Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística

Samuel Vieira Bernardo

Aprimorando a Qualidade de Software com Agilidade: Desenvolvimento de uma UI Intuitiva para Testes de Performance com Gatling

Florianópolis, Brasil

2024

Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística

Samuel Vieira Bernardo

Aprimorando a Qualidade de Software com Agilidade: Desenvolvimento de uma UI Intuitiva para Testes de Performance com Gatling

Trabalho de Conclusão do Curso de Graduação em Ciências da Computação da Universidade Federal de Santa Catarina, requisito parcial à obtenção do título de Bacharel em Ciências da Computação.

Orientador: Raul Sidnei Wazlawick

Florianópolis, Brasil
2024

Resumo

Este trabalho tem como objetivo desenvolver uma solução intuitiva e acessível para a criação e execução de testes de performance utilizando o Gatling, uma ferramenta amplamente reconhecida para testes de carga em sistemas. A solução proposta consiste em uma interface gráfica que permite aos profissionais, mesmo sem conhecimentos avançados de programação, configurar e realizar testes de forma eficiente, eliminando a necessidade de codificação manual. Por meio de recursos como a geração automática de scripts e a visualização interativa dos resultados, busca-se otimizar o processo de testes de carga, aumentando a produtividade e reduzindo a ocorrência de erros. A interface será avaliada qualitativamente em uma equipe de teste, com o objetivo de verificar sua eficácia em termos de usabilidade, tempo de configuração dos testes e impacto na qualidade geral dos sistemas testados. Espera-se, com essa abordagem, facilitar o uso do Gatling, democratizando o acesso a testes de performance e contribuindo para a melhoria contínua da qualidade de software.

Palavras-chave: Testes de performance. APIs REST. Interface gráfica. Gatling. Ferramentas gráficas em testes de performance

Abstract

This study aims to develop an intuitive and accessible solution for creating and executing performance tests using Gatling, a widely recognized tool for load testing in systems. The proposed solution consists of a graphical interface that enables professionals, even those without advanced programming skills, to configure and run tests efficiently, eliminating the need for manual coding. Through features such as automatic script generation and interactive result visualization, the solution seeks to optimize the load testing process, enhancing productivity and reducing the occurrence of errors. The interface will undergo qualitative evaluation within a test team to assess its effectiveness in terms of usability, test configuration time, and impact on overall system quality. This approach aims to facilitate the use of Gatling, democratizing access to performance testing and contributing to the continuous improvement of software quality.

Keywords:

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	Application Programming Interface
GUI	Graphical User Interface
UI	User Interface
REST	Representational State Transfer
HTTP	HyperText Transfer Protocol

Sumário

1 Introdução

1.1 Contexto

Na era dos sistemas conectados, a capacidade de um software de suportar altos volumes de tráfego e manter a estabilidade sob diferentes cargas de trabalho é essencial para garantir sua qualidade e confiabilidade. Testes de carga e performance são fundamentais para avaliar como um sistema se comporta sob condições de uso intensivo, identificando possíveis gargalos e problemas de desempenho que poderiam comprometer a experiência do usuário final. Para esse propósito, ferramentas como o Gatling se destacam devido à sua eficiência e escalabilidade, especialmente no contexto de testes de performance para APIs REST (BANIAș et al., 2021).

O Gatling é amplamente reconhecido por sua capacidade de simular grandes volumes de usuários simultâneos, fornecendo uma visão detalhada do desempenho do sistema sob carga (CAZZOLA; CESARINI; TANSINI, 2022). Sua estrutura escalável e facilidade de integração com outros sistemas tornam-no uma escolha ideal para empresas que buscam soluções robustas para testar a resiliência de suas aplicações. Comparado com outras ferramentas de mercado, como JMeter, K6 e Locust, o Gatling apresenta vantagens significativas em termos de eficiência e facilidade de configuração. O JMeter, apesar de oferecer uma interface gráfica e uma ampla gama de funcionalidades, pode ter dificuldades em cenários de alta concorrência em APIs REST, onde o Gatling mostra-se mais eficaz. Por outro lado, o K6 é uma ferramenta moderna e poderosa para testes de carga, mas sua interface limitada à linha de comando reduz a acessibilidade para usuários que preferem um ambiente gráfico. Já o Locust, embora muito robusto e flexível, apresenta uma curva de aprendizado mais acentuada devido à dependência de Python para configuração de cenários, o que também pode dificultar sua adoção por equipes multidisciplinares.

