천지창조 기술문서

작성자 : 이예은

작성날짜 : 2025.07.23

1. 문서 개요

1.1. 문서 목적

CreationStack 프로젝트의 시스템 아키텍처, 주요 기능, 기술 스택을 명확히 설명합니다.

1.2. 문서 범위

백엔드 애플리케이션, 데이터베이스

1.3. 작성 일자

2025년 7월 22일

1.4. 작성자

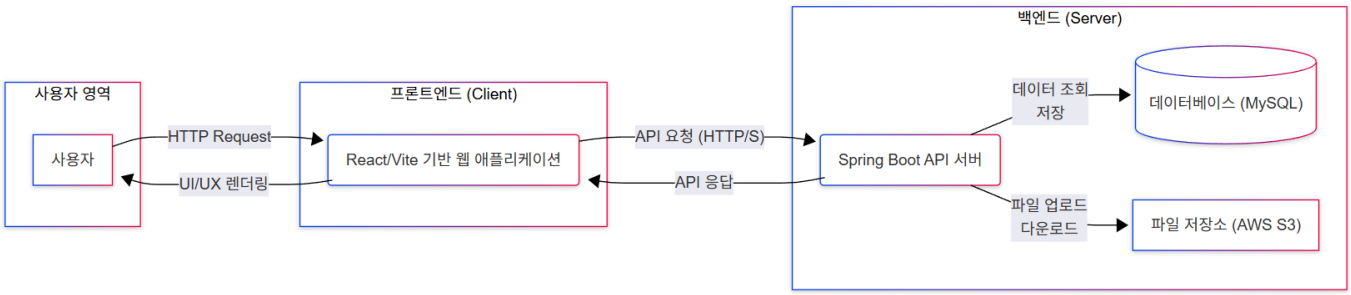
이예은

2. 시스템 개요

2.1. 프로젝트 개요

CreationStack은 실무자들이 자신의 경험과 지식을 담은 콘텐츠를 업로드하고, 취업 준비생과 같은 사용자들이 이를 구독하여 실질적인 도움을 받을 수 있는 플랫폼입니다.

2.2. 시스템 아키텍처



2.2.1. Frontend

React와 Vite를 사용하여 구현된 싱글 페이지 애플리케이션입니다. 사용자에게 UI를 제공하고, Backend API와 통신하여 데이터를 주고받습니다.

2.2.2. Backend

Spring Boot 기반의 RESTful API 서버입니다. 데이터베이스와 연동하여 비즈니스 로직을 처리하고, Frontend에 데이터를 제공합니다.

2.2.3. Database

MySQL을 사용하여 데이터를 저장하고 관리합니다.

2.2.4. Storage

AWS S3를 사용하여 이미지,파일 등 정적 파일을 저장합니다.

2.3. 기술 스택

2.3.1. Backend

- Spring Boot 3.5.3
  - Spring Data JPA
  - Spring Security
  - Spring Web
- MySQL 8.0.41
- Lombok
- JWT (JSON Web Token)
- Portone (iamport-rest-client-java)
- AWS S3
- Maven

2.3.2. Frontend

- React
- Vite
- JavaScript
- CSS
- emoji-mart/react
- qs

3. 기능 명세

3.1. 주요 기능

- 이메일 인증 기반 회원가입 및 로그인
- 카카오 인증 회원가입 및 로그인
- 사용자 프로필 관리
- IT/커리어 관련 콘텐츠 CRUD
- 창작자별 유료 구독 멤버십 결제 및 구독자 전용 콘텐츠 접근 권한 관리
- 결제 내역 확인 및 구독 관리
- 결제 수단 관리
- 콘텐츠 별 좋아요, 댓글 CRUD, 좋아한 콘텐츠 목록 조회 기능
- 창작자의 공지 CRUD 및 구독자 리액션 기능
- 키워드 기반 검색 및 카테고리 검색을 통한 크리에이터/콘텐츠 필터링

3.2. 사용자 및 인증 (User & Authentication)

3.2.1. 기능 설명

사용자의 회원가입, 로그인, 정보 관리 및 시스템 접근 권한 제어를 담당합니다.
사용자는 일반 USER와 CREATOR 역할로 구분되며, JWT(Json Web Token)를 사용하여 인증을 처리합니다.

3.2.2. 주요 처리 흐름

3.2.2.1. 신규 회원가입

1. 사용자
   회원가입 페이지에서 이메일, 비밀번호, 닉네임 등 필수 정보를 입력하고 '가입' 버튼을 클릭합니다.

2. 프론트엔드
    - 입력된 이메일과 닉네임에 대해 중복 확인 API (GET /api/user/check-email, GET /api/user/check-nickname)를 호출하여 실시간으로 사용 가능 여부를 검증합니다.
    - 모든 정보가 유효하면, POST /api/user API를 통해 백엔드로 회원가입 정보를 전송합니다.
3. 백엔드
    - AuthService에서 요청 데이터를 수신하고, 다시 한번 이메일/닉네임 중복 여부를 데이터베이스에서 확인합니다.
    - 비밀번호를 안전하게 저장하기 위해 BCrypt와 같은 해시 함수로 암호화합니다.
    - User 및 UserDetail 엔티티를 생성하여 데이터베이스에 저장합니다.
    - 처리 결과를 프론트엔드에 반환합니다.
4. 프론트엔드
   가입 성공 시, 로그인 페이지로 안내하거나 "가입 완료" 메시지를 표시합니다.

3.2.2.2. 사용자 로그인

3.2.2.2.1. 일반 로그인

1. 사용자
   로그인 페이지에서 이메일과 비밀번호를 입력합니다.
2. 프론트엔드
   POST /api/auth/login API로 사용자 입력 정보를 백엔드에 전송합니다.
3. 백엔드
    - AuthService는 전달받은 이메일로 데이터베이스에서 사용자를 조회합니다.
    - 사용자가 존재하면, 입력된 비밀번호와 데이터베이스에 저장된 암호화된 비밀번호를 비교합니다.
    - 인증에 성공하면, JwtUtil을 통해 Access Token과 Refresh Token을 생성합니다.
    - 생성된 토큰을 LoginResponse에 담아 프론트엔드로 전송합니다.

4. 프론트엔드
    - 서버로부터 받은 JWT 토큰을 브라우저의 안전한 공간(예: localStorage, HttpOnly 쿠키)에 저장합니다.
    - 이후 API 요청 시, HTTP 헤더에 Access Token을 포함하여 인증을 증명합니다.

3.2.2.2.2. 카카오 소셜 로그인

1. 사용자
   로그인 페이지에서 '카카오로 시작하기' 버튼을 클릭합니다.
2. 프론트엔드
   사용자를 카카오 인증 페이지로 리디렉션시킵니다.
3. 카카오 서버 & 사용자
   사용자가 카카오 계정으로 로그인하고 정보 제공에 동의합니다.
4. 카카오 서버
   인증이 완료되면, 미리 지정된 백엔드의 콜백 URI (/api/auth/kakao/callback)로 인가 코드와 함께 사용자를 리디렉션시킵니다.
5. 백엔드 (KakaoAuthService)
    - 전달받은 인가 코드를 사용하여 카카오 API 서버에 액세스 토큰을 요청합니다.
    - 발급받은 액세스 토큰으로 다시 카카오 API 서버에 사용자 정보(이메일, 고유 ID 등)를 요청합니다.
    - 획득한 이메일로 우리 서비스의 데이터베이스를 조회하여 이미 가입된 사용자인지 확인합니다.
6. 백엔드 (분기 처리)
    a. case: 기존 회원
        - 해당 사용자에 대한 우리 서비스의 Access Token과 Refresh Token을 생성합니다. 그 후, 토큰들을 쿼리 스트링에 담아 프론트엔드의 최종 로그인 처리 페이지 (/auth/callback)로 리디렉션시킵니다.
    b. case: 신규 회원
        - 회원가입을 마저 진행하기 위해, 카카오로부터 받은 이메일과 플랫폼 고유 ID를 쿼리 스트링에 담아 프론트엔드의 회원가입 페이지(/register)로 리디렉션시킵니다.

7. 프론트엔드
    - /auth/callback: 리디렉션된 URL에서 토큰 정보를 추출하여 저장하고, 로그인 상태를 활성화한 후 메인 페이지로 이동합니다.
    - /register: 리디렉션된 URL에서 이메일과 플랫폼 정보를 추출하여 회원가입 폼에 자동으로 채워주고, 사용자에게 추가 정보(닉네임, 역할 등)를 입력받아 최종 회원가입을 진행합니다.

3.3. 콘텐츠 (Content)

3.3.1. 기능 설명

콘텐츠 기능은 크리에이터가 자신의 창작물(글, 이미지 등)을 게시하고 관리하는 핵심 기능입니다. 게시된 콘텐츠는 사용자가 권한에 따라 조회하고 상호작용할 수 있도록 제공됩니다.

- **콘텐츠 작성**: 크리에이터는 직관적인 UI를 통해 제목, 본문, 썸네일 이미지, 첨부 파일, 접근 권한(전체 공개/구독자 전용) 등을 설정하여 콘텐츠를 생성할 수 있습니다.
    - **카테고리 설정**: 콘텐츠 작성 시, 미리 정의된 카테고리 중에서 해당 콘텐츠에 가장 적합한 카테고리를 직접 설정할 수 있습니다.
    - **썸네일 이미지**: 썸네일 이미지는 드래그 앤 드롭 방식으로 쉽게 업로드할 수 있습니다.
    - **본문 내용**: 본문 내용은 Toast UI Editor를 활용하여 마크다운 형식으로 작성되며, 다양한 툴바 기능을 통해 서식 지정이 가능합니다. Toast UI Editor를 통해 본문에 삽입되는 이미지는 ImageController를 통해 별도로 관리됩니다.
- **콘텐츠 관리**: 크리에이터는 자신이 생성한 콘텐츠를 수정하거나 삭제할 수 있습니다.
- **콘텐츠 조회**: 사용자는 콘텐츠 목록에서 원하는 콘텐츠를 선택하여 상세 내용을 조회할 수 있으며, 접근 권한에 따라 콘텐츠 열람이 제한될 수 있습니다.

