



안녕하세요. 백엔드 개발에서 사용자 경험의 본질을 찾는 개발자 김승원입니다.

Spring WebFlux기반 리액티브 프로그래밍, 웹 애플리케이션 개발에 관심이 많으며,
전교생이 사용하는 이력서 작성 플랫폼 프로젝트를 리드하여 개발해 안정적인 운영을 이루어냈습니다.

저는 사용자 경험을 개선하고 복잡한 로직을 단순화하는 것을 중요하게 생각하여,
실제 프로젝트에서 비동기 아키텍처 단순화 및 사용자 경험 개선을 위해 고민 해왔습니다.

끊임없이 학습하고 도전하며, 복잡한 로직을 단순화하고, 안정적인 서비스를 제공하는 사용자 중심 서비스를 만들어내는 개발자
로 성장하는 것이 저의 목표입니다.

Contact

- Phone: 010-2977-8517
- Email: kim@seungwon.me
- Linkedin: <https://linkedin.com/in/oriOoOp>
- Github: <https://github.com/oriOoOp>
- Velog: <https://velog.io/@virtualthread>
- Birth: 2007.03.11

Certifications

정보처리기능사 ————— 2023

TOPCIT(소프트웨어 역량 검정) 595점 ————— 2024

Projects

우리의 학교생활 이야기 대마위키에서.

학생들의 정보와 학교에서 일어난 사건을 기록하는 곳,
대마위키에서 함께 정보를 공유하고 학교 생활을 기록해보세요.
나만 몰랐던 우리 학교 이야기.

인기있는 문서들

다양한 주제와 깊이 있는 정보로 학생들의 호기심을 자극하는 인기있는 문서들이 모여있습니다.

99 HOT 99 99

새벽에 기숙사 탈출 험부림 미수 사건 김승원 공략법 v0.3

개발 기간: 2024년 6월 ~ 2025년 2월

서비스(운영) 기간: 운영 준비 중

깃허브 레포지토리: <https://github.com/daemawiki/Claude>

개발 인원: 백엔드 1명, 프론트엔드 2

기술 스택:

- Backend: Java, Spring WebFlux, Spring Security, Spring RSocket
- Database: MongoDB, Redis, AWS S3

대마위키 (운영 준비 중)

프로젝트 개요

대마위기는 대덕소프트웨어마이스터고등학교 학생들을 위한 정보 공유 플랫폼입니다.

Spring WebFlux 기반의 리액티브 웹 애플리케이션으로 구축되어, 높은 성능과 효율적인 리소스 관리
를 목표로 설계되었습니다.

현재는 운영 준비 중이며, 개발 완료 후 내부 테스트를 거쳐 서비스될 예정입니다.

담당 역할: 백엔드 리드를 맡아 전체 API 설계 및 구현을 주도하였으며, 함수형 프로그래밍과 리액티브
프로그래밍 패러다임을 활용하여 고성능 웹 애플리케이션을 구축하였습니다.

주요 개선점(핵심 기여)

1. 실시간 문서 공동 편집 기능 구현

- 문제점: 여러 사용자가 동시에 문서를 편집할 때 발생하는 갱신 손실 발생 (문서 단위로 갱신 손실 발생)
- 해결방안: WebSocket과 작업 동기화를 통해 실시간 문서 공동 편집 기능 구현
 - 커스텀 Flow 객체를 공유하여 데이터 일관성 유지
 - 문서의 열 단위로 객체를 관리

- **Realtime Communication:** WebSocket
- **Architecture:** Project Reactor 기반의 비동기 아키텍처

• 성과:

