Policy Management System — Full-Stack Web Application

A concise, enterprise-grade insurance policy & claims management solution designed for reliability, security and developer productivity. Built with Spring Boot 3 (Java 17) on the backend and Angular 13 (TypeScript) on the frontend. Includes CI-ready database versioning, modular architecture and a developer-friendly codebase.

Project Snapshot

Backend: Spring Boot 3

RESTful APIs, Service layer, Spring Data JPA, Lombok, Bean Validation and Liquibase for migrations.

Frontend: Angular 13

Component-driven UI, PrimeNG components, RxJS for reactive flows, route guards and responsive layouts.

Data: H2 + Liquibase

Fast in-memory development DB, formal change-sets for repeatable schema migrations and automated CI scripts.

Target Users & Roles

Admin

Manage users, roles, policies, view audit logs and approve claims.

End User

Submit claims, view policies, update profile and track claim status.

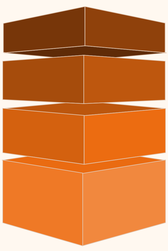
Architecture Overview

Presentation

Angular UI sends requests and collects user input.

Application

Spring Boot REST API processes business logic.



Authentication

Validate credentials and tokens before API access.

Data Layer

H2 Database stores policies and claims.

Presentation (Angular) → REST API (Spring Boot) → Repository (JPA) → H2. Authentication and role validation occur at the API gateway and via Angular Route Guards. Liquibase ensures schema parity across environments.

Core Features

Authentication & Authorization

Secure login, role-based access (ADMIN, USER), token-based session management and server-side endpoint protection.

Policy Management

Full CRUD for policies, versioning, validations and policy lifecycle events.

Claims Processing

Claim submission, status tracking, admin review flows and audit trails.

Interactive Dashboard

Key metrics, recent activity, claim throughput and error monitoring — real-time where feasible.

Technology Details

Backend Highlights

- Spring Boot 3.2.2, Java 17
- Spring Data JPA, DTOs & entity relationships
- Liquibase for deterministic DB migrations
- Validation, exception handling, structured logging

Frontend Highlights

- Angular 13 with TypeScript 4.6
- PrimeNG component library, responsive SASS/CSS
- RxJS-powered state flows and HttpClient services
- Route Guards, reusable components and forms with validation

Security & Database Practices

Role-based access control enforced at both client and server. Hardened endpoints, input validation and structured error responses. Development uses H2; migration to PostgreSQL/MySQL recommended for production. Liquibase ensures repeatable, auditable schema changes.

Operational & Developer Notes

CI / CD

Automated builds, tests, Liquibase checks and containerised deployment pipelines.

Testing

Unit, integration and end-to-end tests for API and UI. Mocked data for isolated development.

Observability

Structured logs, metrics and basic health endpoints; integrate APM for production.

Achievements & Roadmap

Achieved a production-ready layered architecture with secure endpoints, modular frontend and automated DB versioning. Designed for maintainability and rapid feature delivery.

- Clean separation: Controllers → Services → Repositories
- Reusable Angular components and form patterns
- Liquibase ensures migration safety across environments

Next: DB Migration

PDF Reports

Notifications

Document Upload

Closing

Thank you — this implementation balances developer ergonomics with enterprise requirements: secure by design, modular for scale, and ready for production. For implementation details, code samples, or deployment guidance, we can share the repo and run a technical walkthrough.

Contact: Engineering lead | Architecture docs | CI pipeline overview