얄디(Yaldi) 시퀀스

생성일	@2025년 10월 20일 오전 8:46
Prior	☐
기록 타입	기획
수정	☐
최종 편집 일시	@2025년 10월 20일 오전 9:32

Yaldi ERD 도구 시퀀스 다이어그램

본 문서는 Yaldi ERD 협업 도구의 주요 기능별 시퀀스 다이어그램을 포함합니다.

목차

10. 태블릿 스케치

- 10.1 스케치 → ERD 변환 (OCR)

11. 알림 시스템

- 11.1 일일 ERD Diff 알림
- 11.2 Webhook 이벤트

---

1. 인증 및 사용자 관리

1.1 일반 회원가입 및 로그인

기능 명세: 1-1-1
참조 API: POST /api/v1/auth/register, GET /api/v1/auth/verify-email, POST /api/v1/auth/login

```
sequenceDiagram
    participant C as Client
    participant S as Server
    participant DB as Database
    participant E as Email Service

    Note over C,E: 회원가입
    C->>+S: POST /api/v1/auth/register<br/>{email, password, username}
    S->>S: 비밀번호 검증 (8자 이상, 영문+숫자)
    S->>DB: 이메일 중복 확인
    alt 이메일 중복
        S-->>C: 409 Conflict (이미 존재하는 이메일)
    else 성공
        S->>DB: 사용자 정보 저장
        S->>E: 인증 이메일 발송 (24시간 유효)
        S-->>-C: 201 Created {userId, email, username}
    end

    Note over C,E: 이메일 인증
    C->>+S: GET /api/v1/auth/verify-email?token=abc123
```

```
    S->>DB: 토큰 유효성 확인
    alt 토큰 만료 또는 유효하지 않음
        S-->>C: 404 Not Found / 400 Bad Request
    else 성공
        S->>DB: 이메일 인증 완료 처리
        S-->>-C: 200 OK {message: "이메일 인증 완료"}
    end

    Note over C,E: 로그인
    C->>+S: POST /api/v1/auth/login<br/>{email, password}
    S->>DB: 사용자 인증 확인
    alt 이메일 미인증
        S-->>C: 403 Forbidden (이메일 인증 필요)
    else 비밀번호 불일치
        S-->>C: 401 Unauthorized
    else 성공
        S->>S: JWT 토큰 생성<br/>(Access: 1시간, Refresh: 30일)
        S-->>-C: 200 OK {accessToken, refreshToken, user}
    end
```

2. 프로젝트 관리

2.1 프로젝트 생성 및 팀원 초대

기능 명세: 2-1-1, 2-2-1
참조 API: POST /api/v1/projects, POST /api/v1/projects/{id}/members

```
sequenceDiagram
    participant Owner as Owner (Client)
    participant S as Server
    participant DB as Database
    participant E as Email Service
    participant Editor as Editor (Client)
    participant WS as WebSocket
```

```
    Note over Owner,DB: 프로젝트 생성
    Owner->>+S: POST /api/v1/projects<br/>{name, description, visibility, type}
    S->>DB: 프로젝트 생성
    S->>DB: Owner 권한 부여
    S-->>-Owner: 201 Created {projectId, name, owner}

    Note over Owner,E: 팀원 초대
    Owner->>+S: POST /api/v1/projects/{id}/members<br/>{email, role: "edito
r"}
    S->>DB: 초대 대상 사용자 확인
    alt 사용자 미존재
        S-->>Owner: 404 Not Found
    else 이미 팀원
        S-->>Owner: 409 Conflict
    else 성공
        S->>DB: 초대 정보 저장 (pending 상태)
        S->>E: 초대 이메일 발송<br/>(초대 링크 포함)
        S-->>-Owner: 200 OK {invitationId}
    end

    Note over Editor,WS: 초대 수락
    Editor->>+S: POST /api/v1/invitations/{id}/accept
    S->>DB: 초대 정보 확인
    alt 초대 만료 또는 취소됨
        S-->>Editor: 404 Not Found / 410 Gone
    else 성공
        S->>DB: 팀원으로 등록 (role: editor)
        S->>DB: 초대 상태 업데이트 (accepted)
        S->>WS: 팀원 참여 브로드캐스트<br/>{event: "member.joined", user}
        S-->>-Editor: 200 OK {projectId, role}
        WS-->>Owner: 팀원 참여 알림
    end
```

