

**UNIVERSIDADE DO GRANDE RIO**

***PROF. JOSÉ DE SOUZA HERDY***

ESCOLA DE CIÊNCIA E TECNOLOGIA

**BACHARELADO EM SISTEMAS DE INFORMAÇÃO**

Matheus de Oliveira Manhães

Felipe Oliveira Barbosa

**Chaos Engineering**

Duque de Caxias

2019



**UNIVERSIDADE DO GRANDE RIO**

***PROF. JOSÉ DE SOUZA HERDY***

ESCOLA DE CIÊNCIA E TECNOLOGIA

**BACHARELADO EM SISTEMAS DE INFORMAÇÃO**

Matheus de Oliveira Manhães

Felipe Oliveira Barbosa

**Chaos Engineering**

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.

Orientador: Prof. Dr. Thiago Silva de Souza

Duque de Caxias

2019

**Chaos Engineering**

Matheus de Oliveira Manhães – 5306315

Felipe Oliveira Barbosa – 5306312

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:

1. Orientador e Presidente: Prof. Dr. Thiago Silva de Souza
2. Membro interno: Prof. <MEMBRO DA BANCA>
3. Membro interno: Prof. <MEMBRO DA BANCA>

Duque de Caxias

2019

Matheus de Oliveira Manhães  
Felipe Oliveira Barbosa

Chaos Engineering, Duque de Caxias, 2019

<Nº páginas pré-textuais. Exemplo: XIII>, <número total de páginas do texto do projeto> p. 29,7 cm. (Escola de Ciência e Tecnologia, 2019)

Projeto de Final de Curso - Universidade do Grande Rio, Escola de Ciência e Tecnologia.

1. <Assunto 1>
2. <Assunto 2>
3. <Assunto 3>

I. EIN/UNIGRANRIO II. Título (série)

<DEDICATÓRIA DOS ALUNOS>

**AGRADECIMENTOS**

<AGRADECIMENTOS>

“<EPÍGRAFE>”

<AUTOR EPÍGRAFE>

**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 e descobrir pontos fracos.

Este documento mostra como é feito o processo de Engenharia do Caos através de quatro pequenos sistemas criados para este trabalho: 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 nossos experimentos de Caos.

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

**SUMÁRIO**

[1 – INTRODUÇÃO 15](#_Toc25158201)

[1.1 – Motivação 15](#_Toc25158202)

[1.2 – Problema 15](#_Toc25158203)

[1.3 – Hipótese 15](#_Toc25158204)

[1.4 – Objetivos 15](#_Toc25158205)

[1.5 – Organização do Trabalho 15](#_Toc25158206)

[2 – REFERENCIAL TEÓRICO 16](#_Toc25158207)

[2.1 – Netflix 16](#_Toc25158208)

[2.2 – Chaos Engineering 16](#_Toc25158209)

[3 – PROVA DE CONCEITO 17](#_Toc25158210)

[3.1 – Descrição do Problema 17](#_Toc25158211)

[3.2 – Solução Proposta 17](#_Toc25158212)

[3.3 – Arquitetura da Pesquisa 17](#_Toc25158213)

[3.3.1 – Criação das Aplicações 18](#_Toc25158214)

[**3.3.1.1 – Calculadora** 18](#_Toc25158215)

[**3.3.1.2 – Gerenciador de Tarefas** 20](#_Toc25158216)

[**3.3.1.3 – Buscador de Repositórios** 21](#_Toc25158217)

[**3.3.1.4 – API REST de Termos Úteis** 21](#_Toc25158218)

[3.3.2 – Contêinerização das Aplicações 21](#_Toc25158219)

[3.3.3 – Osquestração dos Contêineres 21](#_Toc25158220)

[3.3.4 – Deploy dos Pods 21](#_Toc25158221)

[3.3.5 – Execução dos Testes 21](#_Toc25158222)

[3.3.5.1 – Primeiro Experimento com Gremlin 21](#_Toc25158223)

[3.3.6 – Criação dos Testes 21](#_Toc25158224)

[3.4 – Tecnologias Utilizadas 21](#_Toc25158225)

[3.4.1 – Chaos Toolkit 21](#_Toc25158226)

[3.4.2 – Gremlin 22](#_Toc25158227)

[3.4.3 – HTML5 23](#_Toc25158228)

[3.4.4 – CSS3 23](#_Toc25158229)

