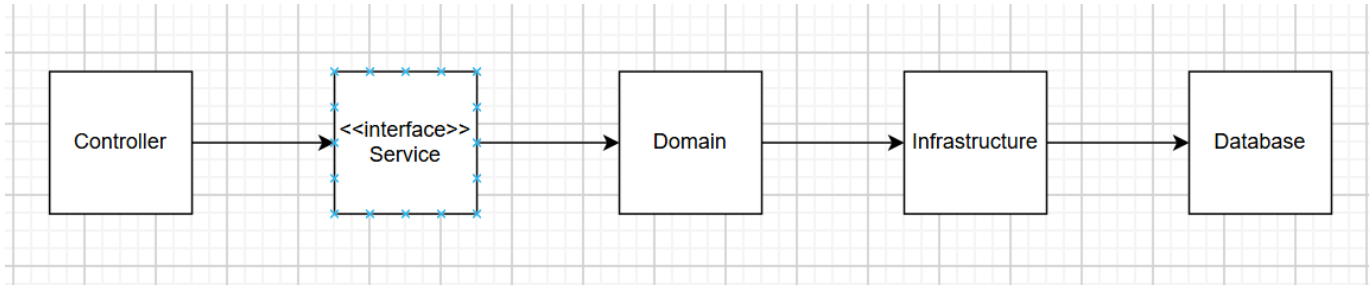
메시지 브로커 선정 기준

- **강결합 문제 해결**
  - 관리자와 사용자 모듈 간 시큐리티 정보(리소스 정보) 동기화는 실시간이지만, 강한 결합은 피해야 했습니다.
  - Pub/Sub 모델을 통해 비동기적으로 전달해 모듈 간 독립성을 유지하려고 도입했습니다.
- **설정이 Kafka에 비해 복잡하지 않다고 생각했습니다**
  - Kafka 는 실무에서 사용한 경험은 없지만 사이드 프로젝트에서 한번 세팅한 적이 있었는데 상당히 세팅하기 복잡했고, 기능도 워낙에 많아서 혼자서 감당하긴 어렵다고 생각했습니다.
- **Spring Boot와의 통합성**
  - Spring AMQP 라이브러리를 통해 애플리케이션에서 쉽게 연결, 메시지 전송, 수신, 에러 처리 등을 추상화해 개발자가 복잡한 저수준 AMQP 프로토콜을 직접 다루지 않게 도와줍니다.

어떤 작업을 수행했나

- 기본적인 세팅의 경우 선임분들의 도움이 있긴했습니다. 기본적인 설계 구상단계 참여하였으며, 메시지 브로커 구조는 다음과 같이 구성했습니다.
  - Exchange 구성
    - `direct exchange`
      - 라우팅 키 기반 특정 큐에 정확히 매칭되는것만 전달
  - DLX 설정
    - 관리자와 사용자간의 메시지 송 수신 작업에는 DLX 발생여부가 그닥 중요하진 않았습니다
    - 제가 개발하진 않았지만 채팅쪽에서 실패하는 작업에 대비하기 위해 DLX 설정은 필수적이였습니다
    - 소비되지 못하거나 재시도 횟수는 넘는 메시지는 DLX 로 전달됩니다.
      - DLX 쌓인 메시지는 rabbit mq 관리자 페이지에서 확인가능하며, 별도 알림은 추후 개발예정으로 후순위로 두었습니다.
    - DLQ 에서 TTL 설정을 하지 않으면 메시지가 사라지지 않는다는 것을 알고, 일단 대략 일주일로 설정해두었는데 요부분은 운영까지에 대한 지식은 부족해서 맞는지 틀린지 판단이 어려웠습니다.

애플리케이션 아키텍처 설계



수정된 아키텍처 설계 기준

- **비즈니스 중심 설계**: 아키텍처는 도메인 계층을 중심으로 의존성 방향이 일관되도록 설계되었습니다. Controller가 Service 인터페이스를 통해 도메인 계층과 상호작용하며, 비즈니스 로직의 변경이 발생하더라도 Infrastructure나 Database 계층에 최소한의 영향을 미치도록 하였습니다.
- **유연성과 확장성**: 의존성 역전 원칙(Dependency Inversion Principle)을 적용하여 Controller가 Service 인터페이스에 의존하도록 설계되었습니다. 이를 통해 계층 간 느슨한 결합을 유지하며, 향후 JPA 기반의 Infrastructure 변경이나 Database 기술 교체가 필요할 때도 도메인 계층에 영향을 최소화하고 유연하게 확장할 수 있습니다.