---

3. ERD 편집

3.1 테이블 생성 및 컬럼 추가

기능 명세: 2-3-1, 2-3-2
참조 API: POST /api/v1/projects/{id}/tables, POST /api/v1/tables/{id}/columns

```
sequenceDiagram
    participant C as Client (Editor)
    participant S as Server
    participant DB as Database
    participant WS as WebSocket
    participant Others as Other Clients

    Note over C,DB: 테이블 생성
    C->>+S: POST /api/v1/projects/{id}/tables<br/>{name, description, position {x,y}}
    S->>DB: 권한 확인 (Owner/Editor)
    S->>DB: 테이블명 중복 확인
    alt 테이블명 중복
        S-->>C: 409 Conflict
    else 성공
        S->>DB: 테이블 생성
        S->>DB: 이력 저장 (action: "table.created")
        S->>WS: 변경사항 브로드캐스트<br/>{event: "erd.tableCreated", table}
        S-->>-C: 201 Created {tableId, name, position}

        WS-->>Others: 테이블 생성 동기화
    end

    Note over C,WS: 컬럼 추가
    C->>+S: POST /api/v1/tables/{id}/columns<br/>{name, type, nullable, defaultValue, isPK}
    S->>DB: 권한 확인
    S->>DB: 컬럼 추가
    alt PK 설정 시
        S->>DB: 기존 PK가 있으면 해제 (단일 PK 정책)
    end
```

```
    S->>DB: 이력 저장 (action: "column.added")
    S->>WS: 변경사항 브로드캐스트<br/>{event: "erd.columnAdded", tableId, column}
    S-->>-C: 201 Created {columnId, name, type}

    WS-->>Others: 컬럼 추가 동기화
```

3.2 관계선 생성

기능 명세: 2-3-3
참조 API: POST /api/v1/projects/{id}/relations

```
sequenceDiagram
    participant C as Client (Editor)
    participant S as Server
    participant DB as Database
    participant AI as AI Service
    participant WS as WebSocket
    participant Others as Other Clients

    Note over C,AI: 관계선 생성 및 검증
    C->>+S: POST /api/v1/projects/{id}/relations<br/>{fromTableId, toTableId, type: "1:N",<br/>relationType: "identifying", onDelete: "CASCADE"}
    S->>DB: 권한 확인 (Owner/Editor)
    S->>DB: 테이블 존재 확인

    Note over S,AI: FK 타입 검증
    S->>AI: FK 타입 일치 검증 요청<br/>{fromColumn, toColumn}
    AI->>AI: PK 타입과 FK 타입 비교
    alt 타입 불일치
        AI-->>S: 경고 {warning: "FK 타입 불일치"}
        S-->>C: 200 Created (with warning)
    else 타입 일치
        AI-->>S: 검증 성공
    end
```

```
    S->>DB: 관계선 생성
    alt 식별 관계 (identifying)
        S->>DB: FK 컬럼을 PK의 일부로 추가
    else 비식별 관계 (non-identifying)
        S->>DB: 일반 FK 컬럼 생성
    end

    S->>DB: 이력 저장 (action: "relation.created")
    S->>WS: 관계선 생성 브로드캐스트<br/>{event: "erd.relationCreated", relation}
    S-->>-C: 201 Created {relationId, fromTable, toTable}

    WS-->>Others: 관계선 생성 동기화
```

3.3 인덱스 생성

기능 명세: 2-3-4
참조 API: POST /api/v1/tables/{id}/indexes

```
sequenceDiagram
    participant C as Client (Editor)
    participant S as Server
    participant DB as Database
    participant WS as WebSocket
    participant Others as Other Clients

    Note over C,DB: 인덱스 생성
    C->>+S: POST /api/v1/tables/{id}/indexes<br/>{name, columns: ["userId",
"createdAt"],<br/>type: "BTREE", isUnique: true}
    S->>DB: 권한 확인 (Owner/Editor)
    S->>DB: 컬럼 존재 확인

    alt 인덱스명 미입력
        S->>S: 인덱스명 자동 생성<br/>(idx_tableName_column1_column2)
    end
```

```
    S->>DB: 인덱스 정보 저장
    S->>DB: 이력 저장 (action: "index.created")
    S->>WS: 인덱스 추가 알림<br/>{event: "erd.indexCreated", tableId, index}
    S-->>-C: 201 Created {indexId, name, columns, type}

    WS-->>Others: 인덱스 추가 동기화

    Note over C: SQL Export 시 CREATE INDEX 문 자동 포함
```

