Capítulo 5

Testes de Desempenho de Software: Teoria e Prática

Thiago Silva de Souza

Abstract

Performance is one of the most important quality features in a software product. Evaluating software performance is not a trivial task because it requires not only practical knowledge of a test automation tool but also understanding of related theoretical aspects. However, the terminology regarding performance testing is quite inconsistent. In addition, the technical literature provides little information as to how performance requirements should be elicited and documented and how performance tests should be planned, implemented, and executed. This mini-course aims to present a performance testing classification as well as provide hands-on experience ranging from eliciting the performance requirement to testing using the Apache JMeter tool.

Resumo

O desempenho é uma das características de qualidade mais importantes em um produto de software. Avaliar o desempenho de um software não é uma tarefa trivial, pois requer não apenas conhecimento prático de uma ferramenta de automação de testes mas, também, compreensão dos aspectos teóricos relacionados. No entanto, a terminologia a respeito de testes de desempenho é bastante inconsistente. Além disso, a literatura técnica traz pouca informação a respeito de como requisitos de desempenho devem ser elicitados e documentados e como testes de desempenho devem ser planejados, implementados e executados. Este minicurso apresenta uma classificação sobre testes de desempenho e proporciona uma experiência prática cobrindo desde a elicitação do requisito de desempenho até o seu teste, utilizando a ferramenta Apache JMeter.

5.1. Introdução

O teste de software é um processo relacionado ao desenvolvimento de software que tem como principal objetivo revelar falhas por meio da análise dinâmica de programas [ISO/IEC/IEEE 2013]. Trata-se de uma análise dinâmica porque há a necessidade de

colocar o software em execução para avaliar se determinados dados de entrada geram determinados dados de saída (resultados esperados), sob determinadas condições de contexto.

Testes de software podem ser classificados em função de diversas dimensões (nível, tipo e técnica de teste, entre outras). Em relação ao seu tipo, os testes podem ser classificados em **testes funcionais** ou **testes não-funcionais**. A definição do tipo de teste depende do tipo de requisito que se deseja avaliar. **Requisitos funcionais** demandam testes funcionais, enquanto que **requisitos não-funcionais** (RNF) exigem a realização de testes não-funcionais.

A classificação em requisitos funcionais e não-funcionais, aparentemente, não é a ideal. Trata-se de uma classificação onde um dos conceitos é dado pela negação do outro. Em outras palavras, todos os requisitos que não são funcionais são classificados como não-funcionais. Assim, temos requisitos totalmente diferentes sob a mesma classificação. Esse tipo de classificação contribui para uma grande confusão conceitual, provocando distorções que se espalham por todo o processo de desenvolvimento do software, inclusive o teste.

Uma dessas distorções diz respeito à estimativa de esforço e custo. Muitas empresas de software têm cobrado pela implementação de RNF de forma genérica, precificando requisitos de desempenho da mesma forma que precificam requisitos de usabilidade ou de segurança, considerando que todos são "requisitos não-funcionais". No entanto, cada tipo de RNF exige esforços e custos específicos, o que justificaria preços diferenciados.

Portanto, há diversos tipos de RNF, classificados das mais variadas formas. Atualmente, a classificação de requisitos de software mais aceita pela indústria é aquela representada no modelo de qualidade de produtos de software da norma ISO 25010 [ISO/IEC 2011], representado na Figura 5.1. Esse modelo é organizado em oito características e 31 subcaracterísticas de qualidade. A maior parte dessas características (sete) e subcaracterísticas (28) se refere a requisitos não-funcionais, dentre as quais se encontra a característica de "Eficiência no desempenho" (ou apenas "Desempenho"), que é o foco deste trabalho.

