

Engenharia do Caos com Gremlin e Chaos Toolkit

Felipe Barbosa de Oliveira e Matheus de Oliveira Manhães



Sistema de Informação – Universidade do Grande Rio Professor José de Souza Herdy

Rua Professor José de Souza Herdy, 1160 – 25073-200 – Duque de Caxias - RJ
Departamento de Sistemas de Informação

Universidade do Grande Rio Professor José de Souza Herdy (UNIGRANRIO) – Duque de Caxias, RJ – Brasil

mmanhaes@unigranrio.br, felipeo@unigranrio.br

Abstract. *Aguardando definição do resumo para escrever esta seção.*

R*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. 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 auxilia 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 dos experimentos de Caos.

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

1. Introdução

1.1. Motivação

Aqui será descrita a motivação.

1.2. Problema

Aqui será descrito o problema.

1.3. Objetivos

Aqui serão descritos os objetivos.

1.4. Organização do Trabalho

Este trabalho está organizado da seguinte forma: o capítulo 1 descreve a parte introdutória do problema, assim como a motivação, objetivos e a forma como o trabalho está organizado; o capítulo 2 descreve o referencial teórico onde é abordado sobre o Netflix, Engenharia do Caos e qual a relação que existe entre esses dois itens; no capítulo 3 é apresentada a prova de conceito descrevendo o problema, a solução proposta e a arquitetura da pesquisa, onde são detalhadamente descritas a forma como as aplicações alvo foram construídas e como funcionam suas interações, assim como a preparação do ambiente para a execução e criação de ataques para o experimento de Engenharia do Caos; o capítulo 4 é apresentada a conclusão deste trabalho abordando as considerações finais, contribuições e trabalhos futuros; finalmente, no capítulo 5 são apresentadas as referências bibliográficas.

2. Referencial Teórico

Este capítulo apresenta um breve histórico sobre a empresa de streaming Netflix, Inc. e como a disciplina de Engenharia do Caos foi incorporada à cultura da empresa até se tornar uma prática comum em seu processo de desenvolvimento. Também será abordado, de uma forma geral, o conceito de Engenharia do Caos.

2.1 Netflix

Segundo a NETFLIX (2019), a empresa é líder em conteúdo digital desde 1997, sendo um dos principais serviços de entretenimento pela internet no mundo, com quase 158 milhões de assinaturas ativas em mais de 190 países e conteúdo que engloba filmes, séries e documentários de diversos gêneros e idiomas.

O que inicialmente era um serviço baseado de aluguéis de DVDs, atualmente disponibiliza aos seus assinantes a possibilidade de assistirem os seus conteúdos quando e onde quiserem, em praticamente qualquer tela com uma conexão à internet.

Observando a demanda cada vez maior pelos serviços oferecidos e a crescente base de usuários, em 2010 a Netflix deu início a um processo de migração de sua infraestrutura, saindo dos data centers para a nuvem, onde os sistemas teriam uma melhor escalabilidade e gerenciamento de recursos.

Devido ao processo de migração da infraestrutura para a nuvem, foi detectado que a escalabilidade horizontal multiplicaria o número de instâncias que executam um dado serviço. Com milhares dessas instâncias em execução, era questão de tempo que uma

dessas máquinas virtuais falham por algum motivo. Havia a necessidade de os serviços serem construídos visando a resiliência à essas falhas.

Desde então foram realizados diversos testes de resiliência em ambiente de produção com o objetivo de comprovar a qualidade do serviço e capacidade de operação em um cenário de uso real. Todo esse processo mais tarde seria conhecido como Engenharia do Caos.

Ferramentas foram criadas com o objetivo de controlar o processo. O Chaos Monkey sendo uma das primeiras a serem utilizadas, seguido do Kaos Kong em uma escala maior e o FIT (Failure Injection Testing) ocupando a lacuna entre as duas primeiras.

Além dos Princípios do Caos foram desenvolvidos, uma plataforma de automação de testes foi implementada para executar continuamente os experimentos sobre a arquitetura de microserviços e foi observado que a Engenharia do Caos tem o objetivo de tornar visível o caos já inerente aos sistemas complexos.

É importante ressaltar que organizações como a Google, Amazon, Microsoft, Dropbox e outras de diversas áreas também vêm adotando a prática da Engenharia do Caos no seu processo de desenvolvimento com o objetivo de tornar seus processos e sistemas mais resilientes.

2.2. Chaos Engineering

A Engenharia do Caos, traduzindo Chaos Engineering para o português, é disciplina que estuda o comportamento de um sistema através de experimentos com o objetivo de construir confiança e resiliência para superar as imprevisibilidades diárias. (Princípios do Caos, 2018).

Ao falar em “caos” podemos ter uma ideia de algo sem organização, sem lógica e totalmente incontrolado, no entanto a Engenharia do Caos não consiste em induzir falhas aleatórias sem qualquer planejamento ou objetivo, muito pelo contrário, entender as condições que podem induzir essas falhas e contingenciar as mesmas é o objetivo principal desta disciplina.

Há um desafio na manutenção e implementação de melhorias em sistemas que atingem certo nível de crescimento e complexidade com o alto número de componentes interagindo entre si. Independente da arquitetura utilizada no sistema, com o desacoplamento de componentes também temos uma série de fatores que podem impactar na sua execução.

Embora não seja viável prever todas as falhas de um sistema, muitas delas podem ser detectadas antes de serem percebidas na execução onde os danos são maiores. Com o diagnóstico em mãos, é possível corrigir as falhas antes de atingirem o usuário final tornando o sistema mais confiável (ROSENTHAL, 2017).

A Engenharia do Caos utiliza testes em alguns pontos específicos do sistema, no entanto ela não se baseia somente nessa prática. Os testes servem como parte do processo de conhecimento sobre o sistema experimentado, indica se uma condição específica está ou não dentro dos padrões, enquanto a Engenharia do Caos busca olhar a interação dos componentes do sistema como um meio para corrigir as falhas e implementar melhorias.

Os experimentos possibilitam revelar detalhes do sistema que normalmente passariam despercebidos ocasionando uma falha futura. Eles não se baseiam apenas em condições de testes binários aplicados em um componente isolado, mas executado de maneira a observar o comportamento do sistema em diversas condições que simulam possíveis falhas reais tendo como foco as áreas críticas desse sistema.

Experimentos ajudam na compreensão do comportamento dos sistemas a partir de uma abordagem empírica, a sua execução gera resultados que nos permite visualizar o seu comportamento em diferentes situações.

Podemos ter como exemplo um crescente número de clientes acessando um site de compras, o que eleva o aumento no tráfego e gera uma receita maior. Neste caso, não temos uma falha em questão, mas é uma boa oportunidade para exploração e experimentação sobre o impacto de aumento do número de usuários no sistema do ponto de vista da Engenharia do Caos.

Quando temos o conhecimento de como os efeitos sistêmicos do caos afetam o sistema e seus componentes em ambiente real de sua execução, podemos implementar melhorias que contribuirão para evitar possíveis falhas e aumentar a confiabilidade deste mesmo sistema (ROSENTHAL, 2017).

Também devemos estar cientes que existem alguns pré-requisitos antes de aplicarmos a engenharia do caos a um determinado sistema. O sistema deve atender requisitos básicos de testes e monitoramento, bem como se encontrar em um estado normal de execução com falhas previamente conhecidas devidamente corrigidas. O objetivo da Engenharia do Caos é dar visibilidade a falhas ocultas.

Outro ponto é a necessidade de alguma forma de monitoramento para visualizar o estado atual do sistema e seu comportamento durante as condições turbulentas dos experimentos executados. Essa comparação de estados permite a análise de resultados dos experimentos, gerando conclusões acerca dos mesmos.

2.2.1. Complexidade Sistêmica

Gerenciar um sistema é um desafio para qualquer arquiteto responsável pela sua execução. Este desafio se multiplica na mesma proporção de tamanho do sistema. Na medida em que são adicionados componentes, a complexidade aumenta ainda mais dificultando a sua manutenção e exigindo um esforço maior das equipes responsáveis por ele.

Ao adotar a arquitetura de microsserviços, a Netflix se deparou com a dificuldade de administrar os inúmeros serviços ativos. Se por um lado eles permitiam que os times trabalhassem de forma independente, a mesma arquitetura gerou um esforço maior de coordenação entre as equipes.

De fato, tanto nas arquiteturas monolíticas como nas arquiteturas com menor acoplamento como as baseadas em microserviços, a medida em que a complexidade aumenta, fica humanamente impossível para um único indivíduo ter uma visão do comportamento e interação de todos os seus componentes. É neste ponto em que a Engenharia do Caos vem para dar suporte, atuando para melhorar a legibilidade, exibindo comportamentos até então ocultos aos olhos humanos. (ROSENTHAL, 2017).

A partir de certo ponto, testes clássicos, funcionais e de integração se tornam insuficientes devido a integração de componentes, uma abordagem que possibilite observar uma gama maior de interação se faz necessária. Se torna preciso ir além das propriedades já conhecidas do sistema.

Neste cenário podemos ter algo conhecido como “efeito chicote” aonde uma pequena perturbação na entrada de dados é capaz de desestabilizar o fluxo de interação entre componentes alterando drasticamente a saída esperada, sem que isto seja detectado em testes isolados, nessa situação, cada componente agiria da forma esperada, porém um evento em cadeia causado por uma certa entrada de dados alteraria a saída

2.2.2. Princípios do Caos

.

2.2.2.1. O estado de prontidão

.

2.2.3. Precursores do caos

Breve descrição das primeiras ferramentas desenvolvidas na Netflix para a aplicação da engenharia do caos e a necessidade por trás de seu desenvolvimento.