3.4 메모/주석 관리

기능 명세: 2-3-5
참조 API: POST /api/v1/projects/{id}/memos, PATCH /api/v1/memos/{id}

```
sequenceDiagram
    participant C as Client (Editor)
    participant S as Server
    participant DB as Database
    participant WS as WebSocket
    participant Others as Other Clients

    Note over C,DB: 메모 추가 (Sticky Note)
    C->>+S: POST /api/v1/projects/{id}/memos<br/>{content, color, position{x,
y}, tableId}
    S->>DB: 권한 확인 (Owner/Editor)
    S->>DB: 메모 생성
    S->>DB: 이력 저장 (action: "memo.created")
    S->>WS: 메모 생성 브로드캐스트<br/>{event: "erd.memoCreated", memo}
    S-->>-C: 201 Created {memoId, content, color, position}

    WS-->>Others: 메모 생성 동기화

    Note over C,WS: 메모 수정 (내용, 색상, 위치)
    C->>+S: PATCH /api/v1/memos/{id}<br/>{content, color, position{x,y}}
```

```
    S->>DB: 권한 확인
    S->>DB: 메모 업데이트
    S->>DB: 이력 저장 (action: "memo.updated")
    S->>WS: 메모 변경 동기화<br/>{event: "erd.memoUpdated", memoId, changes}
    S-->>-C: 200 OK {memo}

    WS-->>Others: 실시간 메모 동기화

    Note over S: 메모는 SQL Export에 포함되지 않음<br/>협업용 설계 의도 공유 목적
```

---

4. SQL Import/Export

4.1 SQL Import (AI 구문 검사)

기능 명세: 2-4-1
참조 API: POST /api/v1/projects/{id}/import/sql

```
sequenceDiagram
    participant C as Client
    participant S as Server
    participant AI as AI Service
    participant DB as Database

    Note over C,AI: SQL 파일 업로드 및 구문 검사
    C->>+S: POST /api/v1/projects/{id}/import/sql<br/>(multipart/form-data: sqlFile)
    S->>S: SQL 파일 읽기

    Note over S,AI: AI SQL Linter 검사
    S->>+AI: SQL 구문 검사 요청<br/>{sql, database: "postgresql"}
    AI->>AI: SQL 파싱 및 구문 분석
    AI->>AI: 문법 오류 검출
    alt 구문 오류 있음
        AI-->>S: 오류 목록 반환<br/>{errors: [{line, message, suggestion}]}
```

```
    S-->>C: 400 Bad Request<br/>{errors, suggestions}
    Note over C: 오류 위치 하이라이트 표시<br/>수정 제안 제공
  else 구문 정상
    AI-->>-S: 검증 성공

    Note over S,DB: ERD 변환
    S->>S: SQL 파싱 후 ERD로 변환<br/>(CREATE TABLE → 테이블/컬럼 생성)
    S->>DB: 테이블, 컬럼, 관계선 저장
    S->>DB: 이력 저장 (action: "sql.imported")
    S-->>-C: 200 OK {projectId, tablesCreated}

    Note over C: ERD 캔버스에 자동 배치
  end
```

4.2 SQL Export (AI 검증)

기능 명세: 2-4-2
참조 API: GET /api/v1/projects/{id}/export/sql

```
sequenceDiagram
    participant C as Client
    participant S as Server
    participant AI as AI Service
    participant DB as Database

    Note over C,DB: SQL Export 요청
    C->>+S: GET /api/v1/projects/{id}/export/sql?database=postgresql
    S->>DB: ERD 데이터 조회 (테이블, 컬럼, 관계선, 인덱스)
    S->>S: SQL DDL 생성 (CREATE TABLE, ALTER TABLE, CREATE INDEX)

    Note over S,AI: AI 설계 검증
    S->>+AI: ERD 설계 검증 요청<br/>{tables, relations, indexes}
    AI->>AI: FK 타입 일치 확인
    AI->>AI: varchar 길이 충분성 검사
    AI->>AI: 정규화 위반 검출
```

```
    alt 설계 오류 발견
        AI-->>S: 경고 목록 반환<br/>{warnings: [<br/>"FK 타입 불일치: users.id(bigint) vs orders.userId(int)",<br/>"nickname(10) 길이 부족, nicknameSub(5) 존재"<br/>]}
        S-->>-C: 200 OK {sql, warnings}

        Note over C: SQL 다운로드 가능<br/>경고 메시지 표시 (빌드 시 오류 가능)
    else 설계 정상
        AI-->>S: 검증 성공
        S->>C: 200 OK {sql}

        Note over C: SQL 파일 다운로드
    end
```

