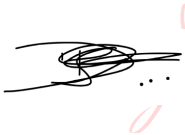

Digitally signed by Dalton Solano dos Reis:66568803987
Date: 2021.12.18 16:54:09 -03'00'

Revisão do Projeto

Disciplina: Trabalho de Conclusão de Curso I – SIS

Caro orientando,

segue abaixo a tabela de cálculo da média das notas obtidas no Pré-Projeto e Projeto, as DUAS revisões do seu projeto contendo a avaliação do professor "avaliador" e professor "TCC1". Lembro que os ajustes indicados nestas revisões não precisam ser feitos no projeto, mas sim quando levarem o conteúdo do projeto para o artigo (se for o caso). Este material contendo todo o histórico das revisões é encaminhado para o professor de TCC2.

Atenciosamente,

	PreProjeto										Projeto										
	TCC1					Avaliador					TCC1					Avaliador					
Nome	A	P	N		**Nota**	A	P	N		**Nota**	A	P	N		**Nota**	A	P	N		**Nota**	**Média**
LucasVanderlinde	18	2	0	20	9,67	9	6	0	15	8,67	20	0	0	20	10,00	15	0	0	15	10,00	9,7

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

FRAMEWORK DE ENGENHARIA DO CAOS: UMA ABORDAGEM GAMIFICADA PARA A ENGENHARIA DO CAOS

Lucas Vanderlinde

Prof. Aurélio Faustino Hoppe – Orientador

Comentado [MH1]: Título foi alterado. Mas a repetição de Engenharia do Caos não ficou legal.
Quem sabe: Framework de Engenharia do Caos: uma abordagem gamificada utilizando Super Breakout.

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, e-mail 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; DOLLIMORE; KINDBERG, 2001).

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. O caos aplicado ao sistema distribuído será representado de forma gamificada através do jogo Super Breakout, combinando a natureza destrutiva do jogo aos experimentos/ataques que serão realizados. Destaca-se que todos os experimentos serão monitorados no cluster buscando identificar e caracterizar as fraquezas que possam aferir a confiança e resiliência do sistema.

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 gamificada a aplicação de Engenharia do Caos a um sistema distribuído;
c) identificar possíveis fraquezas de uma arquitetura distribuída baseada em Kubernetes.

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 em três atividades: conceitualização do problema, design da solução e validação empírica. A conceitualização 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 Tecnologia de Informação (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 e 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 Design Science for Research Methodology in Information System (DSRM) de Peffers *et al.* (2007), 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 localmente as funcionalidades da ferramenta desenvolvida. 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 Unidade Central de Processamento (Central Processing Unit - 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 observação 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, onde cada hipótese permite identificar uma possível fraqueza na arquitetura. Kesim (2019) também verificou se o software possuía fragilidade ao ser testado com um conjunto dados estatísticos

Comentado [MH2]: Onde significa lugar. Troque por "em que"

produzidos pela ferramenta 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 Hyper Text Transfer Protocol (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: antes de executar o experimento no `pod`[1] de configuração do banco de dados ocorreu um erro de falta de memória e o sistema não conseguiu se recuperar para o estado estável. 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 eliminando todos os `pods` de microsserviços de configuração disponíveis. No terceiro foi adicionado um objeto de configuração ao banco de dados (`ID = 2`) solicitando a Application Programming Interface (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 concluíram 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.

[1] Pod no Kuberponetes é um conjunto de um ou mais containers Linux, são usados como a unidade de replicação no Kubernetes (MATOS, 2018)

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 descrevem a fundamentação necessária para exemplificar o tema abordado.

3.1 JUSTIFICATIVA

O Quadro 5 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 5 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 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) se 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 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. Os experimentos ocorrerão em paralelo às ações no jogo proporcionando uma experiência da engenharia do caos mais compreensível e divertida de forma gamificada. Os resultados dos experimentos poderão ser avaliados no Kubernetes Engine Monitoring, oferecido pela plataforma Google Kubernetes Engine (GKE), 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. A importância deste trabalho está em validar o conceito de engenharia do caos em um sistema distribuído de forma gamificada. Monitorando os experimentos do caos que ocorrem em paralelo ao “caos” do jogo Super Breakout.

Comentado [MH3]: , monitorando

3.2 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O *framework* a ser desenvolvido neste trabalho deverá:

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

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

3.3 METODOLOGIA

O trabalho será desenvolvido observando as seguintes etapas:

m) 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;
n) 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;
o) especificação do *framework*: formalizar as estruturas e evoluções da arquitetura do *framework* através de ferramentas de diagramação Lucidchart e Cloudcraft para elaborar os diagramas de classes, sequência e atividades de acordo com a Unified Modeling Language (UML);
p) adaptação do jogo Super Breakout: alterar o código fonte do jogo, utilizando a linguagem de programação Lua, para que as ações de destruição de blocos do jogo provoquem falhas no sistema distribuído alvo do experimento do caos, por meio do *framework;*
q) implementação: implementar o *framework* utilizando a linguagem de programação Java com Spring Boot no ambiente de desenvolvimento Visual Studio;

r) infraestrutura cloud: desenvolver e hospedar a arquitetura de microsserviços na plataforma Google Cloud utilizando Kubernetes;

s) refatoração do sistema distribuído: a partir dos itens (d), (e) e (f), para cada fraqueza identificada na arquitetura do sistema distribuído, documentar e refatorar a solução para assegurar maior resiliência;

t) 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 6.

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										
especificação do *framework*										
adaptação do jogo Super Breakout										
implementação										
infraestrutura cloud										
refatoração do sistema distribuído										
testes										

Fonte: elaborado pelo autor

Comentado [MH4]: Diferentes tamanhos de fonte ao longo do quadro.

4 REVISÃO BIBLIOGRÁFICA

Esta seção tem como objetivo explorar os conceitos mais importantes para a realização deste trabalho. A subseção 4.1 apresenta o contexto da resiliência em sistemas computacionais. A subseção 4.2 contextualiza a engenharia do caos e de que forma ela consegue contribuir para o desenvolvimento de uma arquitetura distribuída de alta resiliência. Por fim, a subseção 4.3 conceitua gamificação.

4.1 RESILIÊNCIA

Segundo Georger *et al*. (2014) 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 *et al.* (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.

> Mesmo nas melhores circunstâncias, projetar um sistema para resiliência é uma tarefa assustadora devido à variedade de questões que precisam ser abordadas. Por exemplo, os envelopes de design precisam ser entendidos com antecedência e, quando um sistema falha, deve fazê-lo de maneira previsível, detectável e elegante para aumentar a capacidade de sobrevivência geral do sistema (ou seja, a capacidade de sobrevivência dos usuários e dos outros subsistemas do sistema). (GEORGER *et al*. 2014).