2.2.3.1. Chaos Monkey

.

2.2.3.1. Chaos Kong

.

3. Prova de Conceito

Este capítulo está dividido em quatro seções. A seção 3.1 aborda a descrição do problema e como isso motiva toda a solução proposta, a seção 3.2 apresenta a solução proposta com o intuito de justificar a pesquisa e o problema, a seção 3.3 apresenta a arquitetura da pesquisa descrevendo o processo de criação e apresentação das aplicações, o setup de contêinerização e gestão das aplicações, assim como a execução e criação dos ataques e, finalmente, a seção 3.4 apresenta as tecnologias utilizadas, ou seja, todos os recursos de software, ferramentas, bibliotecas, frameworks, linguagens de programação e tecnologias em geral que foram necessárias para elaboração do projeto.

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 frequentemente. 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 refletir 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, decidiu-se explorar a maneira como é possível garantir disponibilidade e resiliência de aplicações de grandes empresas como Netflix, analisando o seu processo de Engenharia do Caos a fim de gerar conhecimento levando em consideração seus experimentos e como suas falhas podem se tornar aprendizado no contexto de desenvolvimento de software.

3.2. Solução Proposta

Com o objetivo de explorar este problema, foi proposta a simulação de um ambiente de produção onde são servidas aplicações web construídas a fim de serem alvos de ataques de experimentos realizados baseando-se nos princípios da Engenharia do Caos.

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 quatro aplicações. 1 – Uso do Docker para configurar as quatro aplicações em contêineres. 2 – Uso do Minikube para instalação do Kubernetes localmente afim de gerir os contêineres Docker. 3 – Deploy 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.

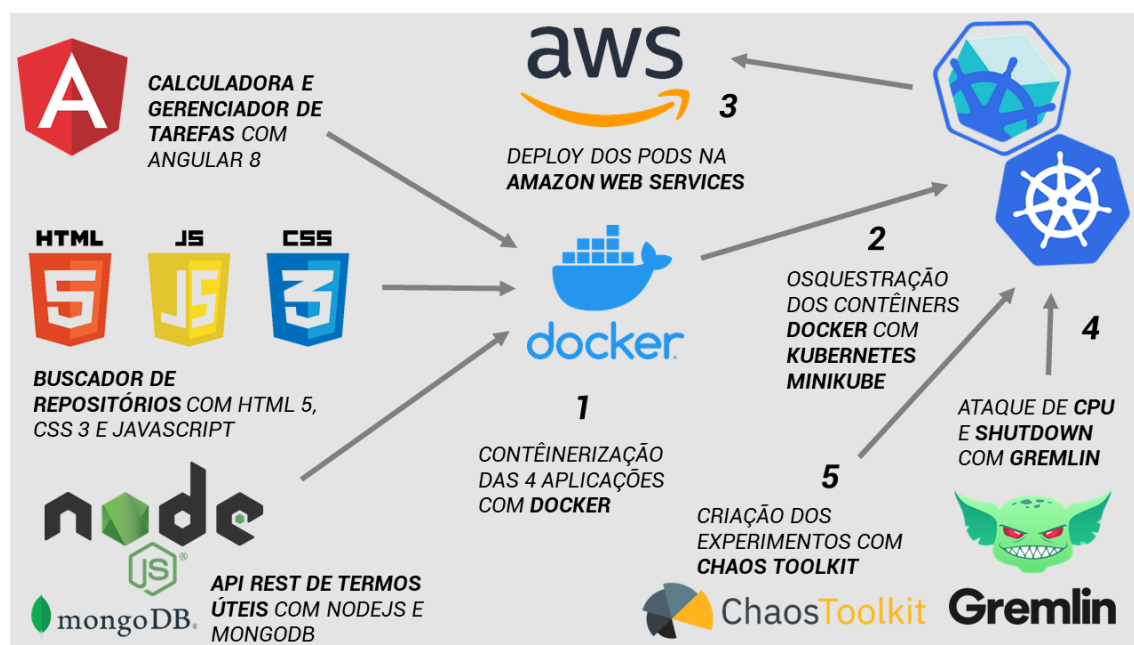


Figura 1 - Arquitetura da Pesquisa.

3.3.1. Aplicações Alvo

Nesta seção serão apresentadas as aplicações que foram criadas para simular um ambiente comum de produção a fim de servirem como alvos para os experimentos do Gremlin e os

experimentos criados com Chaos Toolkit. Estas aplicações estão divididas em seções: a seção 3.3.1.1 descrevendo a calculadora, 3.3.1.2 descrevendo o gerenciador de tarefas, 3.3.1.3 o buscador de repositórios e finalmente a seção 3.3.1.4 descrevendo a API REST que permite a inclusão, remoção, edição, listagem e pesquisa de termos úteis relacionados a Engenharia do Caos.

3.3.1.1. Calculadora

Com Angular 8, a calculadora é uma aplicação simples que realiza tudo o que uma calculadora comum faz: operações de soma, subtração e divisão. Tem o visual responsivo, ou seja, a interface se adapta independentemente do dispositivo que o usuário esteja utilizando (computador, TV, celular, tablet). A Figura 2 mostra a única tela que essa aplicação possui.

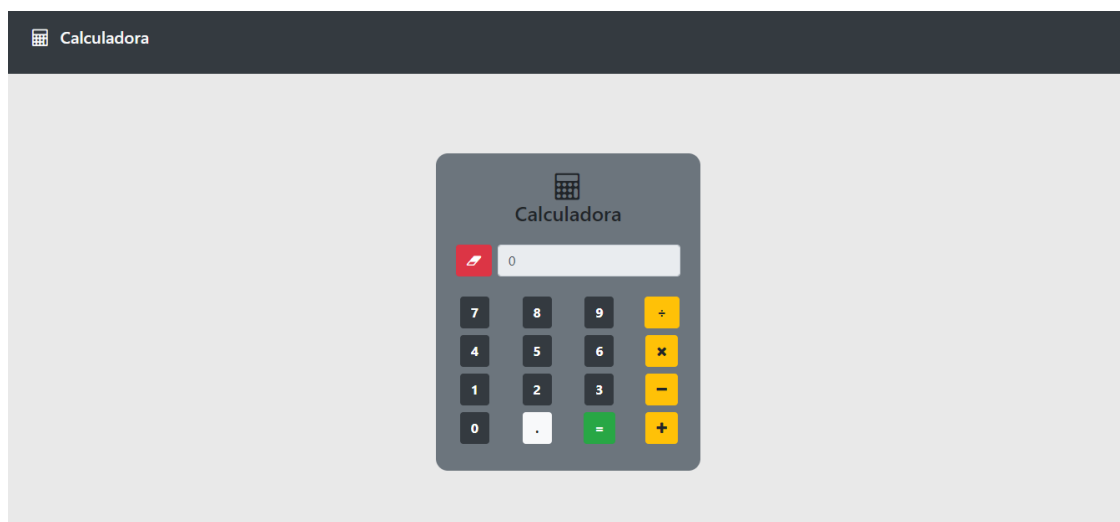


Figura 2 - Calculadora na visão *desktop*.

3.3.1.1. Gerenciador de Tarefas

O Gerenciador de Tarefas é uma aplicação que permite o usuário incluir, remover, editar, pesquisar, listar e alterar o status de tarefas. Desenvolvido com o Angular em sua versão 8, tem a interface responsiva e de fácil interação. A Figura 3 mostra como é a sua interface inicial sem haver tarefa alguma cadastrada. Acima da tabela que exibe as tarefas, existe um botão chamado “Adicionar Tarefa”. Ao clicar neste botão, o formulário de inclusão de tarefas é exibida.

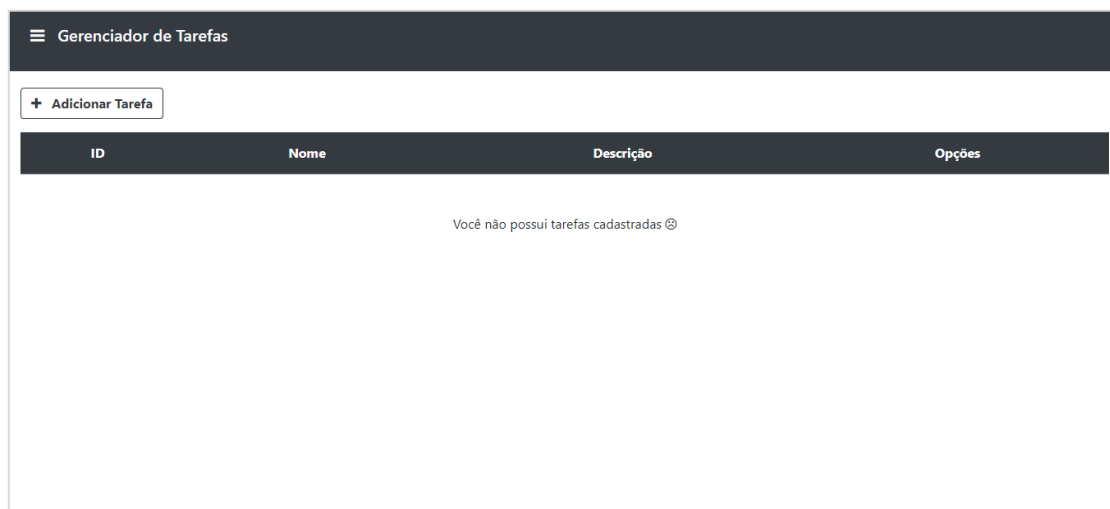


Figura 3 - Tela inicial sem tarefas.