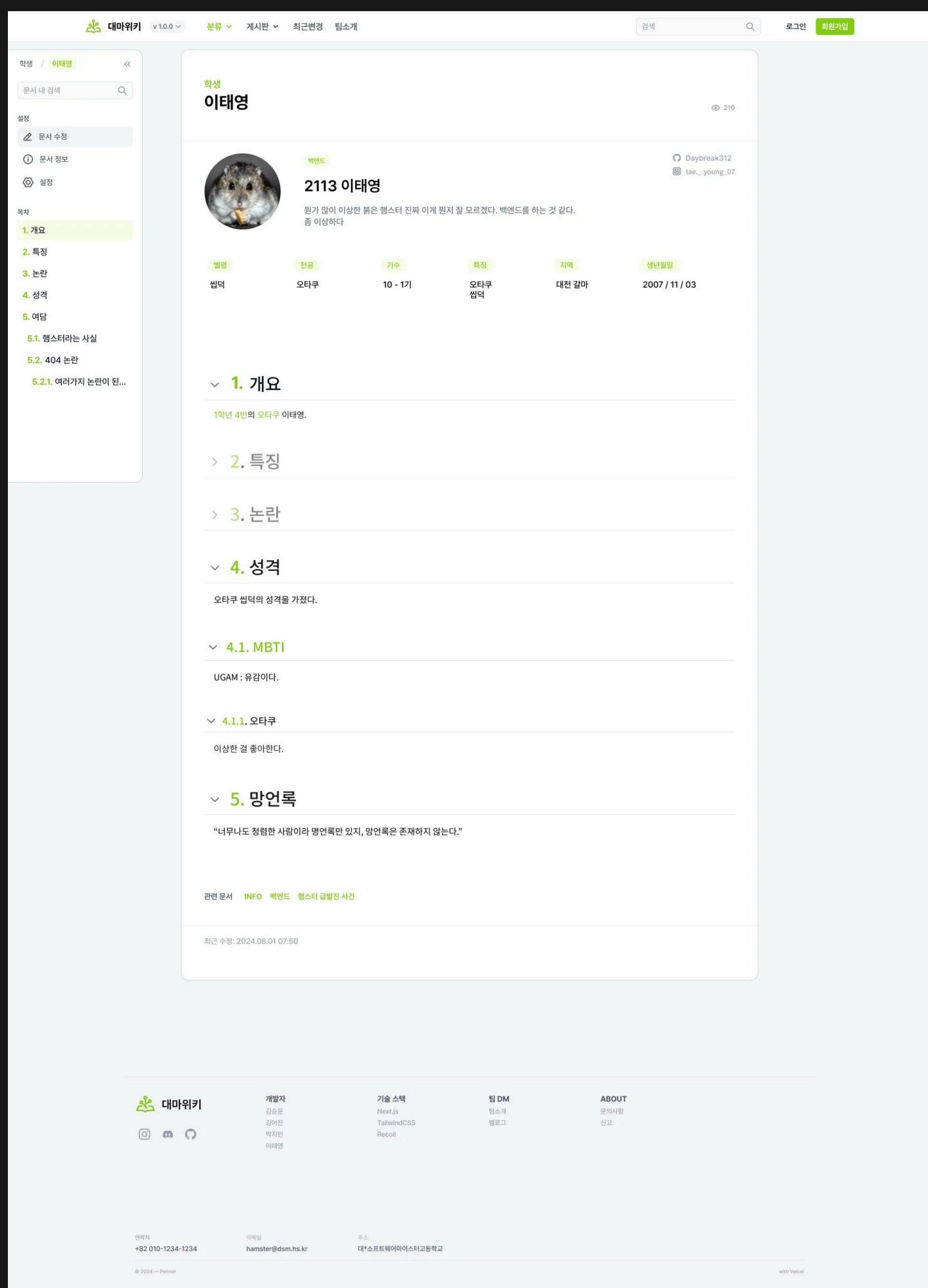
- 여러 사용자의 동시 편집 가능
- 간신 손실 문제를 최대 열 단위로 최소화
- 노션과 유사한 객체 분리 방식(열 단위의 객체 분리)으로 동시성 관리 용이성 향상

• 추후 개선 고민:

- CRDT 알고리즘, OT 알고리즘과 같이 실시간 협업 병합 알고리즘 적용
→ 간신 손실 문제를 최소화?

관련 코드: <https://github.com/Daemawiki/Claude/blob/develop/src/main/java/com/daemawiki/internal/core/domain/service/document/DocumentSocketEditingService.java>

프로젝트 화면:



2. 사용자 인가 관리 개선

- **문제점:** 회원가입 시 이메일 중복 검사와 인증 코드 확인 작업의 순차적 처리로 인한 지연
- **해결방안:** Mono.when()을 활용하여 두 작업을 병렬로 처리

```
public Mono<Void> register(final RegisterDTO dto) {
    return validateRegistration(dto.email())
        .then(Mono.defer(() -> executeRegisterProcess(dto)));
}

private Mono<Void> validateRegistration(final Email email) {
    return Mono.when(
        ensureEmailNotRegistered(email),
        validateEmailAuthentication(email)
    );
}
```

• 성과:

- 전체 처리 시간 약 40% 단축 ⇒ 사용자 경험 개선
- Redis(인증 코드)와 MongoDB(이메일 중복)에 대한 네트워크 요청 병렬 실행
- 해당 두 작업은 서로 영향을 미치지 않기 때문에 병렬 처리 판단을 하였습니다.
- 회원가입 프로세스 Endpoint 응답 속도 P90 기준 300ms 이내 달성

