> **프로젝트명**: MZC 1st Project - 학습관리시스템
>
> **아키텍처**: 서버 분리 (Core API Server + Video Streaming Server)
>
> **기술 스택**: Spring Boot 3.5.7, Java 21, MySQL, Redis
>
> **팀**: MZC Team 2

목차

---

1. 프로젝트 개요

1.1 프로젝트 목적

대학교 학습관리시스템(LMS)으로, 학생과 교수를 위한 **수강신청**, **강의 관리**, **게시판**, **실시간 알림** 등의 기능을 제공합니다.

1.2 주요 사용자

사용자	역할	주요 기능
학생 (Student)	수강생	수강신청, 강의 조회, 게시판 이용, 과제 제출
교수 (Professor)	강의자	강의 개설/관리, 주차별 자료 등록, 과제 출제/채점

1.3 프로젝트 규모

항목	수치
Java 소스 파일	291개
테스트 클래스	42개
도메인 모듈	8개
API 엔드포인트	50+

---

2. 시스템 아키텍처

2.1 시스템 구성

시스템 아키텍처 문서 확인

2.2 서버 역할 분리

서버	포트	역할
Core API Server	8080	학습관리, 인증, 수강신청, 게시판 등 핵심 비즈니스 로직
Video Streaming Server	8090	영상 업로드/스트리밍, 학습 진도 추적, 부정행위 탐지

2.3 적용 패턴

- **Layered Architecture**: Controller - Service - Repository 계층 분리
- **Event-Driven**: Spring ApplicationEventPublisher 활용
- **DTO Pattern**: 계층 간 데이터 전달 시 DTO 사용

---

3. 핵심 기능

3.1 기능 요약

기능	설명	핵심 기술
회원 관리	회원가입, 로그인, 프로필 관리	JWT, BCrypt, AES 암호화
수강신청	강의 검색, 장바구니, 일괄 신청	비관적 락, 이벤트 발행

기능	설명	핵심 기술
강의 관리	강의 개설, 주차별 자료 관리	CRUD, 파일 업로드
게시판	게시글/댓글 CRUD, 해시태그, 좋아요	Soft Delete, 역할 기반 접근 제어
과제 관리	과제 출제, 제출, 채점	파일 업로드, 마감일 관리
알림	알림 목록 조회, 읽음 처리	커서 페이지네이션
메시지	1:1 메시지, 실시간 수신	SSE (Server-Sent Events)

3.2 사용자별 기능 매트릭스

기능	학생	교수
회원가입/로그인	O	O
프로필 관리	O	O
강의 검색/조회	O	O
수강신청	O	-
강의 개설/수정	-	O
주차별 자료 등록	-	O
게시판 글쓰기	O (학생 게시판)	O (교수 게시판)
과제 제출	O	-
과제 출제/채점	-	O
성적 관리	-	O
알림 수신	O	O
메시지 송수신	O	O

---

4. 도메인별 상세 기능

4.1 User (회원 관리)

4.1.1 회원가입

- 학생/교수 구분 가입
- 이메일 인증 (6자리 코드, Redis 저장)
- 비밀번호 정책 검증 (최소 8자, 영문+숫자+특수문자)
- 학과/단과대학 선택

4.1.2 로그인/인증

- JWT 기반 인증 (Access Token + Refresh Token)
- Access Token 만료 시간: 30분 (설정 가능)
- Refresh Token 만료 시간: 7일
- 토큰 갱신 API

4.1.3 프로필 관리

- 프로필 이미지 업로드 (WebP 자동 변환)
- 연락처 정보 수정

4.1.4 사용자 검색

- 이름, 학번, 이메일로 검색
- 학과/단과대학 필터링

4.2 Course (강의 관리)

4.2.1 강의 검색 (학생/교수)

- 학기별 강의 목록 조회
- 필터링: 학과, 이수구분, 학점, 키워드
- 정렬: 과목코드, 과목명, 학점
- 페이지네이션

4.2.2 강의 상세

- 기본 정보 (과목명, 교수, 학점, 정원)
- 공지사항
- 시간표 정보 (요일, 시간, 강의실)
- 주차별 강의 계획
- 선수과목 정보

4.2.3 강의 개설 (교수)