Assim que o botão para adicionar tarefas é selecionado, um formulário que solicita informações como nome e descrição da tarefa é exibido. A Figura 4 ilustra esta tela.

Figura 4 - Formulário de inclusão de tarefas.

No formulário de inclusão de tarefas é obrigatório a inclusão do nome e descrição da tarefa. Ao clicar no botão de cancelar, todos os dados do formulário são perdidos e o usuário é conduzido pelo fluxo da aplicação até a tela de listagem de tarefas. Se o usuário clicar no botão de salvar a tarefa, os dados do formulário são salvos no *local storage*, onde ficam salvos no lado do cliente (navegador), não expiram, só são removidos por ação do cliente e são exibidos formatados e estilizados na tela de listagem de tarefas, conforme mostram a Figura 5 e a Figura 6.

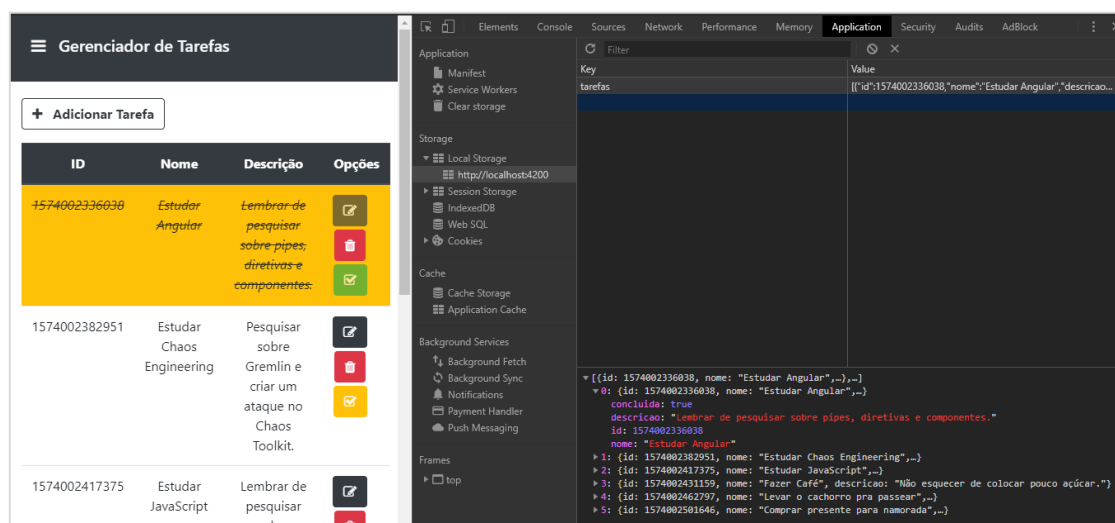


Figura 5 - Listagem de tarefas salvas no *local storage*.



Figura 6 - Listagem de tarefas.

A Figura 6 ilustra a tela de exibição das tarefas criadas em uma lista. É possível perceber que a tabela que faz a listagem de tarefas possui quatro colunas: a primeira exibe os IDs gerados automaticamente para cada tarefa, a segunda exibe os nomes escolhidos pelo usuário no formulário de inclusão de tarefas, assim como na terceira coluna que exibe a descrição. A quarta coluna exibe três botões para que seja possível, nesta ordem, editar, excluir e alterar o status de uma tarefa para marcá-la como pronta. É possível perceber na Figura 6 que algumas tarefas foram marcadas como prontas, ou seja, tiveram seus status alterados, a partir dessa condição, a cor de fundo desta tarefa é alterada para amarelo, o texto também tem seu estilo alterado para que fique em *itálico* e *taxado*. Além disso, é possível perceber que o botão de editar e alterar status ficam desabilitados, somente dando a opção de o usuário excluir aquela tarefa, partindo do pressuposto de que se o usuário concluiu tal tarefa, a mesma não pode ser editada nem ter seu status alterado novamente.

Caso o usuário pressione o botão para editar uma tarefa, o fluxo da aplicação conduz o usuário para o formulário de inclusão de tarefas, porém como se trata de uma

edição, o formulário é exibido preenchido com os dados fornecidos pelo usuário previamente para que seja necessário somente a alteração que o usuário deseja. A Figura 7 ilustra este caso.

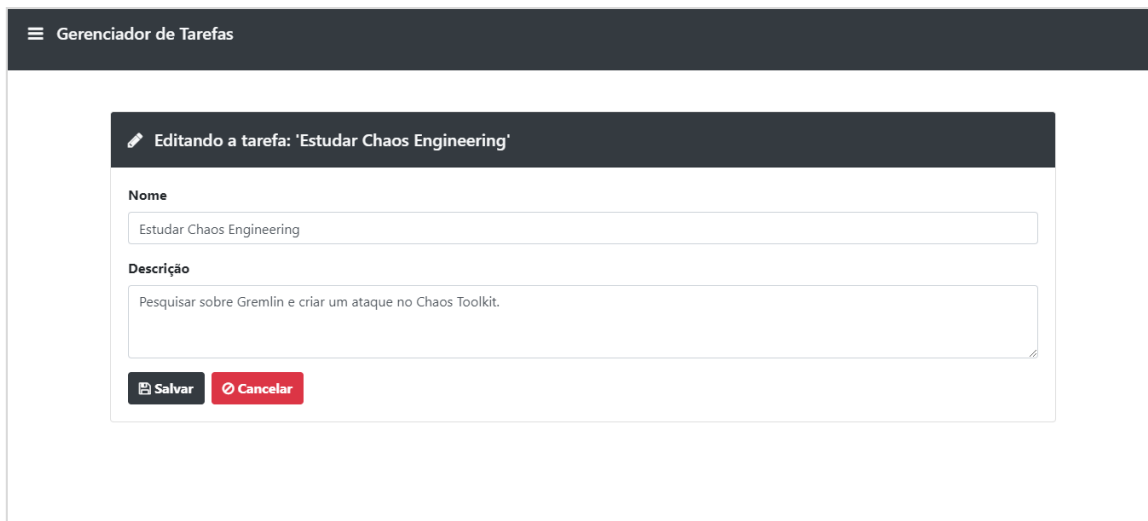



Figura 7 -  de edição de tarefas.

A Figura 7 mostra como a tela de edição de tarefas exibe uma tarefa previamente cadastrada para que seja possível a alteração de seu conteúdo. Caso o usuário clique em “Salvar”, todas as alterações são salvas novamente no *local storage* e a tarefa é exibida atualizada na lista de tarefas, mas caso o cliente clique em “Cancelar” nada é feito e a tarefa volta a ser exibida como estava previamente.

Conforme a Figura 8 é possível perceber que aplicação também disponibiliza que o usuário faça a exclusão de uma tarefa. É exibido um modal de alerta para que o usuário se certifique de que a tarefa será excluída.

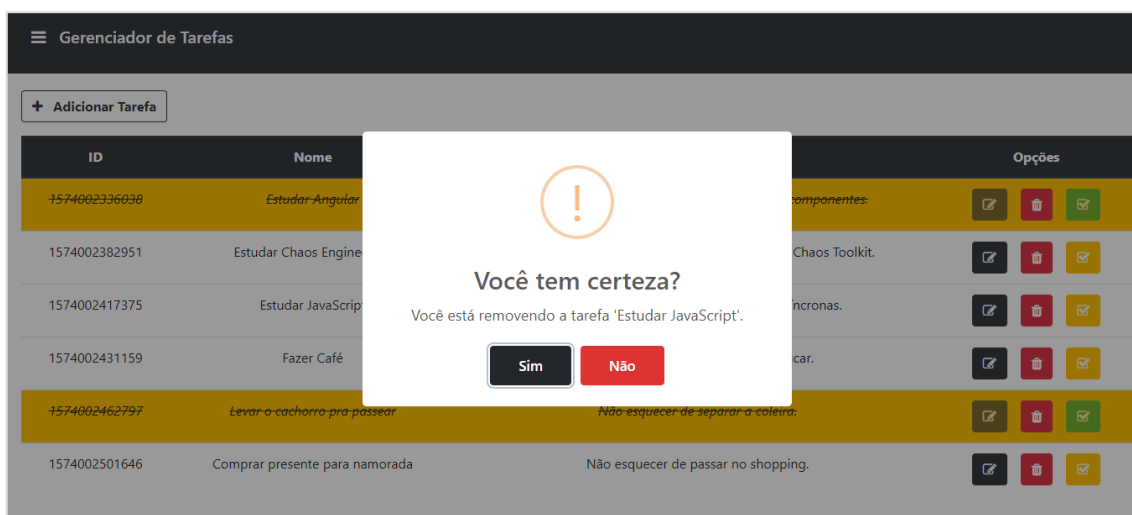


Figura 8 - Remoção de tarefas.

Caso o usuário clique em “Cancelar”, o processo de exclusão da tarefa é revertido e nada acontece, porém, clicando em “Sim” o usuário remove a tarefa da lista de tarefas e também da lista no *local storage*. A Figura 9 mostra como o modal muda seu

comportamento exibindo uma mensagem de sucesso para o usuário caso a opção “Sim” seja escolhida. Clicando em “Ok” o usuário é redirecionado para a tela de listagem de tarefas.

