15지조 발표

발표자 이윤성

목차

1. 프로젝트 소개

2. 팀원 소개 및 역할 분담

3. 목표 달성

4. ERD & API 명세

5. 시연 영상

6. 후기

프로젝트 소개

개발하면서 생긴 문제나 해결법을 누구나
자유롭게 올리고 볼 수 있는 공간

Coderend 

팀원 소개 및 역할 분담

팀장 조현아

- 게시물 CRUD

팀원 김원기

- 회원가입, 회원 탈퇴, 이메일 인증

팀원 이윤성

- Token, 로그인, 로그아웃

팀원 정효진

- 댓글 CRUD

팀원 홍용근

- 유저 프로필 관리, 좋아요 기능

목표 달성


캐릭터 기본 슬롯 8개로 증가

필수 구현 기능

사용자 인증 기능

- 회원 가입, 회원 탈퇴, 로그인, 로그아웃

프로필 관리 기능

- 프로필 조회, 프로필 수정

뉴스피드 게시물 기능

- 게시물 작성, 조회, 수정, 삭제

추가 구현 기능

페이지네이션

정렬 기능

기간별 검색 기능

댓글 CRUD 기능

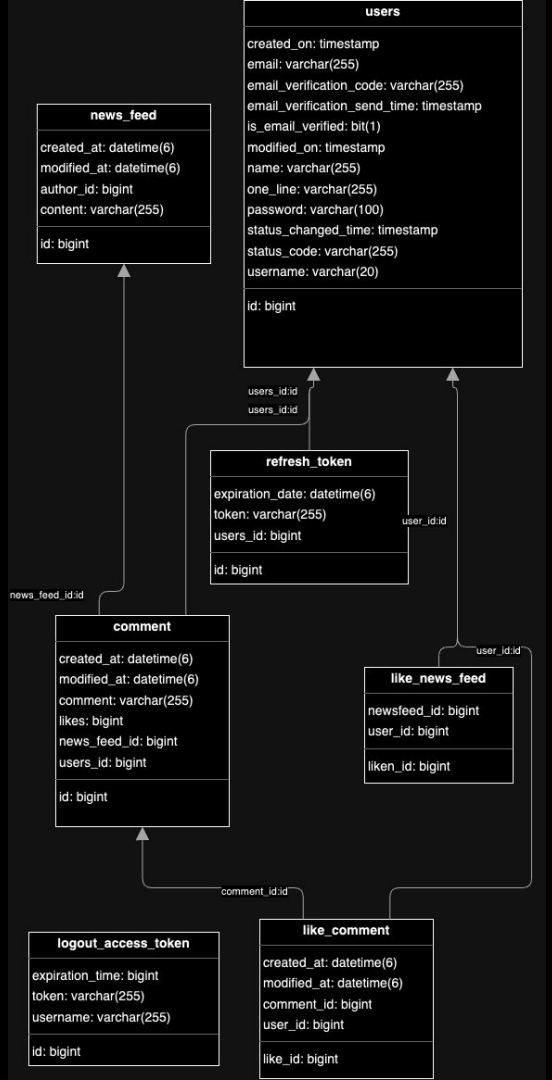
이메일 가입 및 인증 기능

좋아요 기능

Swagger 적용

이메일 가입 및 인증 추가 구현

ERD
&
API 명세



User와 Refresh Token 1:1

User와 Comment 1:N

User와 LikeComment 1:N

Newsfeed와 Comment는 1:N

Comment와 LikeComment는 1:N

LogoutAccessToken은 독립적인 Entity

회원가입 - POST

/api/user

```
//request body


{
    username: String,
    password: String,
    name: String,
    email: email,
    oneline: String
}
```

```
//response header


{
    200: “회원가입에 성공하셨습니다.”,
    400: ”회원가입에 실패하셨습니다.”
}
```

회원탈퇴 - PUT

/api/user

```
//request body


{
    username: String,
    password: String
}
```

```
//response header


{
    200: “회원 탈퇴가 완료되었습니다.”,
    400: ”현재 로그인 한 사용자가 아닙니다.”,
    400: ”아이디 또는 비밀번호를 확인해 주세요.”
}
```

로그인 - POST

/api/user/login

```
//request body


{
    username: String,
    password: String
}
```

```
//response header


{
    200: “회원 탈퇴가 완료되었습니다.”,
    400: ”패스워드가 일치하지 않습니다.”,
    401: ”인증 대기 상태입니다. 이메일 인증을 해주세요.”,
    403: ”탈퇴한 계정입니다.”,
    404: ”존재하지않는 사용자입니다.”,
    500: ”서버 오류가 발생했습니다.”
}
```