[3.4.5 – JavaScript 24](#_Toc25158230)

[3.4.6 – Node.js 24](#_Toc25158231)

[3.4.7 – Angular 25](#_Toc25158232)

[3.4.8 – Docker 25](#_Toc25158233)

[3.4.9 – Kubernetes Minikube 26](#_Toc25158234)

[3.4.10 – AWS 26](#_Toc25158235)

[4 – CONCLUSÃO 27](#_Toc25158236)

[4.1 – Considerações Finais 27](#_Toc25158237)

[4.2 – Contribuições 27](#_Toc25158238)

[4.3 – Trabalhos Futuros 27](#_Toc25158239)

[REFERÊNCIAS BIBLIOGRÁFICAS 28](#_Toc25158240)

[ANEXO I – Glossário 31](#_Toc25158241)

**LISTA DE FIGURAS**

[Figura 1 - Arquitetura do Projeto 2](#_Toc25155733)

[Figura 2 - Calculadora na Visão Desktop 2](#_Toc25155734)

[Figura 3 - Calculadora na Visão Mobile 2](#_Toc25155735)

[Figura 4 - Tela Inicial sem Tarefas 2](#_Toc25155736)

[Figura 5 - Arquitetura do Chaos Tookit 2](#_Toc25155737)

**LISTA DE ABREVIATURAS E SIGLAS**

1. API
2. AWS
3. CSS3
4. FaaS
5. HTML5

# 1 – INTRODUÇÃO

## 1.1 – Motivação

## 1.2 – Problema

## 1.3 – Hipótese

## 1.4 – Objetivos

## 1.5 – Organização do Trabalho

# 

# 

# 2 – REFERENCIAL TEÓRICO

## 2.1 – Netflix

## 2.2 – Chaos Engineering

# 3 – PROVA DE CONCEITO

## 3.1 – Descrição do Problema

Com o avanço da tecnologia o mundo está cada vez mais conectado. Sistemas web estão sendo construídos e novidades em aplicativos estão sendo lançadas praticamente todos os dias. O grande público, independente da faixa etária, cada vez mais tem se adaptado às novas tecnologias para se manter atualizado às novidades.

Percebendo isso, os aplicativos e sistemas web provocaram uma enorme transformação na maneira como lidamos com a nossa rotina pessoal, profissional, sentimental e acadêmica. Observando essa realidade, é importante refletirmos a respeito do que é necessário para garantir que esses aplicativos e sistemas web se mantenham disponíveis sempre que os usuários precisem utilizar, além de garantir que as falhas e turbulências de produção devem ser consideradas e, além disso, serem tratadas da forma correta a fim de não mais ocorrerem ou ocorrerem em menor escala, garantindo a melhor experiência possível para o usuário.

Levando em conta esta problemática...

## 3.2 – Solução Proposta

Com o objetivo de solucionar este problema...

## 3.3 – Arquitetura da Pesquisa

A Figura 1 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 – Uso do Docker para configurar as 4 aplicações em contêineres. 2 – Uso do Minikube para instalação do Kubernetes localmente afim de gerir os contêineres Docker. 3 – D*eploy* dos *pods* no Amazon Web Services. 4 – Execução dos ataques de CPU e Shutdown com Gremlin. 5 – Criação e execução de ataques com o Chaos Toolkit.

