BookDam

책을 읽고, 이야기를 나누는 공간

서비스 소개

독서 커뮤니티 플랫폼
독서 취향을 기반으로, 책에 대한 감상을 나누고 읽기 모임 속에서 관계를 만들어가는 커뮤니티 플랫폼

개요

개발기간 : 2025.12.15 ~ 2026.02.06 (진행중)
개발 인원 : 프론트엔드 1명, 백엔드 2명, 기획 1명, 디자인 1명
배포 : Docker 기반 배포 환경 구성 (Nginx 리버스 프록시 포함)

프로젝트 URL

GITHUB-DBOOK (FE) : https://github.com/orgs/dbook-kr/repository
GITHUB-DBOOK (BE) : https://github.com/orgs/dbook-kr/repository

담당 내용

Back: 아키텍처설계, 백엔드 Front: 퍼블리셔, 디자인

기술 스택 및 선택 이유

프론트엔드

- React — 컴포넌트 기반 UI 구성에 적합하고 생태계가 가장 넓어 레퍼런스 확보 용이
- TypeScript — 정적 타입 체크로 런타임 오류 사전 방지, 유지보수 시 타입 안전성 확보
- tailwindcss — 매뉴얼 없이 빠른 스타일 구현
- react-query (@tanstack/react-query) — 서버 상태 캐싱 및 자동 refetch 처리로 데이터 동기화 단순화
- Recoil — 클라이언트 측 가벼운 상태 관리

백엔드

- Express — 가장 레퍼런스가 많고 빠른 프로토타입 가능
- MySQL — 관계형 데이터 모델이 서비스 도메인에 적합
- Prisma — 타입 안전 ORM, 마이그레이션 관리 간소화

공통

- Figma — 디자인 시스템 공유 및 핸드오프
- Slack — 팀 커뮤니케이션
- Discord — 팀 커뮤니케이션
- GitHub — 코드 리뷰 및 이슈 트래킹
- Notion — 기획 문서 및 회의록 관리

해결하고자 한 문제

기존 독서 플랫폼의 문제점

리뷰만 작성하고 끝나는 독서
대부분 독서 플랫폼은 리뷰 작성까지만 지원하고 그 이후 사용자가 서로 만나는 구조가 없다. 리뷰는 쌓이지만 관계는 쌓이지 않는 구조가 문제라고 판단했다.

개인적인 독서 활동 경험에 커뮤니티가 전혀 부족
독서는 혼자 하는 활동이지만, 같은 책을 읽은 사람들과 이야기하고 싶은 욕구는 분명하다. 기존 서비스는 이 욕구를 충족시키지 못하고 있다.

BookDam의 해결 방안

1. 책장을 중심 메뉴로 — 읽고 있는 책, 읽은 책부터 시작하는 구조
2. 리뷰 작성 후 바로 커뮤니티로 연결 — 리뷰 작성 직후 해당 책의 모임 목록으로 이동
3. 독서 취향 기반 추천 — 읽은 책을 기반으로 비슷한 취향의 사용자를 찾아드림

개발 과정 및 의사결정

기획 → 설계 → 개발 → 테스트

기획
- 아이디어 구체화
- 문제 정의
- 핵심 기능 선정
- 와이어프레임 작성

설계
- 시스템 아키텍처 설계
- 데이터 모델 설계
- 기술 스택 결정

개발
- 환경 구성
- 공통 모듈 개발
- 기능 단위 개발
- 코드 리뷰

테스트
- 단위 테스트
- 통합 테스트
- 버그 수정
- 최종 점검

화면 설계 와이어프레임

사용자 진입 화면 와이어프레임

도서 상세 화면 와이어프레임

개인 독서 관리

주요 API 정의

구분	주요 Endpoint 예시
도서 검색	/books/search
개인 보관	/books/{isbn}/comments
리뷰 작성	/{isbn}/comments/{id}
리뷰 수정	/{isbn}/comments/{id}
리뷰 삭제	/{isbn}/comments/{id}
모임 목록 조회	/communities/{isbn-page}/{size}/{comments}
모임 개설	/{isbn}/communities/{id}/{size}