- 과목 선택 및 분반 지정
- **분반 개설시에도 비관적 락으로 동시성제어**
- 시간표 설정 (충돌 검사)
- 정원 설정

4.2.4 주차별 자료 관리 (교수)

- 주차 등록/수정/삭제
- 주차별 콘텐츠 등록/수정/삭제

4.3 Enrollment (수강신청)

4.3.1 수강신청 기간 조회

- 현재 활성화된 수강신청 기간 조회

4.3.2 장바구니

- 관심 강의 담기 (일괄)
- 장바구니 조회
- 장바구니 삭제 (일괄)

4.3.3 수강신청

- 일괄 신청 지원
- 정원 초과 검사
- 선수과목 이수 여부 검사
- 시간표 충돌 검사
- 학점 제한 검사 (21학점)
- **비관적 락으로 동시성 제어**
- 수강신청 완료 이벤트 발행

4.3.4 수강 취소

- 일괄 취소 지원
- 취소 시 정원 자동 복구
- 취소 이벤트 발행

4.3.5 내 수강 목록 조회

- 학기별 수강 목록
- 총 학점, 남은 학점 표시

4.4 Board (게시판)

4.4.1 게시판 유형

유형	접근 권한	설명
STUDENT	학생만	학생 전용 자유게시판
PROFESSOR	교수만	교수 전용 게시판

4.4.2 게시글 기능

- 게시글 CRUD

- 익명 게시 지원 (게시판 정책에 따라)
- 해시태그 지원
- 첨부파일 업로드 (다중 파일)
- 조회수 카운트
- 좋아요 토글

4.4.3 댓글 기능

- 댓글 CRUD
- 대댓글 지원 (계층형)

4.4.4 해시태그

- 자동 생성 및 연결
- 해시태그별 게시글 검색

4.5 Assignment (과제 관리)

4.5.1 과제 출제 (교수)

- 과제 제목/내용 작성
- 마감일 설정
- 배점 설정

4.5.2 과제 목록/상세 조회

- 강의별 과제 목록
- 과제 상세 조회

4.5.3 과제 제출 (학생)

- 텍스트 답변 작성
- 첨부파일 ID 연결

4.5.4 제출 목록 조회 (교수)

- 과제별 제출 목록

4.5.5 채점 (교수)

- 점수 부여
- 피드백 작성

4.5.6 내 제출 조회 (학생)

- 특정 과제에 대한 내 제출 조회

4.6 Notification (알림)

4.6.1 알림 목록 조회

- 커서 기반 페이지네이션
- 읽지 않은 알림만 필터링

4.6.2 알림 관리

- 알림 상세 조회
- 읽지 않은 알림 개수 조회
- 단일 알림 읽음 처리
- 모든 알림 읽음 처리
- 단일 알림 삭제
- 읽은 알림 일괄 삭제
- 모든 알림 삭제

4.7 Message (메시지)

4.7.1 대화방

- 대화방 목록 조회 (최근 메시지순)
- 대화방 생성 (상대방 ID로)
- 대화방 상세 조회
- 대화방 삭제 (Soft Delete)
- 읽음 처리
- 전체 안읽음 수 조회

4.7.2 메시지

- 메시지 전송 (단일/일괄)
- 대화 내역 조회 (커서 페이지네이션)
- 메시지 삭제
- 읽음 처리

4.7.3 실시간 수신

- SSE 기반 실시간 메시지 수신 (/api/v1/sse/subscribe)

---

5. 영상 강의 시스템 (Video Streaming Server)

> **별도 서버**: Video Streaming Server (Port: 8090) **역할**: 강의 영상 업로드, 스트리밍, 학습 진도 추적, 부정행위 탐지

5.1 영상 업로드

5.1.1 TUS Protocol 기반 청크 업로드

- **재개 가능한 업로드**: 네트워크 오류 시 이어서 업로드 가능
- 최대 파일 크기: 10GB
- 청크 크기: 5MB
- 지원 포맷: mp4, webm, avi, mov, mkv
- 메타데이터 자동 추출 (courseId, weekId, title, duration)
- UUID 기반 파일 저장
- 만료된 업로드 자동 정리 (24시간 TTL)