5. 실시간 협업

5.1 실시간 편집 동기화 (WebSocket)

기능 명세: 2-7-1
참조 API: WebSocket /ws/projects/{id}, SSE /sse/projects/{id}

```
sequenceDiagram
    participant Owner as Owner/Editor (WS)
    participant WS as WebSocket Server
    participant S as Server
    participant DB as Database
    participant Viewer as Viewer (SSE)

    Note over Owner,WS: Owner/Editor WebSocket 연결
    Owner->>+WS: WebSocket 연결<br/>/ws/projects/{id}?token=xxx
    WS->>S: JWT 토큰 검증 및 권한 확인
    alt Viewer 권한
        WS-->>Owner: 403 Forbidden (WebSocket 사용 불가)
    else Owner/Editor
```

```
    WS->>DB: Presence 등록 (userId, projectId, connectedAt)
    WS-->>Owner: 연결 성공<br/>{event: "connected", presence: [users]}
end

Note over Viewer,S: Viewer SSE 연결
Viewer->>+S: SSE 연결<br/>/sse/projects/{id}?token=xxx
S->>DB: 권한 확인 (Viewer)
S-->>Viewer: SSE 스트림 연결<br/>data: {event: "connected"}

Note over Owner,Viewer: 실시간 편집 동기화
Owner->>WS: erd.operation<br/>{type: "table.update", tableId, changes}
WS->>S: Operational Transformation 처리
S->>DB: 변경사항 저장
S->>DB: 이력 기록 (평균 0.3 ops/s)

WS-->>Owner: erd.operationAck (전송 확인)
WS-->>Owner: 변경사항 브로드캐스트<br/>{event: "erd.updated", operation}
S-->>Viewer: SSE 이벤트 전송<br/>data: {event: "erd.updated", operation}

Note over WS: Presence 갱신 주기: 10-15Hz
```

5.2 충돌 방지 및 Merge (OT/CRDT)

기능 명세: 2-7-2
참조 API: WebSocket /ws/projects/{id}

```
sequenceDiagram
    participant ClientA as Client A (Editor)
    participant WS as WebSocket Server
    participant S as Server (OT Engine)
    participant DB as Database
    participant ClientB as Client B (Editor)

    Note over ClientA,ClientB: 동시 편집 시나리오
    ClientA->>WS: operation A<br/>{type: "column.update", columnId: "col1",<
```

```
br/>property: "name", value: "userName"}
    ClientB->>WS: operation B<br/>{type: "column.update", columnId: "col1",<
br/>property: "type", value: "VARCHAR(50)"}

    Note over WS,S: Operational Transformation 처리
    WS->>S: operation A 수신 (timestamp: t1)
    WS->>S: operation B 수신 (timestamp: t2)

    S->>S: OT 변환 적용<br/>(operation A와 B는 독립적이므로 충돌 없음)
    S->>DB: operation A 저장
    S->>DB: operation B 저장

    Note over S,ClientB: 병합된 결과 브로드캐스트
    S->>WS: 변환된 operation A'
    S->>WS: 변환된 operation B'

    WS-->>ClientA: operation B' 전송<br/>(B의 변경사항 반영)
    WS-->>ClientB: operation A' 전송<br/>(A의 변경사항 반영)

    Note over ClientA,ClientB: 동일한 속성 동시 수정 시 충돌
    ClientA->>WS: operation C<br/>{columnId: "col1", property: "name", value:
"email"}
    ClientB->>WS: operation D<br/>{columnId: "col1", property: "name", value:
"userEmail"}

    S->>S: Last-Write-Wins 정책 적용<br/>(timestamp 비교)
    alt operation D가 나중
        S->>DB: operation D 우선 적용
        WS-->>ClientA: 충돌 알림 + operation D 강제 적용
        WS-->>ClientB: operation D 확인
    end
```