상태 관리

- 서버 데이터는 React Query 로 관리
- 로컬 상태는 Recoil 로 분리

공통 API 클라이언트

axiosInstance 를 단일 인스턴스로 생성하고 모든 요청에 Authorization 헤더를 자동 주입하도록 설정

API 공통 형식

- 성공 응답 : { data: ... }
- 에러 응답 : { error: { code, message } }

프론트엔드 에러 처리 시 error.code 기준으로 분기 처리 가능하도록 설계

데이터베이스 ERD

ERD 설계 포인트

books.isbn(varchar(13)) 을 모든 테이블의 외래키로 사용하여 도서 단위 데이터 일관성 확보

데이터 무결성

books.isbn 을 on_delete: RESTRICT 로 설정하여 도서 삭제 시 연관 리뷰가 함께 사라지지 않도록 보호

학기별 데이터 분리

도서 테이블 는 school_year, school_term 컬럼을 추가하여 학기 단위 조회 및 관리 가능하도록 설계

주요 기능

로그인

기능 개요

카카오 OAuth 인증을 사용하여 별도의 회원가입 없이 로그인 할 수 있도록 구현했다. 로그인 성공 시 JWT 토큰을 발급하여 이후 요청에 사용한다.

구현 방안

- 프론트엔드에서 Authorization 헤더로 전달하는 Bearer 토큰 방식을 사용한다. JWT 만료 시 401 응답을 받아 자동 로그아웃 처리한다.
- 백엔드 미들웨어에서 토큰 검증 후 req.user 에 사용자 정보를 주입한다.
- 토큰 만료 시간은 7 일로 설정하고, 갱신 토큰 방식은 추후 개선 사항으로 남겨둔다.

회원가입

기능 개요

닉네임 설정, 프로필 사진 업로드, 관심 장르 선택, 독서 스타일 등록 등 최초 로그인 시 필요한 정보를 입력하는 온보딩 단계를 구현했다. 닉네임 중복 검사와 이미지 업로드를 함께 처리한다.

구현 방안

- 닉네임 중복 검사는 디바운스 300ms 후 서버에 요청하여 실시간 피드백을 제공한다.
- 이미지 업로드는 multer 미들웨어를 사용하고 S3 호환 스토리지에 저장한다.
- 온보딩 완료 여부는 users.onboarded 플래그로 관리하여 이후 로그인 시 바로 메인 화면으로 이동한다.

홈페이지

기능 개요

로그인 후 진입하는 메인 화면. 읽고 있는 책, 최근 활동, 추천 모임, 인기 리뷰 등 여러 섹션을 한 화면에 모아 보여준다. 각 섹션은 독립적인 API 호출로 구성되어 한 섹션이 실패해도 다른 섹션에 영향을 주지 않도록 했다.

구현 방안

- 홈 데이터는 /home 단일 엔드포인트에서 여러 테이블을 조인하여 한 번에 내려준다. 초기 로딩 속도 우선.
- 추천 모임은 사용자 장르 선호도와 최근 활동 기반으로 단순 가중치 계산을 한다. 추후 개선 여지.
- 섹션 순서는 사용자 설정 가능하도록 DB 에 저장한다.

클럽리스트

기능 개요

클럽리스트 도서 등록, 클럽 멤버 리스트 조회, 가입 신청 처리, 공지 작성 등 클럽 운영에 필요한 기본 기능을 묶어 제공한다. 모임 단위 협업의 시작점이 되는 화면이다.

구현 방안

- 가입 신청은 pending 상태로 저장되고 운영자 승인 시 active 로 전환된다.
- 멤버 목록은 cursor 기반 페이지네이션을 사용한다. 대형 클럽 대비.
- 공지는 최신 3 개만 메인에 노출하고 전체 목록은 별도 탭에서 조회한다.

인증형 챌스페이스

기능 개요

챌스페이스 인증 게시판. 인증 사진 업로드, 좋아요, 댓글 기능을 제공한다. 인증 사진은 S3 호환 스토리지에 저장하고 CDN 을 통해 서빙한다.

구현 방안

- 이미지 업로드는 presigned URL 방식을 사용하여 서버 부하를 최소화한다.
- 좋아요는 낙관적 업데이트 적용. 실패 시 롤백.
- 댓글은 depth 1 까지만 허용. 대댓글 UI 단순화.

북마크

기능 개요

북마크 관리. 읽고 있는 책, 읽을 예정, 읽은 책 상태 변경 및 진행 페이지 기록. 독서 습관 형성의 기초 데이터.

구현 방안

- 상태 변경 시 updated_at 갱신하여 최근 활동 정렬 기준으로 사용한다.
- 진행 페이지는 0 ~ total_pages 범위 검증 후 저장한다.
- 북마크 삭제 시 관련 리뷰는 유지한다. 사용자 의사 존중.