Comentado [MH5]: Deveria também informar a página.

Segundo Pereira (2021), "no melhor cenário, a aplicação se recupera sem o usuário/cliente perceber. No pior cenário, a aplicação oferece serviços de forma limitada (que chamamos de *graceful degradation*)". 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).

Para prevenir e mitigar o surgimento de tantos problemas quanto possível, LeRoy (2017) demonstra três estratégias que são prevenção, mitigação e híbrido. A prevenção envolve o teste do sistema, ou partes dele, antes de liberá-lo para a produção. Os principais testes a serem realizados são de correção funcional; capacidade de desempenho sob carga esperada; e resiliência a falhas previsíveis. Já a mitigação, LeRoy (2017) explica que envolve limitar a amplitude do impacto através do isolamento e da degradação harmoniosa (*graceful degradation*). Envolve limitar também a duração do impacto e melhorar o tempo de resposta, diagnóstico e correção. A estratégia híbrida abrange a prevenção e mitigação como a liberação de versões canário para um subconjunto de usuários realizada na produção. Outra forma seria a técnica avançada da Engenharia do Caos para testar e praticar abordagens de prevenção e mitigação em um ambiente de produção.

Tão importante quanto garantir a existência dos recursos necessários para atender as demandas de performance é também adotar práticas para prevenir e mitigar a propagação de falhas (SEVERO JÚNIOR, 2021). "Ser **resiliente** é aceitar que falhas irão acontecer e tratá-las da melhor maneira." (PEREIRA, 2021, grifo do autor).

4.2 ENGENHARIA DO CAOS

Assim como os cientistas conduzem experimentos para estudar fenômenos físicos e sociais, a Engenharia do Caos é uma abordagem para estudar o comportamento de um sistema específico aplicando uma disciplina de exploração empírica (ROSENTHAL *et al*. 2017. p. 2).

Segundo Basiri *et al*. (2017), sendo a utilização da arquitetura de sistemas distribuídos comumente implementada no desenvolvimento de aplicações que possuem

alta complexidade atualmente, a engenharia do caos realiza 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.

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 (ROSENTHAL *et al*. 2017. p. 6). Após 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).

Miles (2019) explica que a engenharia do caos começa com a seguinte pergunta. "Sabemos o que o sistema pode fazer neste caso?". Muitas podem ser as origens dessa pergunta, um incidente que já ocorreu anteriormente ou simplesmente uma dúvida ou preocupação de um integrante da equipe responsável pelo sistema. Com a pergunta feita e avaliado o risco envolvido a engenharia do caos começa formulando uma hipótese como base para o experimento do caos. Os resultados desses experimentos formam uma coleção de observações que evidenciam uma ou mais fraquezas existentes que devem ser consideradas como candidatas a melhorias.

Uma forma convencional de demonstrar a importância da engenharia do caos é citando a história da Engenharia do Caos na Netflix, como esclarece Cavalcanti (2018), que foi quem deu origem a "Engenharia do Caos" que conhecemos. A empresa começou a migração de seu datacenter para a cloud em 2008 e em conjunto com a migração algumas práticas de teste de resiliência na produção foram estabelecidas, o que acabou conhecido como "Engenharia do Caos". Em 2011 a Netflix desenvolveu a ferramenta "Chaos Monkey" para testar a resiliência de sua infraestrutura desligando serviços no ambiente de produção. Foi observado pelos operadores um caso extremo de falha na infraestrutura o que deu origem a ferramenta "Chaos Kong" que desligava uma AWS Region inteira trazendo ainda mais benefícios a resiliência de sua infraestrutura. Com a união de uma ferramenta chamada Failure Injection Testing (FIT) hoje a Netflix possui mais uma ferramenta, a Chaos Automation Platform, que realiza experimentação do caos de forma automatizada em toda a arquitetura de microsserviços de forma intermitente. A partir disso, percebe-se que a confiança em um software está estritamente relacionada à dificuldade em causar impactos ao seu "sistema estável". Fraquezas ou falhas descobertas fornecem metas de melhoria para evitar que esse comportamento se espalhe pelo sistema (PRINCIPLE OF CHAOS, 2018).

Excluído: a

4.3 GAMIFICAÇÃO

Gamificação utiliza mecânicas e dinâmicas de jogos para engajar pessoas, motivando ações e comportamentos em ambientes fora do contexto de jogos (CARVALHO, 2016). Segundo Alves (2015) a gamificação é uma forma cada vez mais utilizada nos negócios para tornar as experiências profissionais mais atrativas.

Suh, Wagner e Liu (2018) explicam que a gamificação aumenta o envolvimento do usuário. No entanto, ainda não está claro, em grande parte devido à falta de uma estrutura teórica. Assim, Ryan, Rigby e Przybylski (2006) propuseram que as pessoas são atraídas por videogames na medida em que experimentam a autonomia, competência e relacionamento enquanto brincam. Autonomia refere-se à liberdade para escolher a atividade do jogo a ser executada e a forma como a realizar. Competência é definido como um sentimento de ser capaz e eficaz no jogo, enquanto relacionamento é uma sensação de conexão com outras pessoas através do jogo. Suh, Wagner e Liu (2018) atribuiram à gamificação, a potencialização do engajamento do usuário, por meio da mediação da satisfação das necessidades psicológicas (autonomia, competência e relacionamento) entre a dinâmica do jogo e a diversão. Gerando diversas dinâmicas de jogo, como recompensas, competição, altruísmo e autoexpressão de uma forma que ajude as pessoas a satisfazer suas necessidades psicológicas.

McGonigal (2012) identificou que todos os jogos têm quatro características: meta, regras, sistema de feedback e participação voluntária. A meta é o motivo pelo qual o usuário está jogando um jogo. As regras são a forma como o jogador deverá se portar dentro do jogo. O sistema de feedback representa a demonstração da progressão de um jogador em relação as metas do jogo. Essa característica tem como objetivo manter o jogador motivado e engajado. O último fator é a participação voluntária, ou seja, a conscientização do usuário com as regras, a meta e sistema de feedback que a atividade tem.

Uma solução de aprendizagem gamificada será tão mais eficaz quanto sua capacidade de engajar adequadamente o público para o qual foi desenhada, levando em consideração seu tipo e a forma como interage com os outros e com o jogo (ALVES, 2015).