5.3 커서 및 선택 영역 공유

기능 명세: 2-7-3
참조 API: WebSocket /ws/projects/{id}

```
sequenceDiagram
    participant C1 as Client 1 (Editor)
    participant WS as WebSocket Server
    participant S as Server
    participant C2 as Client 2 (Editor)
    participant C3 as Client 3 (Viewer)

    Note over C1,C3: 커서 위치 공유 (10-15Hz)
    C1->>WS: presence.update<br/>{userId, cursor: {x: 100, y: 200}, color: "#F
F5733"}
    WS->>S: Presence 정보 갱신

    WS-->>C2: presence.broadcast<br/>{userId: "user1", cursor: {x, y}, color}
    WS-->>C3: presence.broadcast (SSE 통해 전송)

    Note over C2: Client 1의 커서 표시<br/>(색상: #FF5733, 라벨: "User1")

    Note over C1,C3: 선택 영역 공유
    C1->>WS: presence.update<br/>{userId, selection: {<br/>type: "table", tabl
eId: "tbl_123"<br/>}}
    WS-->>C2: presence.broadcast<br/>{userId: "user1", selection}
    WS-->>C3: presence.broadcast (SSE)

    Note over C2,C3: 선택한 테이블 하이라이트 표시<br/>(User1 색상으로 테두리 강
조)

    Note over C2,C3: 컬럼 선택 시
    C1->>WS: presence.update<br/>{selection: {type: "column", columnId: "col
_456"}}
    WS-->>C2: presence.broadcast
    WS-->>C3: presence.broadcast
```

```
    Note over C2,C3: 선택된 컬럼 하이라이트<br/>사용자별 색상 자동 할당
```

---

6. 버전 관리

6.1 버전 저장 (AI 검증)

기능 명세: 3-2-1
참조 API: POST /api/v1/projects/{id}/versions

```
sequenceDiagram
    participant C as Client (Editor)
    participant S as Server
    participant DB as Database
    participant AI as AI Service

    Note over C,DB: 버전 저장 요청
    C->>+S: POST /api/v1/projects/{id}/versions<br/>{name: "v1.0", descriptio
n: "초기 설계"}
    S->>DB: 권한 확인 (Owner/Editor)
    S->>DB: 현재 ERD 스냅샷 생성<br/>(테이블, 컬럼, 관계선, 인덱스 전체 복사)

    Note over S,AI: AI 빌드 가능 여부 검증
    S->>+AI: ERD 설계 검증 요청<br/>{tables, relations, indexes}
    AI->>AI: SQL 빌드 시뮬레이션
    AI->>AI: FK 타입 일치 확인
    AI->>AI: NOT NULL 제약조건 검사
    AI->>AI: 순환 참조 검사

    alt 빌드 오류 발견
        AI-->>S: 오류 목록 반환<br/>{errors: [<br/>"FK 타입 불일치",<br/>"필수 컬럼
에 NULL 허용"<br/>], buildable: false}
        S->>DB: 버전 저장 (buildStatus: "error", errors)
        S-->>-C: 201 Created {versionId, buildStatus: "error", errors}
```

```
    Note over C: 빌드 오류 있음 표시 (🔴)<br/>오류 있어도 저장 가능
  else 빌드 가능
    AI->>-S: 검증 성공 {buildable: true}
    S->>DB: 버전 저장 (buildStatus: "success")
    S->>C: 201 Created {versionId, buildStatus: "success"}

    Note over C: 빌드 가능 표시 (🟢)
  end
```

6.2 버전 복원 (RollBack)

기능 명세: 3-2-3
참조 API: GET /api/v1/versions/{id}, POST /api/v1/versions/{id}/restore

```
sequenceDiagram
    participant C as Client (Owner/Editor)
    participant S as Server
    participant DB as Database
    participant WS as WebSocket
    participant Others as Other Clients

    Note over C,DB: 버전 조회
    C->>+S: GET /api/v1/versions/{id}
    S->>DB: 버전 정보 조회
    S->>-C: 200 OK {version, snapshot, buildStatus}

    Note over C: 버전 ERD 미리보기

    Note over C,WS: 버전 복원 (RollBack)
    C->>+S: POST /api/v1/versions/{id}/restore
    S->>DB: 권한 확인 (Owner/Editor)

    Note over S: 확인 모달 필요<br/>"현재 ERD를 이 버전으로 복원하시겠습니까?<br/>현재 내용이 손실됩니다."
```

```
    S->>DB: 현재 ERD를 임시 백업 (자동 버전 생성)
    S->>DB: 버전 스냅샷으로 ERD 교체<br/>(테이블, 컬럼, 관계선 전체 교체)
    S->>DB: 이력 기록 (action: "version.restored", versionId)

    S->>WS: 복원 알림 브로드캐스트<br/>{event: "erd.restored", versionId}
    S-->>-C: 200 OK {message: "복원 완료"}

    WS-->>Others: ERD 복원 알림<br/>"Owner가 v1.0으로 복원했습니다"

    Note over Others: 캔버스 전체 리로드
```