관련 코드: <https://github.com/Daemawiki/Claude/blob/develop/src/main/java/com/daemawiki/internal/core/domain/service/auth/RegisterService.java>

3. 메일 인증 프로세스 최적화

- **문제점:** 메일 전송 프로세스의 긴 응답 시간
- **해결방안:** Fire-and-Forget 패턴 도입 및 이벤트 기반 아키텍처로 리팩토링
- **성과:**
 - 응답 시간을 4초에서 1초 미만으로 93% 개선
 - 서비스 간 결합도 최소화

관련 코드: <https://github.com/Daemawiki/Claude/blob/develop/src/main/java/com/daemawiki/internal/core/domain/service/mail/MailSendService.java>

성과

- 서버 구현은 100% 완료, 클라이언트 개발 및 테스트 진행 중

후기 개선 계획

현재 실시간 문서 편집의 동시성 수준이 문서 단위에서 최대 열 단위 Sync로 크게 축소했고, 추후에 CRDT, OT 알고리즘을 도입하여 동시성 수준을 더욱 개선할 예정입니다.

또한, Version Vector 도입을 통해 부분 수정을 정확하게 추적하고 싶은 계획이 존재합니다. Backpressure Control Patterns을 더욱 자세하게 공부하여 더 나은 리액티브 패턴을 적용하고 싶습니다.

회고

처음엔 Spring MVC의 스레드 풀 방식이 납득되지 않아 WebFlux를 도입했지만, 오히려 복잡한 구조로 인해 오히려 생산성이 떨어지는 문제를 겪었습니다. 이 경험을 통해 **새로운 기술은 도입 자체보다 트레이드오프**, 그리고 실질적 효과가 더 중요하다는 점을 느꼈습니다.

WebFlux의 동작 원리를 깊이 이해하고 싶어 Project Reactor, Netty까지 학습했고, **Redis/Lettuce**의 **Reactive Command** 부분의 오픈소스에도 기여했습니다. 기술적으로는 큰 성장의 계기였지만, 무조건적인 비동기 구조가 성능 개선으로 이어지지 않는다는 현실적 인사이트도 얻었습니다.

단순히 새로운 걸 해보는 것을 넘어, 어떤 기술이 서비스에 어떻게 기여할 수 있을지 고민하는 태도가 개발자로서 더 중요하다는 걸 배운 프로젝트였습니다.

이력서, 온라인으로 쉽고 간편하게.

개발 기간: 2024년 1월 ~ 2024년 09월
서비스(운영) 기간: 2024월 09월 ~ 운영 중
서비스 URL: <https://www.dsm-repo.com/>
서비스 샘플 (레주메): https://www.dsm-repo.com/resume_viewer/66dc4344fcf9157ccc2e3fd
깃허브 레포지토리: <https://github.com/DSM-Repo/Whopper>
개발 인원: 백엔드 2명, 프론트엔드 2명
기술 스택:

- Backend:** Java 21, Spring Boot, Spring MVC, Spring Security, Spring Data Mongo/Redis
- Database:** MongoDB, Redis
- Documentation:** Swagger
- Architecture:** DDD 기반 (주관적인 핵심만 적용.)

프로젝트 화면:

프로젝트 화면:
1. 홍길동 (Frontend Developer)
2. 홍길동 (Frontend Developer)
3. 프로젝트
4. 랜더링 환경 설정

레포 (운영 중)

프로젝트 개요

레포는 대덕소프트웨어마이스터고등학교 학생들을 위한 이력서 작성 및 관리 플랫폼입니다. 학생들이 작성한 이력서는 PDF로 변환되어 학교와 MOU를 체결한 기업들에 배포됩니다. 현재 130명 이상의 학생이 활발히 사용 중인 서비스입니다.

담당 역할: 백엔드 리드 개발자로서 전체 API 설계 및 구현을 주도하였으며, 아키텍처 개선과 리소스 최적화를 통해 서비스 성능과 확장성을 크게 향상시켰습니다.