Excluído: como

Excluído: ,

Excluído: n

Excluído:

Excluído: propôs

Excluído: a

Excluído: i

Comentado [MH6]: Sua proposta envolve aprendizagem? Ou seja, o framework pode ser considerado uma solução de aprendizagem?

REFERÊNCIAS

ALVES, Flora. **Gamification: como criar experiências de aprendizagem engajadoras**, São Paulo: DVS, 2015.

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

CARVALHO, Rafael. **O que é a gamificação e como ela funciona?** 2016. Disponível em: https://www.edools.com/o-que-e-gamificacao. Acesso em 25 nov. 2021.

CAVALCANTI, Jose C. **A Engenharia do Caos**. 2018. Disponível em: https://josecarloscavalcanti.medium.com/a-engenharia-do-caos-69029c097dea. Acesso em 30 nov. 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.

GOERGER, Simon R. **Engineered Resilient Systems:** a DoD perspective. In: CONFERENCE ON SYSTEMS ENGINEERING RESEARCH (CSER 2014), 12., 2014, Los Angeles. **Procedia Computer Science**. Manchester: Elsevier, 2010. v. 192, p. 865-872. Disponível em: https://www.sciencedirect.com/science/article/pii/S1877050914001665. Acesso em: 05 out. 2021.

JERNBERG, Hugo. **Building 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.

LEROY, Jonny. **Reliability under abnormal conditions.** 2017. Disponível em: https://www.thoughtworks.com/insights/blog/reliability-under-abnormal-conditions. Acesso em: 18 set. 2021.

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.

MCGONIGAL, Jane. **A realidade em jogo**: porque os games nos tornam melhores e como eles podem mudar o mundo. Trad. Eduardo Rieche. Rio de Janeiro: Best Seller, 2012.

MILES, Russ. **Learning Chaos Engineering** Discovering and Overcoming System Weaknesses Through Experimentation. O'Reilly, 2019.

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

PEFFERS, Ken *et al*. **A Design Science Research Methodology for Information Systems Research**. Journal of Management Information Systems, v. 24, n. 3, p.45-77, 2007.

PEREIRA, Maicon C. **Criando Aplicações Resilientes: uma visão geral**. 2021. Disponível em: https://imasters.com.br/desenvolvimento/criando-aplicacoes-resilientes-uma-visao-geral. Acesso em: 05 set. 2021.

PRINCIPLE OF CHAOS. **Princípios de Chaos Engineering**. 2018. Disponível em: http://principlesofchaos.org/?lang=PTBRcontent. Acesso em: 16 set. 2021.

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.

RYAN, Richard M.; RIGBY, C. Scott; PRZYBYLSKI, Andrew. **The motivational pull of video games: A self-determination theory approach. Motivation and emotion**, v. 30, n. 4, p. 344-360, 2006.

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.

SUH, Ayoung; WAGNER, Christian; LIU, Lili. **Enhancing user engagement through gamification**. Journal of Computer Information Systems, v. 58, n. 3, p. 204-213, 2018.

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.

FORMULÁRIO DE AVALIAÇÃO SIS – PROFESSOR AVALIADOR

Avaliador(a): Marcel Hugo

Atenção: quando o avaliador marcar algum item como atende parcialmente ou não atende, deve obrigatoriamente indicar os motivos no texto, para que o aluno saiba o porquê da avaliação.

	ASPECTOS AVALIADOS	atende	atende parcialmente	não atende
ASPECTOS TÉCNICOS	1. INTRODUÇÃO O tema de pesquisa está devidamente contextualizado/delimitado?	X		
	O problema está claramente formulado?	X		
	2. OBJETIVOS O objetivo principal está claramente definido e é passível de ser alcançado?	X		
	Os objetivos específicos são coerentes com o objetivo principal?	X		
	3. TRABALHOS CORRELATOS São apresentados trabalhos correlatos, bem como descritas as principais funcionalidades e os pontos fortes e fracos?	X		
	4. JUSTIFICATIVA Foi apresentado e discutido um quadro relacionando os trabalhos correlatos e suas principais funcionalidades com a proposta apresentada?	X		
	São apresentados argumentos científicos, técnicos ou metodológicos que justificam a proposta?	X		
	São apresentadas as contribuições teóricas, práticas ou sociais que justificam a proposta?	X		
	5. REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO Os requisitos funcionais e não funcionais foram claramente descritos?	X		
	6. METODOLOGIA Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC?	X		
	Os métodos, recursos e o cronograma estão devidamente apresentados e são compatíveis com a metodologia proposta?	X		
	7. REVISÃO BIBLIOGRÁFICA (atenção para a diferença de conteúdo entre projeto e pré-projeto) Os assuntos apresentados são suficientes e têm relação com o tema do TCC?	X		
	As referências contemplam adequadamente os assuntos abordados (são indicadas obras atualizadas e as mais importantes da área)?	X		
ASPECTOS METODOLÓGICOS	8. LINGUAGEM USADA (redação) O texto completo é coerente e redigido corretamente em língua portuguesa, usando linguagem formal/científica?	X		
	A exposição do assunto é ordenada (as ideias estão bem encadeadas e a linguagem utilizada é clara)?	X		

O projeto de TCC ser deverá ser revisado, isto é, necessita de complementação, se:

- qualquer um dos itens tiver resposta NÃO ATENDE;
- pelo menos **5 (cinco)** tiverem resposta ATENDE PARCIALMENTE.

PARECER: (X) APROVADO ( ) REPROVADO

Excluído:

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

FRAMEWORK DE ENGENHARIA DO CAOS: UMA ABORDAGEM GAMIFICADA PARA A ENGENHARIA DO CAOS

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, e-mail 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; DOLLIMORE; KINDBERG, 2001).

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. O caos aplicado ao sistema distribuído será representado de forma gamificada através do jogo Super Breakout, combinando a natureza destrutiva do jogo aos experimentos/ataques que serão realizados. Destaca-se que todos os experimentos serão monitorados no cluster buscando identificar e caracterizar as fraquezas que possam aferir a confiança e resiliência do sistema.

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 gamificada a aplicação de Engenharia do Caos a um sistema distribuído;
c) identificar possíveis fraquezas de uma arquitetura distribuída baseada em Kubernetes.

