Développement d'un framework d'acteurs coopérant via des microservices avec Spring Boot ,inspiré d'akka
Module JEE, ING2, GI

Septembre 2025

Akka est un framework pour Java et Scala qui permet de développer des applications concurrentes et distribuées en se basant sur le modèle des acteurs, où chaque acteur est une boîte noire et possède son propre état. Il communique uniquement par messages. Akka facilite la gestion de la concurrence, de la tolérance aux pannes et de la scalabilité, grâce à l'isolation des erreurs et à la supervision hiérarchique des acteurs. Il est conçu pour créer des systèmes réactifs, robustes et hautement disponibles, capables de gérer de nombreux utilisateurs ou événements simultanément.

Pour ce projet, il vous est demandé de vous inspirer de la philosophie d'Akka afin de concevoir et développer votre propre framework, dédié aux applications d'acteurs distribuées, en utilisant Spring boot. L'objectif est de comprendre le cœur des applications distribuées. Ne cherchez pas à maîtriser Akka, mais plutôt à capturer sa manière de gérer les acteurs et de les faire communiquer. Vous avez compris : akka sera une source d'inspiration pour vous.

Contexte

Dans les architectures microservices modernes, il est souvent nécessaire de gérer des entités autonomes capables de traiter des messages de manière asynchrone et coopérative. L'objectif de ce projet est de créer un *framework d'acteurs distribués*, où chaque acteur représente une entité capable de recevoir et traiter des messages, tout en communiquant avec d'autres acteurs situés dans le même microservice ou dans des microservices différents. Le système doit être :

1. résilient, c'est-à-dire capable de continuer à fonctionner même en cas de panne d'un acteur ;
2. élastique, c'est-à-dire capable de s'adapter dynamiquement à la charge en créant ou en supprimant des instances d'acteurs.

Objectif général

Développer un framework permettant :
— La gestion des acteurs (création, destruction, blocage, déblocage, ..).
— La communication, qu'elle soit asynchrone ou synchrone, d'une part entre les acteurs d'un même microservice et d'autre part avec ceux créés dans d'autres microservices.
— La supervision et la tolérance aux pannes des acteurs.
— La scalabilité des acteurs.
— Ajouter un système de logs permettant de tracer les activités des actions des utilisateurs.

Exemple concret : Acteur restaurant et acteurs clients

Une application très simple pour tester votre framework pourrait consister en un acteur "restaurant" et des acteurs "clients" qui envoient des commandes à l'acteur restaurant.
— **Microservice Restaurant** :

  - Contient un acteur "RestaurantActor" capable de recevoir des commandes de clients.
  - Expose un endpoint HTTP pour recevoir des commandes provenant d'acteurs clients, créés dans un autre micro-service.
- **Microservice Client** :
  - Crée plusieurs acteurs "ClientActor".
  - Chaque acteur envoie des commandes au microservice Restaurant.

Résumés des fonctionnalités attendues

- Gestion des acteurs intra-microservice et inter-microservices.
- Communication synchrone et asynchrone entre acteurs via des messages.
- Supervision des acteurs pour gérer les erreurs et assurer la tolérance aux pannes.
- Scalabilité.
- Gestion des fichiers log des acteurs.
- Définir une application distribuée, autre que l'application restaurant-clients, pour tester votre application. Attention votre framework est indépendant de l'application que vous aurez choisie.
- Définir des tests untaires et d'intégration.
- Et d'autres fonctionnalités que vous jugerez intéressantes pour un système multi-acteurs.

Technologies proposées

- **Spring Boot** : création des microservices et endpoints REST.
- **Spring Cloud Eureka** : découverte et enregistrement des microservices.
- **RestTemplate / WebClient** : communication HTTP asynchrone entre microservices.
- **Optionnel** : système de messaging (Kafka, RabbitMQ) pour la communication distribuée plus robuste.
- Et autres concepts avancés de Spring boot.

Livrables attendus

- Fournissez le lien Git de votre code.
- Déposez vos slides au format PDF et LaTeX. Vous devez fournir une description concise des concepts avancés de Spring Boot que vous avez utilisés. Vous devez présenter de manière élégante la conception de votre framework, ainsi qu'une description très brève de votre code. Explications des résultats, captures d'écran, analyse des performances ou tests unitaires. N'oubliez pas la liste des références bibliographiques que vous avez utilisées.
- Instructions pour exécuter le projet (README clair).
- Scripts de test, données de test, Postman collections si vous avez des APIs.

Grille d'évaluation du projet

Critères	Points max	Points obtenus
Qualité de l'exposé - Bonne répartition de la parole, présentation fluide	2	
Réponses aux questions - Maitrise du sujet et des concepts, - Apporter des modifications à votre code	3	
Qualité du code (Git) - Clarté, organisation, commentaires, modularités - Documentation README	3	
Slides et présentation - Clarté et concision des slides - Présentation de l'architecture et des concepts	3	
Concepts Spring Boot - Utilisation correcte des annotations et patterns - Fonctionnalités avancées mises en œuvre	2	
Conception du framework / architecture - Diagrammes et schémas clairs - Logique et modularité du design	3	
Tests et démonstration - Tests unitaires et fonctionnels - Exécution et validation de l'application	1	
Références et bibliographie - Citations correctes et complètes	1	