주요 개선점(핵심 기여)

1. 아키텍처 전환 (Virtual Thread 기반 아키텍처 재설계)

- 기존 문제: Spring Webflux, 코틀린 코루틴 기반의 코드로 인한 가독성 저하 및 유지보수성 저하
- 개선 사항: Spring Web과 Java 21 가상 스레드(Virtual Thread) 기반으로 아키텍처 전환

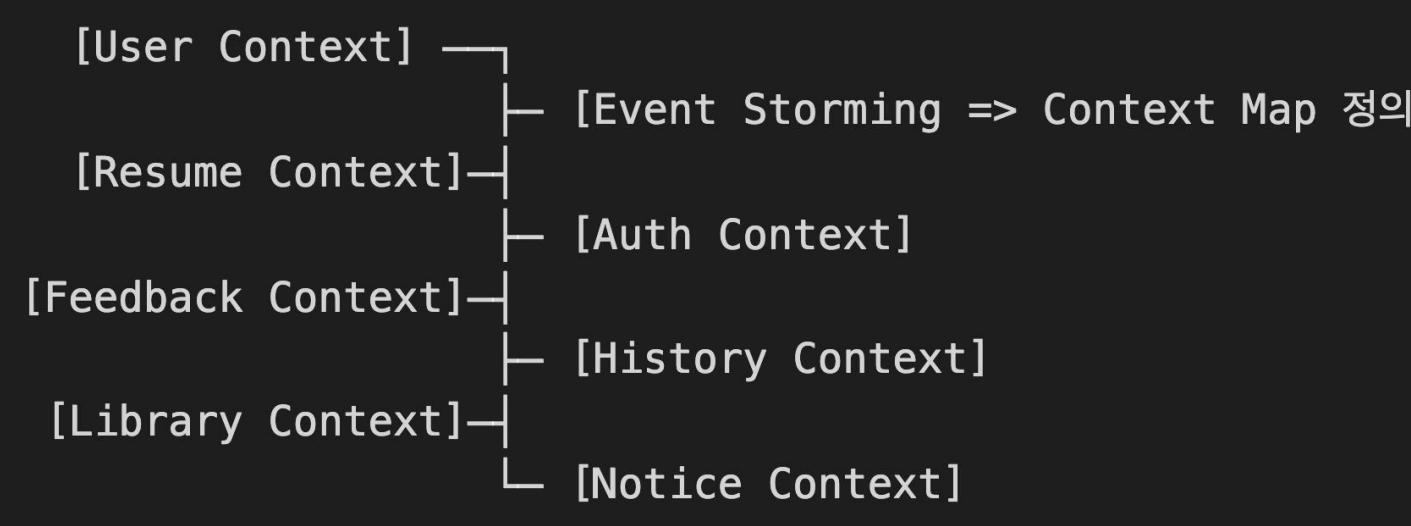
```
# Spring Boot 3.2부터 yml 설정으로 가상 스레드 활성화
spring:
  thread:
    virtual:
      enabled: true
```

성과:

- OS 레벨의 컨텍스트 스위칭 및 스레드 관리를 JVM 수준으로 개선하여 리소스 사용량 최소화
- 코드 가독성과 유지보수성 향상

2. DDD 기반 도메인 설계 개선

- 문제점: 도메인 간 강결합으로 인한 유지보수 어려움과 책임 경계 불명확
- 해결방안: DDD 아주 일부만 적용.
 - 바운디드 컨텍스트 명확화를 통한 도메인 모듈화



성과:

- 팀 전체의 유비쿼터스 랭귀지를 정의하여 팀 내 표준 용어 문서화
- 도메인 간 결합도 감소로 유연한 아키텍처 구축
- 새로운 기능 추가 시 개발 시간 30% 단축
- 팀 커뮤니케이션 효율성 증가