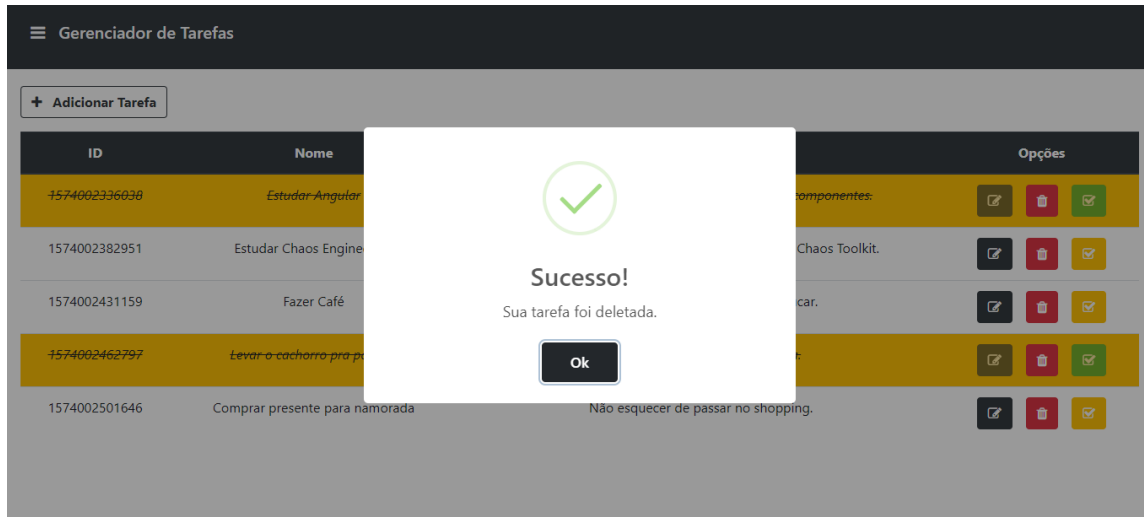


Figura 9 - Sucesso na remoção de tarefas.

3.3.1.1. Buscador de Repositórios

O Buscador de Repositórios é uma aplicação web que permite que o usuário digite o nome de um usuário qualquer do GitHub e clique no Botão “Pesquisar”. A aplicação faz uma requisição na API do GitHub para obter algumas informações úteis a respeito do usuário pesquisado. O perfil deste usuário é exibido na tela contendo seu nome, localidade e avatar (foto cadastrada no GitHub), além de possuir um link que funciona como um accordion permitindo o usuário visualizar todos os repositórios públicos deste membro do GitHub pesquisado. Desenvolvido com HTML, CSS3 com Bootstrap e JavaScript, tem a interface responsiva e de fácil interação. A Figura 10 mostra como é a tela inicial sem haver usuário algum pesquisado.

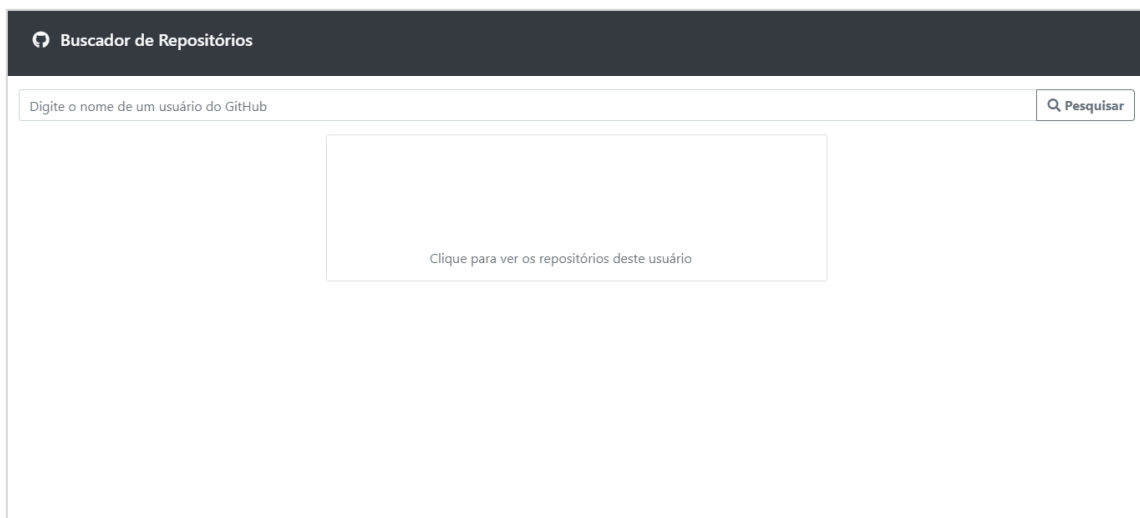


Figura 10 - Tela inicial do Buscador de Repositórios.

Abaixo do card que exibe os usuários do GitHub, existe um link chamado “Clique para ver os repositórios deste usuário”. Ao clicar neste link a tabela de repositórios do usuário é exibida. Caso o usuário não tenha digitado na barra de pesquisa, a mensagem “Primeiro digite o nome do usuário” é exibida. A Figura 11 mostra este comportamento.

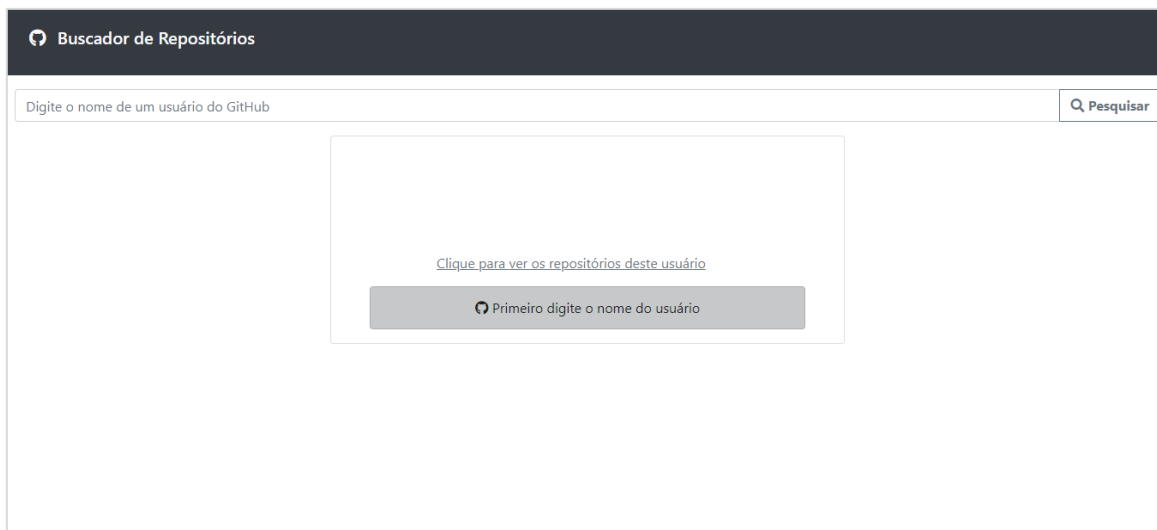


Figura 11 - Accordion sem usuário na barra de pesquisa.

Digitando “momanhaes” na barra de pesquisa é possível perceber o resultado da pesquisa. Uma requisição GET é feita à API do GitHub utilizando a biblioteca Axios passando o usuário “momanhaes” digitado na barra de pesquisa como parâmetro. Caso o usuário não exista, o erro 404 é exibido com um tratamento personalizado para que a mensagem seja mais amigável para o usuário. Se o campo para inserir o usuário não for preenchido e o botão para pesquisar for clicado, um erro é exibido na tela para que o usuário digite um usuário válido do GitHub. É possível perceber um resultado de sucesso analisando a Figura 12.

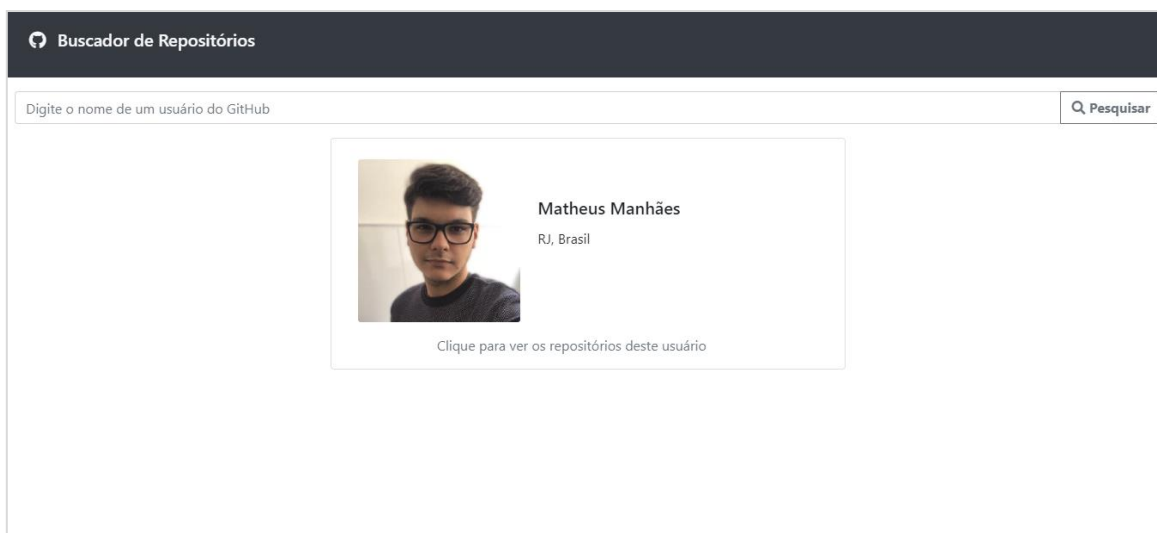



Figura 12 - Resultado de sucesso no Buscador de Repositórios.