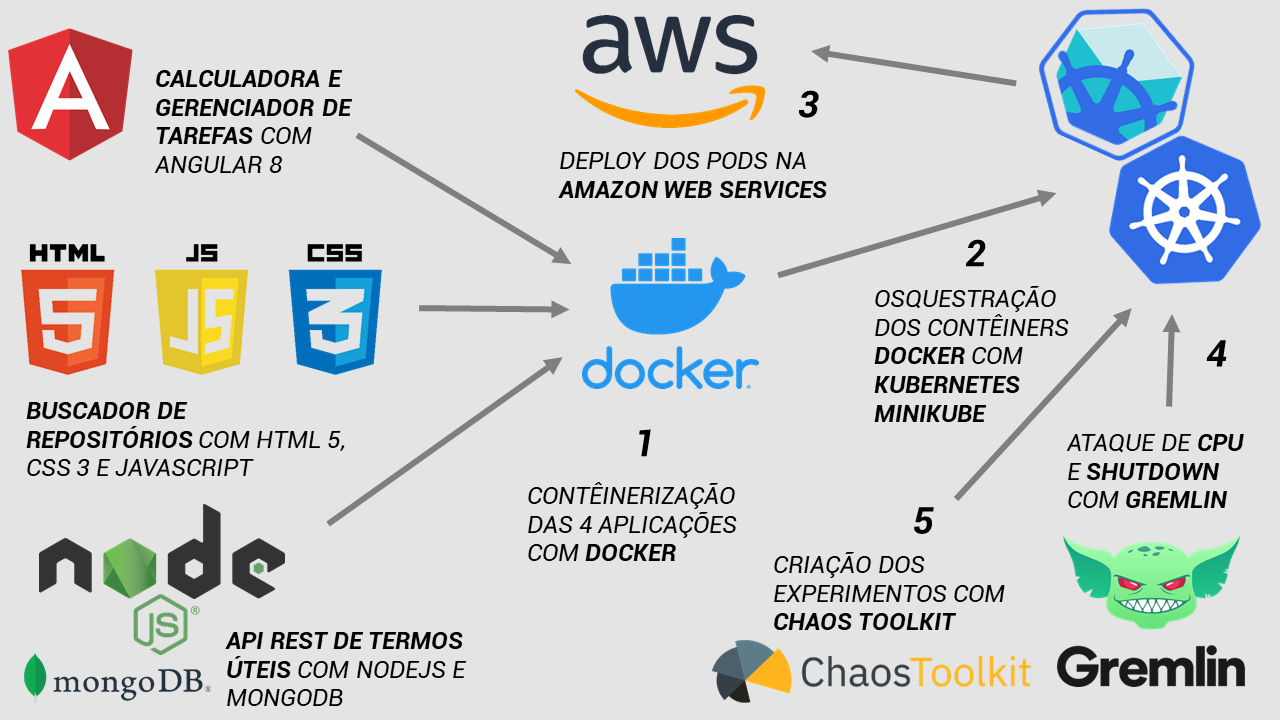


Figura 1 - Arquitetura do Projeto

### 3.3.1 – Criação das Aplicações

#### **3.3.1.1 – Calculadora**

Feita com Angular 8, a calculadora é uma aplicação simples que realiza tudo o que uma calculadora comum faz: operações de soma, adição, subtração e divisão. Tem o visual responsivo, ou seja, a interface se adapta independente do dispositivo que o usuário esteja utilizando (computador, TV, celular, tablet etc.). A Figura 2 e a Figura 3 mostram a única tela que essa aplicação possui, tanto na visão pelo computador, como pelo celular.

