Actor Framework (SAF)
Projet Spring Boot
ING2, GIA, ING2

15 septembre 2025

1 SAF-Résumé

Ce projet vise à concevoir et développer une **plateforme multi-agents distribuée** inspirée du modèle d'acteurs (Akka) mais intégrée dans un écosystème **Spring Boot / microservices**. La plateforme permettra la création, la destruction, la supervision et la communication programmée d'agents. Elle doit être **résiliente**, **responsive**, et **élastique** pour fonctionner en environnement distribué. L'interface utilisateur sera développée en **React**.

2 La plateforme akka

Akka est un framework (Scala/Java) conçu pour développer des applications parallèles, distribuées et tolérantes aux pannes.
Il repose sur le modèle des acteurs, où chaque acteur est une entité indépendante qui échange uniquement par messages, ce qui élimine la complexité des threads et verrous.
Akka peut gérer des milliers d'acteurs en parallèle, offrir une supervision hiérarchique (redémarrage automatique en cas de panne) et s'étendre sur plusieurs machines avec Akka Cluster. Il est particulièrement adapté aux systèmes scalables, réactifs et résilients, comme les applications temps réel, le traitement de flux ou les microservices. En d'autres termes, Akka simplifie la création de systèmes rapides, robustes et évolutifs, grâce à une organisation modulaire et orientée messages.

3 Objectifs fonctionnels principaux du SAF

- Permettre à un utilisateur de créer / détruire dynamiquement des agents.
- Permettre l'envoi/recevoir de messages inter-agents (asynchrone ou synchrone).
- Superviser l'exécution des agents via des threads / superviseurs / stratégies de reprise en cas de panne.
- Supporter la distribution et la tolérance aux pannes (clusterisation, réplication, persistance optionnelle).
- Fournir outils de tests (unitaires, intégration).
- Fournir un exemple d'application distribuée pour valider la plateforme.

4 Périmètre

- Backend : microservices Spring Boot.

- Frontend : application en React (gestion des agents, consoles de logs, visualisation des messages).
- Infrastructure : conteneurisation Docke et orchestration Kubernetes.
- Persistance optionnelle : PostgreSQL / Cassandra selon besoins, et broker de messages.
- Tests : unitaires (JUnit), intégration, tests distribués et tests de résilience.

5 Livrables

- Code source avec README d'installation.
- Images Docker, manifests Kubernetes.
- Application front (React) et backend Spring Boot.
- Scénarios de tests (unitaires et intégration).
- Rapport final
- Démo de l'application que vous aurez choisie.

6 Références / lectures recommandées

- Äkka in Action¨(pour le modèle d'acteurs) ; documentation Akka Typed.
- Documentation Spring Boot, Spring Cloud Gateway.
- Articles et guides sur event sourcing, CQRS, et résilience (Resilience4j).
- Guides Kubernetes (HPA, Pod disruption budgets).