1.2 Problema

Apesar de suas vantagens, o Gatling possui uma limitação importante: a ausência de uma interface gráfica intuitiva, o que restringe seu uso a profissionais com conhecimentos avançados em programação. Atualmente, o uso do Gatling exige a criação de scripts em Scala, o que representa uma barreira significativa para usuários sem habilidades técnicas avançadas. Esse obstáculo torna o Gatling menos acessível para equipes multidisciplinares que exigem uma abordagem colaborativa e ágil para os testes de performance. Nesse sentido, o desenvolvimento de uma interface gráfica para o Gatling é essencial, pois poderia democratizar o acesso à ferramenta e aprimorar a experiência do usuário, permitindo que

profissionais de diferentes perfis configurem e executem testes de carga sem a necessidade de codificação (ZHAO, 2024).

1.3 Justificativa e Motivação

A criação de uma interface gráfica intuitiva para o Gatling pode transformar a forma como os testes de carga são realizados, facilitando a configuração de testes, a visualização de resultados e o gerenciamento de cenários de teste. (CAZZOLA; CESARINI; TANSINI, 2022) destacam a importância de interfaces amigáveis em ferramentas de performance, pois ajudam os usuários a identificar problemas de desempenho de forma mais sistemática e compreensível. Além disso, uma interface gráfica para o Gatling poderia incorporar métodos automatizados de configuração de testes a partir de especificações de API, como as fornecidas pelo OpenAPI, simplificando ainda mais o processo e permitindo que usuários de diferentes níveis técnicos utilizem a ferramenta (BANIAș et al., 2021).

A relevância de uma interface gráfica consistente também é enfatizada por (NORDBY; MALLAM; LüTZHöFT, 2019), que discutem a importância da consistência no design de interfaces para reduzir erros humanos e melhorar a eficiência. Essa consistência é crucial para o desenvolvimento da interface do Gatling, garantindo que os usuários possam navegar e acessar funcionalidades de forma intuitiva. (HOSSAIN et al., 2021) também apontam que a usabilidade é um fator essencial para o sucesso de ferramentas de análise de performance, indicando que uma interface bem projetada pode aumentar o engajamento e a satisfação dos usuários, inclusive para aqueles que acham desafiadora a interface de linha de comando atual do Gatling.

Portanto, o desenvolvimento de uma interface gráfica intuitiva para o Gatling é uma solução necessária para tornar a ferramenta mais acessível e eficiente. Ao integrar metodologias automatizadas e uma experiência de usuário aprimorada, a interface proposta pode democratizar o uso do Gatling, permitindo que equipes de diferentes perfis técnicos adotem a ferramenta com facilidade. Essa democratização do uso dos testes de performance contribui para a melhoria da qualidade e confiabilidade do software em um cenário onde sistemas robustos e escaláveis são cada vez mais essenciais.

1.4 Objetivos

1.4.1 Objetivo geral

Desenvolver uma interface gráfica intuitiva para o Gatling, destinada à criação e execução de testes de carga sem a necessidade de conhecimentos avançados em programação. A interface permitirá que os usuários configurem cenários de teste, definam métricas de análise e visualizem os resultados de maneira simplificada e interativa.

