**Disponibilidade e resiliência de sistemas *web* com *Chaos Engineering*.**

Matheus de Oliveira Manhães – 5306315  
Felipe Barbosa de Oliveira – 5306

Projeto Final de Curso apresentado à Universidade do Grande Rio “*Prof. José de Souza Herdy*” (UNIGRANRIO) como parte dos requisitos para conclusão do curso de Bacharelado em Sistemas de Informação.

**BANCA EXAMINADORA**

Prof. Doutor Thiago Silva – Orientador – UNIGRANRIO

Dedico este trabalho...

Matheus de Oliveira Manhães

Dedico este trabalho...

Felipe Barbosa de Oliveira

**AGRADECIMENTOS**

Somos gratos...

*“Frase inspiradora... ”*

(Autor conhecido)

**RESUMO**

O que antes se considerava um tabu em questões de segurança e privacidade, hoje é possível observar que o uso de aplicativos e sistemas tem sido cada vez mais inerente à cultura e à rotina do ser humano. Vemos, por exemplo, o quão comum é uso do Netflix. A plataforma de vídeos sob demanda tem crescido muito e cada vez mais conquista clientes, ou usuários, ao redor do mundo. É possível perceber o quanto é investido para que os serviços da Netflix permaneçam disponíveis para seus usuários, em situações e dispositivos diversos. Estudar a estrutura de grandes sistemas na *World Wide Web* pode auxiliar na maneira como novos aplicativos e sistemas podem ser construídos garantindo qualidade e disponibilidade para os usuários. Uma das formas em que grandes empresas como Netflix, Amazon, Google, Microsoft, Uber e Dropbox conseguem manter seus serviços com qualidade é usando *Chaos Engineering.*

*Chaos Engineering* é o foco deste trabalho, que mostra como o processo de Engenharia do Caos funciona na execução e criação de ataques simulando rotinas de trabalho comuns, na tentativa de gerar turbulência a fim de preparar os engenheiros de software para as situações onde o *software* pode se tornar indisponível como interrupções de energia, congestionamento de requisições e demora em relação ao tempo de resposta afim de gerar resiliência.

Este documento mostra como é feito o processo de Engenharia do Caos através de quatro pequenos sistemas que criamos: um buscador de repositórios de usuários do GitHub, uma API RESTful de cadastro de termos úteis relacionados a *Chaos Engineering,* uma calculadora comum e um gerenciador de tarefas. A ideia é usar essas aplicações como alvo de nossas experimentações de Caos.

**Palavras-chave:** Aplicativos, Sistemas, Web, Internet, Ataques, Experimentação, Netflix, Engenharia do Caos.

**ABSTRACT**

What was once considered a taboo on security and privacy issues, today we can see that the use of applications and systems was increasingly inherent in the culture and routine of the human being. We see, for example, how common is the use of Netflix. The on-demand video platform has grown a lot and is increasingly catching customers, or users, around the world. You can see how much is invested for Netflix services that are permanently available to your users in situations and across devices. Studying a large system structure on the World Wide Web can help in how new applications and systems can be used with quality and availability for users. One of the ways in big companies like Netflix, Amazon, Google, Microsoft, Uber and Dropbox keeps their services quality using Chaos Engineering.

Chaos Engineering is the focus of this paper, which shows how the Chaos Engineering process works to create and execute attacks simutating common work routine in an attempt to generate turbulence to finalize software engineers' preparation for situations where software can become unavailable such as power outages, requirements congestion and response time delays in order to generate resilience.

This document shows how the Chaos Engineering process is done through four small systems it creates: a GitHub users repositories search engine, a RESTful Chaos Engineering term record API, a common calculator, and a task manager. The idea is to use these applications as a target for our Chaos experiments.