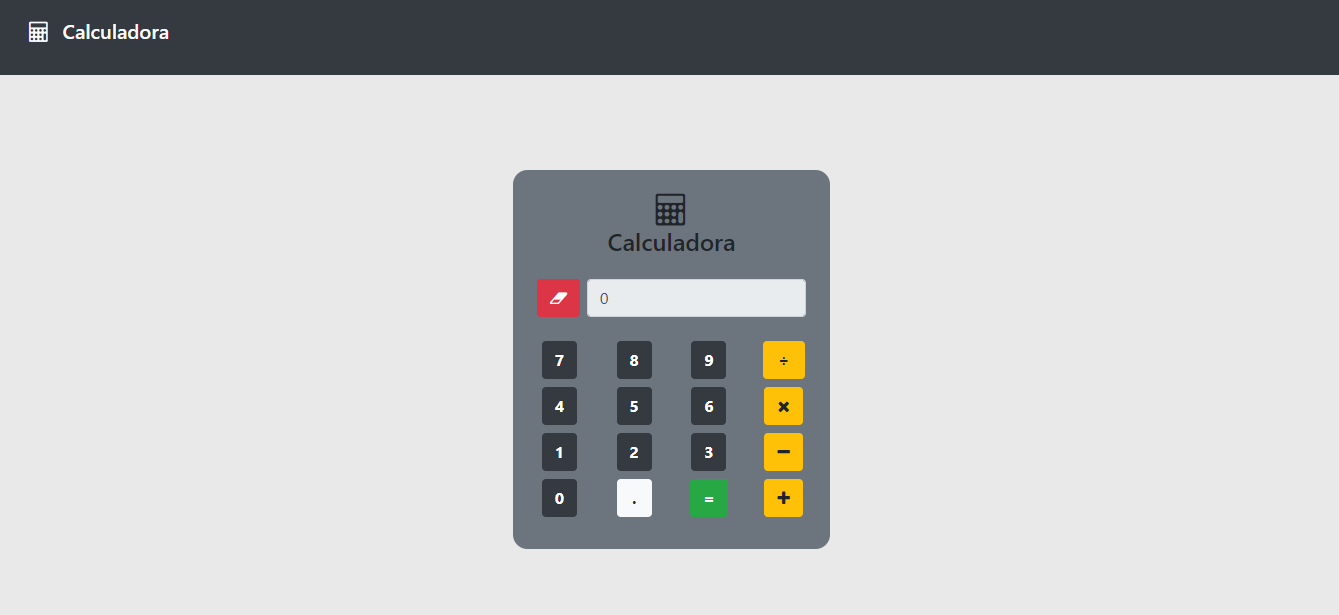


Figura 2 - Calculadora na Visão Desktop



Figura 3 - Calculadora na Visão Mobile

#### **3.3.1.2 – Gerenciador de Tarefas**

O Gerenciador de Tarefas é uma aplicação que permite o usuário incluir, remover, editar, excluir e alterar o status de tarefas. Desenvolvido com o Angular em sua versão 8, tem a interface responsiva e de fácil interação. A Figura 4 mostra como é a sua tela inicial sem haver tarefa alguma cadastrada. Acima da tabela que exibe as tarefas, existe um botão chamado “Adicionar Tarefa”. Ao clicar neste botão a tela de formulário de inclusão de tarefas é exibido.

