CURSO DE SISTEMAS DE INFORMAÇÃO – TCC ACADÊMICO	
(X) PRÉ-PROJETO ( ) PROJETO	ANO/SEMESTRE: 2021/2

FRAMEWORK DE ENGENHARIA DO CAOS: ESTUDO DE CASO SUPER BREAKOUT

Lucas Vanderlinde

Prof. Aurélio Faustino Hoppe – Orientador

1 INTRODUÇÃO

Segundo Oliveira (2002), a larga disseminação dos computadores a partir do surgimento dos microcomputadores na década de 90 impulsionaram a criação de redes de computadores principalmente para o compartilhamento de recursos. A internet pode ser entendida como um grande e disperso sistema distribuído que permite o acesso a determinados serviços como www, email e transferência de arquivo. Então, pode-se definir um sistema distribuído como aquele "formado por componentes de hardware e software, situados em redes de computadores, e que se comunicam e coordenam suas ações apenas através de trocas de mensagens" (COULOURIS, 2001).

De acordo com Espindola *et al*. (2005), distribuir os processos de desenvolvimento é atualmente uma prática comum. Por isso, tem-se criado um cenário no qual projetos de software são desenvolvidos com equipes distribuídas, caracterizando assim o Desenvolvimento Distribuído de Software (DDS). Segundo Gomes (2013), a utilização de DDS proporciona benefícios como a diminuição de riscos e aumento na produtividade, trazendo uma maior vantagem competitiva associada ao custo, qualidade, flexibilidade e desenvolvimento contínuo para as organizações.

Rosenthal *et al*. (2017, p. 1) ressalta que o número de coisas que podem dar errado com sistemas distribuídos é enorme por possuírem tantos componentes e interações. Podem ocorrer falhas na rede de internet, problemas no disco rígido ou até mesmo um aumento repentino no tráfego do cliente etc. Ainda segundo os autores, nunca será possível evitar todas as falhas, mas pode-se identificar muitos dos pontos fracos do sistema de forma que eles possam ser prevenidos.

Como uma estratégia de identificação e análise dos pontos fracos de um sistema, a engenharia do caos torna-se um método de experimentação em infraestrutura (ROSENTHAL *et al*. 2017. p. 1). Já para a Principle of Chaos (2018), a engenharia do caos é uma prática poderosa que aborda especificamente a incerteza sistêmica nos sistemas distribuídos. Quanto

mais difícil for a possibilidade de causar impactos ao estado normal, mais confiança se terá no sistema. O autor também destaca que quando uma fraqueza ou falha são descobertas, é necessário defini-la como uma nova meta de melhoria antes que essa falha se manifeste no sistema.

Severo Júnior (2021) explica que as falhas em um componente, mesmo que pequenas, possuem um grande potencial de destrutividade ao se espalharem e se tornarem defeitos. Ou seja, com os sistemas cada vez maiores e mais complexos, aumentam-se paralelamente os riscos e a necessidade de confiabilidade. Por isso é importante a constante validação dos componentes adotando estratégias que mitigam os impactos das falhas.

Rossi (2021) ressalta a necessidade de monitoramento, definindo-a como um aspecto obrigatório para saber quando um serviço pode falhar ou uma máquina cair. O autor também destaca algumas ferramentas como Elastic, Logstash e Kibana, consideradas por ele como principais para monitoramento, que reúnem vários *logs*, métricas e vestígios das aplicações, permitindo o monitoramento e reações de forma mais efetiva.

Diante deste contexto, este trabalho visa desenvolver um framework de Engenharia do Caos aplicado a um sistema distribuído implantado com Kubernetes, definido por Matos (2018), como um sistema de código aberto para gerenciamento de aplicativos em containers através de múltiplos hosts de um cluster, facilitando a implantação de aplicativos baseados em microsserviços. O caos aplicado ao sistema distribuído será representado de forma gameficada através do jogo Super Breakout, combinando a natureza destrutiva do jogo aos experimentos/ataques que serão realizados, sendo todos os experimentos monitorados no cluster sobre sua disponibilidade e capacidade.

1.1 OBJETIVOS

O objetivo principal deste trabalho é desenvolver um framework de engenharia do Caos que possibilite avaliar a resiliência de um sistema distribuído.