É possível, analisando a Figura 12, que a aplicação trouxe à tela principal o avatar do usuário “momanhaes”, assim como sua localização e seu nome. Clicando no link

abaixo de sua foto é possível verificar os repositórios públicos e s deste usuário. Na Figura 13 é possível analisar este resultado.



Clique para ver os repositórios deste usuário

14 repositórios públicos e forks encontrados
adapter
awesome
awesome-chaos-engineering
calculadora-lmc
chaos-monkey-spring-boot
controle-kids
crud-produtos
curso-js-rocketseat
desafio-b2w
escola-rest
escola-rest-ic
google-glass
momanhaes.github.io
solutions-uri

Figura 13 - Accordion em resultado de sucesso no Buscador de Repositórios.

Caso seja necessário buscar por outro usuário do GitHub na aplicação, todos os dados da pesquisa anterior são descartados e os dados do novo usuário são preenchidos. É possível perceber este comportamento na Figura 14. Com os dados preenchidos anteriormente como resultado da pesquisa do usuário “momanhaes”, foi pesquisado pelo usuário “felipe-b-oliveira” e o resultado dessa nova pesquisa sobrescreveram o resultado da pesquisa anterior.



Figura 14 - Resultado de sucesso sobrescrito no Buscador de Repositórios.

3.3.1.1. API REST de Termos Úteis

A API de Termos Úteis é um servidor RESTful onde são usados os verbos HTTP para disponibilizar os recursos do servidor, sendo possível que uma aplicação *front-end* da camada de interface interaja com esse servidor a fim de consumir seus serviços. Os recursos usados nesta API são termos úteis da disciplina de Chaos Engineering. É possível inserir um novo termo, listar, editar, excluir e buscar um termo pelo seu ID. Os verbos utilizados neste servidor são: GET, POST, PUT, DELETE. Onde GET será usado exclusivamente para busca de termos. POST será utilizado para a criação de novos termos. PUT será utilizado para realizar alterações em termos já existentes. DELETE será usado para a remoção de termos.

Esta API foi desenvolvida com a linguagem de programação JavaScript, utilizando a plataforma Node.js. O servidor foi escrito usando Express, um *framework* utilizado para desenvolvimento de aplicações *web* e que facilita várias rotinas e disponibiliza alguns recursos. Os dados são persistidos no MongoDB, um software de banco de dados gratuito e NoSQL. O MongoDB é orientado a documentos, o que significa que os registros guardados no banco seguem uma estrutura flexível e adaptável. Seus documentos são semelhantes ao formato JSON. O MongoDB não foi instalado em uma máquina local, mas em um contêiner Docker, a fim de não causar uma dependência dos dados persistidos estarem em uma máquina física.

Após fazer toda a configuração para baixar o contêiner do MongoDB, definição do seu nome e redirecionamento de porta é necessário verificar quais imagens estão disponíveis através do comando `docker ps -a`, verificar quais imagens estão rodando através do comando `docker ps` e ir até o navegador e digitar `localhost:27017`. Caso a imagem do MongoDB não esteja rodando, é necessário executar o comando `docker start mongo`. Se o banco de dados estiver disponível, deverá ser exibida uma mensagem no navegador partindo do Docker, conforme ilustra a Figura 15. A mensagem diz: “It looks like you are trying to access mongoDB over HTTP on the native driver port”.

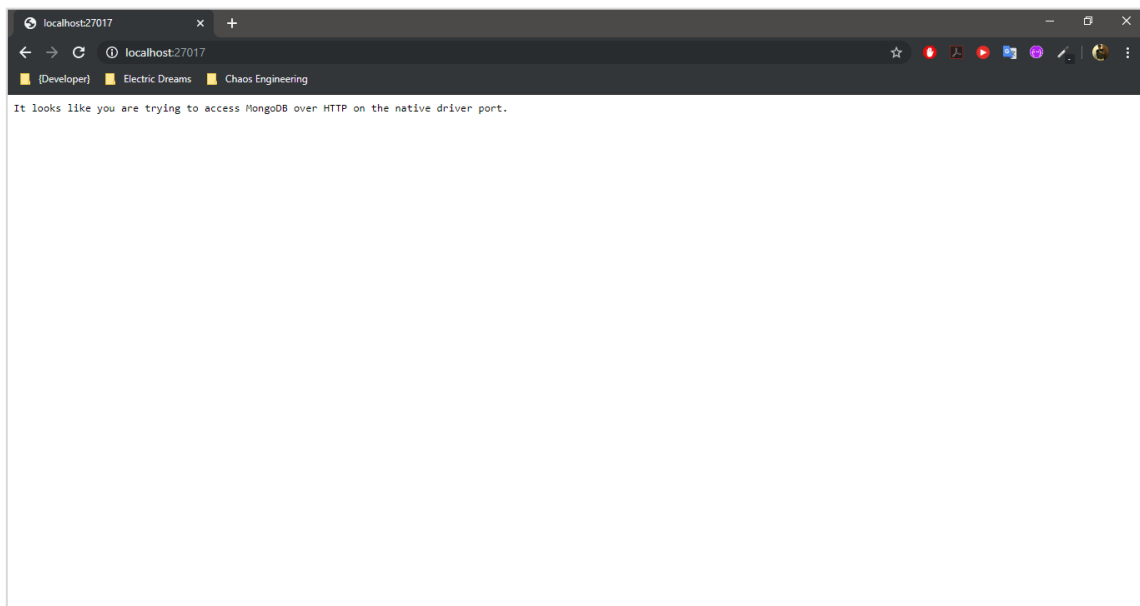


Figura 15 - Mensagem de sucesso do Docker.

Para poder interagir com os termos da API, foi definido um body, que também serve de schema para o banco de dados, para padronizar a forma como os dados vão ser inseridos e editados na API e consequentemente persistidos no banco de dados. Conforme a Figura 16 ilustra, os termos terão um nome, uma descrição e um log de registro que será definido automaticamente a cada inserção, a fim de mostrar a data e hora que cada registro foi criado.

```
name: {
  type: String,
  required: true,
},
description: {
  type: String,
  required: true,
},
createdAt: {
  type: Date,
  default: Date.now,
}
```

Figura 16 - Modelo de definição dos termos da API.

Foram criadas rotas na API para que seja possível interagir e consumir seus recursos. Todo o contexto da aplicação como rotas, esquemas e nomenclatura de arquivos tiveram o nome “attack” usado como prefixo, pois ataque é algo muito comum na disciplina de Chaos Engineering. Portanto, por questões de preferência visual, sem afetar absolutamente em nada o funcionamento da API, foi escolhido o prefixo “attack” ao invés

de “terms”, pois “attack” harmoniza-se melhor com a disciplina de Engenharia do Caos, enquanto “terms” tem uma contextualização mais genérica.

As rotas e seus respectivos verbos HTTP são as seguintes:

GET /attacks – rota utilizada para a listagem de termos.

GET /attacks/:id – rota utilizada para listar o termo que possui o ID recebido como parâmetro, ou seja, irá listar um termo específico de acordo com o ID recebido pelo usuário.

POST /attacks – rota utilizada para a criação de um novo termo.

PUT /attacks/:id – rota utilizada para alterar dados do termo que possui o ID recebido como parâmetro, ou seja, irá editar um termo específico de acordo com o ID recebido pelo usuário.

DELETE /attacks/:id – rota utilizada para excluir o termo que possuir o ID recebido como parâmetro, ou seja, irá excluir um termo específico de acordo com o ID recebido pelo usuário.

Usando o Insomnia, um software responsável por armazenar, organizar e executar requisições de APIs REST, é possível testar cada uma das rotas especificadas acima, permitindo assim, através do Insomnia, criar, listar, editar e excluir termos da disciplina de Engenharia do Caos.

Na Figura 17 é possível analisar um exemplo de criação de termo com o Insomnia, realizando uma requisição POST para criar um novo termo.

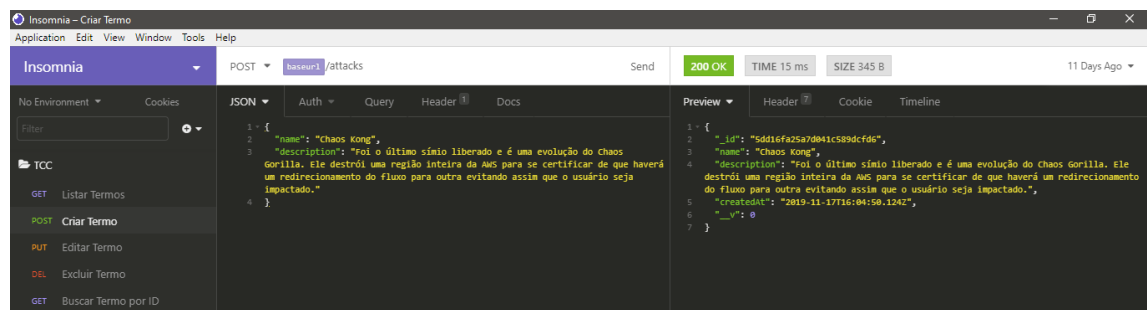


Figura 17 - Criação de termos na API com Insomnia.

Na Figura 18 é possível analisar, em outra requisição e desta vez um GET para listar os termos, o termo anteriormente criado sendo exibido na listagem, também utilizando o Insomnia.