3.3.2. 주요 처리 흐름

3.3.2.1. 콘텐츠 생성

1. 크리에이터
   콘텐츠 작성 페이지에 접속하여 제목, 본문(Toast UI Editor 사용), 썸네일 이미지(드래그 앤 드롭), 첨부 파일, 접근 권한(전체 공개/구독자 전용) 등을 설정합니다.
   콘텐츠에 적합한 카테고리를 선택합니다.
   Toast UI Editor를 통해 본문에 이미지를 삽입하는 경우, 해당 이미지는 ImageController를 통해 별도로 업로드 및 관리됩니다.
   모든 정보 입력 후 '저장' 버튼을 클릭합니다.

2. 프론트엔드
   입력된 데이터와 파일을 multipart/form-data 형식으로 POST /api/content API를 통해 백엔드에 전송합니다.
3. 백엔드
   - ContentController가 요청을 받아 ContentService에 처리를 위임합니다.
   - FileStorageService를 통해 썸네일 이미지와 첨부 파일을 AWS S3에 업로드하고, 해당 파일에 접근할 수 있는 URL을 반환받습니다.
   - 콘텐츠의 텍스트 데이터와 S3 파일 URL을 Content 엔티티에 담아 데이터베이스에 저장합니다.
   - 생성된 콘텐츠의 정보를 프론트엔드에 반환합니다.
4. 프론트엔드
   응답을 받으면, 생성된 콘텐츠의 상세 페이지로 사용자를 이동시킵니다.

3.3.2.2. 콘텐츠 상세 조회

1. 사용자
   콘텐츠 목록에서 특정 콘텐츠를 클릭합니다.
2. 프론트엔드

GET /api/content/{contentId} API를 호출하여 백엔드에 콘텐츠 데이터를 요청합니다.

3. 백엔드
    - ContentService는 요청된 contentId로 데이터베이스에서 콘텐츠를 조회합니다.
    - 콘텐츠의 접근 권한(accessType)을 확인합니다.
        - 만약 '구독자 전용' 콘텐츠이고 현재 요청한 사용자가 구독자가 아니라면, 접근 거부 응답(403 Forbidden)을 반환합니다.
    - 접근 권한이 유효하면, 해당 콘텐츠의 조회수(viewCount)를 1 증가시킵니다.
    - 콘텐츠 상세 정보를 ContentResponse에 담아 프론트엔드에 반환합니다.
4. 프론트엔드
   서버로부터 받은 데이터로 콘텐츠 상세 페이지를 렌더링하여 사용자에게 보여줍니다.

3.3.2.3. 콘텐츠 수정

5. 사용자
   자신이 작성한 콘텐츠의 상세 페이지에서 '수정' 버튼을 클릭합니다. (이 버튼은 작성자 본인에게만 표시됩니다.)
   콘텐츠 수정 페이지에서 제목, 본문, 썸네일 이미지, 첨부 파일, 접근 권한, 카테고리 등을 변경합니다.
   본문 내용의 이미지 변경 시 ImageController를 통해 관리됩니다.
   '수정 완료' 버튼을 클릭합니다.
6. 프론트엔드
   수정된 데이터와 파일을 multipart/form-data 형식으로 PUT /api/content/{contentId} API를 통해 백엔드에 전송합니다.
7. 백엔드
    - ContentController는 요청을 받아 ContentService에 처리를 위임합니다.
    - 요청한 사용자가 해당 콘텐츠의 작성자인지 확인합니다. 작성자가 아닐 경우 접근 거부 응답(403 Forbidden)을 반환합니다.

- 새로운 썸네일 이미지나 첨부 파일이 업로드된 경우, FileStorageService를 통해 기존 파일을 S3에서 삭제하고 새 파일을 업로드한 후 새로운 URL을 반환받습니다.
- Content 엔티티(JPA 기반)를 업데이트하고 데이터베이스에 변경 사항을 반영합니다.
- 수정된 콘텐츠의 정보를 프론트엔드에 반환합니다

8. 프론트엔드
   백엔드로부터 성공적인 응답을 받으면, 수정된 콘텐츠의 상세 페이지를 다시 렌더링하여 사용자에게 보여줍니다.

3.3.2.4. 콘텐츠 삭제

9. 사용자
   자신이 작성한 콘텐츠의 상세 페이지에서 '삭제' 버튼을 클릭합니다. (이 버튼은 작성자 본인에게만 표시됩니다.)
   삭제 확인 모달에서 '확인' 버튼을 클릭하여 최종 삭제를 진행합니다.
10. 프론트엔드
    DELETE /api/content/{contentId} API를 호출하여 백엔드에 콘텐츠 삭제를 요청합니다.
11. 백엔드
    - ContentController는 요청을 받아 ContentService에 처리를 위임합니다.
    - 요청한 사용자가 해당 콘텐츠의 작성자인지 확인합니다. 작성자가 아닐 경우 접근 거부 응답(403 Forbidden)을 반환합니다.
    - 데이터베이스에서 해당 contentId의 Content 엔티티(JPA 기반)를 삭제합니다.
    - 관련된 썸네일 이미지 및 첨부 파일, 그리고 본문에 삽입된 이미지들을 AWS S3에서도 삭제합니다.
    - 성공적인 삭제 응답을 프론트엔드에 반환합니다.
12. 프론트엔드
    백엔드로부터 성공적인 응답을 받으면, 콘텐츠 목록 페이지 또는 메인 페이지로 사용자를 이동시킵니다.

3.4. 구독 및 결제 (Subscription & Payment)

3.4.1. 기능 설명

사용자가 특정 크리에이터를 유료로 구독하고, 정기적으로 결제가 이루어지도록 관리하는 기능입니다. 외부 결제 서비스(PortOne)와 연동하여 처리됩니다.

3.4.2. 주요 처리 흐름

3.4.2.1. 구독 신청 및 최초 결제

1. 사용자
   크리에이터 페이지에서 '구독' 버튼을 클릭하고, 결제 수단을 선택/등록한 후 결제를 진행합니다.
2. 프론트엔드
   - POST /api/subscriptions/pending API를 호출하여 구독 생성 준비를 요청합니다.
   - 백엔드로부터 받은 구독 정보(주문 ID 등)를 사용하여 PortOne 결제 모듈을 초기화하고, 사용자에게 결제 창을 띄웁니다.
3. 백엔드 (/pending 요청 처리)
   SubscriptionService는 creatorId와 subscriberId를 바탕으로 기존 구독 관계가 있는지 데이터베이스에서 조회하고 결제에 필요한 정보를 프론트엔드에 반환합니다.
   a. case: 이미 `ACTIVE` 상태인 구독 존재
      409 Conflict 오류를 반환하여 중복 구독을 방지합니다.
   b. case: ACTIVE가 아닌 상태의 구독 존재
      기존 구독 정보를 재사용하여 결제를 진행합니다.
   c. case: 기존 구독 없음
      새로운 Subscription 엔티티를 생성하고, 상태를 PENDING(결제 대기)으로 하여 데이터베이스에 저장합니다.
4. 사용자

PortOne 결제 창에서 결제를 완료합니다.

5. PortOne & 프론트엔드
   결제가 성공하면, PortOne은 결제 성공 데이터(거래 ID 등)를 프론트엔드에 반환합니다.
6. 프론트엔드
   ortOne으로부터 받은 결제 데이터를 POST /api/subscriptions/{subscriptionId}/activate API를 통해 백엔드에 전송하여 결제 완료를 알립니다.
7. 백엔드 (/activate 요청 처리)
   - SubscriptionService는 해당 구독의 상태를 ACTIVE(활성)으로 변경합니다.
   - Payment 엔티티를 생성하여 결제 정보를 데이터베이스에 기록합니다.
   - 크리에이터의 구독자 수(subscriberCount)를 업데이트합니다.
   - 최종 성공 응답을 프론트엔드에 보냅니다.
8. 프론트엔드
   구독이 완료되었음을 UI에 반영합니다.

3.4.2.2. 정기 구독 갱신 (자동 결제)

1. 스케줄러 (매일 자정)
   SubscriptionScheduler가 정기 결제 로직(SubscriptionBillingService)을 실행합니다.
2. 백엔드
   - 다음 결제일(nextPaymentAt)이 된 모든 ACTIVE 상태의 구독을 조회합니다.
   - 각 구독 건에 대해 등록된 결제 수단(PaymentMethod)으로 자동 결제를 시도합니다.
     a. case: 결제 성공
        구독의 lastPaymentAt을 현재로, nextPaymentAt을 한 달 뒤로 갱신하여 구독 기간을 연장합니다.
     b. case: 결제 실패
        구독 상태를 EXPIRED로 변경합니다.

3.4.2.3. 구독 해지

1. 사용자
   구독 관리 페이지 또는 구독한 크리에이터 메인 페이지에서 '해지하기' 버튼을 클릭합니다.
2. 프론트엔드
   PATCH /api/subscriptions/{subscriptionId} API를 호출하여 해지를 요청합니다.
3. 백엔드
   - SubscriptionService는 해당 구독의 상태를 ACTIVE에서 CANCELLED로 변경합니다.
   - 사용자는 현재 결제 주기가 끝나는 nextPaymentAt까지 서비스를 계속 이용할 수 있습니다.
   - 크리에이터의 구독자 수를 즉시 1 감소시킵니다.
4. 스케줄러 (이후 실행)
   nextPaymentAt의 날짜가 되었을 때, 스케줄러는 CANCELLED 상태인 이 구독을 최종적으로 EXPIRED로 변경하여 완전히 만료시킵니다.

3.5. 검색 (Search)

3.5.1. 기능 설명

사용자가 키워드, 필터, 정렬 옵션을 사용하여 원하는 콘텐츠나 크리에이터를 효율적으로 찾을 수 있는 기능입니다. 통합 검색, 콘텐츠만 검색, 크리에이터만 검색하는 세 가지 모드를 제공합니다.

3.5.2. 주요 처리 흐름

1. 사용자

검색창에 키워드를 입력하고, 필요한 경우 필터(예: 유/무료, 카테고리)와 정렬(예: 최신순, 좋아요순) 옵션을 선택한 후 검색을 실행합니다.