5.1.2 업로드 상태 추적

상태	설명
IN_PROGRESS	업로드 진행 중
COMPLETED	업로드 완료
CANCELLED	업로드 취소
FAILED	업로드 실패
EXPIRED	업로드 만료

5.2 영상 스트리밍

- **HTTP Range Request** 지원 (HTTP 206 Partial Content)
- 바이트 범위 요청으로 효율적인 시킹(seeking)
- Content-Range 헤더 지원
- 8KB 버퍼링

5.3 시청 세션 관리

5.3.1 Redis 기반 세션

- 세션 타임아웃: 30초 (비활성 시 자동 만료)
- Snowflake ID 생성
- 사용자당 단일 활성 세션 (새 세션 시 기존 세션 교체)

5.3.2 세션 상태

상태	설명
ACTIVE	활성 시청 중
TERMINATED	정상 종료
REPLACED	다른 세션으로 교체됨
TIMEOUT	타임아웃 만료

5.4 학습 진도 추적

- 5초 간격으로 진도 보고 (클라이언트 → 서버)
- 진도율 자동 계산: (시청 시간 / 전체 시간) x 100%
- **완료 기준**: 90% 이상 시청 시 완료 처리
- 데이터베이스 영구 저장 (student_content_progress 테이블)
- Core API Server와 진도 데이터 공유

5.5 부정행위 탐지

학습 무결성 보장을 위해 부정 시청 행위를 탐지하고 기록합니다.

5.5.1 탐지 유형

유형	설명	기준
FAST_PLAYBACK	배속 시청	재생 속도 > 1.0x
TAB_HIDDEN	탭 전환/백그라운드	브라우저 탭 숨김
SKIP_SEGMENT	구간 건너뛰기	5초 이상 앞으로 점프
CONCURRENT_SESSION	동시 세션	다중 세션 시도

5.5.2 학습률 계산

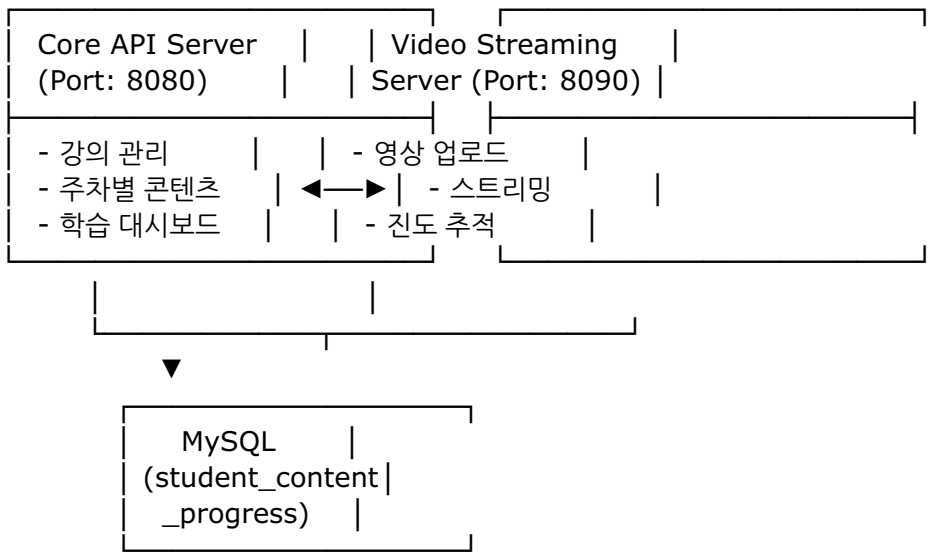
유효 시청 시간 = 전체 시청 시간 - 부정행위 구간
학습률 = (유효 시청 시간 / 전체 영상 길이) x 100%

- 학습률은 감소하지 않음 (최대값 유지)
- 최대 100%로 제한

5.5.3 시청 이벤트 기록

이벤트	설명
PLAY	재생 시작
PAUSE	일시 정지
SEEK	구간 이동
RATE_CHANGE	재생 속도 변경
VISIBILITY_HIDDEN	탭 숨김
VISIBILITY_VISIBLE	탭 복귀