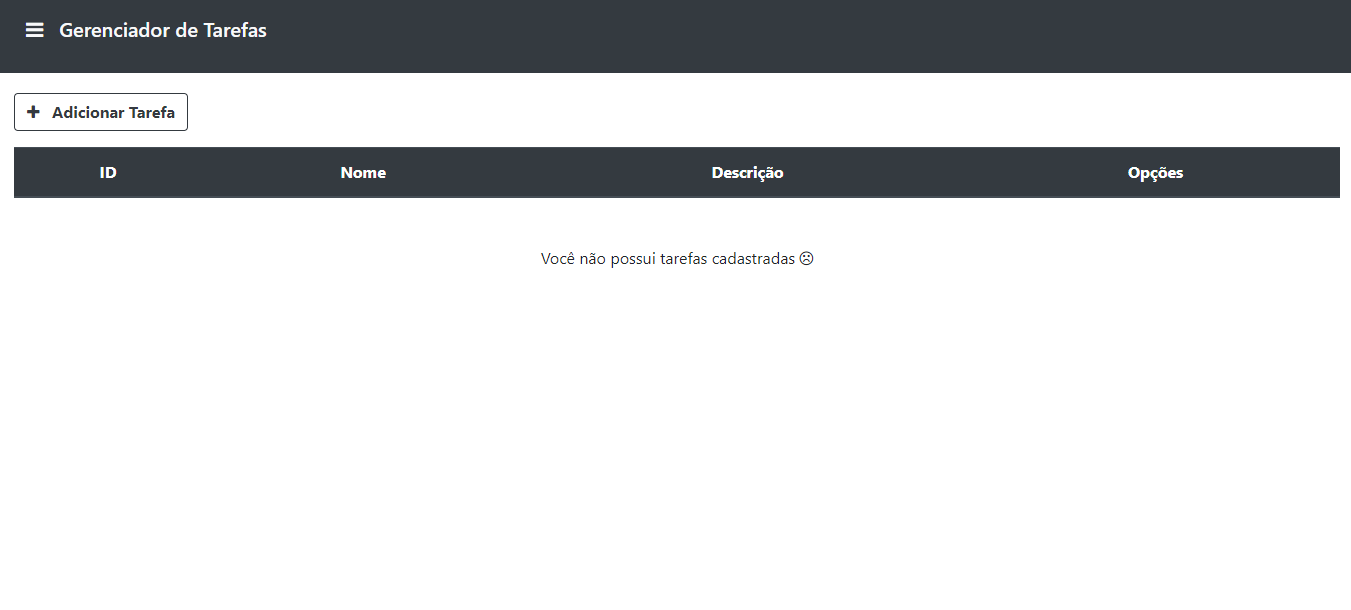


Figura 4 - Tela Inicial sem Tarefas

#### **3.3.1.3 – Buscador de Repositórios**

#### **3.3.1.4 – API REST de Termos Úteis**

### 3.3.2 – Contêinerização das Aplicações

### 3.3.3 – Osquestração dos Contêineres

### 3.3.4 – Deploy dos Pods

### 3.3.5 – Execução dos Testes

#### **3.3.5.1 – Primeiro Experimento com Gremlin**

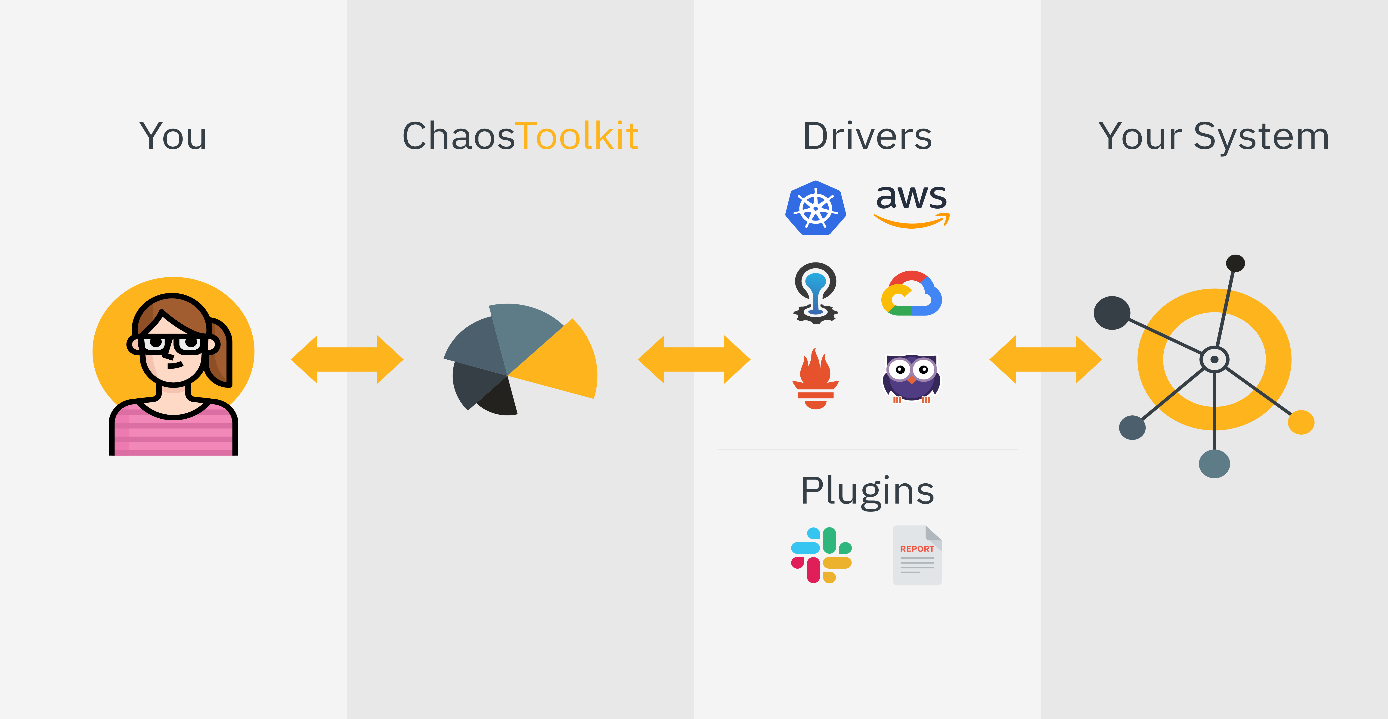
### 3.3.6 – Criação dos Testes

## 3.4 – Tecnologias Utilizadas

Nesta seção serão descritas as tecnologias Chaos Toolkit, Gremlin, HTML5, CSS3, JavaScript, NodeJS, Angular, Docker, Kubernetes Minikube e AWS.

### 3.4.1 – Chaos Toolkit

Chaos Toolkit é um framework de código aberto que disponibiliza um kit de ferramentas para execução e criação de experimentos de Chaos Engineering. O framework tem dois principais propósitos: prover uma ferramenta completa, porém simples a fim de fornecer um ponto de partida fácil para a aplicação da disciplina (*Chaos Engineering*) e disponibilizar uma API aberta para a comunidade para que qualquer experimento de Caos possa ser executado consistentemente usando integrações com outras ferramentas que estão surgindo. A Figura 5 mostra como esse processo é simples.