2. 프론트엔드

사용자의 입력을 조합하여 적절한 검색 API (/api/search, /api/contents, 또는 /api/creators)를 호출합니다.

3. 백엔드 (SearchService)

- JPA의 Specification을 사용하여 사용자의 요청에 맞는 동적 쿼리를 생성합니다.
- 검색 모드에 따라 콘텐츠의 제목/내용, 크리에이터의 닉네임/직업 등 다양한 필드를 대상으로 검색을 수행합니다.
- 요청된 필터와 정렬 조건을 쿼리에 적용합니다.
- 데이터베이스로부터 페이징된 검색 결과를 받아 SearchResponse DTO로 가공하여 프론트엔드에 반환합니다.

4. 프론트엔드

서버로부터 받은 검색 결과를 페이지에 렌더링하여 사용자에게 보여줍니다.

3.6. 프로필 관리 (Profile Management)

3.6.1. 기능 설명

사용자가 자신의 프로필(닉네임, 자기소개, 프로필 이미지 등)을 조회하고 수정하는 기능입니다. 또한 다른 사용자의 공개 프로필을 조회할 수 있습니다.

3.6.2. 주요 처리 흐름

1. 사용자

'마이페이지' 또는 '프로필 설정'으로 이동하여 '수정하기' 버튼을 클릭합니다.

2. 프론트엔드

GET /api/user/me API를 호출하여 현재 로그인된 사용자의 프로필 정보를 가져와 화면에 표시합니다.

3. 사용자

프로필 이미지, 닉네임, 자기소개 등의 정보를 수정한 후 '저장' 버튼을 클릭합니다.

4. 프론트엔드

- 만약 프로필 이미지가 변경되었다면, 먼저 이미지를 서버에 업로드(POST /api/upload/profile-image)하고 반환된 이미지 URL을 확보합니다.
- 변경된 텍스트 정보와 새로운 이미지 URL을 UpdateProfileRequest에 담아 PUT /api/user/me API로 전송합니다.

5. 백엔드 (UserService)

- 요청 데이터를 수신하고, 필요한 경우 닉네임 중복과 같은 유효성 검사를 수행합니다.
- 데이터베이스에서 해당 사용자의 UserDetail 정보를 찾아 업데이트합니다.

6. 프론트엔드

성공적으로 업데이트되었다는 확인 메시지를 사용자에게 보여주고, 화면의 프로필 정보를 갱신합니다.

3.7. 결제 수단 관리 (Payment Method Management)

3.7.1. 기능 설명

사용자가 정기 구독 결제에 사용할 신용카드나 체크카드를 등록, 조회, 삭제하는 기능입니다. 외부 결제 대행 서비스인 PortOne(아임포트)과 연동하여 안전하게 처리됩니다.

3.7.2. 주요 처리 흐름

3.7.2.1. 결제 수단 등록

1. 사용자

'결제 수단 관리' 페이지에서 '새 카드 추가' 버튼을 클릭합니다

2. 프론트엔드
   PortOne의 결제 모듈을 호출하여 사용자에게 카드 정보를 입력받는 UI를 제공합니다. 카드 정보는 우리 서버를 거치지 않고 PortOne으로 직접 전송됩니다.
3. PortOne & 프론트엔드
   PortOne은 카드 정보의 유효성을 검증한 후, 해당 카드를 식별할 수 있는 고유한 빌링키(billing_key)를 생성하여 프론트엔드에 반환합니다.
4. 프론트엔드
   전달받은 빌링키를 POST /api/billings/card API를 통해 백엔드 서버로 전송합니다.
5. 백엔드 (PaymentMethodService)
   - 빌링키를 사용하여 PortOne API를 다시 호출함으로써, 해당 카드의 메타데이터(카드사, 카드 종류, 마스킹된 번호 등)를 안전하게 조회합니다.
   - 조회된 카드 정보와 빌링키를 우리 서비스의 PaymentMethod 테이블에 저장하여 사용자와 연결합니다.
6. 프론트엔드
   성공 응답을 받으면, 사용자의 결제 수단 목록에 새로 등록된 카드를 표시합니다.

3.7.2.2. 결제 수단 삭제

1. 사용자
   결제 수단 목록에서 삭제하고 싶은 카드의 '삭제' 버튼을 클릭합니다.
2. 프론트엔드
   POST /api/billings/keys API를 통해 삭제할 카드의 paymentMethodId를 백엔드로 전송합니다.
3. 백엔드 (PaymentMethodService)
   - 먼저, 삭제하려는 카드가 현재 ACTIVE 상태인 구독에 연결되어 있는지 확인합니다. 만약 그렇다면, "활성화된 구독이 있어 삭제할 수 없습니다"라는 오류를 반환하여 결제 문제를 예방합니다.
   - 활성 구독이 없다면, PortOne API를 호출하여 PortOne 서버에서도 해당 빌링키를 영구히 삭제합니다.

- 우리 서비스의 PaymentMethod 테이블에서도 해당 데이터를 삭제합니다.

4. 프론트엔드
   성공적으로 삭제되었다는 응답을 받으면, UI에서 해당 카드를 제거합니다.

3.8. 크리에이터 공지방 (Creator Notice)

3.8.1. 기능 설명

크리에이터가 자신의 구독자들에게 새로운 소식이나 공지를 전달하는 기능입니다. 사용자들은 공지사항을 조회하고 이모지(emoji)로 반응을 남길 수 있습니다.

3.8.2. 주요 처리 흐름

3.8.2.1. 공지사항 생성

1. 크리에이터
   자신의 관리 페이지에서 '공지사항 작성'을 선택하고, 제목과 내용을 입력한 후 게시합니다.
2. 프론트엔드
   POST /api/creators/{creatorId}/notices API를 호출하여 공지사항 데이터를 백엔드에 전송합니다.
3. 백엔드
   Notice 엔티티를 생성하여 데이터베이스에 저장합니다.
4. 프론트엔드
   성공 응답을 받으면, 공지사항 목록 페이지로 이동하거나 성공 메시지를 표시합니다.

3.8.2.2. 공지사항 조회

1. 사용자
   크리에이터의 프로필 페이지나 공지사항 목록 페이지에 방문합니다.
2. 프론트엔드
   GET /api/creators/{creatorId}/notices API를 호출하여 해당 크리에이터의 공지사항 목록을 받아와 표시합니다.
3. 사용자
   특정 공지사항을 클릭하여 상세 내용을 확인합니다.
4. 프론트엔드
   GET /api/creators/{creatorId}/notices/{noticeId} API를 호출하여 공지사항 상세 내용을 받아와 표시합니다.

3.8.2.3. 공지사항 수정

1. 크리에이터
   특정 공지사항의 '수정' 버튼을 클릭합니다.
2. 프론트엔드
   기존 공지사항 내용을 불러와 수정 폼에 채워줍니다.
3. 크리에이터
   내용을 수정한 후 '저장' 버튼을 클릭합니다.
4. 프론트엔드
   PUT /api/creators/{creatorId}/notices/{noticeId} API를 호출하여 수정된 데이터를 백엔드에 전송합니다.
5. 백엔드
   해당 Notice 엔티티를 찾아 업데이트합니다.

3.8.2.4. 공지사항 삭제

1. 크리에이터
   특정 공지사항의 '삭제' 버튼을 클릭합니다.
2. 프론트엔드
   사용자에게 삭제 확인을 받은 후, DELETE /api/creators/{creatorId}/notices/{noticeId} API를 호출합니다.
3. 백엔드

해당 Notice 엔티티를 데이터베이스에서 삭제합니다.

3.8.2.5. 공지사항에 반응하기 (Emoji Reaction)

1. 사용자
   특정 공지사항을 읽고, 마음에 드는 이모지 버튼을 클릭합니다.
2. 프론트엔드
   POST /api/notices/{noticeId}/reactions?emoji={emoji} API를 호출합니다.
3. 백엔드 (NoticeReactionService)
   - 해당 사용자가 해당 공지사항에 동일한 이모지로 이미 반응했는지 확인합니다.
     a. case: 기존 반응 있음
        기존 NoticeReaction 데이터를 삭제합니다. (토글 OFF)
     b. case: 기존 반응 없음
        새로운 NoticeReaction 데이터를 생성하여 저장합니다. (토글 ON)
4. 프론트엔드
   API 응답에 따라, UI의 이모지 카운트를 업데이트하고 버튼의 활성화 상태를 변경합니다.

3.9. 댓글 및 대댓글 (Comments)

3.9.1. 기능 설명

사용자들이 콘텐츠에 대해 의견을 나누고 소통하는 기능입니다. 대댓글을 지원하여 특정 댓글에 대한 논의를 이어갈 수 있습니다.

3.9.2. 주요 처리 흐름

3.9.2.1. 댓글 목록 조회

1. 사용자
   특정 콘텐츠의 상세 페이지에 접속하여 댓글 섹션을 확인합니다.
2. 프론트엔드
   페이지 로드 시 GET /api/contents/{contentId}/comments API를 호출하여 해당 콘텐츠의 댓글 목록을 요청합니다.
3. 백엔드 (CommentService)
   - 요청받은 `contentId`에 해당하는 댓글들을 데이터베이스에서 조회합니다.
   - 대댓글 관계를 고려하여 계층적으로 댓글 데이터를 구성합니다.
   - 각 댓글에 대한 좋아요 정보(현재 사용자의 좋아요 여부 포함)를 함께 조회하여 CommentResponseDto 형태로 반환합니다.
4. 프론트엔드
   서버로부터 받은 댓글 데이터를 화면에 렌더링하여 사용자에게 보여줍니다.

3.9.2.2. 댓글 작성

1. 사용자
   콘텐츠 하단의 댓글 입력창에 자신의 의견을 작성한 후 '등록' 버튼을 클릭합니다. (대댓글의 경우, 특정 댓글의 '답글' 버튼 클릭 후 입력)
2. 프론트엔드
   POST /api/contents/{contentId}/comments API를 호출하여 댓글 내용을 서버에 전송합니다. (대댓글의 경우, parent_comment_id를 함께 전송합니다.)
3. 백엔드 (CommentService)
   - 요청받은 데이터를 Comment 엔티티로 변환하여 데이터베이스에 저장합니다.
   - 관련된 콘텐츠의 댓글 수(commentCount)를 1 증가시킵니다.
4. 프론트엔드
   성공적으로 등록되면, 댓글 목록을 실시간으로 갱신하여 방금 작성한 댓글을 사용자에게 보여줍니다.

