“다 써봐” 게시판 프로젝트

주제: Spring Boot 게시판 구현하기

개발자: 허재연

깃허브: https://github.com/heo-jaeyeon/spring-boot-board

📝개요

본 프로젝트는 **Spring Boot** 기반의 **블로그 형 게시판 웹 애플리케이션**으로, **소셜 로그인 (OAuth2)을 활용**한 사용자 인증과 게시글 CRUD, 키워드 검색 기능을 제공합니다. 사용자는 **Google 또는 Kakao 계정을 통해 로그인**할 수 있으며, 로그인 후 **게시글 조회, 작성, 수정, 삭제 등 기능**을 자유롭게 사용할 수 있습니다. **검색 기능**은 제목 및 내용을 기준으로 서버에서 **키워드를 필터링**해 사용자 편의성을 높였고, **JWT와 Spring Security를 활용**해 인증된 사용자만 글을 작성할 수 있도록 권한 처리를 구현했습니다. 이 프로젝트를 통해 Spring 기반 웹 서비스의 전반적인 구조 MVC 설계, OAuth 인증 처리, JPA 기반 DB 처리 등 실무에 가까운 백엔드 기술을 직접 구현하고 경험할 수 있었습니다.

💫기능

▼ 카카오, 구글 이메일로 로그인

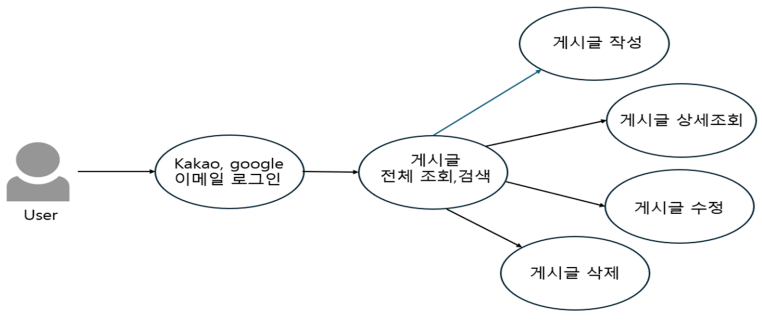
▼ 게시판 글 전제 조회 및 검색 기능

▼ 게시판 글 새로 작성

▼ 특정 게시판 글 상세조회

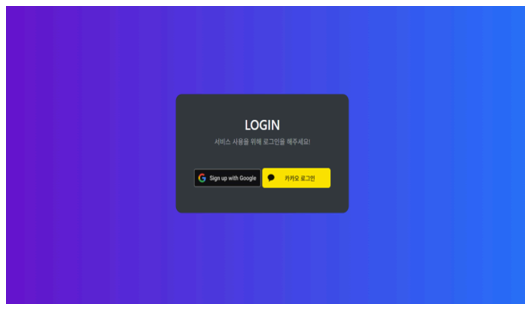
▼ 게시판 글 수정, 삭제

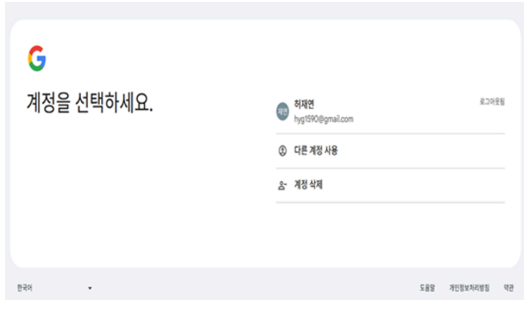
UML Use Case Diagram



👨‍💻기능 구현 화면

1. 로그인 화면





기능 설명:

Spring Security와 **OAuth2 Client**를 사용하여 **Google/Kakao 계정으로 로그인**하는 기능을 구현했습니다.