**Keywords**: Apps, Systems, Web, Internet, Chaos Engineering, Attacks, Experimentation, Netflix.

**SUMÁRIO**

[1 – INTRODUÇÃO 4](#_Toc24733826)

[1.1 – Justificativa da Pesquisa 4](#_Toc24733827)

[1.2 – Problema de Pesquisa 4](#_Toc24733828)

[1.3 – Hipótese 4](#_Toc24733829)

[1.4 – Objetivo 4](#_Toc24733830)

[1.5 – Objeto 4](#_Toc24733831)

[1.6 – Organização do Trabalho 4](#_Toc24733832)

[2 – REFERENCIAL TEÓRICO 4](#_Toc24733833)

[2.1 – Conceito de Sistemas Web 4](#_Toc24733834)

[2.2 – Netflix 4](#_Toc24733835)

[2.3 – Chaos Engineering 4](#_Toc24733836)

[3 – TECNOLOGIAS UTILIZADAS 4](#_Toc24733837)

[3.1 – Chaos Toolkit 4](#_Toc24733838)

[3.2 – Gremlin 4](#_Toc24733839)

[3.3 – HTML 4](#_Toc24733840)

[3.4 – CSS 4](#_Toc24733841)

[3.5 – JavaScript 4](#_Toc24733842)

[3.6 – Node.js 4](#_Toc24733843)

[3.7 – Angular 4](#_Toc24733844)

[3.8 – Docker 4](#_Toc24733845)

[3.9 – Kubernetes Minikube 4](#_Toc24733846)

[3.10 – AWS 4](#_Toc24733847)

[4 – APRESENTAÇÃO DA SOLUÇÃO 4](#_Toc24733848)

[4.1 – Descrição do Problema 4](#_Toc24733849)

[4.2 – Solução Proposta 4](#_Toc24733850)

[4.3 – Arquitetura do Projeto 4](#_Toc24733851)

[4.3.1 – Calculadora 4](#_Toc24733852)

[4.3.1 – Gerenciador de Tarefas 4](#_Toc24733853)

[4.3.1 – Buscador de Repositórios 4](#_Toc24733854)

[4.3.1 – API REST de Termos Úteis 4](#_Toc24733855)

[4.3.1 – Contêinerização das Aplicações 4](#_Toc24733856)

[4.3.1 – Osquestração dos Contêineres 4](#_Toc24733857)

[4.3.1 – Deploy dos Pods 4](#_Toc24733858)

[4.3.1 – Execução dos Testes 4](#_Toc24733859)

[4.3.1 – Criação dos Testes 4](#_Toc24733860)

[5 – CONCLUSÃO 4](#_Toc24733861)

[5.1 – Contribuições 4](#_Toc24733862)

[5.2 – Trabalhos Futuros 4](#_Toc24733863)

[REFERÊNCIAS BIBLIOGRÁFICAS 4](#_Toc24733864)

**LISTA DE FIGURAS**

[**Figura 1 - Arquitetura do Projeto** 4](#_Toc24732430)

**LISTA DE ABREVIATURAS E SIGLAS**

1. HTML –
2. CSS –
3. AWS –
4. API –

# 1 – INTRODUÇÃO

## 1.1 – Justificativa da Pesquisa

## 1.2 – Problema de Pesquisa

## 1.3 – Hipótese

## 1.4 – Objetivo

## 1.5 – Objeto

## 1.6 – Organização do Trabalho

# 

# 

# 2 – REFERENCIAL TEÓRICO

## 2.1 – Conceito de Sistemas Web

## 2.2 – Netflix

## 2.3 – Chaos Engineering

# 3 – TECNOLOGIAS UTILIZADAS

Nesta seção serão descritas as tecnologias Chaos Toolkit, Gremlin, HTML, CSS, JavaScript, NodeJS, Angular, Docker, Kubernetes Minikube e AWS.