3.9.2.3. 댓글 수정

1. 사용자
   자신이 작성한 댓글 옆의 '수정' 버튼을 클릭합니다.
2. 프론트엔드
   댓글 입력 필드를 활성화하고 기존 내용을 채워줍니다.
3. 사용자
   내용을 수정한 후 '저장' 버튼을 클릭합니다.
4. 프론트엔드
   PUT /api/contents/{contentId}/comments/{commentId} API를 호출하여 수정된 댓글 내용을 백엔드에 전송합니다.
5. 백엔드 (CommentService)
   - 요청받은 commentId에 해당하는 댓글을 조회하고, 요청한 userId가 해당 댓글의 작성자인지 권한을 확인합니다.
   - 권한이 유효하면 댓글 내용을 업데이트합니다.
6. 프론트엔드
   성공적으로 수정되면, 해당 댓글을 화면에서 갱신하여 보여줍니다.

3.9.2.4. 댓글 삭제

1. 사용자
   자신이 작성한 댓글 옆의 '삭제' 버튼을 클릭합니다.
2. 프론트엔드
   사용자에게 삭제 확인을 받은 후, DELETE /api/contents/{contentId}/comments/{commentId} API를 호출합니다.
3. 백엔드 (CommentService)
   - 요청받은 commentId에 해당하는 댓글을 조회하고, 요청한 userId가 해당 댓글의 작성자인지 권한을 확인합니다.
   - 권한이 유효하면 댓글의 is_deleted 상태를 true로 변경합니다. (실제 데이터 삭제 대신 논리적 삭제 처리)
   - 관련된 콘텐츠의 댓글 수(commentCount)를 1 감소시킵니다.
4. 프론트엔드

성공적으로 삭제되면, 해당 댓글의 내용은 '삭제된 댓글입니다'라는 메시지로 대체되고, 관련 액션 버튼(좋아요, 답글, 수정, 삭제)은 숨겨집니다.

3.10. 전역 예외 처리 (Global Error Handling)

3.10.1. 기능 설명

백엔드에서 발생하는 모든 종류의 오류(잘못된 요청, 서버 내부 문제, 권한 없음 등)를 일관된 형식으로 처리하여 프론트엔드에 예측 가능한 오류 응답을 제공하는 아키텍처 구성 요소입니다.

3.10.2. 주요 처리 흐름

1. 오류 발생

백엔드 로직 수행 중 CustomException, IllegalArgumentException 등 다양한 예외가 발생합니다.

2. 예외 감지 (@RestControllerAdvice)

GlobalExceptionHandler가 발생한 예외를 감지(catch)합니다.

3. 오류 응답 생성

감지된 예외의 종류에 따라 적절한 HTTP 상태 코드(예: 400, 403, 500)와 정해진 형식의 ErrorResponse DTO(에러 발생 시각, 상태 코드, 메시지 등)를 생성합니다.

4. 프론트엔드

서버로부터 일관된 형식의 오류 응답을 수신하므로, 사용자에게 상황에 맞는 알림(예: "입력값이 올바르지 않습니다", "권한이 없습니다")을 안정적으로 표시할 수 있습니다.

3.11. 좋아요한 콘텐츠 목록 (Liked Content List)

3.11.1. 기능 설명

사용자가 '좋아요'를 누른 모든 콘텐츠를 한 곳에서 모아보고 관리할 수 있는 기능입니다. 사용자의 관심사를 반영하고 콘텐츠 재탐색을 돕습니다.

3.11.2. 주요 처리 흐름

1. 사용자

마이페이지에서 '좋아요한 콘텐츠' 메뉴를 클릭합니다.

2. 프론트엔드

GET /api/content/liked API를 호출하여 현재 로그인된 사용자가 좋아요를 누른 콘텐츠 목록을 요청합니다. 이때 페이징 및 정렬 옵션을 함께 전달할 수 있습니다.

3. 백엔드 (ContentService)

- 요청받은 userId를 기반으로 ContentLike 테이블을 조회하여 해당 사용자가 좋아요를 누른 Content 목록을 가져옵니다.
- 페이징 및 정렬 조건을 적용하여 결과를 필터링합니다.
- 조회된 콘텐츠 데이터를 ContentList DTO 형태로 가공하여 프론트엔드에 반환합니다.

4. 프론트엔드

서버로부터 받은 데이터를 기반으로 '좋아요한 콘텐츠' 목록을 페이지에 렌더링하여 사용자에게 보여줍니다.

3.12. 초기 데이터 로딩 (Initial Data Loading)

3.12.1. 기능 설명

애플리케이션이 시작될 때, 서비스 운영에 필요한 기본 데이터를 데이터베이스에 자동으로 삽입하는 기능입니다. 예를 들어, 크리에이터의 '직업' 목록과 같이 미리 정의되어야 하는 데이터에 사용됩니다.

3.12.2. 주요 처리 흐름

1. 애플리케이션 시작

Spring Boot 애플리케이션이 구동됩니다.

2. 데이터 로더 실행

@Component로 등록된 JobDataLoader와 같이 ApplicationRunner 또는 CommandLineRunner 인터페이스를 구현한 클래스가 자동으로 실행됩니다.

3. 데이터 확인 및 삽입

JobDataLoader는 데이터베이스에 특정 데이터(예: 직업 목록)가 이미 존재하는지 확인합니다.

- case: 데이터 없음
  미리 정의된 데이터를 데이터베이스에 삽입합니다.
- case: 데이터 있음
  추가적인 작업을 수행하지 않고 종료합니다.

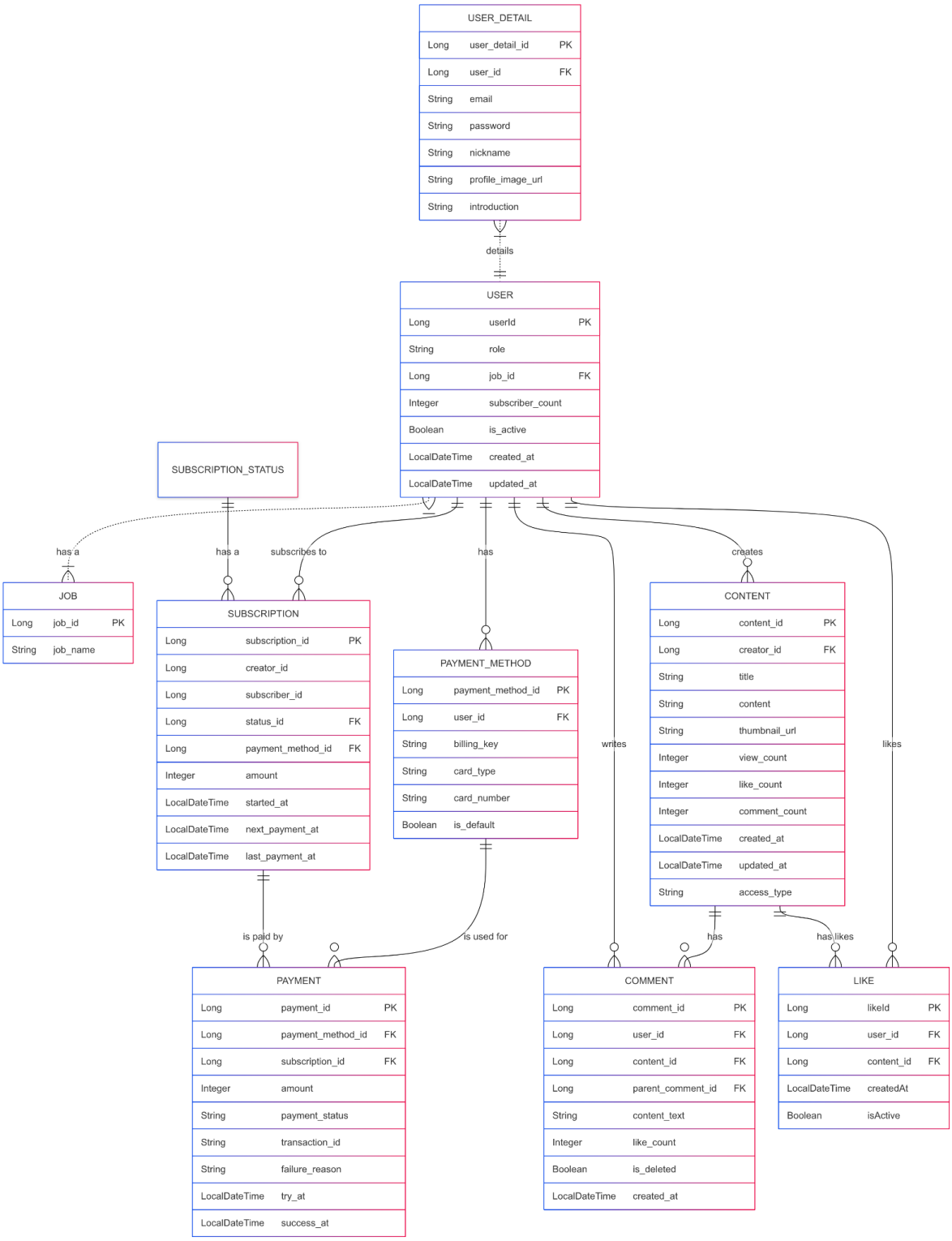
4. 애플리케이션 정상 작동

초기 데이터가 준비된 상태에서 애플리케이션의 다른 서비스들이 정상적으로 동작합니다.