O trabalho será composto pelos seguintes objetivos específicos:

a) adaptar o jogo Super Breakout para utilizá-lo nos experimentos do caos;
b) abordar de forma gameficada a aplicação de Engenharia do Caos a um sistema distribuído;
c) identificar possíveis fraquezas de uma arquitetura distribuída.

2 TRABALHOS CORRELATOS

A seguir estão descritos três trabalhos correlatos que possuem uma proposta semelhante a que será desenvolvida neste trabalho, pois seguem o método da engenharia do caos através de diferentes abordagens. A seção 2.1 descreve como Jernberg (2020) construiu um framework utilizando os conceitos e técnicas da engenharia do caos. Na seção 2.2 é descrito a ferramenta construída por Monge e Matók (2020) que analisa as falhas de um sistema distribuído e os principais fatores que fazem a engenharia do caos reduzi-las. Por fim, a seção 2.3 relata a experiência de Kesim (2019) em utilizar a ferramenta Chaos Toolkit para aplicar experimentos do caos.

2.1 BUILDING A FRAMEWORK FOR CHAOS ENGINEERING

Jernberg (2020) propôs um framework utilizando os conceitos e técnicas da Engenharia do Caos para avaliar e melhorar a resiliência do site ica.se desenvolvidos na ICA Gruppen AB (ICA). A ICA é um grupo de empresas cujo negócio principal é o varejo de alimentos, sendo utilizado como suporte e orientação para as equipes de desenvolvimento da ICA.

O paradigma de pesquisa de Jernberg (2020), consistiu-se de três tipos de atividades: conceitualização do problema, design da solução e validação empírica. A contextualização foi realizada através de um estudo sobre Engenharia do Caos e como ela poderia ser aplicada. Foram feitas entrevistas com desenvolvedores da ICA para entender como o departamento de TI trabalha e onde utilizar Engenharia do Caos. No design da solução utilizou-se o mesmo padrão de pesquisa, mas para buscar onde era adequado aplicar Engenharia do Caos na ICA. Posteriormente, pesquisou-se ferramentas de Engenharia do Caos que seriam apropriadas para o framework desenvolvido. O autor apresentou para a equipe da ICA o que uma ferramenta de Engenharia do Caos poderia realizar, aplicando em seguida um questionário para entender se a Engenharia do Caos poderia ser aplicável na ICA e sobre o quão extensos eram os benefícios percebidos pelos participantes sobre a Engenharia do Caos.

Na validação empírica, utilizou-se uma versão inicial do framework para testar a viabilidade da arquitetura, demonstrando a aplicação da Engenharia do Caos para as partes interessadas. Por fim, os participantes, apontaram a partir de suas experiências melhorias a serem realizadas ou suas dificuldades.

Após o entendimento do cenário de pesquisa, Jernberg (2020) propôs 4 atividades, sendo que cada atividade possui documentos que são usados como base para a sua realização (no total são 9), junto com 12 ferramentas de engenharia do caos que passaram por um processo de seleção e avaliação dentre 27 ferramentas de código aberto. As ferramentas selecionadas foram

Chaos Toolkit, ChaoSlingr, WireMock, Muxy, Toxiproxy, Blockade, Chaos Monkey for Spring Boot, Byte-Monkey, GomJabbar, Litmus, Monkey-Ops, Chaos HTTP Proxy. As atividades propostas no framework são descoberta, implementação, sofisticação e expansão. A descoberta cria um acúmulo de Experimentos do Caos que são possíveis e adequados para serem executados para o aplicativo em teste e a implementação configura e executa apenas um Experimento do Caos. A sofisticação verifica a validade e segurança dos Experimentos do Caos e a expansão adiciona o princípio de aumentar a implementação da Engenharia do Caos de forma incremental ao framework.