독서 기록 페이지

기능 개요

월별 독서 통계, 완독 도서 목록, 평균 독서 기간 등 개인 독서 이력을 한 눈에 보여주는 페이지. 동기 부여 목적.

구현 방안

- 통계는 GROUP BY DATE_TRUNC('month', ...) 쿼리로 집계한다.
- 차트 라이브러리는 recharts 사용. 가볍고 커스텀 가능.
- 데이터 양이 많아지면 materialized view 도입 검토.

책임기(리뷰) 게시글 페이지

구현 방안(FE)

- 게시글 상세 조회 시 GET /api/{isbn}/{id} 호출하여 본문 + 댓글 트리 한 번에 받아옴
- 댓글 작성 시 POST /api/{isbn}/{id}/comments 호출 후 invalidateQueries(['post', id]) 로 캐시 무효화
- 좋아요 토글은 낙관적 업데이트 적용 — 서버 응답 전 UI 먼저 반영, 실패 시 롤백

구현 방안(BE)

- 게시글 조회 시 LEFT JOIN comments 로 댓글 수 함께 집계
- 댓글 트리는 parent_id 재귀 조회 후 depth 2 까지만 노출
- 삭제된 댓글은 "삭제된 댓글입니다" 표시 유지 (대댓글 구조 보존)

도서 검색 페이지

기능 개요

알라딘 API 를 프록시하여 검색 결과를 가공하고 캐싱한다. 외부 API 의 응답 지연 및 장애 대응을 위해 타임아웃 3 초 설정.

구현 방안

- 검색 결과는 isbn 기준 중복 제거 후 응답
- 캐시 키 : search:{query}:{page} , TTL 1 시간
- 외부 API 실패 시 캐시 fallback 으로 빈 결과 대신 에러 메시지 반환

도서 상세 페이지

기능 개요

도서 메타데이터 + 사용자 보관 상태 + 리뷰 목록 + 모임 목록 을 한 화면에 모아 보여주는 허브 페이지. 사용자가 이 페이지에서 모든 행동을 시작하도록 설계.

구현 방안

- 초기 로딩 시 /api/{isbn} 단일 호출로 모든 데이터 수신
- 보관 상태 변경 시 optimistic update 적용
- 리뷰 작성 후 해당 모임 목록 캐시 invalidate

마이페이지

기능 개요

내 활동 전체를 한 곳에 모아보는 페이지. 보관 도서, 작성한 리뷰, 가입한 모임, 최근 활동 기록 등 탭 형태로 구성.

구현 방안

- 각 탭 데이터는 독립 API 호출로 분리하여 한 탭 실패가 다른 탭에 영향 없도록 처리
- 무한 스크롤 적용 — 페이지 사이즈 20 개
- 빈 상태 UI 를 각 탭마다 별도 디자인 적용

트러블 슈팅

1. Prisma Enum과 JWT 로직 불일치

문제 상황

카카오 OAuth 로 신규 사용자를 생성하는 과정에서 Prisma 의 enum 타입이 문자열 리터럴과 호환되지 않아 컴파일 오류가 발생했다. 인증 흐름 전체가 멈춰 있었다.

원인 분석

Prisma 가 생성한 enum 타입은 단순 문자열이 아니라 네임스페이스 객체 형태로 내보내지기 때문에, 토큰 페이로드에 문자열을 그대로 넣으면 타입 불일치가 발생한다.

문제 상황

```
error - src/auth/kakao.ts:42:18
TypeError: Cannot read properties of undefined (reading 'KAKAO')
    at signUpFromKakao (/app/dist/auth/kakao.js:42:18)
    at Module.exports.handler (/app/dist/auth/index.js:7:12)
    at Server.handleRequest (/app/node_modules/express/lib/router/index.js:284:15)
    at next (/app/node_modules/express/lib/router/index.js:137:13)
    at Layer.handle (/app/node_modules/express/lib/router/layer.js:95:5)
    at trim_prefix (/app/node_modules/express/lib/router/index.js:317:13)
    at middleware (/app/node_modules/express/lib/router/index.js:335:45)
    at Function.process_params (/app/node_modules/express/lib/router/index.js:346:12)
    at /app/node_modules/express/lib/router/index.js:280:10
    at Function.handle (/app/node_modules/express/lib/router/index.js:174:5)
```