4. 데이터베이스 설계

04_DBㅁㅕㅇㅅㅖㅅㅓ[2ㅈㅗ]_v.1.4.docx

4.1. ERD


USER_DETAIL
Long user_detail_id PK
Long user_id FK
String email
String password
String nickname
String profile_image_url
String introduction
details
USER
Long userId PK
String role
Long job_id FK
Integer subscriber_count
Boolean is_active
LocalDateTime created_at
LocalDateTime updated_at
SUBSCRIPTION_STATUS
has a
has a
subscribes to
has
writes
creates
likes
JOB
Long job_id PK
String job_name
SUBSCRIPTION
Long subscription_id PK
Long creator_id
Long subscriber_id
Long status_id FK
Long payment_method_id FK
Integer amount
LocalDateTime started_at
LocalDateTime next_payment_at
LocalDateTime last_payment_at
PAYMENT_METHOD
Long payment_method_id PK
Long user_id FK
String billing_key
String card_type
String card_number
Boolean is_default
CONTENT
Long content_id PK
Long creator_id FK
String title
String content
String thumbnail_url
Integer view_count
Integer like_count
Integer comment_count
LocalDateTime created_at
LocalDateTime updated_at
String access_type
is paid by
is used for
has
has likes
PAYMENT
Long payment_id PK
Long payment_method_id FK
Long subscription_id FK
Integer amount
String payment_status
String transaction_id
String failure_reason
LocalDateTime try_at
LocalDateTime success_at
COMMENT
Long comment_id PK
Long user_id FK
Long content_id FK
Long parent_comment_id FK
String content_text
Integer like_count
Boolean is_deleted
LocalDateTime created_at
LIKE
Long likeId PK
Long user_id FK
Long content_id FK
LocalDateTime createdAt
Boolean isActive

4.2. 주요 테이블

4.3. users

- Primary key: user_id
- Foreign key: job_id

Name	Attribute	Type	Value	Description
user ID	user_id	BIGINT	NOT NULL AUTO_INCREMENT PRIMARY KEY	회원 고유 식별자
role	role	ENUM('USER','CREATOR')	DEFAULT NULL	사용자 권한 유형
Job id	job_id	VARCHAR(50)	NOT NULL	선택된 직업
Subscriber count	subscriber_count	INT	NOT NULL	창작자의 구독자 수
Is active	is_active	BOOLEAN	NOT NULL	회원 탈퇴 여부 (TRUE=활동중)
Created at	created_at	DATETIME	DEFAULT NULL	회원가입 날짜
Updated at	updated_at	DATETIME	DEFAULT NULL	회원정보 마지막 수정일

4.4. user_detail

- Primary key: user_id
- Foreign key: user_id

Name	Attribute	Type	Value	Description
user ID	user_id	BIGINT	NOT NULL PRIMARY KEY	회원 고유 식별자
username	username	VARCHAR (50)	NOT NULL	사용자 실명
nickname	nickname	VARCHAR (50)	NOT NULL	사용자 닉네임
Platform ID	platform_id	VARCHAR (255)	DEFAULT NULL	소셜 플랫폼에서 제공한 외부 사용자ID

platform	platform	ENUM('LOCAL','KAKAO')	NOT NULL	가입 플랫폼 유형
email	email	VARCHAR(100)	NOT NULL UNIQUE	사용자 이메일
password	password	VARCHAR(512)	DEFAULT NULL	사용자 비밀번호(암호화 저장)
Profile image	profile_image_url	VARCHAR(255)	NOT NULL	프로필 이미지 URL
bio	bio	TEXT	NOT NULL	사용자 자기소개

4.5. job

- Primary key: job_id

Name	Attribute	Type	Value	Description
Job ID	job_id	INT	NOT NULL AUTO_INCREMENT PRIMARY KEY	직업 고유 식별자
name	name	VARCHAR(50)	NOT NULL UNIQUE	직업 이름

4.6. token

- Primary key: token_id
- Foreign key: user_id

Name	Attribute	Type	Value	Description
Token ID	token_id	BIGINT	NOT NULL AUTO_INCREMENT PRIMARY KEY	토큰 고유 식별자
User ID	user_id	BIGINT	NOT NULL	회원 고유 식별자

Refresh token	refresh_token	VARCHAR (512)	NOT NULL UNIQUE	리프레시 토큰
Issued at	issued_at	DATETIME	NOT NULL DEFAULT CURRENT_TIMESTAMP	토큰 발급 일시
Expire at	expire_at	DATETIME	NOT NULL	토큰 만료 일시
Is revoked	is_revoked	BOOLEAN	NOT NULL DEFAULT FALSE	토큰 기간 만료 체크

4.7. content

- Primary key: content_id
- Foreign key: creator_id

Name	Attribute	Type	Value	Description
Content ID	content_id	BIGINT	NOT NULL AUTO_INCREMENT PRIMARY KEY	콘텐츠 고유 식별자
Creator ID	creator_id	BIGINT	NOT NULL	창작자 고유 식별자
title	title	VARCHAR (255)	NOT NULL	콘텐츠 제목
content	content	TEXT	NOT NULL	콘텐츠 본문
thumbnail_URL	thumbnail_url	VARCHAR (512)	NOT NULL	썸네일 이미지 주소
View count	view_count	INT	NOT NULL DEFAULT 0	조회수
Like count	like_count	INT	NOT NULL DEFAULT 0	좋아요 수
Comment count	comment_count	INT	NOT NULL DEFAULT CURRENT_TIMESTAMP	댓글 수
Created at	created_at	DATETIME	NOT NULL DEFAULT CURRENT_TIMESTAMP	작성일시
Updated at	updated_at	DATETIME	NULL	수정일시
Access type	access_type	ENUM('FREE','SUBSC	NOT NULL DEFAULT 'FREE'	공개/유료 구분

		RIBER)		

4.8. attachment

- Primary key: attachment_id
- Foreign key: content_id

Name	Attribute	Type	Value	Description
Attachment ID	attachment_id	BIGINT	NOT NULL AUTO_INCREMENT PRIMARY KEY	첨부파일 고유 ID
Content ID	content_id	BIGINT	NOT NULL	콘텐츠 고유 ID
FIle URL	file_url	VARCHAR (512)	NOT NULL	첨부파일 경로
Original file name	original_file_name	VARCHAR (100)	DEFAULT NULL	업로드된 원본 파일명
Stored file name	stored_file_name	VARCHAR (255)	NOT NULL	서버에 저장된 파일명 (UUID 등)
File type	file_type	VARCHAR (50)	NOT NULL	파일 MIME 타입
File size	file_size	BIGINT	DEFAULT NULL	파일 크기 (bytes)

4.9. category

- Primary key: category_id
-

Name	Attribute	Type	Value	Description
Category ID	category_id	INT	NOT NULL PRIMARY KEY AUTO_INCREMENT	카테고리 고유 ID
name	name	VARCHAR (50)	NOT NULL UNIQUE	카테고리명 (중복 불가)

4.10. content_category

- Primary key: content_id, category_id
- Foreign key: content_id, category_id

Name	Attribute	Type	Value	Description
Content ID	content_id	BIGINT	NOT NULL PRIMARY KEY	콘텐츠 고유 식별자
Category ID	category_id	INT	NOT NULL PRIMARY KEY	카테고리 고유 식별자

4.11. comment

- Primary key: comment_id
- Foreign key: user_id, content_id, parent_comment_id, root_comment_id

Name	Attribute	Type	Value	Description
Comment ID	comment_id	BIGINT	NOT NULL AUTO_INCREMENT PRIMARY KEY	댓글 고유 ID
User ID	user_id	BIGINT	NOT NULL	회원 고유 식별자
Content ID	content_id	BIGINT	NOT NULL	콘텐츠 고유 식별자
Parent comment ID	parent_comment_id	BIGINT	DEFAULT NULL	부모 댓글 ID (원댓글용)
Content	content	TEXT	NOT NULL	댓글 내용
Like count	like_count	INT	DEFAULT 0	댓글에 달린 좋아요 수
Created at	created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	댓글 작성 일시
delete	is_deleted	BOOLEAN	NOT NULL DEFAULT FALSE	댓글 삭제 여부

4.12. comment_like

- Primary key: like_id
- Foreign key: comment_id, user_id
- Unique key: (comment_id, user_id),

Name	Attribute	Type	Value	Description
Like ID	like_id	BIGINT	AUTO_INCREMENT PRIMARY KEY	댓글 좋아요 고유 식별자
Comment ID	comment_id	BIGINT	NOT NULL UNIQUE KEY	댓글 고유 식별자
User ID	user_id	BIGINT	NOT NULL UNIQUE KEY	회원 고유 식별자
Is active	is_active	BOOLEAN	NOT NULL DEFAULT TRUE	좋아요 취소시 FALSE

4.13. subscription

- Primary key: subscription_id
- Foreign key: subscriber_id, creator_id, payment_method_id, status_id
- Unique key: subscriber_id, creator_id

Name	Attribute	Type	Value	Description
Subscription ID	subscription_id	BIGINT	NOT NULL AUTO_INCREMENT PRIMARY KEY	구독 고유 식별자
Subscriber ID	subscriber_id	BIGINT	NOT NULL	회원 고유 식별자 (구독자)
Creator ID	creator_id	BIGINT	NOT NULL	회원 고유 식별자 (창작자)
Payment method id	payment_method_id	BIGINT	NOT NULL	결제 방법 고유 식별자
status	status_id	INT	NOT NULL DEFAULT 1	구독 상태 고유 식별자

Started at	started_at	DATETIME	NOT NULL	구독 시작일
Next payment at	next_payment_at	DATETIME	NOT NULL	다음 결제 예정일
Last payment at	last_payment_at	DATETIME	DEFAULT NULL	마지막 결제일

4.14. subscription_status

- Primary key: status_id

Name	Attribute	Type	Value	Description
Status ID	status_id	INT	PRIMARY KEY	구독 상태 고유 식별자
name	name	VARCHAR (20)	NOT NULL UNIQUE	구독 상태 이름 (PENDING','ACTIVE', 'CANCELLED', 'EXPIRED')
description	description	VARCHAR (100)	NULL	상태 설정 ('결제 대기 상태', '구독 활성 상태, '사용자에 의한 취소', '유효기간 만료')

4.15. notice

- Primary key: notice_id
- Foreign key: creator_id

Name	Attribute	Type	Value	Description
Notice ID	notice_id	BIGINT	NOT NULL AUTO_INCREMENT PRIMARY KEY	공지 고유 ID
Creator ID	creator_id	BIGINT	NOT NULL	회원 고유 식별자 (창작자)
content	content	TEXT	NOT NULL	공지 내용
Image URL	image_url	VARCHAR(512)	DEFAULT NULL	첨부 이미지 경로