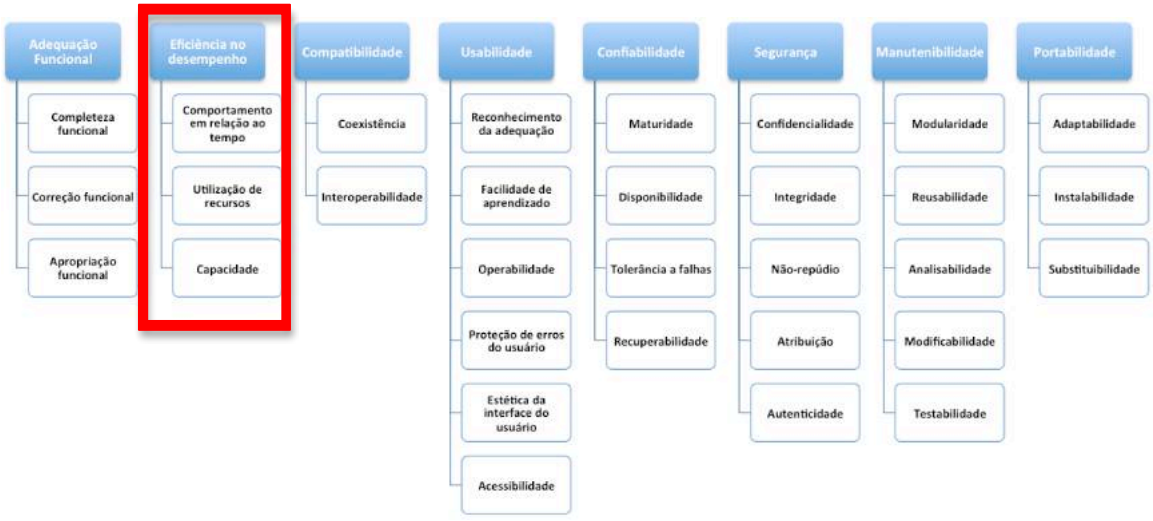


Figura 5.1: Características de qualidade de produtos de software presentes na norma ISO 25010 [ISO/IEC 2011].

O restante deste trabalho está organizado da seguinte maneira: a seção 5.2 aborda aspectos relacionados à elicitação e documentação de requisitos de desempenho; a seção 5.3 descreve os principais conceitos relacionados, com ênfase na classificação de diferentes tipos de testes de desempenho; a seção 5.4 introduz a ferramenta de testes de desempenho Apache JMeter; a seção 5.5 demonstra aspectos do JMeter por meio de um exemplo prático; a seção 5.6 relaciona uma série de recomendações práticas para realização de testes de desempenho; por fim, a seção 5.7 apresenta as conclusões deste trabalho.

5.2. Requisitos de Desempenho de Software

O processo de Engenharia de Requisitos cumpre papel fundamental para o processo de Teste de Software. É nesse processo que os requisitos do sistema são elicitados (identificados), analisados, documentados e validados. A qualidade dos testes é fortemente influenciada pela qualidade dos requisitos. Quando se trata de requisitos de desempenho, é necessário realizar as atividades desse processo com o viés que esse tipo de requisito demanda. Portanto, as subseções 5.2.1 e 5.2.2 apresentam, respectivamente, orientações a respeito das atividades de elicitação e documentação de requisitos de desempenho baseadas na experiência prática do autor.

5.2.1. Elicitação de Requisitos de Desempenho

A literatura de Engenharia de Software descreve diversas técnicas para elicitação de requisitos, tais como entrevistas, questionários, etnografia, etc. A princípio, essas técnicas servem para elicitar qualquer tipo de requisito. Apesar disso, analistas de requisitos costumam utilizá-las com foco em requisitos funcionais. A utilização das técnicas de elicitação de requisitos para identificar RNF exige adaptação de acordo com o contexto apresentado. Cada tipo de RNF pode ter fontes de informação diferentes. Por exemplo, a principal fonte de informação sobre requisitos funcionais é o próprio cliente e os potenciais usuários do sistema. Já os RNF podem ter fontes variadas, podendo ser, inclusive, necessário elicitar requisitos em conjunto com cliente e pessoal técnico. A Figura 5.2 representa essa variedade de fontes em função do tipo de requisito.