3. 어노테이션 기반 권한 검증 시스템 구축

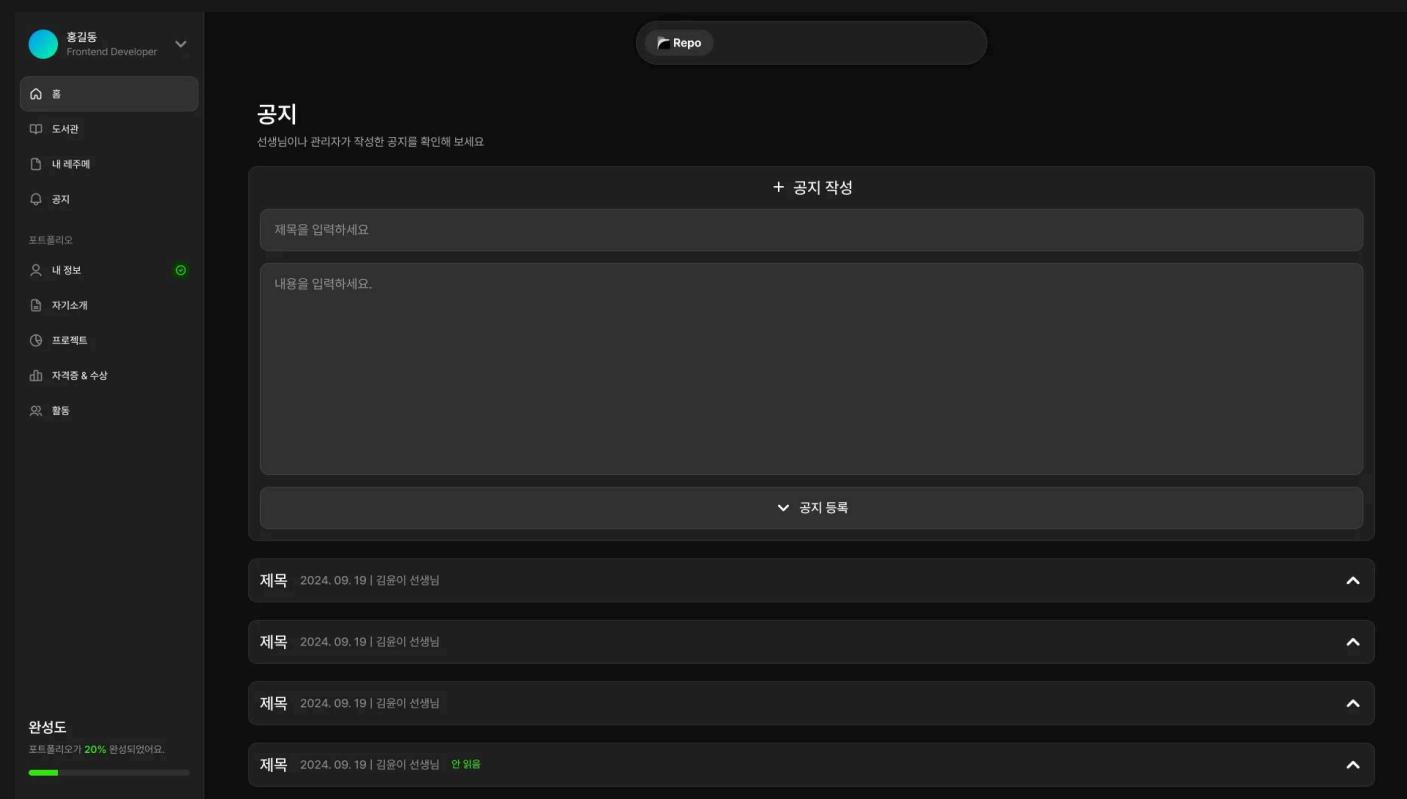
- 문제점: 중복된 권한 검증 로직으로 인한 코드 관리 어려움
- 해결방안: @OnlyRoleName 커스텀 어노테이션을 사용한 사용자 권한 검증 시스템 구축

```
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
@PreAuthorize("hasRole('STUDENT')")
public @interface OnlyStudent {
}

...
@PreAuthorize("hasRole('TEACHER')")
public @interface OnlyTeacher {
```

성과:

- 권한 검증 코드 중복 90% 제거
- 백엔드 개발자들의 도메인 로직 집중도 향상
- 일관된 보안 적용으로 시스템 안정성 향상



이슈

API 명세서 관리 문제

- **문제점:** 유동적인 API 변경으로 인한 명세서 불일치
- **해결방안:**
 - Notion 기반 문서에서 Swagger 자동화 문서로 마이그레이션
 - 코드 리뷰 과정에 문서화 검토 단계 추가
- **성과:**
 - API 스펙과 명세서 간 일치율 100% 달성
 - 프론트엔드-백엔드 협업 소통 문제 해결