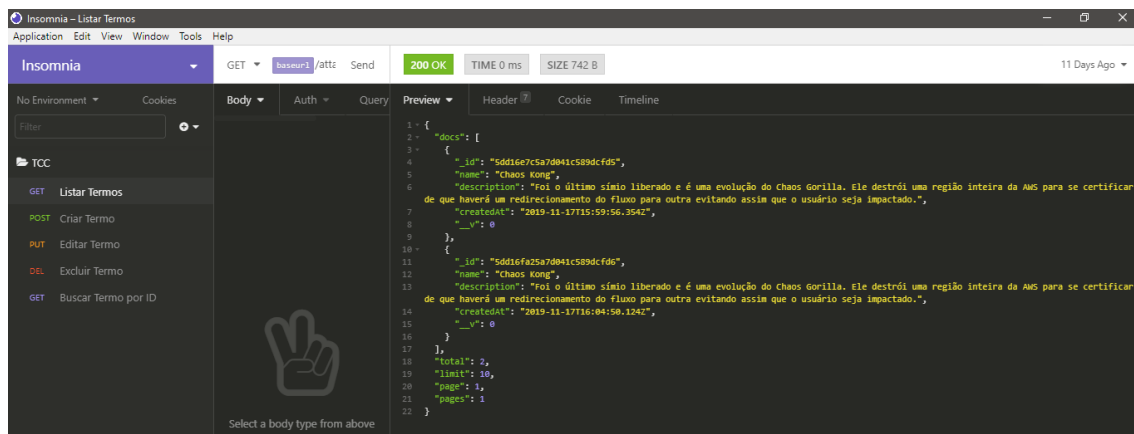


Figura 18 - Listagem de termos na API com Insomnia.

É possível perceber na imagem que, com o Insomnia, também é possível editar, excluir e buscar um termo pelo ID.

Na Figura 19 também é possível perceber o resultado da criação do termo com Insomnia fazendo uma requisição no navegador.

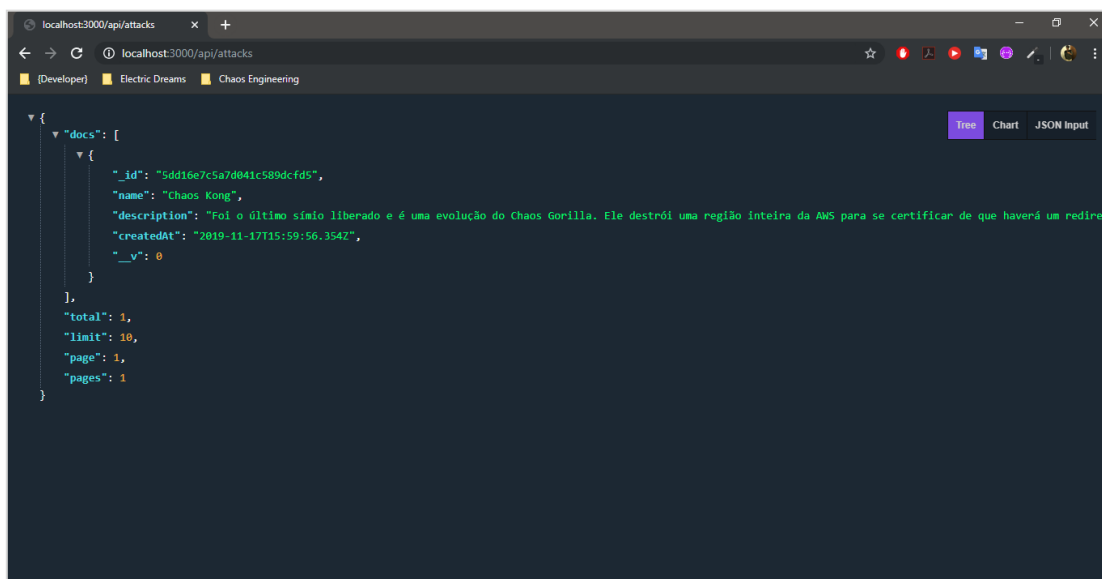


Figura 19 - Listagem de termos da API no navegador.

Também é possível perceber como é o comportamento no MongoDB utilizando o software Robo3T, uma ferramenta que reproduz graficamente os dados do MongoDB, visto que toda interação feita é consequentemente replicada no banco de dados. A Figura 20 mostra como os dados estão sendo persistidos no Mongo.

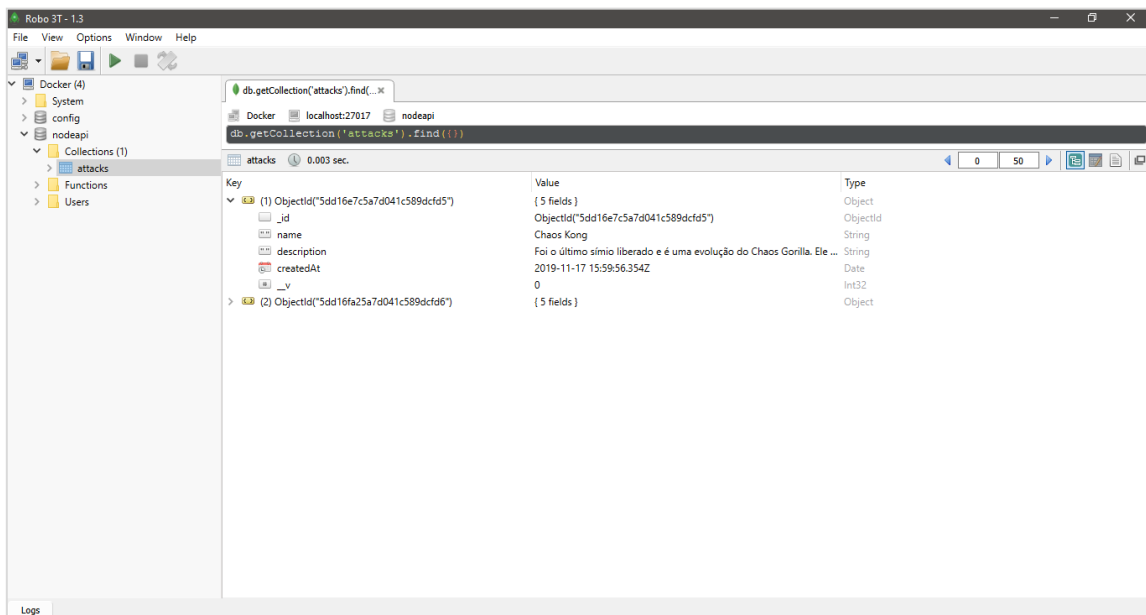


Figura 20 - Termo criado na API e persistido no MongoDB.

3.3.2. Containerização das Aplicações

A Figura 21 abaixo apresenta o painel do Docker chamado Docker Hub. Nele é possível ver todos os repositórios criados a fim de armazenar um contêiner. É possível perceber na imagem que as 4 aplicações estão listadas.

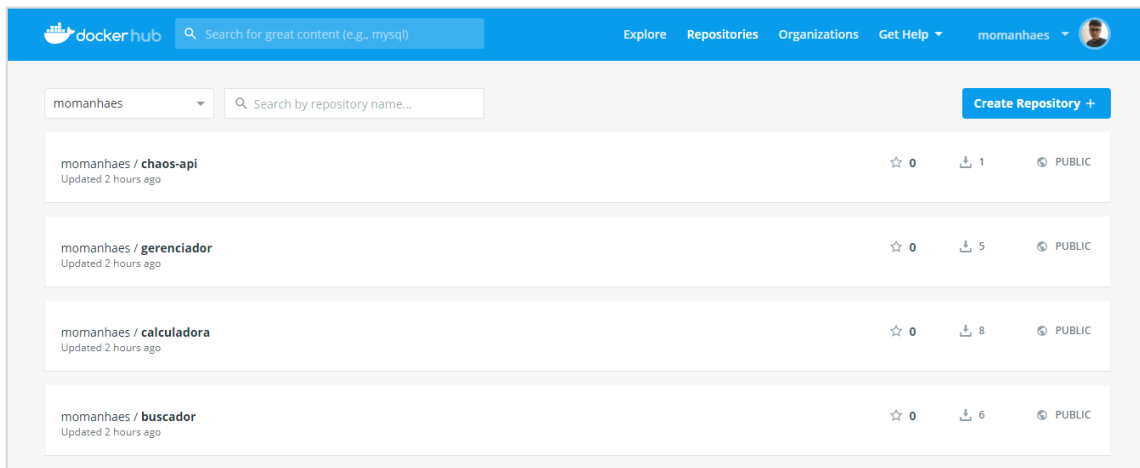


Figura 21 - Docker exibindo as aplicações do projeto.

3.3.3. Orquestração dos Contêineres

Utilizando o Minikube para gerar os pods e fazer o deploy das aplicações no Kubernetes, é necessário verificar se o serviço do Kubernetes está habilitado no painel do Docker Desktop. Com alguns pequenos comandos e passos de configuração é possível ter acesso ao dashboard do Kubernetes e verificar os status das aplicações, verificar como elas estão sendo servidas e qual porta deve-se usar para acessar cada uma delas. A Figura 22 mostra como é exibido este dashboard na tela de overview do painel do Kubernetes.

The screenshot shows the Kubernetes Overview page. On the left is a sidebar with navigation links: Cluster, Cluster Roles, Namespaces, Nodes, Persistent Volumes, Storage Classes, Namespace (set to 'default'), Overview (selected), Workloads, Cron Jobs, Daemon Sets, Deployments, and Jobs. The main content area is divided into two sections: 'Deployments' and 'Pods'.

Deployments Table:

Name	Namespace	Labels	Pods	Age	Images
calculadora	default	-	1 / 1	49 minutes	momanhaes/calculadora:latest
buscador	default	-	1 / 1	an hour	momanhaes/buscador:latest
gerenciador	default	-	1 / 1	an hour	momanhaes/gerenciador:latest

Pods Table:

Name	Namespace	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Age
calculadora-896989f8f-hbrzs	default	app: calculadora name: calculadora	docker-desktop	Running	0	-	-	49 minutes

Figura 22 - Tela de overview do Kubernetes.