1.4.2 Objetivos específicos

- **Facilitar a Configuração de Testes**: Desenvolver uma interface que simplifique o processo de configuração de cenários de teste, permitindo que usuários com diferentes níveis de experiência técnica criem testes de carga de maneira intuitiva.

- **Automatizar a Geração de Scripts**: Implementar uma funcionalidade de geração automática de scripts de teste, eliminando a necessidade de codificação manual e permitindo que os testes sejam executados com base nas interações do usuário.

- **Visualizar Resultados de Forma Intuitiva**: Incorporar elementos visuais que possibilitem a análise dos resultados de desempenho, facilitando a identificação de gargalos e a interpretação das métricas de performance.

- **Ampliar a Acessibilidade da Ferramenta**: Democratizar o uso do Gatling, tornando-o acessível para profissionais sem conhecimentos em programação e para equipes multidisciplinares que precisam realizar testes de performance de forma ágil.

1.5 Metodologia

Para alcançar os resultados desejados neste trabalho, será adotada uma abordagem que combina metodologias de pesquisa e desenvolvimento, estruturadas em etapas alinhadas aos objetivos específicos do projeto. Essas etapas serão conduzidas da seguinte maneira:

Etapa 1 – Fundamentação Teórica: Nesta etapa, será realizada uma análise da literatura acadêmica e técnica para fundamentar os conceitos e métodos utilizados no desenvolvimento da solução. O objetivo é compreender os desafios associados à usabilidade em ferramentas de teste de carga, bem como as melhores práticas em design de interfaces gráficas e automação de testes.

Atividade 1.1: Sintetizar conceitos relacionados a testes de carga e performance, com foco nas especificidades de APIs REST.

Atividade 1.2: Analisar as características técnicas e vantagens do Gatling em comparação com ferramentas como JMeter, K6 e Locust.

Atividade 1.3: Estudar princípios de usabilidade e design centrado no usuário, utilizando referências como Nielsen (1994) e Norman (2013).

Atividade 1.4: Revisar metodologias de desenvolvimento ágil para aplicação no desenvolvimento da interface gráfica.

Etapa 2 – Levantamento do Estado da Arte: Essa etapa consiste em identificar e analisar soluções existentes que tratem de interfaces gráficas para ferramentas de automação de testes ou de problemas semelhantes ao proposto neste trabalho. O objetivo é compreender as lacunas e desafios que o projeto pretende abordar.

Atividade 2.1: Definir o protocolo de busca para levantamento de ferramentas e soluções existentes.

Atividade 2.2: Executar a busca de trabalhos correlatos, considerando artigos acadêmicos, relatórios técnicos e documentações de ferramentas.

Atividade 2.3: Analisar as informações coletadas, focando nas limitações e pontos fortes das interfaces existentes em ferramentas de teste de carga.

- Identificar as técnicas de visualização de resultados utilizadas.
- Mapear as funcionalidades automatizadas disponíveis (e ausentes) em outras ferramentas.
- Verificar como outras ferramentas lidam com a curva de aprendizado e acessibilidade.

Etapa 3 – Desenvolvimento da Solução: Nesta etapa será realizada a modelagem, implementação e validação da interface gráfica proposta, incluindo a integração com o Gatling e a implementação de funcionalidades como geração automática de scripts e visualização de resultados.

Atividade 3.1: Modelar a proposta de solução.

- Definir os requisitos funcionais e não funcionais da interface gráfica, com base na fundamentação teórica e no estado da arte.
- Desenvolver wireframes e protótipos de baixa fidelidade para validar as funcionalidades com potenciais usuários.
- Modelar o fluxo de interações da interface, considerando a usabilidade e a experiência do usuário.

Atividade 3.2: Desenvolver a proposta de solução.

- Implementar a interface gráfica utilizando tecnologias adequadas (por exemplo, JavaScript, frameworks web, ou outros).
- Integrar a interface com o Gatling, garantindo que os scripts gerados sejam compatíveis com o formato padrão da ferramenta.