성과

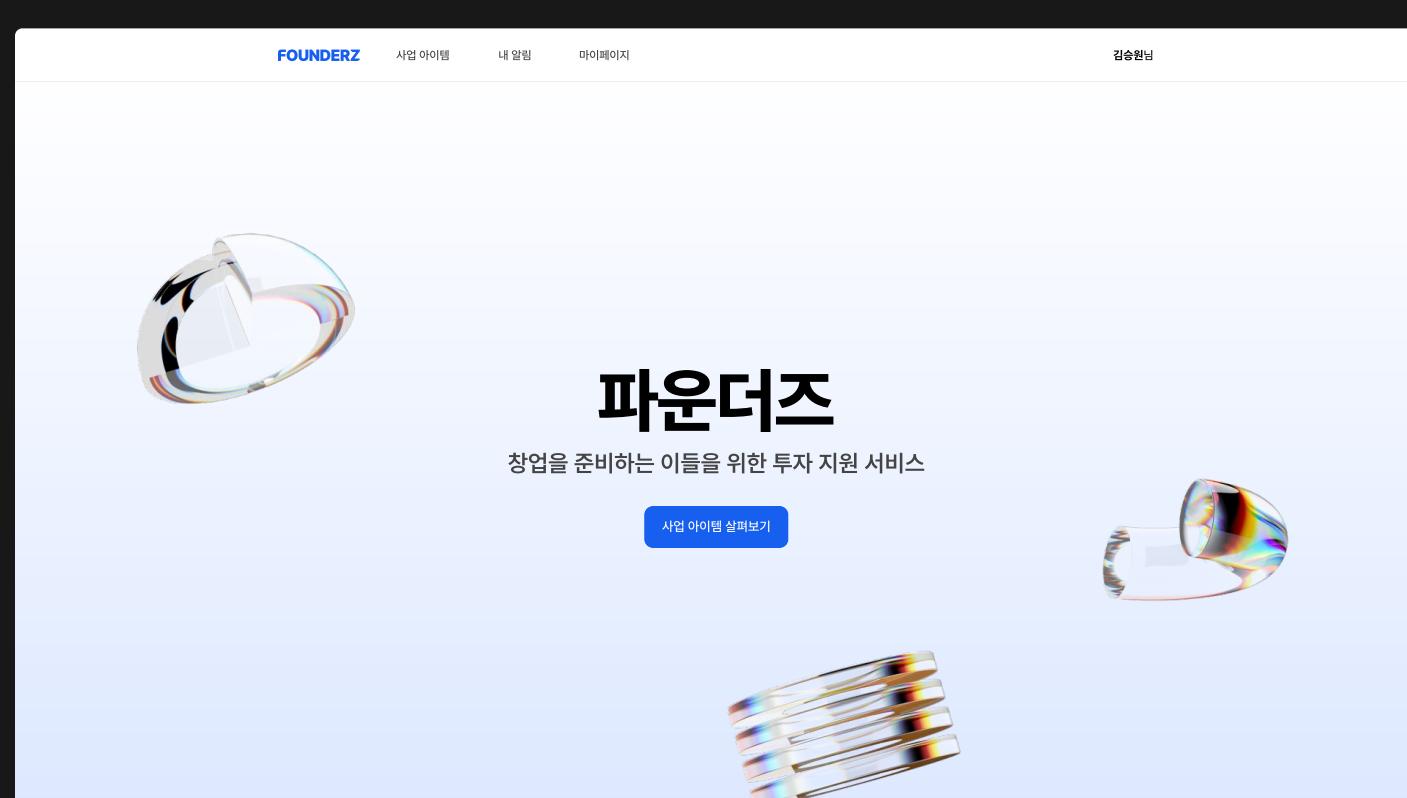
- 교내 130명 이상의 학생을 대상으로 안정적인 서비스 운영 중.
- 작성된 모든 이력서가 PDF로 변환되어 MOU 체결 기업에 성공적으로 배포
- 2024 소프트웨어 전시

회고

실제 사용자를 대상(전교생)으로 직접 운영을 하면서 '그저 기능 구현-배포은 전체 개발의 30% 밖에 안 되었구나'를 깨달았습니다.

처음에는 백엔드 리드 개발자로서 **아키텍처 개선(RP)**, **성능 최적화(Java VT)**, **도메인 설계(DDD)** 등에 집중했는데, 프로젝트가 진행될수록 단순히 많은 사람들이 "좋은 코드"라 칭하는 것을 작성하는 것만으로는 부족하다는 것을 절실히 깨달았습니다.

실제로 운영을 하며 매번 변경, 추가되는 요구사항에 대한 즉각적인 API 개발을 해야했고, 사용자가 실제로 겪는 문제를 해결하는 것이 진짜 개발의 핵심이라는 것을 배웠습니다.
레포 프로젝트를 통해 단순한 기능 개발자가 아닌, 사용자의 문제를 해결하는 개발자로 성장하는 방법을 배웠습니다.



개발 기간: 2024년 5월 ~ 2024년 11월

깃허브 레포지토리: <https://github.com/teamFOUNDERZ/FOUNDERZ-Backend>

개발 인원: 백엔드 2명, 프론트엔드 2

기술 스택:

- **Backend:** Java, Spring Boot, Spring MVC, Spring Security, Spring Data JPA(Hibernate)/Redis
- **Database:** MySQL, Redis
- **Realtime Communication:** SSE
- **Documentation:** Swagger
- **ETC.:** CoolSMS

F 파운더즈 (운영 종료)

프로젝트 개요

FOUNDERZ는 청년 창업가와 투자자를 연결하는 종합 플랫폼입니다.