3.3.4. Deploy dos Pods

Pod é o nome dado pelo Kubernetes para um grupo de um ou mais contêineres que são geridos pelo seu serviço. Como boa prática de uso do Kubernetes, é necessário que seja utilizado um pod por aplicação, ou seja, a calculadora, o gerenciador de tarefas, o buscador de repositórios e a Chaos API terão um pod separado para cada, por mais que seja possível colocar as 4 aplicações em um só pod. Todos os 4 pods foram configurados nos serviços da Amazon, o AWS. O AWS faz o deploy dos 4 pods e oferece uma máquina virtual para que os 4 pods sejam servidos e estejam disponíveis para os usuários utilizarem, sem a necessidade de separar uma máquina física para este fim. O próprio AWS se encarrega de gerar uma URL para que seja possível acessar, através dela, as aplicações que estão nos pods do Kubernetes.

3.3.5. Execução dos Testes

Esta seção irá descrever como testes de CPU e Shutdown foram configurados e executados utilizando o Gremlin.

3.3.5.1. “Alô Mundo” no Gremlin

Neste projeto o Gremlin foi a primeira ferramenta utilizada para gerar conhecimento da disciplina de Engenharia do Caos. Embora ele seja limitado em sua versão free, é possível realizar alguns experimentos que serão bem úteis para analisar o comportamento dos alvos do serviço do Gremlin e o processo de geração de falhas nas aplicações.

Antes de utilizarmos como alvo do Gremlin uma das aplicações criadas, ou até mesmo todas, foi resolvido reproduzir seus ataques em uma máquina virtual criada especialmente para este fim. Foi utilizado o software Oracle VirtualBox para criarmos uma máquina virtual utilizando o sistema operacional Linux em sua distribuição Ubuntu.

Realizando um cadastro no site do Gremlin é possível acessar o seu painel a fim de estudar a documentação da ferramenta, configurar os ataques, criar grupos de ataques e acompanhar o resultado de alguns ataques.

Utilizando um terminal e executando o comando `apt-transport-https` é possível fazer a instalação do Gremlin no sistema operacional Ubuntu, previamente instalado na máquina virtual criada pelo VirtualBox. É necessário executar também os comandos abaixo substituindo a palavra “usuario” pelo usuário criado no site do Gremlin e a palavra “ip_servidor” pelo IP do servidor que se deseja realizar o ataque:

```
ssh usuario@ip_servidor
echo "deb https://deb.gremlin.com/ release non-free" |
sudo tee /etc/apt/sources.list.d/gremlin.list.
```

Para importar a chave GPG é necessário executar o comando abaixo no terminal:

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-
keys
C81FC2F43A48B25808F9583BDF170F324D411349CDB294B29A5B1E2E00
C24C022E8EF3461A50EF6
```

É necessário a execução do comando `sudo apt-get update && sudo apt-get install -y gremlin gremlind` para instalar o *client* e o *daemon* do Gremlin.

Após ter criado uma conta no site do Gremlin, um *e-mail* de verificação será enviado com um link para configuração da mesma. Nessa configuração, será enviado um *Team ID* e uma *Secret Key* para garantir que o ambiente em que os testes estão sendo executados estão isolados. Estas duas chaves também são necessárias para adicionar novos *hosts* que serão alvos dos ataques. É importante ressaltar que cada usuário de um mesmo time possui uma *Secret Key* diferente.

Após passar por esse processo de configuração, utilizando o comando `gremlin init` é possível criar ataques usando o Gremlin.


3.3.5.1.1. CPU Attack

Fazendo o login no site Gremlin e clicando em **Create Attack** é possível ser redirecionado para o fluxo de criação de um ataque.

Na guia **Choose Targets** deve ser escolhido o **target** que foi configurado para ser atacado. É possível marcar a opção **Random** para uma escolha aleatória, caso estejam cadastrados vários **targets**.

Na guia **Choose Gremlin** é possível escolher uma de três categorias disponíveis. São elas **Resource**, **State** e **Network**. Cada categoria possui uma lista de ataques de acordo com cada perfil. Na versão **1.0.0** (versão que está sendo utilizada) estão disponíveis dois ataques: **CPU** na categoria **Resource** e **Shutdown** na categoria **State**. É necessário que seja escolhido **CPU** da categoria **Resource**.

Na opção **Launch** é possível definir quanto tempo (em segundos) a CPU será atacada. Em **Count** é possível definir a quantidade de núcleos da CPU que serão atacados. Em **CPU Capacity** é possível definir a porcentagem de consumo de CPU em cada núcleo. Será definido 120 para **Launch** e 1 para **Count**, visto que a opção **CPU Capacity** não está disponível na versão **1.0.0**.

Na guia Run  attack é possível executar o ataque ou agendá-lo. É necessário clicar em Unleash Gremlin para executar o ataque. Após ter programado o ataque no console do Gremlin no site, é possível verificar o impacto no processador causado pelo Gremlin no Linux Ubuntu usando o comando `top`.

3.3.5.2. Gremlin e Kubernetes

Após realizar o experimento “Alô Mundo” na sessão anterior disparando um ataque à CPU de uma máquina virtual Linux, foi realizado o mesmo procedimento de configuração nesta seção, só que dessa vez os alvos são os pods que estão servindo as aplicações calculadora e buscador de repositórios. Foi criado um ataque de shutdown, que tem por objetivo desligar os pods alvos e um ataque de CPU, que tem por objetivo aumentar o consumo de CPU dos pods alvos. Conforme representa a Figura 23, é possível verificar no console do computador o impacto causado pelo ataque de shutdown nos pods usando o comando `top`.

```
ubuntu@ip-172-31-31-237:~$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
buscador-5c9d6c749b-s5p28	0/1	Error	1	22h
calculadora-c66995fd8-ccpst	1/1	Running	0	52m
calculadora-c66995fd8-kn92v	0/1	Error	0	60m
calculadora-c66995fd8-zjd2t	1/1	Running	0	88m
orange-bumblebee-gremlin-n2h27	1/1	Running	0	27m
orange-bumblebee-gremlin-tkrvk	0/1	Error	0	27m
washed-cheetah-gremlin-4hx5n	0/1	Error	0	3m27s
washed-cheetah-gremlin-8n6fj	1/1	Running	0	3m27s

```
ubuntu@ip-172-31-31-237:~$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
buscador-5c9d6c749b-s5p28	0/1	Error	1	22h
calculadora-c66995fd8-ccpst	1/1	Running	0	52m
calculadora-c66995fd8-kn92v	0/1	Error	0	60m
calculadora-c66995fd8-zjd2t	1/1	Running	0	88m
orange-bumblebee-gremlin-n2h27	1/1	Running	0	27m
orange-bumblebee-gremlin-tkrvk	0/1	Error	0	27m
washed-cheetah-gremlin-4hx5n	0/1	Error	0	3m36s
washed-cheetah-gremlin-8n6fj	1/1	Running	0	3m36s

```
ubuntu@ip-172-31-31-237:~$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
buscador-5c9d6c749b-s5p28	1/1	Running	2	22h
calculadora-c66995fd8-ccpst	1/1	Running	0	53m
calculadora-c66995fd8-kn92v	1/1	Running	1	61m
calculadora-c66995fd8-zjd2t	1/1	Running	0	88m
orange-bumblebee-gremlin-n2h27	1/1	Running	0	27m
orange-bumblebee-gremlin-tkrvk	1/1	Running	1	27m
washed-cheetah-gremlin-4hx5n	1/1	Running	1	4m4s
washed-cheetah-gremlin-8n6fj	1/1	Running	0	4m4s

```
ubuntu@ip-172-31-31-237:~$
```

Figura 23 - Resultado do ataque de shutdown do Gremlin nos pods do Kubernetes.

Analisando a Figura 23 é possível verificar que os pods foram desligados algumas vezes e o Kubernetes se encarrou de reiniciá-los. Também é possível verificar no console do AWS, conforme ilustra a Figura 24, o impacto causado pelo ataque de CPU nos pods.

É possível perceber que no momento em que os testes são disparados, o gráfico exibe alguns picos no consumo de CPU, revelando que o teste criado no painel do Gremlin obteve sucesso.

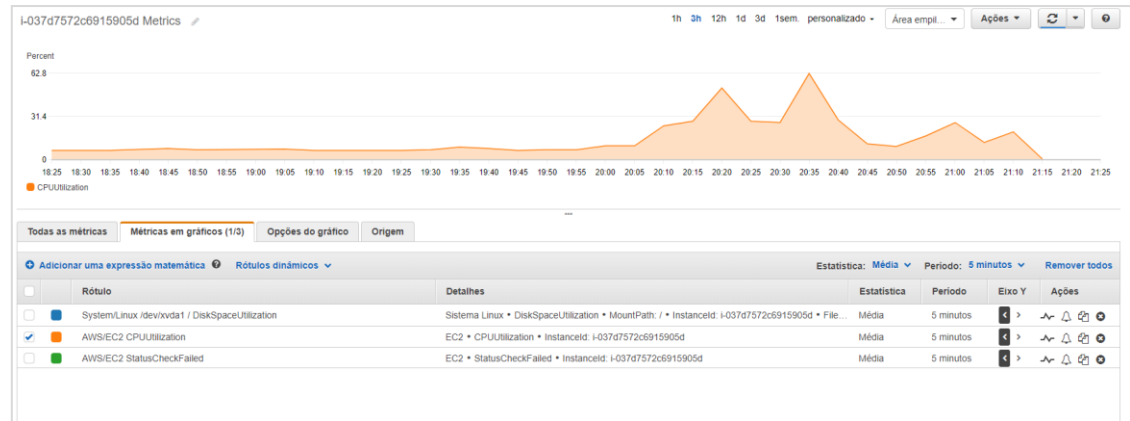


Figura 24 - Console do AWS exibindo o resultado do ataque de CPU do Gremlin.

