전통주 기반 주류 정보 플랫폼을 위한 Keycloak 기반 MSA 인증 시스템 구현

조준현[1], 김병현[1], 박세진[2]
[1] 계명대학교 컴퓨터공학과 학부생
[2] 계명대학교 컴퓨터공학과 교수
wpqlks7@gmail.com, kbo102142@gmail.com, baksejin@kmu.ac.kr

Implementation of a Keycloak-based MSA Authentication System for a Traditional Liquor Information Platform

Jun-Hyeon Cho[1], Byeong-Hyeon Kim[1], Se-Jin Park[2]
[1]Dept. of Computer Engineering, Kei-Myung University
[2]Dept. of Computer Engineering, Kei-Myung University

요 약

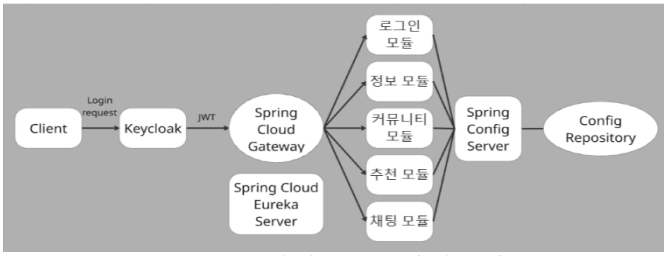
본 논문은 전통주 및 마이너한 주류 정보를 제공하는 사용자 중심 플랫폼에서의 인증 시스템 설계 및 구현 사례를 다룬다. 플랫폼의 다양한 기능 운영을 위해 마이크로 서비스 아키텍처(MSA)를 도입하고, 외부 로그인 기반의 JWT 에 사용자 식별자(userId)를 포함시켜 내부 시스템과 연동하였다. 이메일 기반 인증 구조를 개선하여 보안성과 유지보수성을 강화하였으며, 프로토타입 수준의 적용 흐름과 기대 효과를 제시한다.

1. 서론

전통주 등 마이너 주류에 대한 관심이 증가하며 이를 친숙하게 소개하고 추천하는 플랫폼의 필요성이 제기되고 있다. 본 프로젝트는 AI 기반 전통주 추천, 레시피 제공, 커뮤니티 기능 등으로 사용자 간 정보 공유를 촉진하는 것을 목표로 한다.

초기에는 이메일 기반 인증을 사용했으나, 기능 확장과 구조 복잡도 증가로 MSA 전환 및 인증 시스템 분리가 요구되었다. Keycloak 기반 인증 구조는 독립된 인증 서버 구성과 권한 관리의 효율성을 제공하며 [1][2], 본 논문은 MSA 환경에 맞춘 사용자 인증 흐름의 재설계 사례를 다룬다.

2. 시스템 아키텍처 및 MSA 구조



(그림 1) 전체 MSA 아키텍처

사용자 요청은 Spring Cloud Gateway 를 통해 들어오며, 요청 경로에 따라 각각의 마이크로서비스로 전달된다. 각 서비스는 Spring Boot 기반으로 구성되어 있으며, 공통적으로 Eureka Client 와 Spring Config Client 를 포함하고 있다.

Spring Cloud Eureka Server 는 모든 마이크로서비스의 위치 정보를 등록 및 관리하고, Gateway 는 이 정보를 기반으로 동적 라우팅 및 로드밸런싱을 수행한다.

각 서비스는 자체적인 환경 설정을 보유하지 않고, Spring Config Server 를 통해 외부로부터 설정 값을 가져온다. 이때 Config Server 는 실제 설정 값을 Config Repository(Git 저장소)에서 가져오므로 중앙 집중식 설정 관리가 가능해진다.

이와 같은 구조는 마이크로서비스 간의 결합도를 낮추어 각 서비스의 독립적인 배포 및 확장을 용이하게 한다. 또한 설정 및 서비스 등록/탐색 기능을 중앙에서 처리함으로써 시스템 전반의 관리 편의성과 안정성을 높일 수 있다.

3. Keycloak 기반 인증 흐름 설계

기존에는 사용자가 Google 로그인을 진행한 후, 클라이언트가 전달받은 이메일 정보를 서버로 전송하고, 서버에서는 이를 바탕으로 데이터베이스에서 userId 를 조회하여 유저 정보를 처리했다. 이 방식은 간단하지만, 매 요청마다 userId 를 파라미터로 넘기거나 내부

적으로 DB 조회가 필요해 보안성과 확장성 측면에서 한계가 있었다.

새로운 구조에서는 Keycloak 을 도입하여 인증을 처리하고 있다. 사용자는 외부 OAuth2 기반 로그인(Keycloak)을 통해 인증을 진행하며, 인증에 성공하면 사용자의 고유 식별자인 userId 가 포함된 JWT(JSON Web Token)가 발급된다. 이 JWT 는 클라이언트가 이후 요청 시 HTTP 헤더에 포함시켜 전송하게 되며, Gateway 는 해당 토큰을 검증하여 유효한 요청만 각 마이크로서비스로 전달한다.