Os testes realizados por Jernberg (2020) no framework, ocorreram em duas partes, primeiro durante o seu desenvolvimento na parte de validação empírica da pesquisa, foram realizados testes em aplicativos de amostra com versões iniciais do framework. Para os testes realizados com a versão final do framework, foram aproveitados os profissionais da ICA que não participavam diretamente no desenvolvimento ou teste dos softwares. Jernberg (2020) optou por introduzir apenas a ferramenta Chaos Tooltik na utilização do framework dentro da ICA pois a introdução das 12 ferramentas no mesmo momento teria um grande impacto nos times da organização. O autor relata que durante os testes nenhum dos participantes encontrou alguma parte ausente ou redundante na estrutura do framework. Além disso, todos os comentários indicam que a estrutura mostrou-se viável.

Segundo Jernberg (2020), o framework trouxe benefícios como introduzir maneiras mais simples para as equipes de desenvolvimento começarem a utilizar ou a implementar a Engenharia do Caos. O framework trouxe também uma base comum de conhecimento o que permite diferentes pessoas falarem a mesma língua sobre o tema. Jernberg (2020) sugere a utilização de outras ferramentas, verificando quais poderiam ser utilizadas nas equipes de desenvolvimento da ICA. Além disso, realizar a validação de todas as atividades presentes no framework pois o tempo de realização do projeto não permitiu a validação de todas as atividades apenas a parte da descoberta e implementação.

2.2 DEVELOPING FOR RESILIENCE: INTRODUCING A CHAOS ENGINEERING TOOL

Monge e Matók (2020) analisaram as falhas de um sistema distribuído e os principais fatores que fazem a Engenharia do Caos reduzi-las. Os autores propuseram a utilização de uma metodologia chamada de Ciência do Design para Metodologia de Pesquisa em Sistema de Informação de Peffers composta de seis passos: problematização e motivação, definição dos

objetivos para a solução, design e desenvolvimento, demonstração, avaliação e comunicação, permitindo desenvolver e avaliar a eficácia da geração do caos.

Segundo Monge e Matók (2020), definiu-se quais funcionalidades eram desejadas, buscando identificar relacionamentos e componentes do sistema assim como, sua arquitetura. Para o desenvolvimento da ferramenta utilizou-se o framework Spring Boot com o módulo Spring Boot Starters Application que permite adicionar dependências Maven ou Gradle de outros projetos Java em Spring Boot. Tendo a ferramenta desenvolvida, iniciou-se o processo de experimentação, visando demonstrar como o artefato pode resolver diferentes instâncias do problema.

Monge e Matók (2020) realizaram os testes em um ambiente chamado por eles de "ambiente de preparação". Este ambiente é utilizado pelos desenvolvedores para testar as funcionalidades da ferramenta desenvolvida localmente. O ambiente de preparação simula a comunicação com dispositivos móveis onde suas requisições podem ser encaminhadas para o simulador que realiza o retorno das respostas. Monge e Matók (2020) mapearam todos os ataques implementados pela ferramenta de Engenharia do Caos, sendo executados no ambiente de preparação. Os testes abordaram 19 experimentos nos quais foram validadas a resiliência da ferramenta à latência introduzida artificialmente, valores defeituosos nas requisições, campos ausentes e extras nas requisições, requisições com falha, alto uso de memória, alto uso da CPU e a interromper a execução de ataques.

Monge e Matók (2020) apresentaram uma nova abordagem de como se projetar e arquitetar uma ferramenta de Engenharia do Caos utilizando experiências de praticantes para a implementação dos recursos em sua ferramenta, como por exemplo, a observabilidade sobre as operações, propriedades configuráveis e um "botão de parada de emergência". A ferramenta desenvolvida também provou ser útil para o teste de software pois facilita o trabalho de identificar falhas de tratamento e as fraquezas de um software. Por fim, segundo Monge e Matók (2020), a ferramenta desenvolvida pode ser ampliada em diversas maneiras como adicionar outras maneiras de transmissão de mensagens, conteúdos e tipos, melhorar o controle do usuário sobre os ataques ou até mesmo a automação para a realização de ataques randômicos.

