



Kresil - Kotlin Multi-Platform Library for fault-tolerance

Francisco Engenheiro, n.º 49428, e-mail: a49428@alunos.isel.pt, tel.: 928051992

Orientadores: Pedro Félix, e-mail: pedro.felix@isel.pt

Março de 2023

1 Contexto

1.1 Necessidade de Desenho de Software Resiliente

Grande parte dos sistemas contemporâneos são sistemas distribuídos, ou seja, representam um conjunto de computadores independentes entre si, ligados através de uma rede de dados, que se apresentam aos utilizadores como um sistema único e coerente [1]. No entanto, essa interdependência traz consigo o risco de falhas, e quando um desses componentes falha, toda ou parte da funcionalidade do sistema pode ser comprometida. Tal pode resultar em

perda de dados, indisponibilidade de serviços e outros problemas, dependendo da criticidade do(s) componente(s) afetado(s) [2]. É neste contexto que surge a necessidade de desenhar software resiliente, ou seja, software que é capaz de lidar com falhas e manter a sua funcionalidade, mesmo quando um ou mais dos seus componentes falham ou estão temporariamente indisponíveis.

1.2 Bibliotecas como Mecanismos de Resiliência

Fornecem mecanismos para lidar com as eventuais falhas inerentes a componentes de um sistema distribuído, tentando garantir ao máximo a disponibilidade e confiabilidade dos serviços que estes disponibilizam.

Biblioteca	Linguagem	Plataforma
Netflix's Hystrix [3]	Java	JVM
Resilience4j [4]	Java/Kotlin	JVM
Polly [5]	C#	.NET

Tabela 1: Exemplos de bibliotecas como mecanismos de resiliência

Exemplos de mecanismos de resiliência disponibilizados por estas bibliotecas:

- Retry: Tenta novamente uma operação que falhou, aumentando a probabilidade de sucesso;
- Rate Limiter: Limita a taxa de requisições que um serviço pode receber;
- Circuit Breaker: Interrompe, temporariamente, a comunicação com um serviço que está a falhar, de forma a evitar que o mesmo sobrecarregue o sistema. Semelhante a um disjuntor elétrico;
- Fallback: Fornecer um valor ou executa uma ação alternativa caso uma operação falhe.

A biblioteca Polly [5] divide os mecanismos de resiliência que disponibiliza em duas categorias:

- Resiliência Reativa: Reagir a eventuais falhas e mitigar o seu impacto;
- Resiliência Proativa: Prevenir que as falhas aconteçam.

De referir que, as bibliotecas mencionadas anteriormente serão exploradas ao longo da fase de planeamento e desenvolvimento do projeto, de forma a que se possa tirar proveito do conhecimento inerente à sua implementação. Com especial atenção para a biblioteca Resilience4j, uma vez que apresenta um módulo de interoperabilidade com Kotlin para a plataforma JVM.

1.3 Kotlin Multiplatform

1.4 Ktor

Referências

- [1] FreeCodeCamp contributors. A thorough introduction to distributed systems. <https://www.freecodecamp.org/news/a-thorough-introduction-to-distributed-systems-3b91562c9b3c>, 2024. [Online; accessed 5-March-2024].
- [2] Wikipedia contributors. Cap theorem. https://en.wikipedia.org/wiki/CAP_theorem, 2024. [Online; accessed 5-March – 2024].
- [3] Netflix contributors. Hystrix: Latency and fault tolerance for distributed systems. <https://github.com/Netflix/Hystrix>, 2024. [Online; accessed 6-March-2024].
- [4] resilience4j contributors. Resilience4j: User guide. <https://resilience4j.readme.io/docs/getting-started>, 2024. [Online; accessed 6-March-2024].
- [5] App-vNext contributors. Polly: Resilience strategies. <https://github.com/App-vNext/Pollyresilience-strategies>, 2024. [Online; accessed 6-March-2024].