로그아웃 - POST

/api/user/logout

```
//request header


{
   Authorization: "Bearer
{Access Token}"
}
```

```
//response header


{
   200: “로그아웃 되었습니다.”,
   400: ”로그아웃에 실패하셨습니다.”
}
```

프로필 조회 - GET

/api/user/profile

```
//request header

{
   Authorization: "Bearer {Access Token}"
}
```

```
//response header

{
   200: “프로필 조회에 성공하셨습니다.”,
   400: ”패스워드가 일치하지 않습니다.”
}
```

```
//response body

{
   username: String,
   password: String,
   oneline: String,
   email: email
}
```

프로필 수정 - PUT

/api/user/profile

```
//request header


{
   Authorization: "Bearer
{Access Token}"
}
```

```
//request body

{
   name: String,
   email: email,
   current-password:
String,
   oneline: String,
   new-password: String,
   check-new-password:
String,
}
```

```
//response header


{
   200: “프로필 수정에
성공하셨습니다.”,
   400: ”패스워드가
일치하지 않습니다.”
}
```

뉴스피드 작성 - POST

/api/newsfeed

```
//request header

{
   Authorization: "Bearer {Access Token}"
}
```

```
//request body

{
   content: String
}
```

```
//response header

{
   200: “뉴스피드 작성에 성공하셨습니다.”,
   400: ”뉴스피드 작성에 실패하셨습니다.”
}
```

뉴스피드 조회 - GET

/api/newsfeed/{newsfeedId}

```
//request header

{
   Authorization: "Bearer {Access Token}"
}
```

```
//response header

{
   200: “뉴스피드 조회에 성공하셨습니다.”,
   400: ”뉴스피드 조회에 실패하셨습니다.”
}
```

```
//response body

{
   author-id: bigint,
   content: String,
   created-at: timestamp,
   modified-at: timestamp,
}
```

뉴스피드 수정 - PUT

/api/newsfeed/{newsfeedId}

```
//request header

{
    Authorization: "Bearer {Access Token}"
}
```

```
//request body

{
    content: String
}
```

```
//response header

{
    200: “뉴스피드 수정에 성공하셨습니다.”,
    400: ”뉴스피드 수정에 실패하셨습니다.”
}
```

```
//response body

{
    author-id: bigint,
    content: String,
    created-at: timestamp,
    modified-at: timestamp,
}
```

뉴스피드 삭제 - DELETE

/api/newsfeed/{newsfeedId}

```
//request header


{
    Authorization: "Bearer
{Access Token}"
}
```

```
//response header


{
    200: “뉴스피드 삭제에 성공하셨습니다.”,
    400: ”뉴스피드 삭제에 실패하셨습니다.”
}
```

전체 뉴스피드 조회 - GET

/api/newsfeed/

```
//response header


{
   200: “먼저 작성하여 소식을 알려주세요!”,
   200: “뉴스피드 조회에 성공하셨습니다.”,
   400: ”뉴스피드 조회에 실패하셨습니다.”
}
```

뉴스피드 최신순 정렬 - GET

/api/newsfeed/newest

```
//response header


{
   200: “뉴스피드 조회에 성공하셨습니다.”,
   400: ”뉴스피드 조회에 실패하셨습니다.”
}
```

뉴스피드 좋아요순 정렬 - GET

/api/newsfeed/likes

```
//response header


{
   200: “뉴스피드 조회에 성공하셨습니다.”,
   400: ”뉴스피드 조회에 실패하셨습니다.”
}
```

기간 내 뉴스피드 검색 - GET

/api/newsfeed/search

```
//response header


{
   200: “뉴스피드 조회에 성공하셨습니다.”,
   400: ”뉴스피드 조회에 실패하셨습니다.”
}
```

댓글 작성 - POST

/api/comments/{newsfeedId}

```
//request header

{
   Authorization: "Bearer {Access Token}"
}
```

```
//request body

{
   content: String
}
```

```
//response header

{
   200: “댓글 작성에 성공하셨습니다.”,
   400: ”댓글 작성에 실패하셨습니다.”
}
```