Created at	created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	작성 일시

4.16. notice_reaction

- Primary key: notice_reaction_id
- Foreign key: notice_id, user_id
- Unique key: notice_id, user_id, emoji
-

Name	Attribute	Type	Value	Description
Notice reaction ID	notice_reaction_id	BIGINT	AUTO_INCREMENT PRIMARY KEY	공지 리액션 고유 식별자
Notice ID	notice_id	BIGINT	NOT NULL	공지 고유 식별자
User ID	user_id	BIGINT	NOT NULL	회원 고유 식별자
emoji	emoji	VARCHAR(10)	NOT NULL	반응한 이모지 코드

4.17. payment

- Primary key: payment_id
- Foreign key: subscription_id, payment_method_id
- Unique key: transaction_id

Name	Attribute	Type	Value	Description
Payment ID	payment_id	BIGINT	NOT NULL AUTO_INCREMENT PRIMARY KEY	결제 내역 고유 식별자
Payment mehtod id	payment_method_id	BIGINT	NOT NULL	결제 수단 고유 식별자
Subscription ID	subscription_id	BIGINT	NOT NULL	구독 고유 식별자
amount	amount	INT	NOT NULL	결제 금액

Payment status	payment_status	ENUM ('PENDING', 'SUCCESS', 'FAILED')	NOT NULL	결제 상태
Transaction ID	transaction_id	VARCHAR(255)	DEFAULT NULL	PG사 결제 트랜잭션 고유 ID
Failure reason	failure_reason	TEXT	DEFAULT NULL	실패 사유
Try at	try_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	결제 시도 시각
Success at	success_at	DATETIME	DEFAULT NULL	결제 완료 시각 (실패되었을 경우 NULL)

4.18. payment_method

- Primary key: payment_method_id
- Foreign key: user_id
-

Name	Attribute	Type	Value	Description
Payment method	payment_method_id	BIGINT	NOT NULL AUTO_INCREMENT PRIMARY KEY	결제 수단 고유 식별자
User id	user_id	BIGINT	NOT NULL	결제 수단을 등록한 회원 고유 식별자
billing key	billing_key	VARCHAR (255)	NULLABLE	PG사에서 발급한 정기결제 식별자
Card name	card_name	VARCHAR (100)	NOT NULL	카드사 이름 (삼성, 신한 등)
Card number	card_number	VARCHAR (50)	NOT NULL	카드 번호
Card type	card_type	VARCHAR (50)	DEFAULT NULL	카드 타입 (신용, 체크 등)
Card brand	card_brand	VARCHAR (50)	DEFAULT NULL	카드 브랜드 (MASTER, VISA 등)
Created at	created_at	DATETIME	NOT NULL DEFAULT CURRENT_TIMESTAMP	결제수단 등록 시각

4.19. like

- Primary key: like_id
- Foreign key: user_id, content_id
- Unique key: user_id, content_id

Name	Attribute	Type	Value	Description
like ID	like_id	BIGINT	NOT NULL AUTO_INCREMENT PRIMARY KEY	좋아요 고유 ID
User ID	user_id	BIGINT	NOT NULL	회원 고유 식별자
Content ID	content_id	BIGINT	NOT NULL	콘텐츠 고유 식별자
Created at	created_at	DATETIME	DEFAULT CURRENT_TIMESTAMP	좋아요 누른 시각
Is active	is_active	BOOLEAN	NOT NULL DEFAULT TRUE	좋아요 취소시 FALSE
Unique key	uq_user_content	UNIQUE KEY	(user_id, content_id)	동일 유저가 동일 콘텐츠에 중복 좋아요 방지
index	idx_content_user	INDEX	(user_id, is_active, created_at)	회원별 좋아요 목록 조회 최적화

5. API 명세

5.1. 인증 및 사용자 관리 API

기능	Endpoint	Method	Description	Request Body	Query Parameter	Response
회원가입	/api/user	POST	새로운 사용자를 등록합니다.	Signup Reques	-	SignupResponse

				t		
이메일 중복 확인	/api/user/check-email	GET	이메일 중복 여부를 확인합니다.	-	email (String)	EmailCheckResponse
닉네임 중복 확인	/api/user/check-nickname	GET	닉네임 중복 여부를 확인합니다.	-	nickname (String)	NicknameCheckResponse
사용자 탈퇴	/api/user/me	DELETE	현재 로그인된 사용자를 탈퇴 처리합니다.	-	-	200 OK
내 프로필 조회	/api/user/me	GET	현재 로그인된 사용자의 프로필 정보를 조회합니다.	-	-	UserProfileResponse
내 프로필 수정	/api/user/me	PUT	현재 로그인된 사용자의 프로필 정보를 수정합니다.	UpdateProfileRequest	-	200 OK
공개 프로필 조회	/api/user/public/{nickname}	GET	특정 사용자의 공개 프로필 정보를 조회합니다.	-	nickname (String)	PublicProfileResponse
크리에이터 정보 조회	/api/user/public/creator-manage	GET	현재 로그인된 크리에이터의 관리 정보를 조회합니다.	-	-	SubscriptionCountResponseDto
로그인	/api/auth/login	POST	사용자 로그인을 처리하고 JWT 토큰을 발급합니다.	LoginRequest	-	LoginResponse
토큰 갱신	/api/auth/refresh	POST	Refresh Token을 사용하여 Access Token을 갱신합니다.	TokenRefreshRequest	-	TokenRefreshResponse
로그아웃	/api/auth/logout	POST	현재 로그인된 사용자를 로그아웃 처리합니다.	LogoutRequest	-	LogoutResponse
카카오 로그인 콜백	/api/auth/kakao/callback	GET	카카오 서버로부터 인가 코드를 받아 로그인을 처리합니다. 최종적으로 프론트엔드의 특정 URL로	-	code (String)	302 Found (리디렉션) - (기존 회원) http://localhost:5173/auth/callback?accessToken=...&refreshToken=.

			리디렉션됩니다.			.. - (신규 회원) http://localhost:5173/register?email=...&platform=KAKAO&platformId=...
직업 목록 조회	/api/jobs	GET	회원가입 또는 프로필 수정 시 사용할 직업 목록 전체를 조회합니다.	-	-	JobResponse

5.2. 콘텐츠 API

기능	Endpoint	Method	Description	Request Body	Query Parameter	Response
콘텐츠 생성	/api/content	POST	새로운 콘텐츠를 생성합니다.	ContentCreateRequest (multipart/form-data)	-	ContentResponse
내 콘텐츠 목록 조회	/api/content/my	GET	로그인한 사용자의 콘텐츠 목록을 조회합니다.	-	-	List<ContentResponse>
내 조회수 TOP3 콘텐츠 조회	/api/content/my/top-viewed	GET	로그인한 사용자의 조회수 상위 3개 콘텐츠를 조회합니다.	-	-	List<ContentResponse>
특정 콘텐츠 조회	/api/content/{contentId}	GET	특정 콘텐츠의 상세 정보를 조회합니다.	-	contentId (Long)	ContentResponse
특정 크리에이터 콘텐츠 목록 조회	/api/content/creator/{creatorId}	GET	특정 크리에이터의 콘텐츠 목록을 조회합니다.	-	creatorId (Long), exclud	List<ContentResponse>

					eContentId (Long, optional)	
콘텐츠 수정	/api/content/{contentId}	PUT	특정 콘텐츠를 수정합니다.	ContentUpdateRequest (multipart/form-data)	contentId (Long)	ContentResponse
콘텐츠 삭제	/api/content/{contentId}	DELETE	특정 콘텐츠를 삭제합니다.	-	contentId (Long)	204 No Content
콘텐츠 좋아요 토글	/api/content/{contentId}/like	POST	콘텐츠에 대한 '좋아요' 상태를 토글합니다.	-	contentId (Long)	liked" or "unliked""
좋아요한 콘텐츠 목록 조회	/api/content/liked	GET	현재 로그인된 사용자가 '좋아요'를 누른 콘텐츠 목록을 페이징하여 조회합니다.	-	page, size, sort	Page<ContentList>

5.3. 댓글 API

기능	Endpoint	Method	Description	Request Body	Query Parameter	Response
댓글 목록 조회	/api/contents/{contentId}/comments	GET	특정 콘텐츠의 댓글 목록을 조회합니다.	-	contentId (Long)	List<CommentResponseDto>
댓글 작성	/api/contents/{contentId}/comments	POST	새로운 댓글을 작성합니다.	CommentCreateDto	contentId (Long)	CommentResponseDto
댓글 수정	/api/contents/{contentId}/com	PUT	특정 댓글을 수정합니다.	CommentUpda	contentId	CommentResponseDto

	ments/{commentId}			teDto	(Long), commentId (Long)	
댓글 삭제	/api/contents/{contentId}/comments/{commentId}	DELETE	특정 댓글을 삭제합니다.	-	contentId (Long), commentId (Long)	200 OK
댓글 좋아요 토글	/api/contents/{contentId}/comments/{commentId}/like	POST	댓글에 대한 '좋아요' 상태를 토글합니다.	-	contentId (Long), commentId (Long)	liked" or "unliked""