---

7. AI 기능

7.1 AI 정규화 제안

기능 명세: 2-8-1
참조 API: POST /api/v1/projects/{id}/ai/normalize

```
sequenceDiagram
    participant C as Client (Editor)
    participant S as Server
    participant AI as AI Service
    participant DB as Database

    Note over C,AI: 정규화 제안 요청
    C->>+S: POST /api/v1/projects/{id}/ai/normalize
    S->>DB: 현재 ERD 조회 (테이블, 컬럼, 관계선)

    S->>+AI: 정규화 분석 요청<br/>{tables, relations}

    Note over AI: 제 1정규화 검사
    AI->>AI: 다중값 속성 검출<br/>(예: "tags" 컬럼에 쉼표로 구분된 값)

    Note over AI: 제 2정규화 검사
    AI->>AI: 부분 함수 종속 검출<br/>(복합키의 일부에만 종속된 컬럼)
```

```
    Note over AI: 제 3정규화 검사
    AI->>AI: 이행 함수 종속 검출<br/>(비키 컬럼 간 종속)

    alt 정규화 위반 발견
        AI-->>S: 제안 사항 반환<br/>{suggestions: [{<br/>  type: "제2정규화",<br/>
issue: "부분 함수 종속",<br/>  currentTable: "orders",<br/>  recommendation:
"별도 테이블 분리",<br/>  newTables: ["order_items"],<br/>  sampleData: [{...}]<
br/>}]}
        S-->>-C: 200 OK {suggestions}

        Note over C: 제안 사항 표시<br/>예시 데이터와 함께 이유 설명

        C->>S: POST /api/v1/projects/{id}/ai/normalize/accept<br/>{suggestionI
d}
        S->>DB: ERD 자동 변경 (테이블 분리)
        S-->>C: 200 OK {modified: true}
    else 정규화 정상
        AI-->>S: 정규화 문제 없음
        S-->>C: 200 OK {suggestions: [], message: "정규화 문제 없음"}
    end
```

7.2 AI 더미데이터 생성

기능 명세: 3-3-1
참조 API: POST /api/v1/versions/{id}/ai/dummy-data

```
sequenceDiagram
    participant C as Client (Editor)
    participant S as Server
    participant DB as Database
    participant AI as AI Service (GPT)

    Note over C,DB: 빌드 가능 버전 확인
    C->>+S: GET /api/v1/projects/{id}/versions
```

```
    S->>DB: 버전 목록 조회 (buildStatus 포함)
    S-->>-C: 200 OK {versions: [{versionId, buildStatus}]}

    Note over C: 빌드 가능(🟢) 버전만 '더미데이터 생성' 버튼 활성화

    Note over C,AI: 더미데이터 생성 요청
    C->>+S: POST /api/v1/versions/{id}/ai/dummy-data<br/>{<br/>  requirements: "한국 이커머스",<br/>  region: "Korea",<br/>  category: "Electronics",<br/>  count: 100<br/>}
    S->>DB: 버전 빌드 상태 확인
    alt 빌드 오류 있음
        S-->>C: 400 Bad Request<br/>"빌드 오류가 있는 버전은 더미데이터 생성 불가"
    else 빌드 가능
        S->>DB: 버전 스냅샷 (ERD 구조) 조회

        Note over S,AI: GPT 기반 더미데이터 생성
        S->>+AI: 더미데이터 생성 요청<br/>{schema, requirements, count}
        AI->>AI: 테이블 구조 분석
        AI->>AI: 한국 이커머스 맥락 반영<br/>(서울, 부산 등 지역, 한국 이름, 전자제품 카테고리)
        AI->>AI: FK 관계 고려한 데이터 생성
        AI-->>-S: 더미데이터 반환<br/>{sql: "INSERT INTO...", json: [{...}]}

        S-->>-C: 200 OK {sql, json, downloadUrl}

        Note over C: SQL 또는 JSON 형식으로 다운로드
    end
```