창업 아이디어 게시, 투자 계약 자동화, 법적 서류 관리 등 창업 과정의 복잡성을 간소화합니다. 투명한 투자 관리와 맞춤형 추천 시스템을 통해 안전하고 효율적인 창업 생태계를 구축하며, 청년 창업가들의 성공적인 창업을 지원합니다.

창업과 투자의 모든 단계를 한 곳에서 해결할 수 있는 플랫폼입니다.

담당 역할: 유저 관리 및 알림 부분의 백엔드 개발을 맡아 안정적인 시스템을 구축했습니다. 특히, 스프링 멀티 모듈 구조를 주도하여 도입하였습니다.

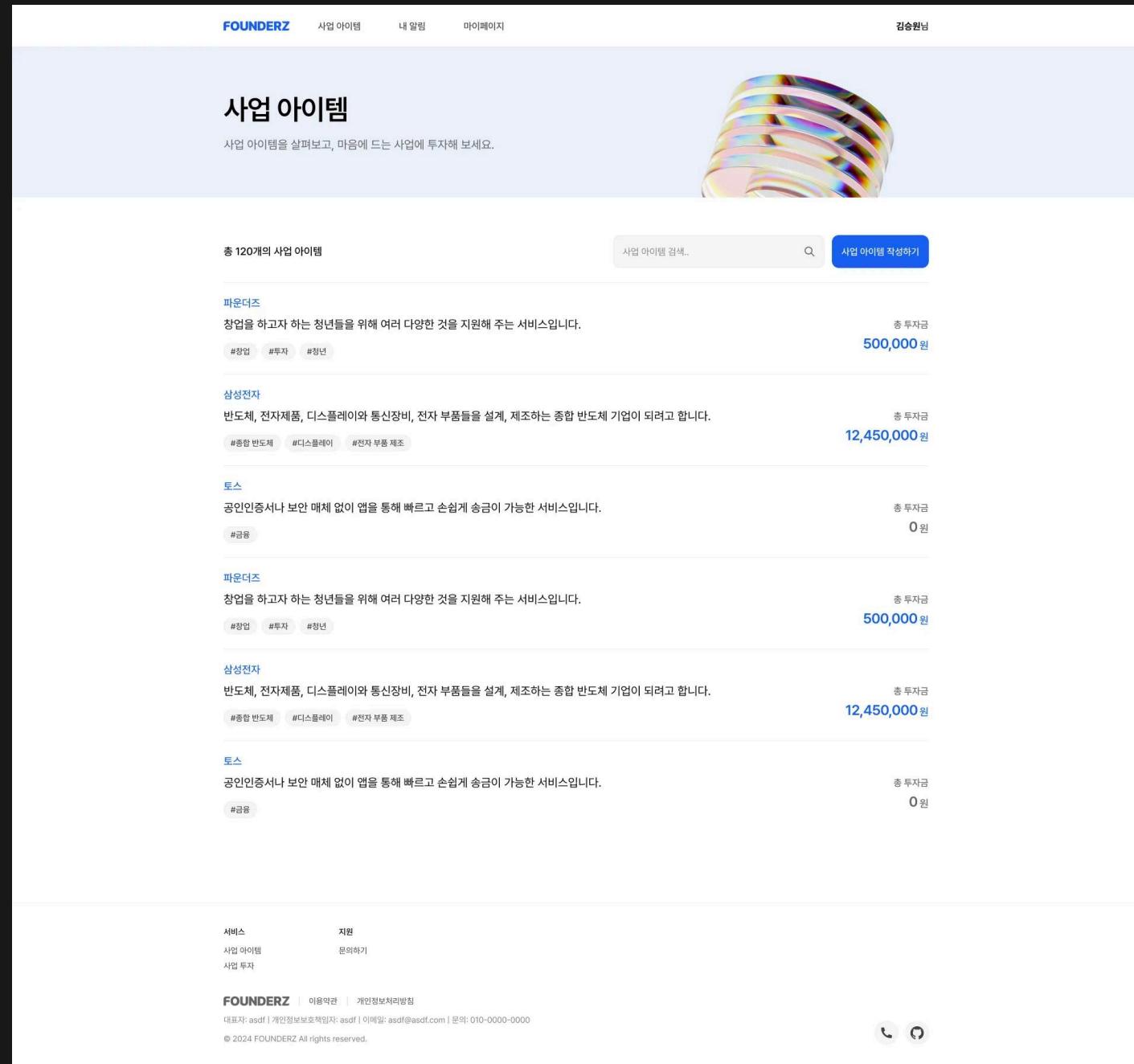
주요 개선점(핵심 기여)