- Implementar funcionalidades de automação, como a geração de scripts de teste baseados em ações do usuário e a configuração de cenários de teste.

Atividade 3.3: Validar a proposta de solução.

- Realizar testes de usabilidade para avaliar a intuitividade e eficiência da interface.
- Conduzir testes de integração para garantir que a interface funcione corretamente com o Gatling.
- Coletar feedback de potenciais usuários para ajustes finais na interface.

Etapa 4 – Avaliação da Solução: Essa etapa foca na análise dos impactos da interface gráfica desenvolvida. Será realizada uma avaliação qualitativa e quantitativa com uma equipe de testes para validar a eficácia da solução em termos de usabilidade e produtividade.

Atividade 4.1: Identificar métricas para avaliação, como:

- Tempo médio para criação de um teste.
- Taxa de erros durante a configuração e execução de testes.
- Nível de satisfação dos usuários com a interface.

Atividade 4.2: Coletar e analisar os dados obtidos durante a avaliação.

Atividade 4.3: Elaborar um relatório com os resultados, discutindo as contribuições da interface desenvolvida e identificando áreas de melhoria.

2 Fundamentação Teórica

Neste capítulo, são apresentados os conceitos teóricos necessários para contextualizar e embasar o desenvolvimento da interface gráfica para o Gatling, ferramenta amplamente utilizada em testes de performance e carga. São abordados de forma concisa os fundamentos de testes de performance, com ênfase no uso de APIs REST, bem como uma análise comparativa das principais ferramentas disponíveis no mercado.

Além disso, são discutidos princípios de desenvolvimento de interfaces gráficas com foco em usabilidade e experiência do usuário, assim como os benefícios da automação na configuração e execução de testes. Por fim, o capítulo explora as metodologias ágeis aplicadas ao desenvolvimento da solução proposta, justificando sua escolha como estratégia para garantir maior flexibilidade, colaboração e entrega contínua ao longo do projeto.

2.1 Testes de Carga e Performance

2.2 APIs REST e Sua Importância nos Sistemas Modernos

2.3 Gatling como Ferramenta de Teste de Performance

2.4 Desenvolvimento de Interfaces Gráficas

2.5 Automação de Testes e Geração de Scripts

2.6 Metodologias Ágeis no Desenvolvimento de Software

3 Trabalhos Correlatos

4 Projeto

A interface gráfica proposta para o Gatling busca superar as limitações atualmente enfrentadas por profissionais que não possuem conhecimentos avançados em programação. Como discutido nos capítulos anteriores, o uso de ferramentas de teste de carga é indispensável para garantir a qualidade de software em sistemas modernos, mas muitas dessas ferramentas apresentam barreiras de acessibilidade, limitando sua adoção em equipes multidisciplinares.

Atualmente, o Gatling, apesar de sua robustez e eficiência em cenários de testes de APIs REST, exige que os usuários criem scripts em Scala, uma linguagem que apresenta uma curva de aprendizado significativa para não programadores. Além disso, a falta de uma interface gráfica que simplifique a configuração e execução de testes dificulta sua utilização por equipes compostas por profissionais de diferentes áreas. Essa limitação reduz o potencial do Gatling como uma solução abrangente e acessível para realização de testes de carga.

Dessa forma, este projeto propõe o desenvolvimento de uma interface gráfica que combina usabilidade, automação e visualização de dados. A proposta visa democratizar o uso do Gatling, permitindo que usuários com diferentes níveis de conhecimento técnico configurem cenários de teste, gerem scripts automaticamente e analisem resultados de maneira clara e interativa.

As funcionalidades previstas para a interface estão detalhadas na Tabela 1, que apresenta os requisitos funcionais essenciais e desejáveis para a solução. Entre os principais requisitos estão a configuração visual de cenários de teste (RF-01), a geração automática de scripts compatíveis com o Gatling (RF-02) e a visualização gráfica de métricas de desempenho (RF-03).