핵심 구현 내용:

- `spring-boot-starter-oauth2-client` 의존성을 활용하여 OAuth 인증 흐름을 간편하게 구성
- **구글**은 단일 키(JSON 구조) 방식, **카카오**는 중첩된 JSON 형태이므로 각각 다른 방식으로 이메일/닉네임을 추출하도록 분기 처리
- OAuth2 로그인 성공 시 **‘OAuth2USerService’**에서 사용자 정보를 받아 **JWT 토큰 생성 및 자동 로그인 처리**
- JWT는 AccessToken 기반으로 클라이언트에 저장되며, 이후 요청 시 인증 헤더를 통해 인증
- 로그인 후 사용자는 게시글 등록, 수정, 삭제 등의 기능에 접근 가능하며, 비로그인 상태에서는 접근이 제한

2. 게시글 조회

기능 설명:

로그인 후 작성되어 있는 **모든 게시글을 볼 수 있고 검색**을 통해 찾고자 하는 내용을 빠르게 찾을 수 있도록 구현했습니다. **"보러 가기"** 버튼을 누르면 해당 게시글을 좀 더 자세히 볼 수 있습니다.

핵심 구현 내용:

- **'BlogApiController'**에 `@Controller`에서 클라이언트 요청을 받아, **'BlogService'**에서 비즈니스 로직을 처리하고, **'BlogRepository'**를 통해 DB에 접근하는 3계층 구조로 구현
- 전체 게시글 조회 시 `ArticleEntity` 를 `ArticleListViewResponse`, `ArticleResponseDTO`로 변환하여 View에 필요한 데이터만 전달함으로써 **불필요한 정보 노출을 방지**하고 **레이어 간 역할 분리**를 명확히 유지
- 단일 게시글 조회 시 `@PathVariable`을 통해 ID 값을 받아 해당 게시글을 반환하고, `ArticleViewResponse` DTO로 변환 후 상세 화면에 렌더링
- 검색 기능은 `@RequestParam`으로 입력된 키워드를 기준으로 `findByTitleContainingIgnoreCaseOrContentContainingIgnoreCase(...)` 메서드를 사용해 **제목 또는 본문 내 키워드를 포함하는 게시글만 필터링**
- 전체 게시글과 검색 결과는 동일한 Thymeleaf 템플릿(`articleList.html`)에서 일관되게 처리되어 유지 보수가 용이

3. 게시글 작성

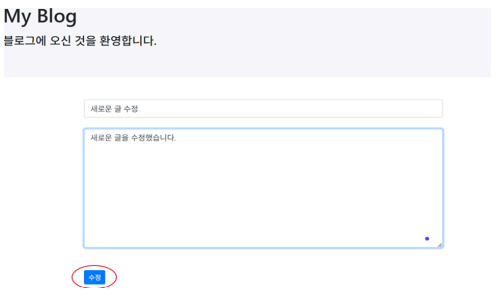
기능 설명:

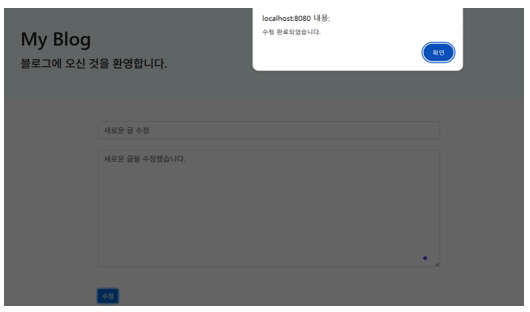
"글 등록" 버튼을 누르면 새로운 게시글을 작성할 수 있는 화면이 나오면서 제목과 내용을 작성 후 **"등록"**을 누르면 새로운 게시글이 저장됩니다.

핵심 구현 내용:

- **'BlogApiController'**에 존재하는 `addArticle()` 에서 AddArticleRequest DTO를 받아 처리
- **'BlogService'**에 존재하는 `save()` 에서 `userName` 을`save(request.toEntity(userName))` 로 받아 **'BlogRepository'**를 통해 **데이터베이스에 저장**
- Spring Security의 Principal로 로그인한 사용자를 작성자로 설정
- 데이터베이스에 저장된 데이터를 바탕으로 새로 추가된 게시글 확인 가능