각 마이크로서비스는 JWT 내부에 포함된 userId 를 기반으로 사용자 별 데이터를 분리하여 처리할 수 있다. 이로 인해 기존처럼 userId 를 RestAPI 파라미터로 전달하지 않아도 되어 보안성이 향상되었고, JWT 만으로도 유저 식별이 가능하므로 불필요한 DB 조회 없이 효율적인 처리가 가능하다.

또한, Keycloak 을 통해 발급되는 JWT 는 MSA 구조에서도 동일한 인증 흐름을 공유할 수 있어, 여러 마이크로서비스 간 일관된 인증 체계를 유지할 수 있게 된다. 이 구조는 보안성과 확장성을 동시에 확보할 수 있는 장점을 제공한다.

유사한 연구로, 이정은 외 [3]는 Ceph 오브젝트 스토리지 환경에 Keycloak 기반의 인증, 인가 체계를 설계하였다. 해당 연구 역시 인증 서버와 서비스 기능을 분리한 구조를 적용하였으며, Keycloak 의 유연성과 다양한 환경에서의 확장 가능성을 입증한 사례로 볼 수 있다.

4. 도입 효과 및 구현 결과

Keycloak 도입으로 JWT 의 userId 클레임을 기반으로 사용자 식별을 통합하고, 인증 서버를 분리하여 보안성과 MSA 적합성을 확보하였다.

기존에는 클라이언트가 직접 userId 를 전달하거나, 이메일을 통해 DB 에서 사용자 정보를 조회하는 방식이었으나, 이는 매 요청마다 DB 접근이 필요하고 사용자 식별 정보가 클라이언트에 노출된다는 단점이 있었다. JWT 기반 구조에서는 토큰의 클레임 정보를 통해 DB 접근 없이 사용자 식별이 가능해져 구조적인 단순화와 보안성 향상이 가능해졌다.

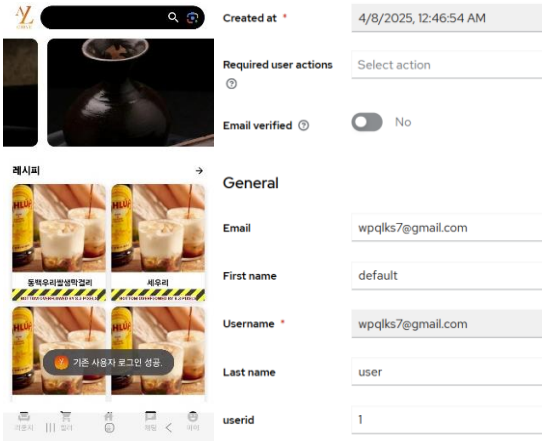
실제 /getUserId (DB 조회)와 /jwt/private/me (JWT 클레임 추출) API 를 각각 50 회 호출하여 평균 응답 시간을 비교한 결과, 이메일 기반 방식은 평균 37.02ms, JWT 기반 방식은 32.92ms 로 측정되었다.

<표 1> 사용자 식별 방식별 평균 응답 시간 비교

방식	평균 응답 시간(ms)	최대 응답 시간(ms)	최소 응답 시간(ms)
Email 기반	37.02	418	25
JWT 기반	32.96	172	25

두 방식 모두 대부분의 요청은 30ms 내외로 응답되었으며, 첫 요청에서 각각 418ms 와 172ms 로 응답 편차가 발생했으나, 이는 초기 서버 상태의 영향으로 판단된다. 이후 요청은 일정하게 수렴하였으며, JWT 방식은 DB 접근이 생략되어 경량화된 처리 구조로 효율성을 입증하였다.

또한 Keycloak Admin API 로 Google 로그인 시 닉네임만 입력하면 사용자 정보가 DB 와 Keycloak 에 자동 등록되도록 구현하였다.



(그림 2) 사용자 등록 후 Keycloak 저장 화면

5. 실제 적용 사례

서비스에서는 사용자가 앱을 실행하면 Google 로그인이 진행되고, 처음 로그인한 경우 Email 정보와 새로운 userId 를 Keycloak EC2 서버에 연결한 RDS 에 저장한다. 이후에는 JWT 를 발급받고, 사용자 본인의 정보가 필요할 때마다 토큰에 있는 userId 를 통해 사용자 요청을 처리한다.

6. 결론 및 향후 과제

본 논문에서는 전통주 기반 주류 플랫폼에 Keycloak 을 활용한 인증 시스템을 적용하고, 기존 이메일 기반 구조를 MSA 환경에 적합하도록 재설계한 사례를 제시하였다. 향후에는 AI 를 활용한 추천 알고리즘과의 연계와 클라이언트 최적화, 최종적으로는 앱 출시를 목표로 한다.

참고문헌

[1] 박채림, 전우재, 박진형, 박성훈, “클라우드 네이티브 IAM(Identity and Access Management) 솔루션”, 2022 년 한국정보과학회 동계학술대회 논문집, 제주, 2022, pp. 913-915.

[2] Keycloak 공식 홈페이지, Single-Sign-On, https://www.keycloak.org/

[3] 이정은, 이승윤, “효율적인 Ceph 인증과 인가를 지원하는 Keycloak 기반 사용자 접근 제어 모델,” 한국정보과학회 컴퓨터시스템 및 이론 학술대회, 제주, 2023, pp. 1392-1394.