5.6 데이터 연동



- **공유 테이블**: student_content_progress
- Video Server가 진도 기록 → Core API Server에서 조회
- courseId, weekId, contentId 등 ID 공유

---

6. 기술 스택

6.1 Backend

분류	기술	버전
Framework	Spring Boot	3.5.7
Language	Java	21
ORM	Spring Data JPA	-
Migration	Flyway	-

분류	기술	버전
Security	Spring Security	-
Validation	Spring Validation	-

6.2 Database & Cache

분류	기술	용도
RDBMS	MySQL 8.0	메인 데이터베이스
Cache	Redis	이메일 인증 코드 저장, 사용자 정보 캐싱

6.3 인증 & 보안

분류	기술	용도
Token	JWT (jjwt 0.11.5)	Access/Refresh Token
Password	BCrypt	비밀번호 해싱
Encryption	AES-256	민감정보 암호화

6.4 기타 라이브러리

분류	기술	용도
Image Processing	Scrimage 4.1.3	WebP 이미지 변환
ID Generator	ULID Creator 5.2.3	정렬 가능한 고유 ID
API Docs	SpringDoc OpenAPI 2.7.0	Swagger UI
Scheduler Lock	ShedLock 6.0.2	분산 환경 스케줄러

6.5 코드 품질

도구	버전	용도
CheckStyle	10.21.1	코드 스타일 검사
PMD	7.0.0	정적 코드 분석
SpotBugs	6.0.26	버그 패턴 탐지

6.6 Video Streaming Server 전용

분류	기술	용도
Upload		

분류	기술	용도
	TUS Protocol (tus-java-server 1.0.0-3.0)	청크 업로드, 재개 가능
Session	Redis	시청 세션 관리 (30초 TTL)
ID Generator	Snowflake	세션 ID 생성

7. API 엔드포인트

7.1 인증 API (/api/auth)

Method	Endpoint	설명
POST	/signup/email-verification	이메일 인증 코드 발송
POST	/signup/verify-code	인증 코드 확인
POST	/signup	회원가입
POST	/login	로그인
POST	/refresh	토큰 갱신
POST	/logout	로그아웃
GET	/check-email	이메일 중복 확인

7.2 프로필 API (/api/v1/profile)

Method	Endpoint	설명
GET	/me	내 프로필 조회
PATCH	/me	프로필 수정
POST	/me/image	프로필 이미지 업로드
DELETE	/me/image	프로필 이미지 삭제

7.3 사용자 검색 API (/api/v1/users)

Method	Endpoint	설명
GET	/search	사용자 검색
GET	/colleges	단과대학 목록
GET	/colleges/{id}/departments	단과대학별 학과 목록
GET	/departments	전체 학과 목록

7.4 강의 API (/api/v1/courses)

Method	Endpoint	설명
GET	/	강의 목록 조회
GET	/{courseId}	강의 상세 조회

7.5 교수 강의 관리 API (/api/v1/professor/courses)

Method	Endpoint	설명
POST	/	강의 개설
PUT	/{courseId}	강의 수정
DELETE	/{courseId}	강의 삭제
GET	/	내 강의 목록
GET	/{courseId}	내 강의 상세

7.6 주차별 콘텐츠 API (/api/v1/professor/courses/{courseId}/weeks)

Method	Endpoint	설명
GET	/	주차 목록 조회
POST	/	주차 등록
PUT	/{weekId}	주차 수정
DELETE	/{weekId}	주차 삭제
GET	/{weekId}/contents	콘텐츠 목록
POST	/{weekId}/contents	콘텐츠 등록
PUT	/{weekId}/contents/{contentId}	콘텐츠 수정
DELETE	/{weekId}/contents/{contentId}	콘텐츠 삭제
PUT	/{weekId}/contents/reorder	콘텐츠 순서 변경

7.7 과목 API (/api/v1/subjects)

Method	Endpoint	설명
GET	/	과목 목록 조회
GET	/{subjectId}	과목 상세 조회
GET	/search	과목 검색

7.8 수강신청 API (/api/v1/enrollments)