또한, 기존 팀에서 처음 도입하는 구조이기 때문에 기존 계층형 아키텍처의 장단점을 분석하고, 팀 구성원들이 새로운 DDD 기반 아키텍처를 이해하고 수용할 수 있도록 개발 가이드 문서화와 팀 내부 발표 등의 활동을 통해 성공적으로 도입하였습니다.

은행 API 연동 광고 기반 플랫폼 (단독 개발)

- 프로젝트 개략 설명
  - 신한카드 API 를 활용하여 관리자에서 광고 등록시, 신한카드앱에서 이어진 웹앱으로 기간에 맞는 광고가 노출되며 포인트등을 적립받는 서비스입니다.
  - 약 90 qps 정도의 서비스입니다.

2025.01 ~ 2025.05

`Spring Boot 2.7` `Java 17` `JPA` `QueryDSL` `MySQL 8.0`

프로젝트 목적

- 사용자 광고 참여 유도
  - 포인트 적립이라는 보상을 통해 사용자가 서비스를 더 자주 이용하도록 장려
- 외부 API 연동을 통한 서비스의 확장
  - 외부 은행 API 를 활용해 포인트 적립 시스템 구현, 사용자에게 더 다양한 혜택을 제공함

포인트 적립관련 Race Condition 문제 해결

상황

광고 시청 후 포인트를 적립하는 기능에서, 동일 사용자에게 짧은 시간 안에 여러 요청이 동시에 들어올 경우 `Race Condition(경쟁 상태)` 이 발생하여 포인트가 중복으로 적립되는 문제를 발견했습니다.

실행

- 문제의 원인이 '상태 확인'과 '데이터베이스 업데이트' 사이의 시간차로 인해 발생하는 `Race Condition`임을 파악했습니다.
- 데이터의 강력한 정합성이 필수적인 포인트 시스템의 특성을 고려하여 JPA의 `비관적 락(Pessimistic Lock)` 을 해결책으로 선택했습니다. 비관적 락은 데이터에 접근하는 순간 락을 걸어 다른 트랜잭션의 동시 수정을 원천적으로 차단하므로, 충돌이 빈번할 수 있는 환경에서 안정적인 처리를 보장할 수 있다고 생각했습니다.
- 사용자 포인트와 관련된 트랜잭션 시작 시, `SELECT ... FOR UPDATE` 쿼리를 통해 해당 사용자의 레코드에 락을 획득하고 포인트 적립 로직을 수행하도록 구현했습니다.

> 비관적락 프로세스 예시
>
> 트랜잭션 시작 `->` 광고 시청 기록 확인 및 저장 `->` 비관적 락 획득 `->` 포인트 적립 `->` 트랜잭션 종료

결과

`Race Condition`으로 인한 **포인트 중복 적립 문제를 해결**하고, 다중 요청 상황에서도 **데이터의 정합성을 100% 보장**하는 안정적인 포인트 시스템을 구축했습니다.

아이템베이 메인배너 레디스 캐싱

상황

- 아이템베이 메인페이지의 **배너 데이터**는 매번 DB에서 직접 조회하고 있었음
- 메인페이지는 DB 부하는 최대한 주지 않는 선에서 운영해야하는 것으로 인지. 해당 내용 개선이 필요했습니다.

실행 - 결과

- 메인배너 데이터를 Redis에 캐싱하여, **DB 직접 호출 → Redis 조회 방식으로 전환**
- 관리자가 배너를 수정하거나 갱신 버튼을 누르면, **DB에서 데이터를 재조회 후 Redis에 동기화**되도록 구현
- TTL 의 경우 관리자에서 설정된 게시기간이 종료시까지 설정
- 캐싱 적용 이후 메인 배너 영역 DB 부하 감소

Open Feign 재시도 관련 문제 인식

상황

외부 은행 API 를 통해 회원에 대한 UUID 등, 포인트 적립 등의 행동을 하는데, Open Feign 을 사용했으나 재시도 설정을 하지 않아 일시적인 네트워크 오류나 외부 서비스 응답 지연(503) 시 요청이 바로 실패로 처리되었습니다.