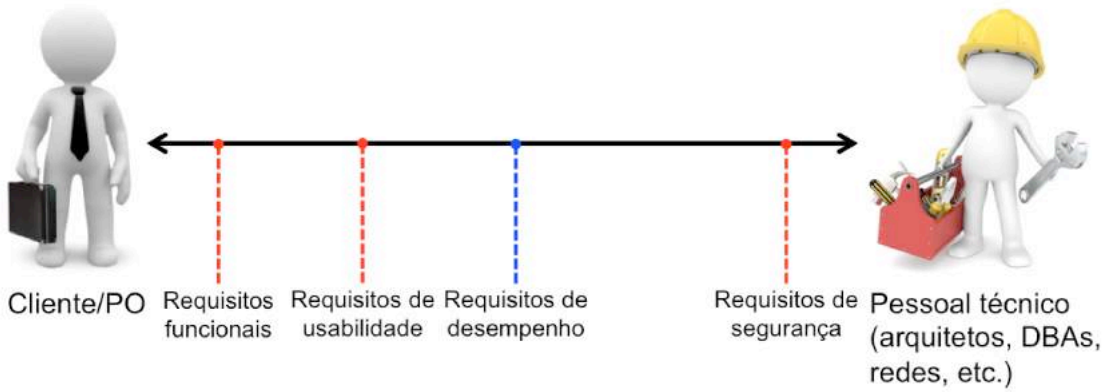


Figura 5.2: Fontes de diferentes tipos de requisitos de software.

Vale destacar que a Figura 5.2 representa um espectro e o fato de um tipo de requisito estar mais próximo de uma das extremidades não significa que a outra não possa contribuir com informações sobre determinado tipo de requisito. Outro ponto importante é que a responsabilidade por determinar todos os requisitos de um projeto é

exclusiva do cliente. Ser uma fonte de informações sobre um determinado tipo de requisito não significa ser o responsável por determinar quais serão os requisitos daquele tipo. Cabe, no entanto, ao pessoal técnico (tanto do lado da equipe de desenvolvimento quanto do lado do cliente, quando pertinente) fornecer informações que apoiem a decisão do cliente sobre cada tipo de requisito. Por exemplo, um arquiteto pode sinalizar que um determinado requisito funcional é inviável de se alcançar, devido às restrições tecnológicas da solução. Assim, o requisito funcional em questão deve ser renegociado com o cliente.

Requisitos de desempenho, em especial, são difíceis de elicitar porque nem sempre o cliente sabe responder a questionamentos como "Qual a expectativa de acessos concorrentes a esta funcionalidade?" ou "Qual o tempo de resposta tolerado para esta consulta?". Em contrapartida, para um membro da equipe técnica do projeto, pode ser um desafio propenso a erros tentar inferir requisitos desse tipo sem a contribuição do cliente. Recomenda-se, portanto, aplicar uma combinação de técnicas de elicitação de requisitos, tais como:

- **Entrevistas** com o cliente, *product owner* (PO) e demais *stakeholders*;
- **Analogia** com sistemas existentes (realizando a análise de *logs* de produção);
- ***Brainstorming*** envolvendo especialistas no negócio e pessoal técnico;
- **Testes de desempenho exploratórios**, submetendo o sistema sob teste a cargas definidas aleatoriamente, definindo os requisitos de desempenho em função dos resultados apresentados. É importante destacar que neste caso os requisitos de desempenho são definidos tardiamente, somente após a disponibilização de uma versão executável do sistema.

5.2.2. Documentação de Requisitos de Desempenho

Há diversas técnicas disponíveis para especificar (documentar) requisitos funcionais, tais como casos de uso e histórias de usuário. Além disso, há vasta publicação a respeito de critérios de qualidade de especificação de requisitos funcionais. No entanto, não se encontram tantos padrões industriais para se especificar RNF.