4. 게시글 수정





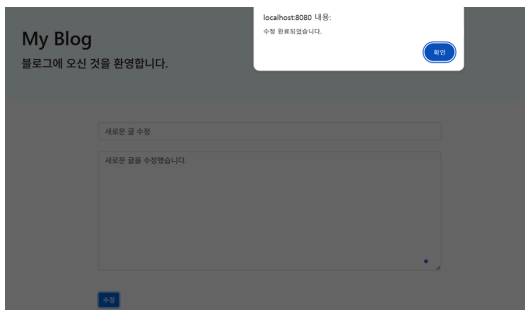
기능 설명:

작성한 게시글을 수정할 수 있는 기능을 구현했습니다. 수정하고자 하는 제목 또는 내용을 수정 후 **"수정"**버튼을 누르면 기존에 게시글 제목과 내용이 수정한 대로 업데이트됩니다.

핵심 구현 내용:

- 'BlogApiController'에 존재하는 `updateArticle()` 에서 `id`와 `UpdateArticleRequest` DTO를 받아 처리
- 'BlogService'에 존재하는 `update()`가 `id` 를 받아 `authorizeArticleAuthor()` 를 통해 작성자를 확인 후 `article.update(request.getTitle(), request.getContent());`로 수정
- `@Transactional` 를 사용해 데이터베이스 일관성 보장
- Spring Security로 작성자만 수정 가능

5. 게시글 삭제



기능 설명:

내가 작성한 글을 선택 후 "삭제"를 누르면 게시글이 삭제됩니다.

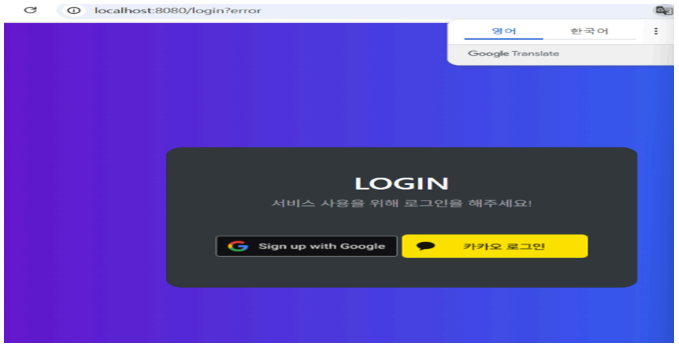
핵심 구현 내용:

- **'BlogApiController'**에 존재하는 `deleteArticle()` 에서 `id`를 받아 처리
- **'BlogService'**에 존재하는 `delete()` 에서 `authorizeArticleAuthor()` 를 통해 작성자를 확인 후 `blogRepository.deleteById(id);`로 삭제
- Spring Security로 작성자만 삭제 가능

🚀트러블 슈팅

문제상황 1

OAuth2 기반 소셜 로그인 연동 중, 카카오 로그인 시 `email == null` 문제가 발생했습니다.



문제 해결

카카오에서 제공하는 사용자 정보의 구조는 다음과 같이 중첩된 **JSON 객체로 구성**되어 있는데

`{"kakao_account": {"email": "user@example.com","profile": {"nickname": "홍길동"}}}`
기존 코드는
`OAuth2User.getAttribute("email")` 식으로만 **‘email’**을 추출하고 있었고,
이는 구글처럼 단일 키 구조에서는 유효하지만, 카카오처럼
중첩 객체 구조에서는 null을 반환한다는 사실을 알았습니다. 그래서 저는 `getAttributes("email")` 대신 `Map(kakao_account)`에서 직접 추출하도록 수정하여 해결하여 구글/카카오 통합 로그인 환경에서 사용자 인증 및 JWT 발급까지 **성공적으로 구현**했습니다.

문제상황 2

새로운 글을 등록하려는데 계속 **“등록실패”**를 했습니다.

문제해결

“article.js”에서 `httpRequest( )` 함수에
`"Content-type": "application/json"`을 보면 **‘type’**의 **‘t’**가 소문자임을 발견했고,

`"Content-type"` → `"Content-Type"`**로 수정**했습니다.
또한
`"header"`에서 마지막에 **‘s’**가 빠진 걸 발견하여

`"header"` → `"headers"` 로 수정 후 **글 등록에 성공**했습니다.