이로 인해 회원 정보 호출 실패 등의 UX 가 저하되는 문제가 있었습니다.

실행

- Open Feign 의 재시도 설정을 추가해 외부 API 호출 안정성을 높였습니다.
- 기본적으로 Open Feign 은 재시도를 하지 않기에, `Retryer` 와 `ErrorDecoder` 를 설정해 일시적 오류 (503 Service Unavailable) 에 대해 재시도를 수행하도록 했습니다.

결과

이러한 변경으로 외부 API 호출의 안정성이 개선되었습니다.
일시적인 오류가 발생해도 최대 3번 재시도하여 성공 확률이 높아졌으며, 회원정보 누락의 문제가 발생할 가능성이 매우 적어졌기에 UX 가 향상되었습니다.

사내 솔루션 소셜 로그인 업그레이드

2022. ~ 2022.07

`Spring Boot 2.4` `Java 8` `MyBatis` `MySQL 8.0`

프로젝트 목적

스프링부트 버전의 사내 프로젝트 버그 픽스 및 소셜 로그인 기능 코드 효율성 증대

상황

사내 프로젝트에서 소셜 로그인(카카오, 구글 등)은 레거시 구조로 구현되어 있었습니다.
각 소셜 플랫폼별로 별도의 로그인 로직이 하드코딩되어 있어 새로운 소셜 플랫폼을 추가하거나 기존 로직을 수정하는 데 어려움이 많았습니다. 또한, 인증/인가 처리가 일관되지 않아 유지보수가 힘들었고, 사용자 경험이 저하되었습니다.

실행

레거시 소셜 로그인 시스템을 Spring Boot OAuth2 로 전환하기 위해 다음과 같은 작업을 수행했습니다.

- Spring Security OAuth2 를 설정해 카카오, 구글 등 다양한 소셜 로그인을 통합적으로 지원하도록 구성했습니다. 각 플랫폼의 클라이언트 ID 와 시크릿 정보를 환경 설정 파일에 정의해서 관리
- OAuth2 사용자 서비스를 커스터마이징하여 소셜 로그인 후 사용자 정보를 통합적으로 처리했습니다. 소셜 로그인 성공 시 기존 사용자와 매핑하거나 신규 사용자로 등록하는 로직을 구현했습니다.

결과

Spring Boot OAuth2로 소셜 로그인 시스템을 포팅하면서 통합된 인증/인가 시스템을 구축했습니다. 이를 통해 새로운 소셜 플랫폼 추가가 쉬워져 확장성이 향상되었습니다.

레거시 청산 및 기존 비즈니스 유지보수

프로젝트 마이그레이션시 트랜잭션 관련 누락

상황

레거시 청산 및 기존 비즈니스 유지보수 과정에서, 신규 스프링 부트 CMS 프로젝트로 마이그레이션을 진행했습니다. 기존 레거시 시스템에서는 글로벌 트랜잭션 설정이 적용되어 있어 별도로 @Transactional 어노테이션을 메서드 단위로 붙이지 않아도 트랜잭션이 자동으로 관리되었습니다.

하지만 신규 스프링 부트 CMS 프로젝트로 마이그레이션하면서 글로벌 트랜잭션 설정이 누락되었고, @Transactional 어노테이션도 개별 메서드에 추가되지 않았습니다.

이로 인해 서비스 오류 발생 시 롤백이 이루어지지 않는 문제를 발견했습니다. 예를 들어, CMS 콘텐츠 업데이트 중 오류가 발생했을 때 데이터베이스 변경 사항이 롤백되지 않아 데이터 불일치가 발생했습니다.

실행

트랜잭션 문제를 해결하기 위해 다음과 같은 작업을 수행했습니다.