Cada tipo de RNF demanda um formato de especificação diferente. Um requisito de acessibilidade poderá exigir a identificação do tipo de deficiência do usuário e suas respectivas necessidades - por exemplo, um recurso de alto contraste em uma aplicação Web pode ser muito útil para uma pessoa com baixa visão, porém não terá utilidade para uma pessoa com cegueira total. Já um requisito de desempenho poderá exigir a especificação de um modelo de carga (*workload model*), incluindo, para cada funcionalidade do sistema uma expectativa de usuários concorrentes em momentos de uso típico e de pico e o tempo de resposta desejável ou aceitável.

Geralmente, RNF são documentados em linguagem natural, usando texto livre. Essa característica pode acarretar em descrições insuficientes, ambíguas e inconsistentes, diminuindo sua utilidade no contexto do projeto. Os três exemplos de RNF a seguir foram coletados em projetos reais na indústria. Todos se referem a requisitos de desempenho e apresentam problemas que confundem e, até mesmo, inviabilizam o trabalho de arquitetos, testadores e demais interessados nesses requisitos.

- "A quantidade de acessos **simultâneos** estimada **mínima** é de 200 usuários."
    - Provavelmente, o analista de requisitos quis dizer "acessos concorrentes", o que é diferente de "simultâneos" (paralelos).
    - O que de fato importa é saber a quantidade máxima de acessos concorrentes, para que tanto o hardware quanto o software sejam dimensionados para suportar as cargas de pico.
- "O sistema pode acomodar **vários** usuários ou transações."
    - O requisito está incompleto e impreciso. O que significa "vários"? É necessário indicar claramente quantos acessos concorrentes o sistema deve suportar e em qual espaço de tempo.
- "O tempo **médio** de resposta às operações de atualização e consultas **simples** deverá ser no máximo de **trinta segundos**."
    - Recomenda-se não usar o tempo médio como parâmetro para um requisito de desempenho, tendo em vista que a média é muito sensível a valores extremos (*outliers*). Neste caso, é preferível utilizar medidas relacionadas aos percentis 90, 95 ou 99 de tempo de resposta, dependendo do grau de criticidade do sistema.
    - O requisito é genérico e impreciso, quando agrupa e não indica quais são as "operações de atualização e consultas simples". Além disso, se há operações "simples", provavelmente também há operações "complexas", as quais não são mencionadas no requisito.
    - Provavelmente, um tempo médio de trinta segundos para responder a operações classificadas como "simples" não será tolerado pelos usuários.

O exemplo a seguir, por sua vez, reúne um conjunto de boas práticas de especificação de requisitos de desempenho que viabilizam a sua testabilidade:

- "A funcionalidade 'Consultar pedidos de compra' deve fornecer tempos de resposta de no máximo 2s, considerando 100 usuários concorrentes, com uma rampa de subida de 10s e 200 registros a serem retornados do banco de dados."
    - O requisito é específico: refere-se a apenas uma funcionalidade.
    - Indica objetivamente o tempo de resposta tolerado e em que condições ele deve ser observado, considerando o número de usuários concorrentes, a rampa de subida e a quantidade de registros retornados.

Esse exemplo, no entanto, não representa um padrão ou modelo a ser seguido em qualquer contexto. Características específicas de um projeto poderão exigir um formato de especificação diferenciado visando garantir a testabilidade do requisito de desempenho.

5.3. Testes de Desempenho de Software

As subseções 5.3.1, 5.3.2 e 5.3.3 descrevem, respectivamente, o processo típico de teste de software, as principais medidas de desempenho de software e os tipos de teste de desempenho mais comuns. A definição precisa desses conceitos é fundamental para a compreensão das práticas que envolvem um projeto de testes dessa natureza.