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 em três atividades: conceitualização do problema, design da solução e validação empírica. A conceitualização 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 Tecnologia de Informação (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 e 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 Design Science for Research Methodology in Information System (DSRM) de Peffers *et al.* (2007), 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 localmente as funcionalidades da ferramenta desenvolvida. 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 Unidade Central de Processamento (Central Processing Unit - 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 observação 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, onde cada hipótese permite identificar uma possível fraqueza na arquitetura. Kesim (2019) também verificou se o software possuía fragilidade ao ser testado com um conjunto dados estatísticos

produzidos pela ferramenta 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 Hyper Text Transfer Protocol (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: antes de executar o experimento no `pod`[2] de configuração do banco de dados ocorreu um erro de falta de memória e o sistema não conseguiu se recuperar para o estado estável. 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 eliminando todos os `pods` de microsserviços de configuração disponíveis. No terceiro foi adicionado um objeto de configuração ao banco de dados (`ID = 2`) solicitando a Application Programming Interface (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 concluíram 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.

[2] Pod no Kuberponetes é um conjunto de um ou mais containers Linux, são usados como a unidade de replicação no Kubernetes (MATOS, 2018)

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 descrevem a fundamentação necessária para exemplificar o tema abordado.

3.1 JUSTIFICATIVA

O Quadro 5 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 3 – 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 5 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 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) se 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 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. Os experimentos ocorrerão em paralelo às ações no jogo proporcionando uma experiência da engenharia do caos mais compreensível e divertida de forma gamificada. Os resultados dos experimentos poderão ser avaliados no Kubernetes Engine Monitoring, oferecido pela plataforma Google Kubernetes Engine (GKE), 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. A importância deste trabalho está em validar o conceito de engenharia do caos em um sistema distribuído de forma gamificada. Monitorando os experimentos do caos que ocorrem em paralelo ao “caos” do jogo Super Breakout.

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 aos elementos do jogo Super Breakout (falhas de hardware, parada de servidores, falhas de software e falhas de rede) (RF);
f) apresentar os resultados 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) especificação do *framework*: formalizar as estruturas e evoluções da arquitetura do *framework* através de ferramentas de diagramação Lucidchart e Cloudcraft para elaborar os diagramas de classes, sequência e atividades de acordo com a Unified Modeling Language (UML);
d) adaptação do jogo Super Breakout: alterar o código fonte do jogo, utilizando a linguagem de programação Lua, para que as ações de destruição de blocos do jogo provoquem falhas no sistema distribuído alvo do experimento do caos, por meio do *framework;*
e) implementação: implementar o *framework* utilizando a linguagem de programação Java com Spring Boot no ambiente de desenvolvimento Visual Studio;

f) infraestrutura cloud: desenvolver e hospedar a arquitetura de microsserviços na plataforma Google Cloud utilizando Kubernetes;
g) refatoração do sistema distribuído: a partir dos itens (d), (e) e (f), para cada fraqueza identificada na arquitetura do sistema distribuído, documentar e refatorar a solução para assegurar maior resiliência;
h) 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 6.

Quadro 4 – 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										
especificação do *framework*										
adaptação do jogo Super Breakout										
implementação										
infraestrutura cloud										
refatoração do sistema distribuído										
testes										

Fonte: elaborado pelo autor

4 REVISÃO BIBLIOGRÁFICA

Esta seção tem como objetivo explorar os conceitos mais importantes para a realização deste trabalho. A subseção 4.1 apresenta o contexto da resiliência em sistemas computacionais. A subseção 4.2 contextualiza a engenharia do caos e de que forma ela consegue contribuir para o desenvolvimento de uma arquitetura distribuída de alta resiliência. Por fim, a subseção 4.3 conceitua gamificação.

4.1 RESILIÊNCIA

Segundo Georger *et al*. (2014) 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 *et al.* (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.

> Mesmo nas melhores circunstâncias, projetar um sistema para resiliência é uma tarefa assustadora devido à variedade de questões que precisam ser abordadas. Por exemplo, os envelopes de design precisam ser entendidos com antecedência e, quando um sistema falha, deve fazê-lo de maneira previsível, detectável e elegante para aumentar a capacidade de sobrevivência geral do sistema (ou seja, a capacidade de sobrevivência dos usuários e dos outros subsistemas do sistema). (GEORGER *et al*. 2014).

Segundo Pereira (2021), "no melhor cenário, a aplicação se recupera sem o usuário/cliente perceber. No pior cenário, a aplicação oferece serviços de forma limitada (que chamamos de *graceful degradation*)". 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).

Para prevenir e mitigar o surgimento tantos problemas quanto possível, LeRoy (2017) demonstra três estratégias que são prevenção, mitigação e híbrido. A prevenção envolve o teste do sistema, ou partes dele, antes de liberá-lo para a produção. Os principais testes a serem realizados são de correção funcional; capacidade de desempenho sob carga esperada; e resiliência a falhas previsíveis. Já a mitigação LeRoy (2017) explica que envolve limitar a amplitude do impacto através do isolamento e da degradação harmoniosa (*graceful degradation*). Envolve limitar também a duração do impacto e melhorar o tempo de resposta, diagnóstico e correção. A estratégia híbrida abrange a prevenção e mitigação como a liberação de versões do canário para um subconjunto de usuários realizada na produção. Outra forma seria a técnica avançada da Engenharia do Caos para testar e praticar abordagens de prevenção e mitigação em um ambiente de produção.

Tão importante quanto garantir a existência dos recursos necessários para atender as demandas de performance é também adotar práticas para prevenir e mitigar a propagação de falhas (SEVERO JÚNIOR, 2021). "Ser **resiliente** é aceitar que falhas irão acontecer e tratá-las da melhor maneira." (PEREIRA, 2021, grifo do autor).

4.2 ENGENHARIA DO CAOS

Assim como os cientistas conduzem experimentos para estudar fenômenos físicos e sociais, a Engenharia do Caos é uma abordagem para estudar o comportamento de um sistema específico aplicando uma disciplina de exploração empírica (ROSENTHAL *et al*. 2017. p. 2).

Segundo Basiri *et al*. (2017), sendo a utilização da arquitetura de sistemas distribuídos comumente implementada no desenvolvimento de aplicações que possuem

alta complexidade atualmente, a engenharia do caos realizada 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.

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 (ROSENTHAL *et al*. 2017. p. 6). Após 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).

Miles (2019) explica que a engenharia do caos começa com a seguinte pergunta. "Sabemos o que o sistema pode fazer neste caso?". Muitas podem ser as origens dessa pergunta, um incidente que já ocorreu anteriormente ou simplesmente uma dúvida ou preocupação de um integrante da equipe responsável pelo sistema. Com a pergunta feita e avaliado o risco envolvido a engenharia do caos começa formulando uma hipótese como base para o experimento do caos. Os resultados desses experimentos formam uma coleção de observações que evidenciam uma ou mais fraquezas existentes que devem ser consideradas como candidatas a melhorias.