3.3.4. Criação dos Testes

Assim como é possível executar ataques através da plataforma do Gremlin, o Chaos Toolkit também oferece uma plataforma para execução e criação de testes personalizados. Para este projeto foi criado um caso de teste bem parecido com o teste de shutdown do Gremlin. Através de um documento chamado “*chaos-test.json*”, configuram-se coisas como o nome do ataque, que neste caso é “*Pod should be automatically killed and restarted when unhealthy*”, as tags, os alvos e vários outros itens de configuração. É possível verificar na Figura 25 a estrutura deste ataque. Executando o comando *chaos run chaos-test.json* no terminal é possível verificar os resultados do ataque percebendo que a aplicação ficou indisponível na URL que a serve, assim como no console do Kubernetes também é possível perceber que a aplicação foi desligada e o próprio Kubernetes se encarregou de reiniciá-la. É importante ressaltar que para executar o teste do Chaos Toolkit é necessário que o computador possua o Docker, o Minikube, o Python e seu gerenciador de pacotes PIP.

```

{
  "version": "1.0.0",
  "title": "Pod should be automatically killed and restarted when unhealthy",
  "description": "Can we trust Kubernetes to restart our microservice when it detects it is unhealthy?",
  "tags": [
    "microservice",
    "kubernetes",
    "python"
  ],
  "steady-state-hypothesis": {
    "title": "Services are all available and healthy",
    "probes": [
      {
        "name": "all-our-microservices-should-be-healthy",
        "type": "probe",
        "tolerance": "true",
        "provider": {
          "type": "python",
          "module": "chaosk8s.probes",
          "func": "microservice_available_and_healthy",
          "arguments": {
            "name": "calculadora",
            "ns": "default"
          }
        }
      },
      {
        "type": "probe",
        "name": "application-must-responde",
        "tolerance": 200,
        "provider": {
          "type": "http",
          "verify-tls": "false",
          "url": "http://localhost:3030/"
        }
      }
    ]
  },
  "method": [
    {
      "type": "action",
      "name": "terminate-calculadora-pod",
      "provider": {
        "type": "python",
        "module": "chaosk8s.pod.actions",
        "func": "terminate_pods",
        "arguments": {
          "label_selector": "app=calculadora",
          "name_pattern": "calculadora-[0-9]$",
          "rand": true,
          "ns": "default"
        }
      },
      "pauses": {
        "after": 5
      }
    }
  ],
  "rollbacks": [
  ]
}

```

Figura 25 - Ataque criado com Chaos Toolkit.

O Chaos Toolkit possui integração com o Latex, software responsável pela criação de PDFs automáticos de acordos com os arquivos de log que o Chaos Toolkit gera no resultado dos seus testes. O Latex lê todo o arquivo e monta um PDF, assim gerando um log um pouco mais apresentável e legível para o usuário. É necessário instalar os *plugins* *textlive-latex-base*, *textlive-fonts-recommended*, *textlive-fonts-extra*, *textlive-latex-extra* e *pdflatex* para gerar o documento PDF conforme ilustra a Figura 26, Figura 27, Figura 28, Figura 29 e Figura 30.

Chaos Engineering Report	
11 October 2019	
Contents	
Summary	2
Experiment	3
Pod should be automatically killed and restarted when unhealthy . . .	3
Summary	3
Definition	3
Result	4
Appendix	5

Figura 26 - Chaos Engineering Report do Chaos Toolkit - Página 1.

Summary

This report aggregates 1 experiments spanning over the following subjects:

python, microservice, kubernetes

Figura 27 - Chaos Engineering Report do Chaos Toolkit - Página 2.

Experiment

Pod should be automatically killed and restarted when unhealthy

Can we trust Kubernetes to restart our microservice when it detects it is unhealthy?

Summary

Status	completed
Tagged	microservice, kubernetes, python
Executed From	ip-172-31-31-237
Platform	Linux-4.15.0-1051-aws-x86_64-with-Ubuntu-18.04-bionic
Started	Fri, 11 Oct 2019 20:56:39 GMT
Completed	Fri, 11 Oct 2019 20:56:45 GMT
Duration	6 seconds

Definition

The experiment was made of 1 actions, to vary conditions in your system, and 0 probes, to collect objective data from your system during the experiment.

Steady State Hypothesis

The steady state hypothesis this experiment tried was “**Services are all available and healthy**”.

Before Run

The steady state was verified

Probe	Tolerance	Verified
application-must-respond	200	True

After Run

The steady state was verified

Figura 28 - Chaos Engineering Report do Chaos Toolkit - Página 3.

Probe	Tolerance	Verified
Probe	Tolerance	Verified
application-must-respond	200	True

Method

The experiment method defines the sequence of activities that help gathering evidence towards, or against, the hypothesis.

The following activities were conducted as part of the experimental's method:

Type	Name
action	terminate-calculadora-pod

Result

The experiment was conducted on Fri, 11 Oct 2019 20:56:39 GMT and lasted roughly 6 seconds.

Action - terminate-calculadora-pod

Status	succeeded
Background	False
Started	Fri, 11 Oct 2019 20:56:39 GMT
Ended	Fri, 11 Oct 2019 20:56:39 GMT
Duration	0 seconds
Paused After	5s

The action provider that was executed:

Type	python
Module	chaos8s.pod.actions
Function	terminate_pods
Arguments	{'label_selector': 'app=calculadora', 'name_pattern': 'calculadora.*', 'rand': True, 'ns': 'default'}

Figura 29 - Chaos Engineering Report do Chaos Toolkit - Página 4.

Appendix
Action - terminate-calculadora-pod
The <i>action</i> returned the following result:
None

Figura 30 - Chaos Engineering Report do Chaos Toolkit - Página 5.

3.4. Tecnologias Utilizadas

Nesta seção serão descritas as tecnologias Chaos Toolkit, Gremlin, 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 Chaos possa ser executado consistentemente usando integrações com outras ferramentas que estão surgindo. A Figura 31 mostra como esse processo é simples.

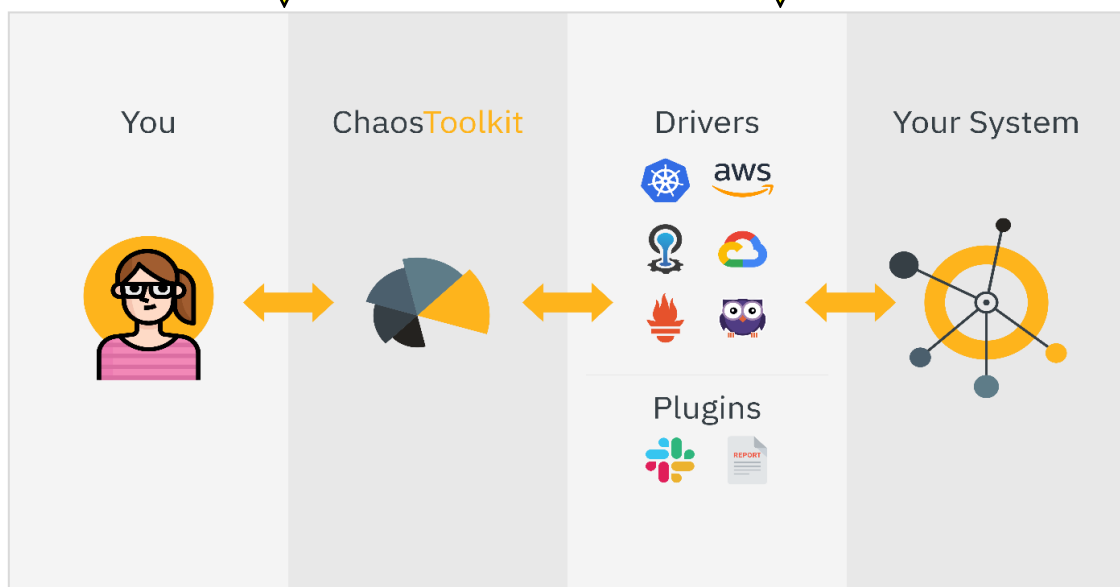


Figura 31 - Arquitetura do Chaos Toolkit

Fonte: Documento do Chaos Toolkit.

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. 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 é ideal para que possa “subir” as quatro aplicações em contêineres, visto que também seria fácil a integração com o Kubernetes e o AWS, que serão tratados nos próximos tópicos.

3.4.4. 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.

3.4.5. 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

Aqui serão descritas as considerações finais.

4.2. Contribuições

Aqui serão descritas as contribuições.

4.3. Trabalhos Futuros

Aqui serão descritos os trabalhos futuros.

7. Referência bibliográficas

ROSENTHAL, Casey. **Chaos Engineering: Building Confidence in System Behavior through Experiments**. O'Reilly, 2017. Disponível em: <<https://www.oreilly.com/ideas/chaos-engineering>>. Acesso em: 11 mar. 2019.

Links ssados

Netflix. Sobre a Netflix. Site. Disponível em: <https://media.netflix.com/pt_br/about-netflix>. Acesso em: 5 de outubro de 2019.

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

GitHub Chaos Toolkit. Chaos Toolkit 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.