Web Applications 2

Lab 5: Signup and Observability

Learning Objectives

By successfully completing this lab, students will develop the following skills:

- Enabling signup for users and experts
- Enabling observability
- Using docker compose

Description

In this lab, we will implement a signup method in Kotlin using Spring Boot 3 that communicates with Keycloak for user registration. Additionally, we will add a new endpoint to allow managers to create expert users.

Implementing observability and tracing in our applications helps in monitoring, troubleshooting, optimizing performance, planning capacity, detecting faults, and ensuring the reliability and scalability of your systems. It empowers you to make data-driven decisions, improve system performance, and deliver a better user experience.

We will incorporate observability and tracing using Loki, Tempo, Prometheus and Grafana.

Loki: Loki is a horizontally scalable, highly available, and multi-tenant log aggregation system. It is designed to store and index log data efficiently while providing fast and cost-effective log querying capabilities. It allows to centralize and analyze logs from various sources, enabling effective monitoring, troubleshooting, and debugging of distributed systems.

Tempo: Tempo is an open-source, scalable, and highly efficient distributed tracing backend. It is designed to store and query trace data in a cost-effective and scalable manner. Tempo supports the OpenTelemetry standard for distributed tracing, allowing to instrument your applications and collect trace spans. It provides a unified view of request flows across microservices, enabling end-to-end tracing and analysis. Tempo integrates seamlessly with observability tools like Prometheus and Grafana, allowing you to visualize and analyze trace data alongside other observability metrics.

Prometheus: Prometheus is a popular open-source monitoring and alerting system. It collects time-series data, such as metrics, from various sources, including applications, services, and infrastructure components. Prometheus uses a pull-based model to scrape and store metric data, and it provides a flexible query language for querying and analyzing the collected metrics. Prometheus supports a wide range of exporters, allowing you to collect metrics from different systems. It also provides alerting capabilities to notify you of critical events or threshold breaches.

Grafana: Grafana is an open-source visualization and monitoring platform that provides rich and interactive dashboards for visualizing time-series data. It supports various data sources, including Prometheus, Loki, Tempo, and many others, allowing you to consolidate and visualize data from different systems in a single dashboard. Grafana offers a user-friendly interface for creating and customizing dashboards, as well as advanced features like alerts, annotations, and templating. It is widely used for building monitoring and observability solutions, providing real-time insights into system health and performance.

All those will be deployed using Docker Compose

Steps

1. Starting from lab 4 repository, import the keycloak-admin-client dependency. Implement a @PostMapping method for the /signup endpoint. In the signup method, create a UserRepresentation object and set the user details. Create a CredentialRepresentation object and set the password. Use the Keycloak admin client to create the user in Keycloak and return an appropriate response indicating the signup status.
2. Implement a new @PostMapping method for the /createExpert endpoint adding appropriate security measures to ensure only managers can access this endpoint.
3. Following this blog post, using docker compose deploy the monitoring and observability stack and add micrometer dependencies in order to allow your application to send data to the stack https://medium.com/spring-boot/spring-boot-3-observability-monitor-application-on-the-method-level-8057abec5926

Submission rules

Download a copy of the zipped repository and upload it in the “Elaborati” section of “Portale della didattica”. Label your file “Lab5-Group<N>.zip” (one copy, only - not one per each team member).
Work is due by Friday June 02, 23.59 for odd numbered teams, and by Friday June 09, 23.59 for even numbered ones.