**Figura 5 - Arquitetura do Chaos Tookit**

Fonte: Documentação do Chaos Tookit

O Chaos Toolkit visa tornar simples e direto, seguindo os Princípios da Engenharia do Caos, a execução de experimentos contra o seu sistema ativo, a fim de criar confiança em seu comportamento e aprender sobre possíveis pontos fracos.

Foi escolhida esta ferramenta para poder escrever casos de testes similares ao Gremlin. O uso do Chaos Toolkit permite a criação dos experimentos, execução dos mesmos e integração com os outros serviços como o Gremlin, Docker, Kubernetes e AWS.

### 3.4.2 – Gremlin

O Gremlin é uma ferramenta paga, mas oferece alguns recursos grátis, porém limitados. Criada a fim de “Transformar o fracasso em resiliência”, como diz em sua documentação, o Gremlin oferece uma estrutura a fim simular interrupções reais com segurança e com uma crescente biblioteca de ataques. Gremlin se encaixa na categoria FaaS, onde oferece uma plataforma para os usuários encontrarem pontos fracos em seus sistemas antes que eles possam causar problemas.

Foi escolhido o Gremlin, pois é possível realizar alguns experimentos que estão divididos pelas categorias “Resource”, “State” e “Network”. Cada categoria possui uma lista de ataques de acordo com cada perfil. Na versão grátis estão disponíveis dois ataques: CPU na categoria “Resource” e “Shutdown” na categoria “State”. Escolhendo o Gremlin como ferramenta deste trabalho é possível executar experimentos de aumento do consumo de CPU do servidor assim como desligá-lo programadamente com alguns passos de configuração.

### 3.4.3 – HTML5

HTML é sigla para HyperText Markup Language, que significa Linguagem de Marcação de Hipertexto. HTML é a linguagem de marcação padrão para páginas Web. Com ela é possível criar páginas web estáticas e dinâmicas. HTML descreve a estrutura de uma página da Web que consiste em uma série de elementos que informam ao navegador como exibir o conteúdo a quem está acessando a página.

Os elementos HTML são representados por *tags* que rotulam partes do conteúdo, como "cabeçalho", "parágrafo", "tabela" e assim por diante. Os navegadores não exibem as tags HTML, mas as usam para exibir o conteúdo da página.

Para este projeto escolhemos utilizar o HTML5 que é a versão mais atual da linguagem que possui novos elementos, atributos e comportamentos. Com ela, segundo sua documentação na página do Mozilla, é possível descrever mais precisamente o nosso conteúdo por conta da inserção das *tags* semânticas, além de permitir que páginas web armazenem dados localmente no lado do cliente (navegador)entre outras muitas funcionalidades que auxiliam e muito na rotina do programador.

### 3.4.4 – CSS3

CSS é uma sigla para Cascading Style Sheets, Folhas de Estilo em Cascata traduzindo para a Língua Portuguesa. CSS é a linguagem que define o estilo (formatação, cores, espaçamento, posição de elementos etc.) de um documento HTML, ou seja, descreve como os elementos HTML devem ser exibidos.

O estilo da página com CSS pode ser feito de várias maneiras: usando uma *tag* de estilono documento inteiro, usando uma propriedade de estilo para cada *tag* ou criando um link para um arquivo *.css* externo ao documento de conteúdo (HTML).

Neste projeto 3 aplicações utilizam páginas *web* feitas com HTML, então foi decidido utilizar CSS3 que é a versão mais atual da ferramenta junto com o framework do Twitter chamado Bootstrap, que é um conjunto de estilos previamente definidos e configurados onde podemos ter acesso usando sua documentação na internet. Optamos por utilizar CSS3 com Bootstrap pelo fato de ser uma biblioteca muito utilizada em muitos sites renomados e por possuir estilos criativos e funcionais.

### 3.4.5 – JavaScript

Uma linguagem de programação que surgiu em 1995 que tinha como objetivo ser linguagem de programação para o lado do cliente (navegador), algo que nos últimos anos vem sendo resignificado desde o surgimento do Node.js, um interpretador javascript para o lado do servidor.