Method	Endpoint	설명
GET	/periods/current	현재 수강신청 기간 조회
GET	/courses	수강신청 가능 강의 조회
POST	/bulk	수강신청 (일괄)
DELETE	/bulk	수강 취소 (일괄)
GET	/my	내 수강 목록

7.9 장바구니 API (/api/v1/carts)

Method	Endpoint	설명
GET	/	장바구니 조회
POST	/bulk	장바구니 추가 (일괄)
DELETE	/bulk	장바구니 삭제 (일괄)
DELETE	/	장바구니 전체 삭제

7.10 게시판 API (/api/v1/board)

Method	Endpoint	설명
POST	/{boardType}/posts	게시글 작성
GET	/{boardType}/posts	게시글 목록
GET	/{boardType}/posts/{id}	게시글 상세
PUT	/posts/{id}	게시글 수정
DELETE	/posts/{id}	게시글 삭제
POST	/posts/{id}/like	좋아요 토글
GET	/posts/{id}/liked	좋아요 여부 조회

7.11 댓글 API (/api/v1/board)

Method	Endpoint	설명
POST	/comments	댓글 작성
GET	/comments	댓글 목록 (postId 필터)
PUT	/comments/{id}	댓글 수정
DELETE	/comments/{id}	댓글 삭제

7.12 첨부파일 API (/api/v1/attachments)

Method	Endpoint	설명
POST	/upload	단일 파일 업로드
POST	/upload/multiple	다중 파일 업로드
GET	/{attachmentId}/download	파일 다운로드
GET	/{attachmentId}	첨부파일 정보 조회
DELETE	/{attachmentId}	첨부파일 삭제

7.13 과제 API (/api/v1/assignments)

Method	Endpoint	설명
POST	/	과제 등록 (교수)
GET	/	강의별 과제 목록
GET	/{id}	과제 상세
PUT	/{id}	과제 수정 (교수)
DELETE	/{id}	과제 삭제 (교수)
POST	/{id}/submit	과제 제출 (학생)
GET	/{id}/submissions	제출 목록 (교수)
PUT	/submissions/{submissionId}/grade	채점 (교수)
GET	/{id}/my-submission	내 제출 조회 (학생)

7.14 알림 API (/api/notifications)

Method	Endpoint	설명
GET	/	알림 목록 (커서 기반)
GET	/{notificationId}	알림 상세
GET	/unread-count	안읽은 알림 개수
PATCH	/{notificationId}/read	읽음 처리
PATCH	/read-all	모든 알림 읽음 처리
DELETE	/{notificationId}	알림 삭제
DELETE	/read	읽은 알림 삭제
DELETE	/all	모든 알림 삭제

7.15 대화방 API (/api/v1/conversations)

Method	Endpoint	설명
GET	/	대화방 목록
POST	/with/{otherUserId}	대화방 생성/조회
GET	/{conversationId}	대화방 상세
DELETE	/{conversationId}	대화방 삭제
POST	/{conversationId}/read	읽음 처리
GET	/unread-count	전체 안읽음 수

7.16 메시지 API (/api/v1/messages)

Method	Endpoint	설명
POST	/	메시지 전송
POST	/bulk	메시지 일괄 전송
GET	/conversations/{conversationId}	대화 내역 조회
DELETE	/{messageId}	메시지 삭제
POST	/conversations/{conversationId}/read	읽음 처리

7.17 SSE API (/api/v1/sse)

Method	Endpoint	설명
GET	/subscribe	SSE 연결 (실시간 알림/메시지)

7.18 Video Streaming API (Port: 8090)

> **Video Streaming Server** 전용 API

영상 업로드 (TUS Protocol)

Method	Endpoint	설명
OPTIONS	/api/v1/videos/upload	TUS 프로토콜 지원 확인
POST	/api/v1/videos/upload	업로드 생성 (Location 헤더 반환)
HEAD	/api/v1/videos/upload/**	업로드 상태 확인
PATCH		청크 업로드

Method	Endpoint	설명
	/api/v1/videos/upload/**	
DELETE	/api/v1/videos/upload/**	업로드 취소