5.4. 공지사항 API

기능	Endpoint	Method	Description	Request Body	Query Parameter	Response
공지사항 생성	/api/creators/{creatorId}/notices	POST	크리에이터가 새로운 공지사항을 작성합니다.	NoticeCreateDto	creatorId (Long)	NoticeResponseDto
공지사항 목록 조회	/api/creators/{creatorId}/notices	GET	특정 크리에이터의 공지사항 목록을 조회합니다.	-	creatorId (Long)	List<NoticeResponseDto>
공지사항 상세 조회	/api/creators/{creatorId}/notices/{noticeId}	GET	특정 공지사항의 상세 내용을 조회합니다.	-	creatorId (Long), noticeId (Long)	NoticeDetailDto
공지사항 수정	/api/creators/{creatorId}/notices/{noticeId}	PUT	기존 공지사항을 수정합니다.	NoticeUpdateDto	creatorId (Long),	NoticeResponseDto

					noticeId (Long)	
공지사항 삭제	/api/creators/{creatorId}/notices/{noticeId}	DELETE	공지사항을 삭제합니다.	-	creatorId (Long), noticeId (Long)	204 No Content
공지사항 반응 토글	/api/creators/{creatorId}/notices/{noticeId}/reaction	POST	공지사항에 대한 이모지 반응을 추가하거나 삭제합니다.	-	creatorId (Long), noticeId (Long), emoji (String)	200 OK
공지사항 반응 조회	/api/creators/{creatorId}/notices/{noticeId}/reaction	GET	특정 공지사항에 대한 반응 목록을 조회합니다.	-	creatorId (Long), noticeId (Long)	List<NoticeReactionDto>
공지사항 반응 조회 (NoticeReactionController)	/api/notices/{noticeId}/reactions	GET	특정 공지사항에 대한 반응 목록을 조회합니다. (레거시 또는 다른 목적)	-	noticeId (Long)	List<NoticeReactionDto>
공지사항 반응 토글 (NoticeReactionController)	/api/notices/{noticeId}/reactions	POST	공지사항에 대한 이모지 반응을 추가하거나 삭제합니다. (레거시 또는 다른 목적)	-	noticeId (Long), emoji (String)	200 OK

5.5. 구독 API

기능	Endpoint	Method	Description	Request Body	Query Parameter	Response
구독 생성	/api/subscriptions/pending	POST	구독을 생성하고 대기 상태로 설정합니다.	SubscriptionRequestDto	-	SubscriptionResponseDto
구독 활성화	/api/subscriptions/{subscriptionId}/activate	POST	결제 완료 후 구독을 활성화합니다.	ActivateSubscriptionRequestDto	subscriptionId (Long)	구독이 활성화되었습니다.""
구독 실패 처리	/api/subscriptions/{subscriptionId}/fail	POST	구독 결제 실패 시 처리합니다.	-	subscriptionId (Long)	구독 실패 처리가 완료되었습니다." "
내 구독 목록 조회	/api/users/me/subscriptions	GET	로그인한 사용자의 구독 목록을 조회합니다.	-	-	Map<String, Object>
구독한 크리에이터 목록 조회	/api/users/{nickname}/subscriptions	GET	특정 사용자가 구독한 크리에이터 목록을 조회합니다.	-	nickname (String)	Map<String, Object>
구독 해지	/api/subscriptions/{subscriptionId}	PATCH	구독을 해지합니다.	-	subscriptionId (Long)	200 OK

5.6. 결제 API

기능	Endpoint	Method	Description	Request Body	Query Parameter	Response
빌링키 결제 처리	/api/billings/payments	POST	빌링키를 사용하여 결제를 처리합니다.	BillingKeyPaymentRequestDto	-	BillingKeyPaymentResponseDto
결제 수단 등록	/api/billings/card	POST	사용자의 결제 수단(카드)을 등록합니다.	SavePaymentMethodRequestDto	-	SavePaymentMethodResponseDto
결제 수단 조회	/api/payments	GET	사용자가 등록한 모든 결제 수단을 조회합니다.	-	-	List<PaymentMethodResponseDto>
결제 수단 삭제	/api/billings/keys	POST	등록된 결제 수단을 삭제합니다.	DeletePaymentMethodRequestDto	-	DeletePaymentMethodResponseDto

5.7. 검색 API

기능	Endpoint	Method	Description	Query Parameter	Response
통합 검색	/api/search	GET	콘텐츠와 크리에이터를 통합하여 검색합니다.	SearchDto (ModelAttribute), sort (String)	IntegratedSearchResponse
콘텐츠 검색	/api/contents	GET	콘텐츠(제목, 내용)를 검색합니다.	SearchDto (ModelAttribute), page, size, sort	SearchResponse<SearchResultDto>
크리에이터 검색	/api/creators	GET	크리에이터(닉네임, 직업)를 검색합니다.	SearchDto (ModelAttribute), page, size, sort	SearchResponse<SearchResultDto>

5.8. 파일 업로드 API

기능	Endpoint	Method	Description	Request Body	Query Parameter	Response
에디터 이미지 업로드	/api/upload/image	POST	Toast UI Editor에서 사용되는 이미지를 업로드합니다.	image (MultipartFile)	-	Map<String, String> (imageUrl)
프로필 이미지 업로드	/api/upload/profile-image	POST	프로필 이미지를 업로드합니다.	image (MultipartFile)	-	Map<String, String> (imageUrl)

6. 보안 및 인증

6.1. 보안 및 인증 개요

CreationStack 프로젝트는 Spring Security와 JWT(JSON Web Token)를 활용하여 사용자 인증 및 권한 부여를 처리합니다.

6.2. 로그인 및 인증 흐름

사용자 인증은 일반 로그인과 카카오 소셜 로그인을 통해 이루어지며, JWT를 사용하여 세션 없이 상태를 관리합니다.

6.2.1. 일반 로그인 흐름

1. 사용자

로그인 페이지에서 이메일과 비밀번호를 입력하고 로그인 요청을 보낸다.

2. 프론트엔드

POST /api/auth/login API로 사용자 입력 정보를 백엔드에 전송한다.

3. 백엔드

- AuthService는 이메일로 데이터베이스에서 사용자를 조회한다.

- BCryptPasswordEncoder를 사용하여 입력된 비밀번호와 저장된 암호화된 비밀번호를 비교한다.
- 인증에 성공하면 JwtUtil을 통해 Access Token과 Refresh Token을 생성한다.
- Refresh Token은 데이터베이스에 저장되어 토큰 갱신에 사용된다.
- 생성된 토큰 정보를 LoginResponse 형태로 프론트엔드에 응답한다.

4. 프론트엔드

- 서버로부터 받은 JWT 토큰(Access Token, Refresh Token)을 브라우저의 안전한 저장소(localStorage 또는 HttpOnly 쿠키)에 저장한다.
- 이후 보호된 API 요청 시, HTTP Authorization 헤더에 Access Token을 Bearer 토큰 형식으로 포함해 전송한다.

6.2.2. 카카오 소셜 로그인 흐름

1. 사용자

로그인 페이지에서 '카카오로 시작하기' 버튼을 클릭합니다.

2. 프론트엔드

사용자를 카카오 인증 페이지로 리디렉션시킵니다.

3. 카카오 서버 & 사용자

사용자가 카카오 계정으로 로그인하고 정보 제공에 동의합니다.

4. 카카오 서버

인증이 완료되면, 미리 지정된 백엔드의 콜백 URI (/api/auth/kakao/callback)로 인가 코드와 함께 사용자를 리디렉션시킵니다.

5. 백엔드 (KakaoAuthService)

- 전달받은 인가 코드를 사용하여 카카오 API 서버에 액세스 토큰을 요청합니다.
- 발급받은 액세스 토큰으로 다시 카카오 API 서버에 사용자 정보(이메일, 고유 ID 등)를 요청합니다.
- 획득한 이메일로 우리 서비스의 데이터베이스를 조회하여 이미 가입된 사용자인지 확인합니다.
    a. case: 기존 회원

해당 사용자에 대한 우리 서비스의 Access Token과 Refresh Token을 생성하고, 토큰들을 쿼리 스트링에 담아 프론트엔드의 최종 로그인 처리 페이지 (/auth/callback)로 리디렉션시킵니다.

b. case: 신규 회원

회원가입을 마저 진행하기 위해, 카카오로부터 받은 이메일과 플랫폼 고유 ID를 쿼리 스트링에 담아 프론트엔드의 회원가입 페이지 (/register)로 리디렉션시킵니다.

6. 프론트엔드

- /auth/callback

  리디렉션된 URL에서 토큰 정보를 추출하여 저장하고, 로그인 상태를 활성화한 후 메인 페이지로 이동합니다.

- /register

  리디렉션된 URL에서 이메일과 플랫폼 정보를 추출하여 회원가입 폼에 자동으로 채워주고, 사용자에게 추가 정보(닉네임, 역할 등)를 입력받아 최종 회원가입을 진행합니다.

6.3. 보안 설정 (Spring Security & JWT)

CreationStack 백엔드는 Spring Security를 사용하여 API 요청에 대한 인증 및 권한 부여를 관리하며, JWT를 통해 상태 비저장(Stateless) 인증을 구현합니다.

6.3.1. 주요 구성 요소

- SecurityConfig.java: Spring Security의 핵심 설정 파일입니다.
  - CSRF 비활성화: REST API 서버이므로 CSRF 보호를 비활성화합니다.

- CORS 설정: CorsConfigurationSource Bean을 통해 프론트엔드(http://localhost:5173)로부터의 요청을 허용합니다. 허용되는 HTTP 메서드, 헤더, 자격 증명(credentials) 등이 설정됩니다.
- 세션 관리: SessionCreationPolicy.STATELESS를 설정하여 서버가 세션을 유지하지 않도록 합니다. 이는 JWT 기반 인증의 핵심입니다.
- URL 권한 설정: requestMatchers().permitAll()을 통해 인증 없이 접근 가능한 공개 API 경로를 정의합니다. 그 외의 모든 요청(anyRequest().authenticated())은 인증된 사용자만 접근할 수 있도록 설정합니다.
- BCryptPasswordEncoder: 비밀번호 암호화를 위해 BCrypt 해싱 알고리즘을 사용합니다.

● JwtAuthenticationFilter.java:

OncePerRequestFilter를 상속받아 모든 HTTP 요청에 대해 한 번씩 실행되는 커스텀 필터입니다.

- 토큰 추출: HTTP Authorization 헤더에서 Bearer 토큰을 추출합니다.
- 토큰 유효성 검증: JwtUtil을 사용하여 추출된 Access Token의 유효성을 검증합니다.
- 인증 정보 설정: 토큰이 유효하면, 토큰에서 사용자 ID, 이메일, 역할을 추출하여 Spring Security SecurityContextHolder에 UsernamePasswordAuthenticationToken 형태로 인증 정보를 설정합니다. 이를 통해 @AuthenticationPrincipal 어노테이션 등으로 컨트롤러에서 사용자 정보를 쉽게 가져올 수 있습니다.
- UsernamePasswordAuthenticationFilter.class 이전에 실행되도록 설정되어, 실제 인증 처리 전에 JWT를 통한 인증을 시도합니다.