1. DDD 기반 아키텍처 설계

- 목표: 도메인 간 결합도 최소화 및 유지보수성 향상
- 구현:
 - 유비쿼터스 랭귀지 정의로 팀 내 용어 표준화
 - 멀티 모듈을 활용한 바운디드 컨텍스트 분리 (계약, 투자, 알림)
 - 이벤트 기반 통신으로 컨텍스트 간 의존성 관리
- 성과:
 - 도메인 수정 시 영향 범위 50% 감소
 - VO(Value Object)와 DP(Domain Primitive) 적용으로 데이터 무결성 보장

2. 실시간 알림 시스템 (SSE)

- 목표: 외부 의존성이 없이 실시간 알림 구현
- 구현:
 - FCM 대신 HTTP 기반 SSE(Server-Sent Events) 채택
 - 알림 이벤트 트리거 설계 (투자 요청, 계약 체결, 문서 업데이트)
- 성과: 외부 의존성이 없어 시스템 복잡도 그대로 유지하면서 실시간 알림 구현



3. 프론트엔드 개발 지원

- 문제: 프론트엔드 개발 지연
- 해결: 알림 API 연동 함수 직접 개발 (React)
- 결과:
 - 알림 모듈 개발 기간 2주 → 3일 단축
 - 직접 구현한 API를 연동하여 빠른 개발 가능

성과

- Software Future & Dream Challenge 2024 본선 진출
- 2024 빛가람 에너지밸리 소프트웨어 작품 경진대회 본선 진출

회고

도메인 주도 설계를 처음 적용했을 땐, 책과 블로그에서 본 개념을 그대로 가져와 사용했습니다. 하지만 결과적으로 불필요하게 복잡한 구조를 만들었고, 팀원 간 소통에도 문제가 생겨 개발 초기 안정성이 떨어지는 아쉬움이 있었습니다.

이 경험을 통해 '기술을 도입할 때 무엇을 해결하고 싶은지, 서비스에 어떻게 기여를 하는지'를 먼저 고민해야 한다는 것을 배웠습니다.

이후에는 우리 팀의 개발 방식에 적합한 개념만 선별하고 단순화해 적용했습니다. 또한, 멀티 모듈 구조를 도입할 때는 팀원들의 이해를 돋기 위해 개발 가이드를 직접 문서화하고 공유했습니다.

이 과정을 통해 **기술의 난이도보다 팀이 함께 이해하고 유지할 수 있는 구조가 더 중요하다는 사실을 체감** 했고, 기술 도입의 관점을 실무적으로 바꿀 수 있었습니다.

Technical Skills

Java

Kotlin

Spring Boot

Spring MVC

Spring WebFlux

Spring Data JPA

Spring Data Mongo

MongoDB

MySQL

Redis

Contests & Awards

2024 SW 마이스터고등학교 연합 해커톤 대상(과학기술정보통신부 장관상)

- (주)서북에서 제시한 과제인 AI 기반 포토이즘 서비스의 백엔드 개발을 담당하여 **대상을 수상했습니다**.

대한민국 소프트웨어대전 소프트웨이브 2024 프로젝트 전시

Software Future & Dream Challenge 2024 본선 진출

2024 빛가람 에너지밸리 소프트웨어 작품 경진대회 본선 진출

Study & Mentoring

교내 스프링 웹 플러스 스터디 진행 (팀장 참여)

교내 신입생 전공 분야 멘토링 진행 (멘토 참여), 약 23시간

블록체인 점프업 캠프 수강

Contributes

Redis 자바 클라이언트 `Lettuce` 오픈소스 기여 #3061, #3262

- `mget` 메서드를 함수형 접근법을 사용하여 파티션 처리 단순화하였고, 수동으로 퍼블리셔 리스트를 생성하던 부분을 Stream API로 대체하였습니다.
- 또한, Project Reactor의 `flatMapMany(Flux::fromIterable)`를 사용하여 반응형 체인 개선을 하여 코드 가독성과 유지보수성 개선에 기여했습니다.
- 내부적으로 불필요하게 Collection을 생성하는 메서드를 생성하지 않도록 수정하여 성능 개선을 했습니다.

워크플로 자동화 플랫폼 `Kestra` 오픈소스 기여 16.5k Star #6073

- 일관성 없는 `instanceof` 패턴을 패턴 매칭으로 개선하여 코드 가독성 향상과 불필요한 캐스트 제거를 진행하여 코드 스타일 일관성 유지에 기여했습니다.

개발자 블로그 `Velog` 프론트엔드 오픈소스 기여 #49

- 화이트 모드에서 텍스트 요소가 보이지 않는 문제를 해결하기 위해 스타일 설정을 제안하고, Pull Request를 통해 문제 해결에 기여했습니다.