영상 스트리밍

Method	Endpoint	설명
GET	/api/v1/videos/stream/{videoId}	영상 스트리밍 (Range 헤더 지원)

시청 세션

Method	Endpoint	설명
POST	/api/v1/sessions	시청 세션 시작
DELETE	/api/v1/sessions/{sessionId}	시청 세션 종료
GET	/api/v1/sessions/{sessionId}	세션 정보 조회
GET	/api/v1/sessions/active	활성 세션 조회 (userId)

진도 보고

Method	Endpoint	설명
POST	/api/v1/progress	진도 보고 (5초 간격)
GET	/api/v1/progress/{contentId}	진도 조회 (studentId)

시청 이벤트

Method	Endpoint	설명
POST	/api/v1/watch-events	이벤트 기록
GET	/api/v1/watch-events/session/{sessionId}	세션별 이벤트 조회
GET	/api/v1/watch-events/session/{sessionId}/type	이벤트 유형별 조회

8. 기술적 특징

8.1 동시성 제어

수강신청 시 정원 초과를 방지하기 위해 **두 가지 락 전략**을 지원합니다.

8.1.1 JPA 비관적 락 (현재 적용)

단일 DB 환경에서 사용하는 방식입니다.

```
// CourseRepository
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("SELECT c FROM Course c WHERE c.id = :id")
Optional<Course> findByIdWithLock(@Param("id") Long id);

// EnrollmentServiceImpl
Course course = courseRepository.findByIdWithLock(courseId)
    .orElseThrow(() -> new IllegalArgumentException("강의를 찾을 수 없습니
            다"));

if (course.getCurrentStudents() >= course.getMaxStudents()) {
    // 정원 마감 처리
}

// 정원 증가 (락이 걸린 상태에서 안전하게)
course.setCurrentStudents(course.getCurrentStudents() + 1);
```

8.1.2 Redisson 분산 락 (스케일 아웃 대응)

다중 서버 환경에서 사용할 수 있는 Redis 기반 분산 락입니다.

```
// DistributedLockService 인터페이스
public interface DistributedLockService {
    void executeWithLock(String lockKey, Runnable task);
    <T> T executeWithLock(String lockKey, Supplier<T> supplier);
    <T> T executeWithLock(String lockKey, long waitTime, long
            leaseTime,
                    TimeUnit timeUnit, Supplier<T> supplier);
    boolean tryExecuteWithLock(String lockKey, Runnable task);
}

// 사용 예시: 수강신청
distributedLockService.executeWithLock(
    "enrollment:course:" + courseId,
    () -> {
        Course course = courseRepository.findById(courseId)
            .orElseThrow(() -> new IllegalArgumentException("강의를 찾을 수
              없습니다"));

        if (course.getCurrentStudents() >= course.getMaxStudents()) {
            throw new IllegalStateException("정원이 마감되었습니다");
```

```
        }

        // 수강신청 처리
        Enrollment enrollment = Enrollment.builder()
            .student(student)
            .course(course)
            .build();
        enrollmentRepository.save(enrollment);

        course.setCurrentStudents(course.getCurrentStudents() + 1);
    }
);

// 반환값이 있는 경우
EnrollmentResult result = distributedLockService.executeWithLock(
    "enrollment:course:" + courseId,
    () -> processEnrollment(studentId, courseId)
);

// 커스텀 타임아웃 설정
distributedLockService.executeWithLock(
    "enrollment:course:" + courseId,
    10,  // waitTime: 락 대기 시간 (초)
    30,  // leaseTime: 락 유지 시간 (초)
    TimeUnit.SECONDS,
    () -> processEnrollment(studentId, courseId)
);
```

특징:

- Redis 기반 분산 환경 지원
- 데드락 방지 (Lock Timeout)
- 재시도 로직 내장
- Redisson 3.40.2 사용

8.2 AOP 기반 자동 알림 시스템

비즈니스 로직과 알림 로직을 분리하기 위해 **@NotifyEvent 어노테이션**을 사용합니다.

```
// 어노테이션 정의
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
public @interface NotifyEvent {
    NotificationEventType type();
    String titleExpression() default "";
    String messageExpression() default "";
    String recipientIdExpression() default "";
    String recipientIdsExpression() default "";
    String conditionExpression() default "";
    boolean async() default true;
```

```
}

// 사용 예시 1: 과제 등록 알림
@NotifyEvent(
    type = NotificationEventType.ASSIGNMENT_CREATED,
    titleExpression = "'새 과제가 등록되었습니다'",
    messageExpression = "#result.title + ' 과제가 등록되었습니다. 마감일: ' +
            #result.dueDate",
    recipientIdsExpression = "#courseStudentIds",
    courseIdExpression = "#courseId"
)
public AssignmentResponseDto createAssignment(Long courseId,
            AssignmentCreateRequestDto request) {
    // 비즈니스 로직만 작성 - 알림은 AOP에서 자동 처리
    return assignmentService.create(request);
}

// 사용 예시 2: 수강신청 완료 알림
@NotifyEvent(
    type = NotificationEventType.ENROLLMENT_SUCCESS,
    titleExpression = "'수강신청 완료'",
    messageExpression = "'[' + #result.courseName + '] 수강신청이 완료되
            었습니다.'",
    recipientIdExpression = "#studentId"
)
public EnrollmentResult enroll(Long studentId, Long courseId) {
    return enrollmentService.enroll(studentId, courseId);
}

// 사용 예시 3: 조건부 알림 (점수가 80점 이상일 때만)
@NotifyEvent(
    type = NotificationEventType.ASSIGNMENT_GRADED,
    titleExpression = "'과제 채점 완료'",
    messageExpression = "'점수: ' + #result.score + '점'",
    recipientIdExpression = "#result.studentId",
    conditionExpression = "#result.score >= 80"
)
public GradeResult gradeAssignment(Long submissionId, int score) {
    return gradeService.grade(submissionId, score);
}
```

SpEL 표현식 변수:

- `#result`: 메소드 반환값
- `#arg0`, `#arg1`, ...: 메소드 인자
- `#args`: 전체 인자 배열

8.3 이벤트 기반 아키텍처

수강신청/취소 시 **Spring ApplicationEventPublisher**를 활용하여 이벤트를 발행합니다.

```
// 이벤트 발행
eventPublisher.publishEvent(new EnrollmentCreatedEvent(
    studentId,
    course.getId(),
    course.getSubject().getSubjectName(),
    course.getSectionNumber()
));

// 이벤트 리스너에서 처리
@EventListener
public void onEnrollmentCreated(EnrollmentCreatedEvent event) {
    // 알림 발송 등 후속 처리
}
```

8.4 Soft Delete 구현

데이터 복구 가능성과 참조 무결성을 위해 **Soft Delete** 패턴을 적용했습니다.

```
@Entity
@Where(clause = "deleted_at IS NULL")
public class User {
    @Column(name = "deleted_at")
    private LocalDateTime deletedAt;
}
```

8.5 커서 기반 페이지네이션

대용량 데이터 조회 성능을 위해 알림, 메시지 목록에 **커서 기반 페이지네이션**을 적용했습니다.

```
// NotificationController
@GetMapping
public ResponseEntity<?> getNotifications(
    @RequestParam(required = false) Long cursor,
    @RequestParam(defaultValue = "20") int size,
    @RequestParam(defaultValue = "false") boolean unreadOnly
) {
    // cursor 이후 데이터만 조회
}
```

8.6 SSE 기반 실시간 통신

메시지, 알림의 실시간 수신을 위해 **Server-Sent Events**를 사용합니다.

```
@GetMapping(value = "/subscribe", produces =
            MediaType.TEXT_EVENT_STREAM_VALUE)
public SseEmitter subscribe(@AuthenticationPrincipal Long userId) {
    return sseService.subscribe(userId);
}
```

8.7 코드 품질 자동화

빌드 시 자동으로 코드 품질 검사를 수행합니다.

```
./gradlew build
# checkstyleMain, checkstyleTest
# pmdMain, pmdTest
# spotbugsMain, spotbugsTest
```

> **문서 작성일**: 2025/12/22 **버전**: 1.0
>
> **작성자** : 송명주(2025-12-22)