Uma forma convencional de demonstrar a importância da engenharia do caos é citando a história da Engenharia do Caos na Netflix, como esclarece Cavalcanti (2018), que foi quem deu origem a "Engenharia do Caos" que conhecemos. A empresa começou a migração de seu datacenter para a cloud em 2008 e em conjunto com a migração algumas práticas de teste de resiliência na produção foram estabelecidas, o que acabou conhecido como "Engenharia do Caos". Em 2011 a Netflix desenvolveu a ferramenta "Chaos Monkey" para testar a resiliência de sua infraestrutura desligando serviços no ambiente de produção. Foi observado pelos operadores um caso extremo de falha na infraestrutura o que deu origem a ferramenta "Chaos Kong" que desligava uma AWS Region inteira trazendo ainda mais benefícios a resiliência de sua infraestrutura. Com a união de uma ferramenta chamada Failure Injection Testing (FIT) hoje a Netflix possui mais uma ferramenta, a Chaos Automation Platform, que realiza experimentação do caos de forma automatizada em toda a arquitetura de microsserviços de forma intermitente. A partir disso, percebe-se que a confiança em um software está estritamente relacionada a dificuldade em causar impactos ao seu "sistema estável". Fraquezas ou falhas descobertas fornecem metas de melhoria para evitar que esse comportamento se espalhe pelo sistema (PRINCIPLE OF CHAOS, 2018).

4.3 GAMIFICAÇÃO

Gamificação utiliza mecânicas e dinâmicas de jogos para engajar pessoas, motivando ações e comportamentos em ambientes fora do contexto de jogos (CARVALHO, 2016). Segundo Alves (2015) a gamificação é uma forma cada vez mais utilizada nos negócios para tornar as experiências profissionais mais atrativas.

Suh, Wagner e Liu (2018) explica que como a gamificação aumenta o envolvimento do usuário, no entanto, ainda não está claro, em grande parte devido à falta de uma estrutura teórica. Assim, Ryan, Rigby e Przybylski (2006) propôs que as pessoas são atraídas por videogames na medida em que experimentam a autonomia, competência e relacionamento enquanto brincam. Autonomia refere-se a liberdade para escolher a atividade do jogo a ser executada e a forma como a realizar. Competência é definido como um sentimento de ser capaz e eficaz no jogo, enquanto relacionamento é uma sensação de conexão com outras pessoas através do jogo. Suh, Wagner e Liu (2018) atribui à gamificação, a potencialização do engajamento do usuário, por meio da mediação da satisfação das necessidades psicológicas (autonomia, competência e relacionamento) entre a dinâmica do jogo e a diversão. Gerando diversas dinâmicas de jogo, como recompensas, competição, altruísmo e autoexpressão de uma forma que ajude as pessoas a satisfazer suas necessidades psicológicas.

McGonigal (2012) identificou que todos os jogos têm quatro características: meta, regras, sistema de feedback e participação voluntária. A meta é o motivo pelo qual o usuário está jogando um jogo. As regras são a forma como o jogador deverá se portar dentro do jogo. O sistema de feedback representa a demonstração da progressão de um jogador em relação as metas do jogo. Essa característica tem como objetivo manter o jogador motivado e engajado. O último fator é a participação voluntária, ou seja, a conscientização do usuário com as regras, a meta e sistema de feedback que a atividade tem.

Uma solução de aprendizagem gamificada será tão mais eficaz quanto sua capacidade de engajar adequadamente o público para o qual foi desenhada, levando em consideração seu tipo e a forma como interage com os outros e com o jogo (ALVES, 2015).

REFERÊNCIAS

ALVES, Flora. **Gamification: como criar experiências de aprendizagem engajadoras**, São Paulo: DVS, 2015.

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

CARVALHO, Rafael. **O que é a gamificação e como ela funciona?** 2016. Disponível em: https://www.edools.com/o-que-e-gamificacao. Acesso em 25 nov. 2021.

CAVALCANTI, Jose C. **A Engenharia do Caos**. 2018. Disponível em: https://josecarloscavalcanti.medium.com/a-engenharia-do-caos-69029c097dea. Acesso em 30 nov. 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.

GOERGER, Simon R. **Engineered Resilient Systems:** a DoD perspective. In: CONFERENCE ON SYSTEMS ENGINEERING RESEARCH (CSER 2014), 12., 2014, Los Angeles. **Procedia Computer Science**. Manchester: Elsevier, 2010. v. 192, p. 865-872. Disponível em: https://www.sciencedirect.com/science/article/pii/S1877050914001665. Acesso em: 05 out. 2021.

JERNBERG, Hugo. **Building 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.

LEROY, Jonny. **Reliability under abnormal conditions.** 2017. Disponível em: https://www.thoughtworks.com/insights/blog/reliability-under-abnormal-conditions. Acesso em: 18 set. 2021.

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.

MCGONIGAL, Jane. **A realidade em jogo**: porque os games nos tornam melhores e como eles podem mudar o mundo. Trad. Eduardo Rieche. Rio de Janeiro: Best Seller, 2012.

MILES, Russ. **Learning Chaos Engineering** Discovering and Overcoming System Weaknesses Through Experimentation. O'Reilly, 2019.

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

PEFFERS, Ken *et al*. **A Design Science Research Methodology for Information Systems Research**. Journal of Management Information Systems, v. 24, n. 3, p.45-77, 2007.

PEREIRA, Maicon C. **Criando Aplicações Resilientes: uma visão geral**. 2021. Disponível em: https://imasters.com.br/desenvolvimento/criando-aplicacoes-resilientes-uma-visao-geral. Acesso em: 05 set. 2021.

PRINCIPLE OF CHAOS. **Princípios de Chaos Engineering**. 2018. Disponível em: http://principlesofchaos.org/?lang=PTBRcontent. Acesso em: 16 set. 2021.

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.

RYAN, Richard M.; RIGBY, C. Scott; PRZYBYLSKI, Andrew. **The motivational pull of video games: A self-determination theory approach. Motivation and emotion**, v. 30, n. 4, p. 344-360, 2006.

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.

SUH, Ayoung; WAGNER, Christian; LIU, Lili. **Enhancing user engagement through gamification**. Journal of Computer Information Systems, v. 58, n. 3, p. 204-213, 2018.

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.

