FIU Knight Foundation School of Computing and Information Sciences
Florida International University

Centralized RBAC & Identity Management Server

Carson Farinella · Pablo Murillo · Guilherme Fulco · Noor Almasarweh · Danni Benitez · Instructor: Prof. Masoud Sadjadi

INFORMATION TECHNOLOGY
Capstone 2 · Summer 2026

Problem

- Every service decides access on its own — that's where breaches start.
- Per-service permission checks drift out of sync.
- Access is tied to people, not roles — moves and exits get missed.
- No single place to grant, revoke, or audit access.
- Most breaches come from getting “what may you do?” wrong, not “who are you?”

Requirements

- Built for engineering teams that need one trusted authorization boundary shared across every microservice.
- Functional: authenticate users · issue signed tokens · validate every call · enforce the required permission · audit every decision.
- Non-functional: no plaintext credentials · tamper-proof tokens · least-privilege networking · one-command reproducible deploy.
- Feasibility: team of 5, five two-week sprints, Docker Compose runs identically on any machine.

Our Solution

- RS256 signing — only the auth server mints tokens; any service can verify one, none can forge one.
- user → role → permission model — change a role once and every assignment updates.
- Downstream services delegate every check upstream — one audited source of truth.

Architecture

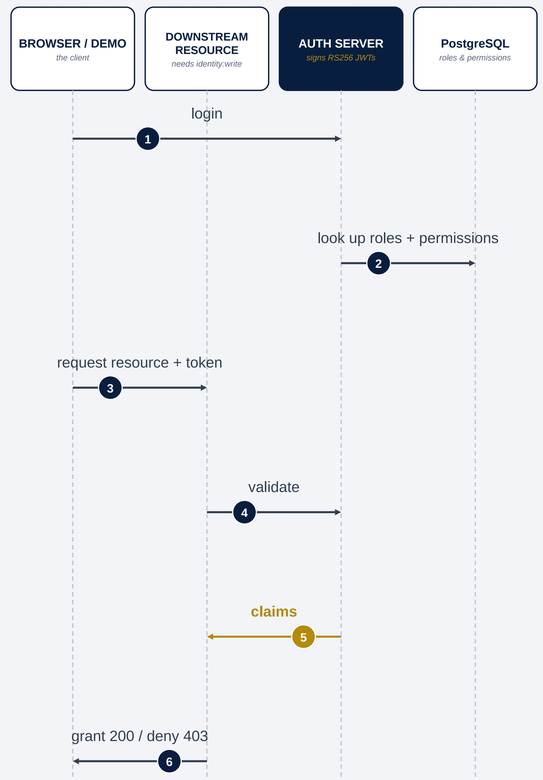

Figure 1. Request flow: the browser logs in, the auth server issues an RS256 JWT, the downstream service validates every call against Postgres-backed roles and permissions, and each decision (200 grant / 403 deny) is returned and logged.

Demo & Screenshots

- admin request → 200 GRANTED (has identity:write).
- viewer request → 403 DENIED (valid user, wrong role).
- Tampered or missing token → 401 (signature checked on every request).
- Same login flow, opposite outcomes — enforced by the server, not the UI.

Results & Evaluation

- 200 vs. 403 — the same request, opposite outcome, once the caller's role changes.
- 0 secrets in code · 0 plaintext passwords stored.
- 5 tables · 3 containers · 2 seeded roles.
- 9.4 s cold boot to first successful login.
- Tested by driving the full allow path and full deny path end-to-end; every decision is written to the audit log.

Program Depth (SO6)

- Selected RS256, bcrypt, and PostgreSQL for real, auditable security guarantees.
- 3 containers across 2 isolated networks — the database is unreachable from the downstream service.
- Secrets injected at runtime; signing keys mounted read-only; zero secrets in source control.
- Versioned migrations, seeded roles, live audit log — one command (docker compose up) boots it all.

What's Next

- Token refresh and revocation lists.
- A roles/permissions management console.
- Finer-grained, resource-scoped permissions.
- TLS termination and rate limiting for production.

People & Acknowledgments

Team Leader: *Carson Farinella*
Team Members: *Carson Farinella, Pablo Murillo, Guilherme Fulco, Noor Almasarweh, Danni Benitez*
Instructor: *Prof. Masoud Sadjadi, FIU KFSCIS Capstone 2*
Acknowledgments: *Prof. Masoud Sadjadi and the KFSCIS Capstone Program.*

Ethics & Professional Practice (SO4)

Security & privacy by design: bcrypt-only credential storage, least-privilege networking, no secrets in source control, demo/seed data only (no real PII). Named limitation: no token revocation before expiry.

References

[1] JWT, RFC 7519, IETF, 2015. [2] PKCS #1 (RSA), RFC 8017, IETF, 2016. [3] Provos & Mazières, “A Future-Adaptable Password Scheme,” USENIX, 1999. [4] PostgreSQL 15 Documentation, PGDG. [5] Docker Compose Documentation, Docker Inc. [6] OWASP Top 10: Broken Access Control, 2021.

Dr. Masoud Sadjadi, Ph.D. — Professor of Capstone 2 · Questions, comments, or follow-up: sadjadi@fiu.edu
FIU Knight Foundation School of Computing & Information Sciences · Capstone Program · capstone.cs.fiu.edu