Além disso, o desenvolvimento da interface considera os diferentes perfis de usuários-alvo, conforme descrito na Tabela 2. Esses perfis incluem profissionais de QA, que precisam de uma ferramenta acessível para realizar testes de performance sem conhecimentos de programação; desenvolvedores, que buscam agilidade na configuração de cenários complexos; e gerentes de produto, que necessitam de relatórios claros para tomar decisões estratégicas. A interface também busca atender equipes multidisciplinares, promovendo maior colaboração e integração no contexto de testes de performance.

Com base nas funcionalidades e nos perfis identificados, espera-se que a solução proposta atenda às demandas dos usuários, eliminando barreiras técnicas e otimizando os processos de teste de carga.

Tabela 1 – Requisitos funcionais

Identificador	**Requisito**	**Classificação**
RF-01	A interface deve permitir a configuração visual de cenários de teste de carga.	Essencial
RF-02	A interface deve gerar automaticamente scripts compatíveis com o Gatling.	Essencial
RF-03	A interface deve oferecer visualização gráfica de métricas de desempenho, como tempos de resposta e throughput.	Essencial
RF-04	A interface deve permitir o gerenciamento de múltiplos cenários de teste.	Essencial
RF-05	A interface deve ser compatível com sistemas operacionais populares.	Essencial
RF-06	A interface deve suportar a personalização de testes por meio de parâmetros configuráveis.	Essencial
RF-07	A interface deve permitir a exportação de relatórios em formatos como PDF ou CSV.	Desejável

Tabela 2 – Perfil de Usuários

Usuário	**Uso**
Profissionais de QA	Configurar e executar testes de performance sem necessidade de conhecimento avançado em programação.
Desenvolvedores	Avaliar o desempenho de APIs REST e simular cenários de alta carga de forma rápida e eficiente.
Gerentes de Produto	Interpretar resultados de performance em relatórios claros para a tomada de decisões estratégicas.
Equipes Multidisciplinares	Colaborar na criação e execução de testes sem barreiras técnicas, permitindo maior integração entre áreas.

5 Cronograma

	2024					2025						
Etapas	**Ago**	**Set**	**Out**	**Nov**	**Dez**	**Jan**	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**
Revisão do estado da arte & prática	x											
Proposta de solução	x											
Levantamento de requisitos	x	x										
Design de Interface		x										
Desenvolvimento da UI e desenvolvimento do relatório		x	x	x								
Entrega do relatório de Projeto I					fim							
Desenvolvimento da UI e testes						x						
Documentação e coleta de dados						x	x					
Aplicação da UI dentro de uma equipe de qualidade de software							x	x				
Coleta e análise de dados								x	x			
Relatório de análise								x	x	x		
Entrega do rascunho do TCC											metade	
Preparação da defesa pública											x	
Defesa pública												inicio
Ajustes no relatório final do TCC												x

Referências

BANIAș, O. et al. Automated specification-based testing of rest apis. *Sensors*, v. 21, p. 5375, 2021. Citado 2 vezes nas páginas 13 e 14.

CAZZOLA, W.; CESARINI, F.; TANSINI, L. Performerl: a performance testing framework for erlang. *Distributed Computing*, v. 35, p. 439–454, 2022. Citado 2 vezes nas páginas 13 e 14.

HOSSAIN, M. et al. Web performance analysis: an empirical analysis of e-commerce sites in bangladesh. *International Journal of Information Engineering and Electronic Business*, v. 13, p. 47–54, 2021. Citado na página 14.

NORDBY, K.; MALLAM, S.; LüTZHöFT, M. Open user interface architecture for digital multivendor ship bridge systems. *Wmu Journal of Maritime Affairs*, v. 18, p. 297–318, 2019. Citado na página 14.

ZHAO, S. A new web performance testing framework for industrial internet platform. v. 2, p. 1–4, 2024. Citado na página 14.