FORMULÁRIO DE AVALIAÇÃO SIS ACADÊMICO
PROFESSOR TCC I – PROJETO

Avaliador(a): Dalton Solano dos Reis

	ASPECTOS AVALIADOS	atende	atende parcialmente	não atende
ASPECTOS TÉCNICOS	1. INTRODUÇÃO O tema de pesquisa está devidamente contextualizado/delimitado?	X		
	O problema está claramente formulado?	X		
	2. OBJETIVOS O objetivo principal está claramente definido e é passível de ser alcançado?	X		
	Os objetivos específicos são coerentes com o objetivo principal?	X		
	3. TRABALHOS CORRELATOS São apresentados trabalhos correlatos, bem como descritas as principais funcionalidades e os pontos fortes e fracos?	X		
	4. JUSTIFICATIVA Foi apresentado e discutido um quadro relacionando os trabalhos correlatos e suas principais funcionalidades com a proposta apresentada?	X		
	São apresentados argumentos científicos, técnicos ou metodológicos que justificam a proposta?	X		
	São apresentadas as contribuições teóricas, práticas ou sociais que justificam a proposta?	X		
	5. REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO Os requisitos funcionais e não funcionais foram claramente descritos?	X		
	6. METODOLOGIA Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC?	X		
	Os métodos, recursos e o cronograma estão devidamente apresentados e são compatíveis com a metodologia proposta?	X		
	7. REVISÃO BIBLIOGRÁFICA (atenção para a diferença de conteúdo entre projeto e pré-projeto) Os assuntos apresentados são suficientes e têm relação com o tema do TCC?	X		
	As referências contemplam adequadamente os assuntos abordados (são indicadas obras atualizadas e as mais importantes da área)?	X		
ASPECTOS METODOLÓGICOS	8. LINGUAGEM USADA (redação) O texto completo é coerente e redigido corretamente em língua portuguesa, usando linguagem formal/científica?	X		
	A exposição do assunto é ordenada (as ideias estão bem encadeadas e a linguagem utilizada é clara)?	X		
	9. ORGANIZAÇÃO E APRESENTAÇÃO GRÁFICA DO TEXTO A organização e apresentação dos capítulos, seções, subseções e parágrafos estão de acordo com o modelo estabelecido?	X		
	10. ILUSTRAÇÕES (figuras, quadros, tabelas) As ilustrações são legíveis e obedecem às normas da ABNT?	X		
	11. REFERÊNCIAS E CITAÇÕES As referências obedecem às normas da ABNT?	X		
	As citações obedecem às normas da ABNT?	X		
	Todos os documentos citados foram referenciados e vice-versa, isto é, as citações e referências são consistentes?	X		

O projeto de TCC ser deverá ser revisado, isto é, necessita de complementação, se:

- qualquer um dos itens tiver resposta NÃO ATENDE;
- pelo menos **5 (cinco)** tiverem resposta ATENDE PARCIALMENTE.

PARECER: (X) APROVADO ( ) REPROVADO

Revisão do Pré-projeto

Disciplina: Trabalho de Conclusão de Curso I – SIS

Caro orientando,

segue abaixo o Termo de Compromisso, as DUAS revisões do seu pré-projeto contendo a avaliação do professor "avaliador" e professor "TCC1". É muito importante que revise com cuidado e discuta possíveis dúvidas decorrente das revisões com o seu professor orientador, e com o professor de TCC1. Sempre procure fazer todos os ajustes solicitados, até mesmo os menores detalhes, pois todos são importantes e irão refletir na sua nota nesta disciplina.
Mas, caso o professor orientador julgue que algumas anotações das revisões não devam ser feitas, ou mesmo que sejam feitas de forma diferente a solicitada pelo revisor, anexe ao final do seu projeto a ficha "Projeto: Observações – Professor Orientador" disponível no material da disciplina, e justifique o motivo.
Lembrem que agora o limite de páginas do projeto é no máximo 16 (dezesseis) páginas.

Atenciosamente,

UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE SISTEMAS DE INFORMAÇÃO

TERMO DE COMPROMISSO

I – IDENTIFICAÇÃO DO ALUNO	
Nome:	Lucas Vanderlinde
CV Lattes:	http://lattes.cnpq.br/4854217004078831
E-mail:	lucasv.eu@gmail.com / lucvanderlinde@furb.br
Telefone:	47 98840-9785

II – IDENTIFICAÇÃO DO TRABALHO	
Título provisório:	FRAMEWORK DE ENGENHARIA DO CAOS: ESTUDO CASO SUPER BREAKOUT
Orientador:	Aurélio Faustino Hoppe
Coorientador (se houver):	
Linha de Pesquisa:	☐ Tecnologias aplicadas à informática na educação ☒ Tecnologias aplicadas ao desenvolvimento de sistemas

III – COMPROMISSO DE REALIZAÇÃO DO TCC

Eu (aluno), Lucas Vanderlinde
comprometo-me a realizar o trabalho proposto no semestre 2022/1 , de acordo com as normas e os prazos determinados pela FURB, conforme previsto na resolução nº.20/2016.
Assinatura: NÃO É NECESSÁRIO – Encaminhar por mail ao orientador

IV – COMPROMISSO DE ORIENTAÇÃO

Eu (orientador), Aurélio Faustino Hoppe
comprometo-me a orientar o trabalho proposto no semestre 2022/1 , de acordo com as normas e os prazos determinados pela FURB, conforme previsto na resolução nº.20/2016.
Assinatura: NÃO É NECESSÁRIO – Encaminhar por mail ao professor de TCC I

Blumenau, 16 de Agosto de 2021

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

Comentado [MH7]: Após ler a proposta, sugiro alterar o título pois dá a impressão que o estudo de caso terá como alvo o SuperBreakout e não que ele é uma parte da interação com o framework.

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.

Comentado [MH8]: O que Desenvolvimento Distribuído de Software tem a ver com Sistemas Distribuídos?
Acho que este parágrafo não agrega ao tema.

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.

Comentado [MH9]: Ficou estranho colocar a definição no meio da redação do objetivo.
E não completou o objetivo, pois não disse para que irá desenvolver um framework: para testar, para monitorar,...

Comentado [MH10]: Apesar de ser um neologismo e poder ser escrito desta forma, normalmente se utiliza "gamificada", por vir de gamificação, pois o próprio termo em inglês é gamification (com i e não com e).

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.

Excluído: e

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.

Comentado [MH11]: distribuída baseada em Kubernetes.

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 de três 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

Excluído: ,

Excluído: -se

Excluído: tipos de

Comentado [MH12]: No início da linha é falado em conceitualização.
Qual dos dois é o termo correto?