## 3.1 – Chaos Toolkit

## 3.2 – Gremlin

## 3.3 – HTML

## 3.4 – CSS

## 3.5 – JavaScript

## 3.6 – Node.js

## 3.7 – Angular

## 3.8 – Docker

## 3.9 – Kubernetes Minikube

## 3.10 – AWS

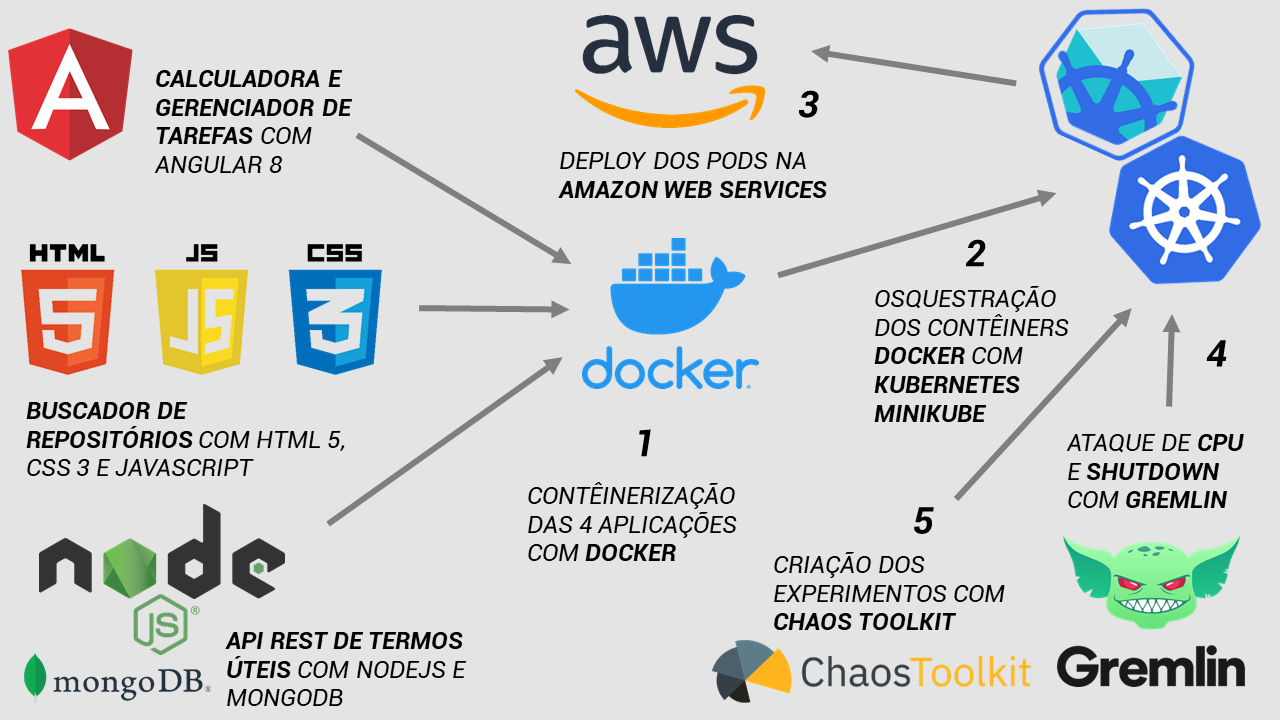
# 4 – APRESENTAÇÃO DA SOLUÇÃO

## 4.1 – Descrição do Problema

## 4.2 – Solução Proposta

## 4.3 – Arquitetura do Projeto

A Figura 1 abaixo representa a arquitetura de todo o projeto, desde a criação das aplicações até a execução e criação dos ataques. O processo é dividido em 5 etapas após a criação das 4 aplicações. 1 – Subir as 4 aplicações em contêineres. 2 – Usar o Minikube para instalação do Kubernetes localmente afim de gerir os contêineres Docker. 3 – Subir os *pods* no Amazon Web Services. 4 – Execução de ataques de CPU e Shutdown com Gremlin. 5 – Criação e execução de ataques com o ChaosToolkit.



**Figura 1 - Arquitetura do Projeto**

Fonte: Autores

## 4.3.1 – Calculadora

## 4.3.1 – Gerenciador de Tarefas

## 4.3.1 – Buscador de Repositórios

## 4.3.1 – API REST de Termos Úteis

## 4.3.1 – Contêinerização das Aplicações

## 4.3.1 – Osquestração dos Contêineres

## 4.3.1 – Deploy dos Pods

## 4.3.1 – Execução dos Testes

## 4.3.1 – Criação dos Testes

# 5 – CONCLUSÃO

## 5.1 – Contribuições

## 5.2 – Trabalhos Futuros

# REFERÊNCIAS BIBLIOGRÁFICAS