JavaScript é uma linguagem multi-paradigma de tipagem dinâmica que não faz uso de compilador por ser uma linguagem interpretada, portanto a maioria de seus erros podem apenas ser descobertos em tempo de execução.

Foi o escolhido o JavaScript como linguagem de programação, pois faz a integração perfeita com as tecnologias HTML5 e CSS3. JavaScript é a linguagem mais utilizada na web para fazer a interação dos elementos HTML e estilo CSS como botões, barras de navegações, fundos animados e muitas outras possibilidades.

### 3.4.6 – Node.js

É um interpretador de código que tem como propósito interpretar o JavaScript no lado do servidor, coisa que só era possível no lado do cliente (navegador). Isso é possível hoje, pois Ryan Dahl, engenheiro por trás do Node.js, utilizou o motor *open-source* do Google Chrome, chamado V8, no lado do servidor, assim sendo capaz de interpretar código JavaScript não somente no lado do cliente, como era até 2009.

Foi o escolhido o Node.js como tecnologia para o desenvolvimento da API de termos úteis de Chaos Engineering pela sua simplicidade para “subir” um servidor e, como a lógica desta API é algo simples, ou seja, somente um CRUD de termos úteis com ID, título e descrição, foi escolhido o Node.js, junto com a biblioteca Express e o banco de dados não relacional MongoDB para construção.

### 3.4.7 – Angular

Angular é um framework usado para construção da interface de aplicações Web usando HTML, CSS e, principalmente JavaScript. O framework é mantido pela Google e a sua primeira versão chamada AngularJS foi totalmente reescrita e passou, a partir da versão 2, a ser chamado somente de “Angular”. A partir da versão 2 fizeram um framework totalmente novo. Entre as muitas mudanças do AngularJS para as versões superiores, destaca-se a linguagem padrão de desenvolvimento do Angular que passou a ser o TypeScript, um superconjunto da Microsoft que insere tipagem no JavaScript.

Para este projeto foi escolhido o uso do Angular para o desenvolvimento da calculadora e do gerenciador de tarefas, pois é um framework mantido por uma empresa de alta credibilidade, além de ser um framework excelente para criação de SPAs, ou Single Page Applications.

### 3.4.8 – Docker

O Docker é uma plataforma aberta de desenvolvimento, envio e execução de aplicações. Fornece a capacidade de empacotar e executar um aplicativo em um ambiente pouco isolado chamado contêiner. O isolamento e a segurança permitem executar muitos contêineres simultaneamente em um determinado host.

Foi escolhido o Docker para este projeto, pois com o Docker é possível criar imagens (containers prontos para *deploy*) a partir de arquivos de definição chamados Dockerfiles com facilidade, além do Docker utilizar como *backend* *default* o LXC, sendo possível definir limitações de recursos por container (memória, CPU e I/O, por exemplo). Sendo assim, o Docker “caiu como uma luva” para que pudéssemos “subir” as 4 aplicações em contêineres, visto que também seria fácil a integração com o Kubernetes e o AWS, que serão tratados nos próximos tópicos.

### 3.4.9 – Kubernetes Minikube

O Kubernetes, também chamado de k8s, é uma plataforma de código aberto que automatiza as operações dos containers Linux. Essa plataforma elimina grande parte dos processos manuais necessários para implantar e escalar as aplicações em containers. O Kubernetes oferece um serviço para que os *pods* (grupo de um ou mais contêineres) sejam facilmente gerenciados.

Já o Minikube é uma ferramenta que facilita a execução local do Kubernetes. O Minikube executa um cluster Kubernetes de nó único dentro de uma Máquina Virtual (VM).

Esse projeto faz uso do Kubernetes com Minikube, pois ele tem a responsabilidade de gerenciar as aplicações com facilidade e eficiência. O Kubernetes foi feito para gerenciar *clusters* que podem incluir hosts em *clouds* públicas, privadas ou híbridas. Por isso, o Kubernetes é a plataforma ideal para hospedar aplicações nativas em *cloud* que exigem escalabilidade rápida, algo que é essencial para o projeto, visto que serão realizados testes e experimentos de Chaos Engineering, onde será necessário, por exemplo, “subir” serviços que foram derrubados, analisar indisponibilidade dos serviços e configurar redirecionamento de fluxo.