7.3 AI 설계 검증

기능 명세: 여러 기능에서 공통 사용
참조 API: 내부 AI 서비스 호출

```
sequenceDiagram
    participant S as Server
    participant AI as AI Service
    participant DB as Database

    Note over S,AI: AI 설계 검증 (공통 모듈)
    S->>+AI: ERD 검증 요청<br/>{tables, relations, indexes, context}

    Note over AI: SQL 구문 검사
    AI->>AI: SQL DDL 생성 시뮬레이션
    AI->>AI: 구문 오류 검출

    Note over AI: FK 타입 검사
    AI->>AI: PK-FK 타입 일치 확인<br/>(bigint vs int, varchar 길이)
    alt 타입 불일치
        AI->>AI: 오류 목록 추가<br/>"users.id(bigint) vs orders.userId(int)"
    end

    Note over AI: varchar 길이 검사
    AI->>AI: 관련 컬럼 길이 비교<br/>(nickname(10) + nicknameSub(5) 검출)
    alt 길이 부족
        AI->>AI: 경고 추가<br/>"nickname(10) 길이 부족, nicknameSub(5) 고려 시 최소 15 필요"
    end

    Note over AI: 정규화 검사
    AI->>AI: 제1~3정규화 위반 검출

    Note over AI: 순환 참조 검사
    AI->>AI: 관계선 그래프 분석

    alt 오류/경고 있음
        AI-->>-S: {<br/>  errors: [...],<br/>  warnings: [...],<br/>  buildable: false/true<br/>}
    else 정상
```

```
    AI-->>S: {buildable: true, errors: [], warnings: []}
  end

  Note over S: 검증 결과를 호출한 기능에 반환<br/>(버전 저장, SQL Export 등)
```

---

8. DTO 관리

8.1 ERD → DTO 생성

기능 명세: 2-6-1, 4-1-2
참조 API: POST /api/v1/projects/{id}/dtos

```
sequenceDiagram
  participant C as Client (ERD 페이지)
  participant S as Server
  participant DB as Database
  participant DTO as DTO 페이지

  Note over C,DB: DTO 생성 요청 (ERD 페이지에서)
  C->>+S: POST /api/v1/projects/{id}/dtos<br/>{<br/>  tableId: "users",<br/>dtoType: "Response",<br/>  fields: {<br/>    userId: {include: true},<br/>    email: {include: true},<br/>    password: {include: false},<br/>    createdAt: {include: true}<br/>  },<br/>  nested: {<br/>    posts: {tableId: "posts", fields: {...}}<br/>  }<br/>}
  S->>DB: 권한 확인 (Owner/Editor)
  S->>DB: 테이블 구조 조회

  Note over S: DTO 구조 생성
  S->>S: 컬럼 comment → DTO 필드 설명 매핑
  S->>S: Nested DTO 구조 생성 (다른 테이블 참조)
  S->>DB: DTO 정보 저장
  S-->>-C: 201 Created {dtoId, name: "UserResponse"}

  Note over C,DTO: DTO 페이지로 이동
  C->>DTO: Navigate to DTO 페이지
```

```
    DTO->>S: GET /api/v1/dtos/{id}
    S-->>DTO: 200 OK {dto, fields, nested}

    Note over DTO: DTO 구조 표시<br/>실시간 코드 프리뷰 (TypeScript/Java)
```

8.2 DTO Export

기능 명세: 4-2-1
참조 API: GET /api/v1/dtos/export

```
sequenceDiagram
    participant C as Client
    participant S as Server
    participant DB as Database

    Note over C,DB: Public 프로젝트 검색
    C->>+S: GET /api/v1/explore/projects? | search=ecommerce&page=1&limit=
20
    S->>DB: Public 프로젝트 검색, (visibility: "public")
    S->>DB: 프로젝트명, 설명 기반 키워드 검색, (Full-text search)

    S->>DB: ERD 미리보기 이미지 조회, (캐싱된 썸네일)

    S-->>-C: 200 OK {projects: [{,  projectId,,  name: "E-commerce ERD",,  de
scription: "온라인 쇼핑몰 ERD",,  owner: {username, profileImage},,  thumbnail:
"https://cdn.../preview.png",,  tablesCount: 25,,  createdAt,,  updatedAt, }], t
otal, page}

    Note over C: 프로젝트 목록 표시 | ERD 미리보기 포함

    Note over C,DB: 프로젝트 상세 보기 (Viewer 권한)
    C->>+S: GET /api/v1/projects/{id} | (Public 프로젝트)
    S->>DB: 프로젝트 정보 조회
    S->>DB: ERD 데이터 조회 (읽기 전용)
    S-->>-C: 200 OK {project, erd}
```

```
    Note over C: ERD 상세 페이지 | Viewer 권한으로 열람 | (편집 불가, 조회만 가능)
```