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 et al, composta de seis passos:

Excluído: ,

Excluído: , p

Comentado [MH13]: Sugiro deixar no original: Design Science for Research Methodology in Information System (DSRM)

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 localmente as funcionalidades da ferramenta desenvolvida. 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 observação 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

Excluído: ,
Excluído: ,
Excluído: ,
Excluído: ,
Excluído: ,
Excluído: localmente
Excluído: bilidade

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.

Comentado [MH14]: Melhorar a redação. Tive que ler a frase algumas vezes para compreender.

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, antes de executar o experimento no pod de configuração do 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, eliminando todos os pods de microsserviços de configuração disponíveis. 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).

Excluído: , no qual,

Excluído: a

Comentado [MH15]: Informação importante, mas sugiro colocar como nota de rodapé.

Excluído: , o

Excluído: ele eliminou

Excluído: e n

Kesim (2019) conseguiu aplicar com sucesso meios para identificar fraquezas e falhas potenciais na arquitetura do sistema e os resultados da análise de risco concluíram 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

Excluído: implicaram

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 5 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 5 – 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 5 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 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.

Excluído: palpável

Formatado: Fonte: 12 pt

Já os cenários no qual a solução ou o estudo proposto foi aplicado 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 ocorrerão em paralelo às 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.

Excluído: se

Formatado: Fonte: 12 pt

Excluído: C

Excluído: C

Comentado [MH16]: Esta ideia precisa ser mais trabalhada, explicando melhor o motivo desta vinculação.
A ideia apresentada no final da introdução (gamificação) precisa ser retomada aqui. Veja que o termo gamificação nem aparece agora.

Excluído: em

Excluído: a

Comentado [MH17]: GKE não é a sigla para o extenso apresentado. GKE = Google Kubernetes Engine.

Comentado [MH18]: Contudo é uma conjunção adversativa, isto é, dá a noção de que a próxima ideia ocorre em oposição ao que foi falado até aqui.

Comentado [MH19]: Validar EC em contexto de jogo? Dá a impressão que será aplicado EC em um jogo.

Comentado [MH20]: Se os experimentos ocorrerão a partir das ações do jogo não me parece que serão realizados de forma controlada. Ao vincular ao jogo, parece que o contexto do jogo é que ditará a sequência de ações, portanto aleatório do ponto de vista da experimentação.

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 aos elementos do jogo 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).

Comentado [MH21]: Para ter certeza: o framework vai avaliar a resiliência ou vai apresentar os resultados das métricas para que o usuário faça a avaliação da resiliência?
Se vai avaliar a resiliência, em algum lugar você deve mencionar a existência ou criação de um método que considere as diferentes métricas para alcançar um resultado/conclusão da resiliência.

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

Comentado [MH22]: Já será possível alterar o jogo antes mesmo da especificação do framework?

Comentado [MH23]: Dá a impressão que vai inserir falhas no jogo e não que vai fazer com que o jogo seja o gatilho para inserir/provocar falhas no sistema distribuído alvo do experimento por meio do framework.

Comentado [MH24]: Não existe um diagrama de estrutura na UML. Existe uma categoria de diagramas chamada diagramas estruturais. Mas qual deles você pretende utilizar?

programação Java com Spring Boot no ambiente de desenvolvimento Visual Studio. Desenvolver e hospedar a arquitetura de microsserviços 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 6.

Quadro 6 – 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), embora a resiliência esteja cada vez mais se tornando algo comum em uma variedade de campos, 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).

Excluído: e

Comentado [MH25]: Deveria ser outra etapa, pois selecionar e implantar o sistema distribuído alvo dos experimentos não tem relação com o desenvolvimento do framework.

Comentado [MH26]: Pela introdução, gamificação também é um assunto importante.

Excluído: a resiliência

Excluído: a

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 devem ser corrigidas as fraquezas conhecidas nos serviços. Por 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).

Comentado [MH27]: Para o projeto não esqueça de dizer o que é Engenharia do Caos, qual sua composição, etc.

Excluído: Depois

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 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.

Excluído: Building a Framework for Chaos Engineering

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.

FORMULÁRIO DE AVALIAÇÃO – PROFESSOR AVALIADOR

Comentado [MH28]: Gostei do tema e da proposta. Mas precisa melhorar o projeto.

Avaliador(a): **Marcel Hugo**

Atenção: quando o avaliador marcar algum item como atende parcialmente ou não atende, deve obrigatoriamente indicar os motivos no texto, para que o aluno saiba o porquê da avaliação.

	ASPECTOS AVALIADOS[1]	atende	atende parcialmente	não atende
ASPECTOS TÉCNICOS	1. INTRODUÇÃO O tema de pesquisa está devidamente contextualizado/delimitado?	X		
	O problema está claramente formulado?		X	
	2. OBJETIVOS O objetivo principal está claramente definido e é passível de ser alcançado?	X		
	Os objetivos específicos são coerentes com o objetivo principal?	X		
	3. TRABALHOS CORRELATOS São apresentados trabalhos correlatos, bem como descritas as principais funcionalidades e os pontos fortes e fracos?	X		
	4. JUSTIFICATIVA Foi apresentado e discutido um quadro relacionando os trabalhos correlatos e suas principais funcionalidades com a proposta apresentada?	X		
	São apresentados argumentos científicos, técnicos ou metodológicos que justificam a proposta?		X	
	São apresentadas as contribuições teóricas, práticas ou sociais que justificam a proposta?		X	
	5. REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO Os requisitos funcionais e não funcionais foram claramente descritos?	X		
	6. METODOLOGIA Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC?		X	
	Os métodos, recursos e o cronograma estão devidamente apresentados e são compatíveis com a metodologia proposta?		X	
	7. REVISÃO BIBLIOGRÁFICA (atenção para a diferença de conteúdo entre projeto e pré-projeto) Os assuntos apresentados são suficientes e têm relação com o tema do TCC?		X	
	As referências contemplam adequadamente os assuntos abordados (são indicadas obras atualizadas e as mais importantes da área)?	X		
ASPECTOS METODOLÓGICOS	8. LINGUAGEM USADA (redação) O texto completo é coerente e redigido corretamente em língua portuguesa, usando linguagem formal/científica?	X		
	A exposição do assunto é ordenada (as ideias estão bem encadeadas e a linguagem utilizada é clara)?	X		

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.

Comentado [DSdR29]: Não entendi a relação de Sistemas Distribuídos com DDS.

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

Comentado [DSdR30]: 2018 ou 2019

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.

Comentado [DSdR31]: Itálico.

Comentado [DSdR32]: Melhor gamificada
Se trocar, rever em todo o texto.

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.

Comentado [DSdR33]: Itálico.