👨‍🏫사용 스택

백엔드	Java 17, Spring Boot, Spring Security, Spring Data JPA, JWT
OAuth 인증	Google OAuth2, Kakao OAuth2 (Spring OAuth2 Client 기반)
데이터베이스	H2 Database (개발 단계), JPA 기반 ORM 설계
빌드 도구	Gradle
테스트 및 디버깅	Postman, curl
개발 도구	IntelliJ IDEA, Git

- **Java**는 계속 사용한 언어로 익숙한 점과 **코드 재사용성, 유지 보수성, 확장성**이 뛰어난 점에서 사용했습니다.
- **Spring Boot**를 선택한 이유는 이전 팀 프로젝트에선 spring을 사용하면서 설정 작업(XML, 자바 설정 클래스 등)을 직접 수동으로 하다 보니 번거롭게 느껴졌는데
  Spring Boot는 복잡한 **설정 작업을 자동화**하여 더 쉽고 빠르게 애플리케이션을 개발할 수 있도록 한다는 점이 편리하게 느껴져 선택했습니다.
- **Spring Security**를 사용한 이유는 홈페이지에 인증 및 권한 기능을 빠르게 부여하고 사용자의 정보 및 거래 데이터를 승인되지 않는 접근으로부터 **보호**하고 **인증** 기능을 손쉽게 할 수 있기 때문에 사용했습니다.
- **Google OAuth2, Kakao OAuth2**은 각 소셜(google, Kakao, Naver 등)에서 제공하는 API를 활용하여 인가 및 코드를 직접 받는 방법으로 대부분의 웹에서 로그인을 할 때 많이 사용하는 기능이기 때문에 연습해보고 싶어 사용했습니다.
- **Spring Data JPA**는 반복 작성되는 메서드를 자동화하여 기본 CRUD 연산, 쿼리 메서드 생성, 페이징 처리 등을 자동으로 지원하여 개발 속도를 빠르게 하고, 코드량도 줄여주는 장점이 있어 사용했습니다.
- **H2 Database**를 사용한 이유는 원래 Mysql을 사용해봤지만 별도의 설치나 관리 없이 바로 사용할 수 있다는 점, 메모리 상에 데이터베이스를 구축하여 빠르게 개발하고 테스트할 수 있어서 사용했습니다.
- **Gradle**은 Groovy 문법으로 간결하게 스크립트를 작성할 수 있다는 점과 더 빠르게 빌드 할 수 있기 때문에 사용했습니다.
- **Postman**은 API의 기능 검증과 오류 디버깅을 위해 사용했습니다.
- **IntelliJ**는 사용해보지 않았던 IDE이기도 하고 Java 개발을 위한 최적화된 환경을 제공한다고 해서 사용했습니다.

🤦‍♂️이 프로젝트를 하며 느낀 점

이번 프로젝트에서 가장 인상 깊었던 기능은 **OAuth2 기능을 활용해 구글, 카카오로 로그인이 가능하게 구현한 점**입니다. 이번 프로젝트를 하기 이전에는 회원 가입을 할 때, 사용자가 정보를 일일이 입력하고, 입력 받은 데이터를 DB에 저장 한 뒤, 저장되어 있는 데이터가 입력 받은 데이터와 일치 한지 비교하는 기능을 구현하는 점이 번거로웠는데 OAuth2 로그인 기능은 별도 회원 가입 및 로그인 없이 애플리케이션이 사용자를 대신해 특정한 작업을 수행할 수 있도록 하는 점이 편리했습니다. 아쉬운 점은 중간 중간 저의 실수로 오타가 생겨 디버깅에 시간이 많이 들었다는 점입니다. 하지만 디버깅을 하는 능력도 프로젝트를 하기 전보다 더욱 성장했다는 것을 느꼈습니다. 앞으로 더욱 많은 프로젝트를 경험하면서 부족한 부분들에 대해 성장 할 수 있도록 꾸준히 공부해야겠다는 생각이 들었습니다.