9. 프로젝트 탐색

9.1 Public 프로젝트 검색

기능 명세: 5-1-1
참조 API: GET /api/v1/explore/projects

```
sequenceDiagram
    participant C as Client
    participant S as Server
    participant DB as Database

    Note over C,DB: Public 프로젝트 검색
    C->>+S: GET /api/v1/explore/projects?<br/>search=ecommerce&page=1&limit=20
    S->>DB: Public 프로젝트 검색<br/>(visibility: "public")
    S->>DB: 프로젝트명, 설명 기반 키워드 검색<br/>(Full-text search)

    S->>DB: ERD 미리보기 이미지 조회<br/>(캐싱된 썸네일)

    S-->>-C: 200 OK {projects: [{<br/>  projectId,<br/>  name: "E-commerce ERD",<br/>  description: "온라인 쇼핑몰 ERD",<br/>  owner: {username, profileImage},<br/>  thumbnail: "<https://cdn>.../preview.png",<br/>  tablesCount: 25,<br/>  createdAt,<br/>  updatedAt<br/>}], total, page}

    Note over C: 프로젝트 목록 표시<br/>ERD 미리보기 포함

    Note over C,DB: 프로젝트 상세 보기 (Viewer 권한)
    C->>+S: GET /api/v1/projects/{id}<br/>(Public 프로젝트)
    S->>DB: 프로젝트 정보 조회
    S->>DB: ERD 데이터 조회 (읽기 전용)
    S-->>-C: 200 OK {project, erd}
```

```
    Note over C: ERD 상세 페이지<br/>Viewer 권한으로 열람<br/>(편집 불가, 조회만 가능)
```

9.2 유사 프로젝트 추천 (Vector Search)

기능 명세: 5-2-1
참조 API: GET /api/v1/projects/{id}/similar

```
sequenceDiagram
    participant C as Client
    participant S as Server
    participant DB as Database
    participant VDB as Vector DB

    Note over C,VDB: 유사 프로젝트 추천
    C->>+S: GET /api/v1/projects/{id}/similar?limit=10
    S->>DB: 프로젝트 ERD 구조 조회

    Note over S,VDB: Graph-aware Embedding 검색
    S->>S: ERD 구조를 그래프로 변환<br/>(테이블: 노드, 관계선: 엣지)
    S->>+VDB: Vector Search 요청<br/>{graphStructure, embeddings}

    VDB->>VDB: 관계형 구조 기반 유사도 계산<br/>(테이블명, 관계 패턴, 도메인)
    VDB->>VDB: 유사도 높은 프로젝트 검색<br/>(Cosine Similarity)

    VDB-->>-S: 유사 프로젝트 목록<br/>{projects: [{projectId, similarity: 0.85}]}

    S->>DB: 프로젝트 정보 조회<br/>(Public 프로젝트만)
    S-->>-C: 200 OK {similarProjects: [{<br/>  projectId,<br/>  name,<br/>  similarity: 0.85,<br/>  thumbnail,<br/>  matchReason: "유사한 관계 구조 (User-Order-Product)"<br/>}]}

    Note over C: "이 ERD와 비슷한 프로젝트" 섹션 표시<br/>유사도 점수 및 이유 표시
```

10. 태블릿 스케치

11. 알림 시스템

문서 정보

- **작성일**: 2025-02-03
- **버전**: 1.0
- **기반 문서**:
  - 기능 명세서.csv
  - API_연동_규격서.csv
- **총 시퀀스 다이어그램 수**: 29개

사용 방법

1. 각 시퀀스 다이어그램은 Mermaid 형식으로 작성되어 있습니다.
2. GitHub, GitLab, Notion, VS Code (Mermaid 플러그인) 등에서 렌더링 가능합니다.
3. 목차의 링크를 클릭하여 원하는 시퀀스 다이어그램으로 이동할 수 있습니다.

주요 액터 (Actor)

- **Client**: 웹/모바일 클라이언트
- **Server**: 백엔드 서버
- **Database**: 데이터베이스
- **WebSocket**: WebSocket 서버 (실시간 협업)
- **AI Service**: AI 분석 서비스 (GPT 기반)
- **Email Service**: 이메일 발송 서비스
- **OAuth Provider**: Google, GitHub 등 OAuth 제공자
- **OCR Service**: 손글씨 인식 서비스
- **Vector DB**: 벡터 데이터베이스 (유사도 검색)

- **Push Service**: 모바일 푸시 알림 서비스 (FCM/APNs)
- **Storage**: 파일 저장소 (S3/CDN)
- **Scheduler**: Cron 스케줄러