### 3.4.10 – AWS

Amazon Web Services, também conhecido como AWS, é uma plataforma de serviços de computação em nuvem. É a maior empresa fornecedora de serviços em nuvem, oferecendo cerca de 165 serviços completos incluindo série de aplicativos, incluindo computação, armazenamento, bancos de dados, redes, análises, machine learning e inteligência artificial (IA), Internet das Coisas (IoT), segurança e desenvolvimento, implantação e gerenciamento de aplicativos.

Foi escolhido o uso do AWS para este projeto para não precisar fazer o *setup* e o *deploy* de toda a configuração e recursos do projeto em uma máquina física. Seria extremamente complicado ter que deixar um computador disponível e ligado para que pudéssemos realizar as atividades deste trabalho, além de termos a possibilidade de analisarmos os gráficos de monitoramento do serviço da Amazon, visto que o AWS oferece serviços *cloud* de maneira excepcional, onde grandes empresas utilizam de seus recursos, além de oferecer serviços grátis com fins experimentais.

# 4 – CONCLUSÃO

## 4.1 – Considerações Finais

## 4.2 – Contribuições

## 4.3 – Trabalhos Futuros

# REFERÊNCIAS BIBLIOGRÁFICAS

**Links Acessados**

Chaos Toolkit. **Chaos Tookit Site.** Disponível em: <https://docs.chaostoolkit.org/>. Acesso em: 15 de novembro de 2019.

GitHub Chaos Toolkit. **Chaos Tookit Documentation.** Disponível em: <https://github.com/chaostoolkit/chaostoolkit/>. Acesso em: 15 de novembro de 2019.

Gremlin. **Gremlin Documentation.** Disponível em: <https://www.gremlin.com/docs/>. Acesso em: 15 de novembro de 2019.

W3Schools. **HTML Tutorial.** Disponível em:

<https://www.w3schools.com/html/>. Acesso em: 15 de novembro de 2019.

MDN Web Docs. **HTML5 Documentation.** Disponível em:

<https://developer.mozilla.org/pt-BR/docs/Web/HTML/HTML5/>. Acesso em: 15 de novembro de 2019.

W3Schools. **CSS Tutorial.** Disponível em:

<https://www.w3schools.com/css/>. Acesso em: 15 de novembro de 2019.

Bootstrap. **Bootstrap Documentation.** Disponível em:

<https://getbootstrap.com/docs/4.3/getting-started/introduction/>. Acesso em: 15 de novembro de 2019.

MDN Web Docs. **Documentação JavaScript.** Disponível em:

<https://developer.mozilla.org/pt-BR/docs/Aprender/JavaScript/>. Acesso em: 15 de novembro de 2019.

Node.js. **Documentação Node.js.** Disponível em:

<https://nodejs.org/pt-br/about/>. Acesso em: 15 de novembro de 2019.

Angular. **Documentação Angular.** Disponível em:

<https://angular.io/docs/>. Acesso em: 16 de novembro de 2019.

Docker. **Documentação Docker.** Disponível em:

<https://docs.docker.com/>. Acesso em: 16 de novembro de 2019.

Mundo Docker. **O que é Docker?** Disponível em: <https://www.mundodocker.com.br/o-que-e-docker/>. Acesso em: 16 de novembro de 2019.

Kubernetes Concepts. **O que é um pod?** Disponível em: <https://kubernetes.io/docs/concepts/workloads/pods/pod/>. Acesso em: 16 de novembro de 2019.

Red Hat. **O que é Kubernetes?** Disponível em: <https://www.redhat.com/pt-br/topics/containers/what-is-kubernetes/>. Acesso em: 16 de novembro de 2019.

Kubernetes. **Site Oficial.** Disponível em: < https://kubernetes.io/pt/docs/home/>. Acesso em: 16 de novembro de 2019.

Kubernetes. **Getting Started – Installing Kubernetes with Minikube.** Disponível em: < https://kubernetes.io/docs/setup/learning-environment/minikube/>. Acesso em: 16 de novembro de 2019.

AWS. **Documentação da AWS.** Disponível em: <https://docs.aws.amazon.com/index.html?nc2=h\_ql\_doc\_do\_v />. Acesso em: 16 de novembro de 2019.

Site Oficial do AWS. **O que é AWS?** Disponível em: <https://aws.amazon.com/pt/what-is-aws/>. Acesso em: 16 de novembro de 2019.

# ANEXO I – Glossário