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.

Comentado [DSdR34]: Itálico.

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.

Comentado [DSdR35]: Itálico.

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.

Comentado [DSdR36]: Tecnologia da Informação (TI)

Comentado [DSdR37]: Itálico.

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.

Comentado [DSdR38]: Itálico.

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

Comentado [DSdR39]: foram:

Comentado [DSdR40]: Trocar virgula por e

Comentado [DSdR41]: Itálico.

Comentado [DSdR42]: são:

Comentado [DSdR43]: Itálico.

Comentado [DSdR44]: Itálico.

Comentado [DSdR45]: Itálico.

Comentado [DSdR46]: Itálico.

Comentado [DSdR47]: Itálico.

Comentado [DSdR48]: Itálico.

Comentado [DSdR49]: Itálico.

Comentado [DSdR50]: Itálico.

Comentado [DSdR51]: Itálico.

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

Comentado [DSdR52]: Itálico.

Comentado [DSdR53]: da Unidade Central de Processamento (Central Processing Unit - CPU)

Comentado [DSdR54]: observação

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.

Comentado [DSdR55]: Hyper Text Transfer Protocol (HTTP)

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).

Comentado [DSdR56]: Estilo fonte: TF-Courier

Comentado [DSdR57]: Estilo fonte: TF-Courier

Comentado [DSdR58]: Estilo fonte: TF-Courier

Comentado [DSdR59]: Estilo fonte: TF-Courier

Comentado [DSdR60]: Estilo fonte: TF-Courier

Comentado [DSdR61]: Application Programming Interface (API)

Comentado [DSdR62]: Estilo fonte: TF-Courier

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.

Comentado [DSdR63]: Remover.

3.1 JUSTIFICATIVA

O Quadro 5 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.

Comentado [DSdR64]: A borda do quadro não pode ultrapassar a margem direita.

Quadro 7 – 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.

Comentado [DSdR65]: Características

Comentado [DSdR66]: Itálico.

A partir do Quadro 5 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.

Comentado [DSdR67]: Formato da fonte.

Comentado [DSdR68]: Formato da fonte.

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.

Comentado [DSdR69]: Formato da fonte.

Comentado [DSdR70]: Formato da fonte.

Comentado [DSdR71]: Itálico.

Comentado [DSdR72]: Itálico.

Comentado [DSdR73]: Itálico.

Comentado [DSdR74]: Itálico.

Comentado [DSdR75]: Itálico.

3.2 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O framework a ser desenvolvido neste trabalho deverá:

Comentado [DSdR76]: Itálico.

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);

Comentado [DSdR77]: Não

h) utilizar a linguagem Java com Spring Boot para desenvolver o framework (RNF);

Comentado [DSdR78]: Itálico.

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);

Comentado [DSdR79]: Itálico.

Comentado [DSdR80]: Itálico.

e) implementação: implementar o framework utilizando a linguagem de

Comentado [DSdR81]: Itálico.

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 6.

Quadro 8 – 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).

Comentado [DSdR82]: Itálico.

Comentado [DSdR83]: Itálico.

Comentado [DSdR84]: Centralizar entre margens esq/dir, e não parágrafo.

Comentado [DSdR85]: Esta seção

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.

Comentado [DSdR86]: Remover.

Comentado [DSdR87]: Remover.

Comentado [DSdR88]: Não negrito.

Comentado [DSdR89]: Sul

Comentado [DSdR90]: Remover.

Comentado [DSdR91]: Remover.

Comentado [DSdR92]: Remover.

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.

Comentado [DSdR93]: Distribuídos

Comentado [DSdR94]: Sul

Comentado [DSdR95]: Remover

Comentado [DSdR96]: Remover

Comentado [DSdR97]: Remover

Comentado [DSdR98]: Remover

Comentado [DSdR99]: Remover

Comentado [DSdR100]: Remover

FORMULÁRIO DE AVALIAÇÃO SIS – PROFESSOR TCC I

Avaliador(a): Dalton Solano dos Reis

	ASPECTOS AVALIADOS	atende	atende parcialmente	não atende
ASPECTOS TÉCNICOS	1. INTRODUÇÃO O tema de pesquisa está devidamente contextualizado/delimitado?	X		
	O problema está claramente formulado?	X		
	2. OBJETIVOS O objetivo principal está claramente definido e é passível de ser alcançado?	X		
	Os objetivos específicos são coerentes com o objetivo principal?	X		
	3. TRABALHOS CORRELATOS São apresentados trabalhos correlatos, bem como descritas as principais funcionalidades e os pontos fortes e fracos?	X		
	4. JUSTIFICATIVA Foi apresentado e discutido um quadro relacionando os trabalhos correlatos e suas principais funcionalidades com a proposta apresentada?	X		
	São apresentados argumentos científicos, técnicos ou metodológicos que justificam a proposta?	X		
	São apresentadas as contribuições teóricas, práticas ou sociais que justificam a proposta?	X		
	5. REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO Os requisitos funcionais e não funcionais foram claramente descritos?	X		
	6. METODOLOGIA Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC?	X		
	Os métodos, recursos e o cronograma estão devidamente apresentados e são compatíveis com a metodologia proposta?	X		
	7. REVISÃO BIBLIOGRÁFICA (atenção para a diferença de conteúdo entre projeto e pré-projeto) Os assuntos apresentados são suficientes e têm relação com o tema do TCC?	X		
	As referências contemplam adequadamente os assuntos abordados (são indicadas obras atualizadas e as mais importantes da área)?	X		
ASPECTOS METODOLÓGICOS	8. LINGUAGEM USADA (redação) O texto completo é coerente e redigido corretamente em língua portuguesa, usando linguagem formal/científica?	X		
	A exposição do assunto é ordenada (as ideias estão bem encadeadas e a linguagem utilizada é clara)?	X		
	9. ORGANIZAÇÃO E APRESENTAÇÃO GRÁFICA DO TEXTO A organização e apresentação dos capítulos, seções, subseções e parágrafos estão de acordo com o modelo estabelecido?	X		
	10. ILUSTRAÇÕES (figuras, quadros, tabelas) As ilustrações são legíveis e obedecem às normas da ABNT?		X	
	11. REFERÊNCIAS E CITAÇÕES As referências obedecem às normas da ABNT?	X		
	As citações obedecem às normas da ABNT?		X	
	Todos os documentos citados foram referenciados e vice-versa, isto é, as citações e referências são consistentes?	X		

Comentado [DSdR101]: Indicadas no texto.

Comentado [DSdR102]: Indicadas no texto.