2.3 ASSESSING RESILIENCE OF SOFTWARE SYSTEMS BY APPLICATION OF CHAOS ENGINEERING – A CASE STUDY

Kesim (2019) analisou o planejamento e realização de experimentos do caos, visando compreender o seu comportamento em um sistema distribuído, sendo apoiado por métodos de análise de risco. Segundo o autor, para entender o funcionamento do sistema, realizou-se

levantamento sobre a arquitetura, modelagem e comportamento do sistema assim como, informações de desenvolvedores e dados de arquivos. As entrevistas foram transcritas e comparadas para não gerar repetição de informação. Cada experimento do caos foi avaliado considerando testes de hipótese, verificando se o software satisfez o que foi previsto ou não, e através de dados estatísticos via JMeter, que é responsável por montar um ambiente de estresse com o software alvo para simular o comportamento do usuário.

A partir dos resultados obtidos nas entrevistas, Kesim (2019) desenvolveu um protótipo sob uma arquitetura de microsserviços utilizando componentes do Kubernetes, utilizando o Postgres SQL como banco de dados e o protocolo HTTP como serviço de comunicação juntamente com a arquitetura REST.

Para hospedar o protótipo, Kesim (2019) optou pela plataforma Microsoft Azure utilizando o serviço Azure Kubernetes Service (AKS), configurando a ferramenta JMeter para as análises estatísticas. Já os experimentos, segundo o autor, foram realizados utilizando a ferramenta Chaos Toolkit observando os resultados da análise de risco. Para observar um impacto potencial no estado estacionário do sistema, os resultados dos experimentos de engenharia do caos foram analisados aplicando o teste de hipótese de Kolmogorov-Smirnov.

No total foram realizados 4 experimentos, obtendo sucesso nos dois primeiros. Já no terceiro foi detectada uma fraqueza, no qual, antes de executar o experimento no pod de configuração ao banco de dados ocorreu um erro de falta de memória e o sistema não conseguiu se recuperar para o estado estável. Um pod do Kubernetes é um conjunto de um ou mais containers Linux, sendo a menor unidade de uma aplicação Kubernetes. O quarto foi desconsiderado pois é recomendado consertar qualquer fraqueza antes de realizar novos experimentos. O primeiro teste foi sobre a hipótese de que os tempos de resposta não aumentariam após matar um pod do Kubernetes, o segundo foi mais destrutivo, ele eliminou todos os pods de microsserviços de configuração disponíveis e no terceiro foi adicionado um objeto de configuração ao banco de dados (ID = 2) solicitando a API do administrador e encerrado o objeto de configuração inicial (ID = 1).

Kesim (2019) conseguiu aplicar com sucesso meios para identificar fraquezas e falhas potenciais na arquitetura do sistema e os resultados da análise de risco implicaram que o sistema alvo é considerado bastante frágil. As principais dificuldades encontradas foram em avaliar os resultados sem métricas de resiliência bem definidas, pois os meios existentes para determinar se um experimento do caos teve impacto significativo não são transparentes, faltam mecanismos de monitoramento adequados. Por fim, Kesim (2019) aponta que a visualização é um importante campo a ser explorado pois está intimamente ligado ao aspecto de

monitoramento e que ferramentas que aplicam a engenharia do caos poderiam ter seu uso avaliado além da exploração de configurações padrões de resiliência.

3 PROPOSTA DO FRAMEWORK

A seguir será descrito qual foi a motivação para o desenvolvimento deste trabalho em conjunto com seus principais pilares e a relações entre os trabalhos correlatos a este que que descrevem a fundamentação necessária para exemplificar o tema abordado.

3.1 JUSTIFICATIVA

O Quadro 1 detalha de forma comparativa a relação entre os trabalhos correlatos que serão utilizados para dar embasamento à proposta deste projeto, onde as linhas representam as características e as colunas os trabalhos.

Quadro 1 – Comparativo entre os trabalhos correlatos

Correlatos / Características	Jernberg (2020)	Monge e Matók (2020)	Kesim (2019)
Objetivo	Construir um framework de Engenharia do Caos robusto e com um longo período de utilização	Analisar falhas de um sistema distribuído e os benefícios da Engenharia do Caos através da construção de uma ferramenta	Analisar o planejamento e a experimentação da Engenharia do Caos
Cenário de aplicação	Novos aplicativos em versões iniciais e no site ica.se desenvolvidos na ICA Gruppen AB	Ambiente de preparação ou de pré-produção que simula a comunicação com um dispositivo móvel	Protótipo desenvolvido pelo autor
Ferramentas utilizadas	Chaos Toolkit	Própria	Chaos Toolkit
Arquitetura utilizadas	Não foi implementada uma ferramenta de Engenharia do Caos	Linguagem Java utilizando Framework Spring Boot com o módulo Spring Boot Starters Application	Não foi implementada uma ferramenta de Engenharia do Caos

Fonte: elaborado pelo autor.