- 기존 레거시 시스템의 글로벌 트랜잭션 설정을 분석하여, 신규 스프링 부트 CMS 프로젝트에 동일한 설정이 누락되었음을 확인했습니다.
- 스프링 부트에서 글로벌 트랜잭션 관리를 활성화하기 위해 @EnableTransactionManagement 어노테이션을 메인 애플리케이션 클래스 또는 설정 클래스에 추가했습니다.
- 서비스 레이어의 주요 메서드들(예: 콘텐츠 저장, 수정, 삭제)에 @Transactional 어노테이션을 명시적으로 적용하여 트랜잭션이 필요한 작업에 대해 명확히 관리되도록 했습니다.
- 트랜잭션 매니저가 올바르게 설정되었는지 확인하고, 데이터베이스 작업 시 트랜잭션 경계가 적절히 설정되도록 검토했습니다.
- 테스트 환경에서 의도적으로 오류를 발생시켜 롤백 동작을 검증했습니다.

결과

CMS 콘텐츠 업데이트 중 오류가 발생했을 때 변경 사항이 롤백되어 데이터 일관성이 유지되었습니다. 이를 통해 신규 스프링 부트 CMS 프로젝트의 안정성이 향상되었고, 기존 비즈니스 로직이 안정적으로 동작함을 확인했습니다. 또한, 글로벌 트랜잭션 설정 누락으로 인한 잠재적 문제를 방지할 수 있었습니다.

메일 발송 프로세스 비동기 처리로 리팩토링

상황

기존 시스템에서 메일 발송 프로세스는 동기적으로 구성되어 있었습니다. 사용자가 특정 작업(예: 회원 가입, 주문 완료 등)을 완료하면 메일 발송 로직이 동기적으로 실행되어,

메일 서버의 응답을 기다리는 동안 사용자 요청이 지연되었습니다.

이로 인해 사용자 응답 시간이 길어졌고, 메일 발송이 실패하거나 지연될 경우 전체 프로세스가 영향을 받아 사용자 경험이 저하되는 일이 자주는 아니지만 간혹 있었습니다.

실행

메일 발송 프로세스를 비동기 처리로 리팩토링하기 위해 다음과 같은 작업을 수행했습니다.

- 메일 발송 로직을 별도의 비동기 메서드로 분리했습니다.
- 스프링의 @Async 어노테이션을 활용하여 메일 발송 메서드를 비동기로 실행하도록 설정했습니다.
- 비동기 작업이 올바르게 처리되는지 확인하기 위해, 메일 발송 실패 시 예외를 로깅하고 재시도 로직을 추가로 검토했습니다.
- 테스트 환경에서 동기와 비동기 처리의 응답 시간을 비교하여 성능 개선 여부를 검증했습니다.

추가중에 기술적인 문제가 몇 개 있었습니다.

1. **비동기 메서드 호출 실패**

   `@Async` 메서드가 비동기로 실행되지 않고 동기적으로 동작하는 문제가 있었습니다.

   동일 클래스 내에서 `@Async` 메서드를 직접 호출하는 경우 프록시가 적용되지 않아 비동기 처리가 동작하지 않는 다는 점을 발견했습니다.
   메일 발송 로직을 별도 서비스 클래스로 분리 + 해당 클래스의 메서드에
   `@Async` 를 적용했습니다.

2. **트랜잭션 누락 문제**

   `@Async` 메서드와 별도로 트랜잭션이 필요한 데이터베이스 작업에 `@Transactional` 어노테이션을 명시적으로 추가했습니다.

   메일 발송과 로그 저장은 별도의 트랜잭션으로 분리해야 했습니다. 메일이 발송되었음에도 불구하고 로그 저장이 오류로 인해 저장되지 않는 케이스가 존재했기에, 로그 저장의 트랜잭션 propagation 설정을 `REQUIRES_NEW` 로 설정했습니다.

결과

@Async를 적용하여 메일 발송 프로세스를 비동기로 변경한 결과, 사용자 요청 처리 시간이 크게 단축되었습니다

실제 서비스의 경우 사용자가 사실 엄청나게 많지 않아 티는 별로 안났지만 Jmeter 같은 측정 도구로 확인한 결과 기대 응답 시간이 약 50% 이상 빨라진 것을 확인했습니다.