● JwtUtil.java: JWT 토큰의 생성, 유효성 검증, 정보 추출 등을 담당하는 유틸리티 클래스입니다.

- Access Token 및 Refresh Token 생성 로직을 포함합니다.
- 토큰의 만료 시간, 서명 키 등을 관리합니다.

6.3.2. 인증 흐름 요약

1. 클라이언트가 로그인 API를 호출하여 Access Token과 Refresh Token을 받습니다.
2. 클라이언트는 이후 보호된 API 요청 시 Access Token을 Authorization: Bearer <AccessToken> 형태로 헤더에 포함하여 전송합니다.
3. JwtAuthenticationFilter가 요청을 가로채 Access Token을 추출하고 유효성을 검증합니다.
4. 토큰이 유효하면, 필터는 토큰에서 사용자 정보를 파싱하여 SecurityContextHolder에 저장합니다.
5. 컨트롤러는 SecurityContextHolder에 저장된 인증 정보를 사용하여 현재 요청을 보낸 사용자를 식별하고 권한을 확인합니다.
6. Access Token이 만료되면, 클라이언트는 Refresh Token을 사용하여 새로운 Access Token을 발급받습니다.

6.4. 예외 처리 및 보안 관련 예외

백엔드 시스템은 GlobalExceptionHandler를 통해 모든 예외를 일관되게 처리하여 안정적인 API 응답을 제공합니다. 보안 관련 예외 또한 이 핸들러를 통해 처리됩니다.

6.4.1. 전역 예외 처리

- GlobalExceptionHandler.java: @RestControllerAdvice 어노테이션을 사용하여 애플리케이션 전역에서 발생하는 예외를 중앙 집중적으로 처리합니다.
  - 다양한 예외 타입(예: IllegalArgumentException, MethodArgumentNotValidException, CustomException, HttpClientErrorException 등)에 대해 @ExceptionHandler 메서드를 정의하여 특정 HTTP 상태 코드와 ErrorResponse DTO를 반환합니다.
  - 이를 통해 프론트엔드는 일관된 오류 응답 형식을 기대하고 사용자에게 적절한 피드백을 제공할 수 있습니다.

6.4.2. 보안 관련 예외 처리

- AccessDeniedException: Spring Security에 의해 발생하는 접근 거부 예외입니다. GlobalExceptionHandler에서 이를 처리하여 403 Forbidden 상태 코드와 "접근 권한이 없습니다." 메시지를 반환합니다.
- JWT 관련 예외: JwtAuthenticationFilter 내에서 JWT 토큰 검증 실패 시 발생하는 예외(예: SignatureException, ExpiredJwtException)는 필터 내부에서 로깅되고 SecurityContextHolder를 비워 인증 실패를 처리합니다.
  클라이언트에게는 일반적으로 401 Unauthorized 응답이 반환됩니다.

7. 개발 환경 설정 및 실행 방법

7.1. Backend (Spring Boot)

7.1.1. 의존성 설치

프로젝트 루트의 backend 디렉토리에서 다음 명령어를 실행하여 Maven 의존성을 설치합니다.

```
cd backend
mvn install
```

7.1.2. 환경 변수 설정

backend/src/main/resources/application.yml 파일을 열어
데이터베이스 연결 정보, JWT 시크릿 키, AWS 자격 증명 등 필요한 환경 변수를 설정합니다.

```yaml
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/creationstack
    username: your-db-username
    password: your-db-password

jwt:
  secret: your-jwt-secret-key

cloud:
  aws:
    credentials:
      access-key: your-aws-access-key
      secret-key: your-aws-secret-key
    s3:
      bucket: your-s3-bucket-name
    region:
      static: ap-northeast-2
```

7.1.3. 애플리케이션 실행

다음 명령어를 사용하여 Spring Boot 애플리케이션을 실행합니다.

```
mvn spring-boot:run
```

7.2. Frontend (React)

7.2.1. 의존성 설치

프로젝트 루트의 frontend 디렉토리에서 다음 명령어를 실행하여 npm 의존성을 설치합니다.

```
cd frontend
npm install --legacy-peer-deps
```

7.2.2. 개발 서버 실행

다음 명령어를 사용하여 Vite 개발 서버를 실행합니다.

```
npm run dev
```

이제 웹 브라우저에서 http://localhost:5173 (또는 Vite가 지정한 다른 포트)으로 접속하여 프론트엔드 애플리케이션을 확인할 수 있습니다.

8. 기타 참고 사항

8.1 환경 변수 관리

민감한 정보를 보호하고 다양한 실행 환경(local, dev, prod 등)에 맞춰 유연하게 설정을 적용하기 위해 .env 파일을 사용했습니다. .env 파일은 .gitignore에 등록하여 Git에 커밋되지 않도록 하였습니다.

8.1.2 백엔드 .env

backend/ 폴더 내의 application.yml에서 dotenv-java 라이브러리를 통해 로드됩니다.

```
# PortOne 설정
PORTONE_HOSTNAME=https://api.portone.io
PORTONE_API_SECRET=...
PORTONE_STORE_ID=...

# JWT 설정
Secret_token=...
Refresh_token=...

# Kakao 로그인
Kakao_Client_Id=...
Kakao_Redirect_Uri=http://localhost:8080/api/auth/kakao/callback

# AWS S3
AWS_S3_BUCKET_NAME=...
AWS_ACCESS_KEY_ID=...
AWS_SECRET_ACCESS_KEY=...
```

8.1.2 프론트엔드 .env

package.json 있는 프로젝트 루트에서 VITE_ 접두어를 붙여 import.meta.env로 접근합니다

```
# PortOne
VITE_STORE_ID=...
VITE_CHANNEL_KEY=...

# Kakao 로그인
VITE_KAKAO_REST_API_KEY=...
VITE_KAKAO_REDIRECT_URI=http://localhost:8080/api/auth/kakao/callback
```

8.2 AWS S3 버킷 정책 관리

썸네일, 프로필 이미지, 에디터 이미지 등 다양한 정적 리소스를 AWS S3에 저장하고 웹에서 직접 불러오기 위해 S3 버킷은 다음과 같은 정책과 CORS를 설정하였습니다 .

8.2.1 버킷 정책 설정 (정적 파일 공개 읽기 허용)

사용자가 업로드한 이미지 파일을 브라우저에서 직접 로드할 수 있도록 S3 객체 조회(s3:GetObject) 권한을 퍼블릭에게 허용해야 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadAccess",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::creationstack-bucket/*"
    }
  ]
}
```

8.2.2 application.yml 및 .env 연동

Spring Boot 애플리케이션은 S3와 연동하여 파일 업로드 및 다운로드 URL을 구성하고, 프론트엔드는 해당 URL을 통해 이미지를 직접 렌더링할 수 있습니다.

```
cloud:
  aws:
    s3:
      bucket-name: ${AWS_S3_BUCKET_NAME}
    credentials:
      access-key: ${AWS_ACCESS_KEY_ID}
      secret-key: ${AWS_SECRET_ACCESS_KEY}
    region:
      static: ap-northeast-2
```

9. 참고 자료 및 외부 라이브러리 목록

9.1. 외부 라이브러리 및 프레임워크

9.1.1. 백엔드 (Java/Spring Boot)

- Spring Boot (3.5.3): 애플리케이션 개발을 위한 핵심 프레임워크
  - spring-boot-starter-data-jpa: JPA(Java Persistence API)를 통한 데이터베이스 연동
  - spring-boot-starter-security: 강력한 인증 및 권한 부여 기능 제공
  - spring-boot-starter-validation: 데이터 유효성 검사
  - spring-boot-starter-web: RESTful API 개발을 위한 웹 기능 제공
  - spring-boot-devtools: 개발 중 빠른 재시작 및 코드 변경 감지
  - spring-boot-starter-test: 테스트 코드 작성 지원
  - spring-security-test: Spring Security 관련 테스트 지원
- MySQL Connector/J: MySQL 데이터베이스 연결 드라이버
- Lombok: Getter, Setter, 생성자 등을 어노테이션으로 자동 생성하여 코드량 감소
- JWT (jjwt-api, jjwt-impl, jjwt-jackson): JSON Web Token 생성 및 검증 라이브러리
- PortOne (iamport-rest-client-java): 아임포트(PortOne) 결제 시스템 연동을 위한 클라이언트 라이브러리
- AWS SDK for S3 (aws-java-sdk-s3): AWS S3 버킷에 파일 업로드/다운로드 등 객체 스토리지 서비스 연동
- dotenv-java: .env 파일에서 환경 변수를 로드하는 라이브러리

9.1.2. 프론트엔드 (React/Vite)

- Vite (7.0.4): 빠르고 가벼운 프론트엔드 빌드 도구
- axios: HTTP 클라이언트 라이브러리 (API 요청)
- react-router-dom: React 애플리케이션 내 라우팅 관리
- qs: 쿼리 스트링 파싱 및 직렬화 라이브러리
- jwt-decode: JWT 토큰 디코딩
- @emoji-mart/data, @emoji-mart/react, emoji-mart: 이모지 선택 및 표시 라이브러리
- @portone/browser-sdk: 아임포트(PortOne) 결제 시스템 프론트엔드 SDK
- @radix-ui/: UI 컴포넌트 라이브러리 (dialog, label, radio-group, select 등)
- @toast-ui/editor, @toast-ui/react-editor: 웹 에디터 컴포넌트
- lucide-react: React 컴포넌트로 제공되는 아이콘 라이브러리
- prismjs: 코드 블록 구문 강조
- react-markdown, remark-gfm: Markdown 렌더링 및 GitHub Flavored Markdown 지원
- uuid: UUID(Universally Unique Identifier) 생성
- class-variance-authority, clsx: Tailwind CSS와 같은 유틸리티 우선 CSS 프레임워크와 함께 사용되는 유틸리티
- dotenv: .env 파일에서 환경 변수를 로드하는 라이브러리 (프론트엔드용)