A partir do Quadro 1 é possível identificar que todos os trabalhos têm como objetivo a implantação da Engenharia do Caos em um ambiente no qual ela não existia previamente. Percebe-se que o desenvolvimento de uma ferramenta de Engenharia do Caos palpável foi abordado apenas por Monge e Matók (2020). Já Kesim (2019) e Jernberg (2020) utilizam uma ferramenta Engenharia do Caos de código aberto já existente, a Chaos Toolkit.

Já os cenários no qual a solução ou o estudo proposto foi aplicado se divergiram nos três

trabalhos. Jernberg (2020) optou por aplicá-la com foco no site ica.se que é um site para buscar receitas desenvolvido no ICA Gruppen AB (ICA), mas nas primeiras versões do framework foram utilizados aplicativos em versões iniciais para teste. Monge e Matók (2020) visavam promover a identificação e análise de falhas em um sistema distribuído. Já Kesim (2019) desenvolveu um protótipo com base em um sistema distribuído para analisar e planejar a experimentação da Engenharia do Caos em um software em desenvolvimento.

Apenas Monge e Matók (2020) desenvolveram uma ferramenta que aplica o Caos em um sistema distribuído. O framework de Jernberg (2020) é uma metodologia de trabalho para a utilização de ferramentas já existentes e Kesim (2019) desenvolveu um protótipo de sistema distribuído, aplicando o Caos com o uso da ferramenta Chaos Tooltik. O desenvolvimento da ferramenta de Monge e Matók (2020) utilizou a linguagem Java e framework Spring Boot com o módulo Spring Boot Starters Application, uma arquitetura robusta e bastante difundida no mercado.

A partir deste contexto, percebe-se que na engenharia de software moderna, os sistemas distribuídos, experimentação e confiabilidade são pontos cruciais no seu desenvolvimento. Diante disso, este trabalho torna-se relevante pois busca agregar a essa nova área uma ferramenta que forneça confiabilidade e resiliência a sistemas distribuídos através da aplicação da Engenharia do Caos. Para isso, será utilizada a linguagem Java com o framework Spring Boot para o desenvolvimento da ferramenta, sendo responsável por realizar experimentos do caos em conjunto com os serviços da Google Cloud Plataform (GCP) para Kubernetes. Será adaptado o jogo Super Breakout para comunicar com o framework conectando a destruição dos objetos no jogo a ataques realizados ao sistema alvo, tornando a experiência da engenharia do caos mais atrativa, pois os experimentos ocorrerem em paralelo as ações no jogo. Os resultados dos experimentos poderão ser avaliados no Kubernetes Engine Monitoring (GKE), oferecido pela plataforma Google, que agrega registros, eventos e métricas do ambiente monitorado auxiliando na compreensão do comportamento do sistema alvo. O sistema alvo será selecionado podendo ser qualquer sistema distribuído que utilize Kubernetes. Contudo, este trabalho torna-se importante pois irá validar o conceito de engenharia do caos, em um contexto de jogo, realizando experimentos do caos de forma controlada e monitorada em um sistema distribuído.

3.2 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O framework a ser desenvolvido neste trabalho deverá:

a) permitir que a infraestrutura seja criada e destruída de forma automatizada (Requisito Funcional - RF);

b) utilizar uma arquitetura de microsserviços e um orquestrador de contêineres (RF);
c) permitir a configuração do cluster Kubernetes (RF);
d) gerenciar o ciclo de vida e orquestrar os contêineres dos serviços (RF);
e) aplicar a injeção de falhas ao Super Breakout (falhas de hardware, parada de servidores, falhas de software e falhas de rede) (RF);
f) avaliar a resiliência a partir das métricas (Rolling Update, Liveness Probe, Retry, Time Out e Circuit Breaker) utilizando o Kubernetes Engine Monitoring (RF);
g) utilizar a linguagem Lua para adaptar uma versão de código aberto do jogo Super Breakout (Requisito não Funcional - RNF);
h) utilizar a linguagem Java com Spring Boot para desenvolver o framework (RNF);
i) utilizar a plataforma Google Cloud Platform (RNF).