댓글 수정 - PUT

/api/comments/{commentId}

```
//request header

{
   Authorization: "Bearer {Access Token}"
}
```

```
//request body

{
   content: String
}
```

```
//response header

{
   200: “댓글 수정에 성공하셨습니다.”,
   400: ”댓글 수정에 실패하셨습니다.”
}
```

댓글 삭제 - POST

/api/comments/{commentId}

```
//request header

{
   Authorization: "Bearer {Access Token}"
}
```

```
//response header

{
   200: “댓글 작성에 성공하셨습니다.”,
   400: ”댓글 작성에 실패하셨습니다.”
}
```

```
//response body

{
   newsfeed-id: bigint,
   author-id: bigint,
   content: String,
   like: int,
   created-at: timestamp,
   modified-at: timestamp,
}
```

뉴스피드 별 전체 댓글 조회 - GET

/api/comments/{newsfeedId}

```
//response header


{
   200: “댓글 조회에 성공하셨습니다.”,
   400: ”댓글 조회에 실패하셨습니다.”
}
```

뉴스피드 좋아요 토글 - POST

/api/comments/{newsfeedId}/likeToggle

```
//request header

{
   Authorization: "Bearer
{Access Token}"
}
```

```
//response header

{
   200: “좋아요에 성공하셨습니다.”,
   400: ”좋아요에 실패하셨습니다.”
}
```

댓글 좋아요 토글 - POST

/api/comments/{newsfeedId}/comments/{commentId}/likeToggle

```
//request header

{
   Authorization: "Bearer
{Access Token}"
}
```

```
//response header

{
   200: “좋아요에 성공하셨습니다.”,
   400: ”좋아요에 실패하셨습니다.”
}
```

토큰 갱신 - POST

/api/refresh

```
//request header


{
    Authorization: "Bearer
{Access Token}"
}

//cookie
refreshToken
```

```
//response header


{
    200: “newAccessToken.”,
    400: ”Invalid Refresh Token.”
}
```

시연 영상

팀 프로젝트 후기

조현아

기본적인 CRUD는 어렵지 않았지만 좋아요 순으로 기능을 정렬할 때 예상했던 엔티티랑 바뀌어서 막혔던 것 같다.

하지만 같은 문제를 겪었던 팀원의 도움으로 해결할 수 있었다 !

또 access 토큰을 적용해서 CRUD를 해보는 것이 처음이라 그 부분에서도 어려움을 겪었던 것 같다.

김원기

초기 구현 시 구현된 토큰들을 User와 연관 관계를 지어 물리적으로 데이터베이스에서 삭제하는 작업을 했는데 그 때 당시에 토큰과 필터 등 인증 과정의 개념을 정확히 인지하지 못해 물리적인 삭제만 고민했던 점이 있었다.

→ 이 부분은 윤성님이 전체적으로 해결해주심… 다만 공부를 조금 더 했다

이윤성

Refresh token을 처음 구현하다보니 실수가 있어 다시 작업하는 시간이 있었다.

이메일 인증 기능을 테스트 해보려했는데 메일 서버에 접속 할 수 없는 문제가 생겨서 문제를 해결하려 많은 시간을 쏟았다. 공유기를 사용하고 있던 것을 공유기 사용을 하지 않으니 해결됐다.

정효진

데이터베이스를 create하는 과정에서 새로 entity가 변경될 때 run 하는 과정에서 오류가 많이 생겼습니다.

application run 시, 데이터베이스를 생성하고, 종료 시 데이터베이스를 삭제하도록 spring.jpa.hibernate.ddl-auto=create-drop로 설정하거나, run하기 전에 데이터베이스를 전부 drop해서 해결했습니다.

홍용근

데이터베이스 만들 때 자꾸 열이름 관련 부분이 바뀌는 부분이 있는데 좋아요는 아무래도 다른 부분이랑 연관 부분이 많다 보니 자꾸 수정하다 보니 테이블의 문제가 많았다.

특히 기억에 남는 건 갑자기 테이블 생성이 안됐던 부분 사실 아직도 왜 그런지는 잘 모르겠다. 그래서 앞에 했던 것 롤백해서 했다.

그래도 jpa가 생성되는 순서가 언제인지 정확하게 알아서 좋았다. 그리고 이거 계속 똑같이 반복하는 문젠데..

자꾸 어노토테이션 빼먹는 문제가 많은 거 같다. 이거 땜에 너무 불필요한 시간 낭비가 많았던 거 같다. 이부분에 대해서는 한번 정리해줄 필요가 있을 거 같다.

감사합니다!