3.3 METODOLOGIA

O trabalho será desenvolvido observando as seguintes etapas:

a) levantamento bibliográfico: realizar levantamento bibliográfico sobre resiliência, Engenharia do Caos e suas ferramentas. Estudar o Kubernetes e a plataforma Google Cloud Platform para fornecer a infraestrutura necessária para hospedar uma aplicação de sistema distribuído a ser utilizada nos experimentos, assim como pesquisar trabalhos correlatos;
b) elicitação de requisitos: com base nas informações da etapa anterior, realizar reavaliação dos requisitos, e caso necessário, especificar novos requisitos a partir das necessidades identificadas a partir da revisão bibliográfica;
c) adaptação do jogo Super Breakout: alterar o código fonte do jogo para permitir a inserção de falhas utilizando a linguagem de programação Lua;
d) especificação do framework: formalizar as estruturas e evoluções da arquitetura do framework através de ferramentas de diagramação Lucidchart e Cloudcraft para elaborar o diagrama de estrutura de acordo com a Unified Modeling Language (UML);
e) implementação: implementar o framework utilizando a linguagem de programação Java com Spring Boot no ambiente de desenvolvimento Visual Studio. Desenvolver e hospedar a arquitetura de microsserviçoes na plataforma Google Cloud. Para tanto, a partir dos itens (c) e (d), para cada fraqueza identificada na arquitetura do framework, documentar e reprojetar a solução para assegurar maior resiliência;
f) testes: para cada hipótese de fraqueza da arquitetura, validar a eficiência da

resiliência do sistema através dos experimentos projetados pela engenharia do caos, a partir do jogo Super Breakout. Deverá ser utilizado ferramentas específicas de engenharia do caos para maior amplitude e assertividade dos testes.

As etapas serão realizadas nos períodos relacionados no Quadro 2.

Quadro 2 – Cronograma de atividades a serem realizadas

	2022									
	fev.		mar.		abr.		maio		jun.	
etapas / quinzenas	1	2	1	2	1	2	1	2	1	2
levantamento bibliográfico										
elicitação dos requisitos										
adaptação do jogo Super Breakout										
especificação do framework										
implementação										
testes										

Fonte: elaborado pelo autor

4 REVISÃO BIBLIOGRÁFICA

Este capítulo tem como objetivo explorar brevemente os conceitos e fundamentos mais importantes para a realização deste trabalho: resiliência em sistemas computacionais e engenharia do caos.

Segundo Severo Júnior (2021), a resiliência embora esteja cada vez mais se tornando algo comum em uma variedade de campos, a sua definição e medição varia muito com o domínio do problema. A resiliência foi definida por Wu (2013, p. 1) como “a capacidade de se adaptar com sucesso em face do estresse e adversidade”. Deconti (2021) conta que a resiliência não se aplica somente às situações extremas. Uma série de situações de rotina podem ter como sua resposta a resiliência.

A resiliência implica na contenção das falhas para que elas não se tornem erros. Por isso é importante tratá-las e prevê-las para que não “se espalhem”. Em sistemas complexos é importante cuidar dos pontos de integração mitigando impactos de falhas, erros ou defeitos de um componente nos demais (SEVERO JÚNIOR, 2021).

Segundo Basiri (2017), sendo a utilização da arquitetura de sistemas distribuídos comumente implementada no desenvolvimento de aplicações que possuem alta complexidade atualmente, a engenharia do caos tem o papel de garantir a resiliência desses sistemas. A injeção de falhas, como experimentação da engenharia do caos, de forma empírica foca em identificar possíveis falhas ocultas no sistema.

De acordo com Rosenthal *et al*. (2017), a engenharia do caos é utilizada para expor fraquezas desconhecidas em um sistema de produção. Quando se tem certeza de que um

experimento do caos acarretará um problema significativo não faz sentido a utilização da engenharia do caos. Por isso primeiro deve ser corrigidas as fraquezas conhecidas nos serviços. Depois isso é importante definir um "sistema estável", uma medida que indica o comportamento normal do sistema para então dar início a construção de experimentos do caos (PRINCIPLE OF CHAOS, 2018).

REFERÊNCIAS

BASIRI, Ali *et al*. **Chaos Engineering.** 2017. Disponível em: <https://www.infoq.com/articles/chaos-engineering>. Acesso em 27 ago. 2021.

COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. **Distributed Systems: Concepts and Design**. 3. ed. Boston: Addison Wesley, 2000. 800 p.

DECONTI, Rosemeire. **Criando sistemas resilientes**. 2021. Disponível em: <https://digitalinnovation.one/artigos/criando-sistemas-resilientes/ >. Acesso em: 05 set. 2021.

ESPINDOLA, Rodrigo *et al*. **Uma Abordagem Baseada em Gestão do Conhecimento para Gerência de Requisitos em Desenvolvimento Distribuído de Software**: rafael prikladnick. 2005. 13 f. Monografia (Especialização) - Curso de Ciência da Computação, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre.

GOMES, Vanessa M. **Um Estudo Empírico Sobre o Impacto da Confiança no Desempenho de Projetos Distribuídos de Desenvolvimento de Software**. 2013. 110 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Pontifícia Universidade Católica do Rio Grande do Su, Porto Alegre.

JERNBERG, Hugo. **Building a Framework for Chaos EngineeringBuilding a Framework for Chaos Engineering**. 2020. 108 f. Dissertação (Doutorado) - Curso de Ciência da Computação, Departamento de Ciência da Computação, Universidade de Lund, Lund.

KESIM, Dominik. **Assessing Resilience of Software Systems by Application of Chaos Engineering – A Case Study**. 2019. 187 f. TCC (Graduação) - Curso de Engenharia de Software, Universidade de Estugarda, Estugarda.

MATOS, David. **Kubernetes: Pods, Nodes, Containers e Clusters.** 2018. Disponível em: <https://www.cienciaedados.com/kubernetes-pods-nodes-containers-e-clusters>. Acesso em: 08 set. 2021.

MONGE, Ignacio; MATÓK, Enikő. **Developing for Resilience**: Introducing a Chaos Engineering tool. 2020. 93 f. Dissertação (Mestrado) - Curso de Faculdade de Tecnologia e Sociedade, Departamento de Ciência da Computação e Tecnologia de Mídia, Universidade de Malmö, Malmö.

OLIVEIRA, Rômulo. **Programação em Sistemas Distríbuidos**. Florianópolis: Escola de Informática da Sbc-Su, 2002. 49 p. Disponível em: http://www.romulosilvadeoliveira.eng.br/artigos/Romulo-Joni-Montez-Eri2002.pdf. Acesso em: 06 out. 2021.

PRINCIPLE OF CHAOS. **Princípios de Chaos Engineering**. Disponível em: <http://principlesofchaos.org/?lang=PTBRcontent>. Acesso em: 16 set. 2019.

ROSENTHAL, Casey *et al*. **Chaos Engineering:** Building Confidence in System Behavior through Experiments. O'Reilly, 2017.

ROSSI, Rodrigo. **Entrando no Mundo de Microsserviços:** Parte 1. 2021. Disponível em: < https://www.linkapi.solutions/blog/entrando-no-mundo-de-microsservicos-parte-1/>. Acesso em 04 set. 2021.

SEVERO JÚNIOR, Elemar R. **Fundamentos para arquiteturas de sistemas resilientes**. 2021. Disponível em: < https://arquiteturadesoftware.online/fundamentos-para-arquiteturas-de-sistemas-resilientes-capitulo-13-v-1-01/#Taticas_para_previnir_falhas/>. Acesso em 04 set. 2021.

WU, Gang *et al*. Understanding stress resilience: understanding resilience. **Behavioral Neuroscience**. Boulder, p. 1-1. 15 fev. 2013. Disponível em: https://www.frontiersin.org/articles/10.3389/fnbeh.2013.00010/full. Acesso em: 05 out. 2021.