5.3.1. Processo de Teste de Software

As atividades que compõem um processo de teste de software podem variar de acordo com a organização. No entanto, de acordo com a norma ISO/IEC/IEEE 29119-2 [ISO/IEC/IEEE 2013], um processo de testes típico costuma incluir as seguintes atividades:

- **Planejamento de testes**: atividade caracterizada pela elaboração do Plano de Teste, documento frequentemente utilizado para descrever o escopo do projeto de testes e para definir a estratégia de testes, os recursos alocados e o cronograma;
- **Design e implementação de testes**: atividade na qual são derivados os casos e procedimentos de teste a partir dos requisitos do software. Esta atividade requer que os testadores apliquem uma ou mais técnicas de design de testes com o objetivo de alcançar os critérios de conclusão estabelecidos, tipicamente descritos em termos de medidas de cobertura. Esta atividade também contempla a implementação dos procedimentos de teste na forma de *scripts* de teste automatizados;
- **Configuração do ambiente de testes**: atividade em que se estabelece o ambiente operacional no qual os testes serão executados, incluindo o software sob teste e ferramental de apoio a sua operação;
- **Execução de testes**: atividade em que os procedimentos de teste definidos na atividade de design e implementação de testes são executados sobre o ambiente de testes estabelecido, de forma manual ou automatizada;
- **Comunicação de incidentes**: atividade em que os resultados observados com a execução de testes são analisados e eventuais incidentes de teste são registrados e comunicados às partes interessadas;
- **Conclusão do projeto de testes**: esta atividade marca o encerramento do projeto de testes e consiste em uma série de tarefas, incluindo o registro e comunicação dos resultados do projeto de testes às partes interessadas, a "limpeza" do ambiente de testes e a identificação de lições aprendidas.

Testes de desempenho podem ser realizados de acordo com esse processo-base, com as devidas adaptações considerando as características desse tipo de teste. Atividades de diagnóstico de gargalos e *tuning* do sistema, apesar de importantes nesse contexto, não fazem parte do processo de testes - elas compõem o processo de desenvolvimento.

5.3.2. Medidas de Desempenho

Dentre os diversos tipos de RNF representados na Figura 5.1, na forma de características e subcaracterísticas de qualidade, há o requisito de "eficiência no desempenho" ou apenas "**desempenho**". De acordo com esse modelo, o desempenho de um produto de software pode ser observado em função de três dimensões: o comportamento em relação ao tempo, a utilização de recursos e a capacidade do sistema.

O **comportamento em relação ao tempo** é definido como o "grau em que os tempos de resposta e de processamento, bem como as taxas de *throughput* do sistema, atingem seus requisitos, quando as funções do sistema são exercitadas". Já a **utilização de recursos** é definida como o "grau em que a quantidade e variedade de recursos usados pelo sistema atingem seus requisitos, quando as funções do sistema são exercitadas". Por fim, a **capacidade** é definida como o "grau em que os limites máximos de um parâmetro do sistema, incluindo o número de itens a serem armazenados, número de usuários concorrentes, largura de banda, *throughput* das transações e tamanho do banco de dados, atingem seus requisitos" [ISO/IEC 2011].

Cada uma dessas dimensões pode ser medida de diferentes maneiras, conforme sugere a norma ISO/IEC/IEEE 25023 [ISO/IEC/IEEE 2015], como pode ser observado na Tabela 5.1.

Tabela 5.1. Dimensões e medidas de desempenho de software [ISO/IEC/IEEE 2015].

Comportamento no tempo	Utilização de recursos	Capacidade
Tempo médio de resposta	Utilização média do processador	Capacidade de processamento de transações
Adequação do tempo de resposta	Utilização média da memória	Capacidade de acessos de usuários
Tempo médio de entrega	Utilização média dos dispositivos de E/S	Adequação do aumento do número de acessos
Adequação do tempo de entrega	Utilização da largura de banda	
Throughput médio		

Para avaliar se um produto de software atende a essa variedade de requisitos é necessário realizar **testes de desempenho**. De acordo com a norma ISO/IEC/IEEE 29119-1, o teste de desempenho (ou *performance*) mede e avalia o grau em que um item de teste desempenha suas funções designadas dentro de determinados limites de tempo, de uso de recursos (CPU, memória, disco, rede) e de capacidade [ISO/IEC/IEEE 2013].

Testes de desempenho também servem para identificar eventuais "gargalos" (*bottlenecks*) no sistema. Um **gargalo** é uma parte do sistema que limita o desempenho ou capacidade de todo o sistema, afetando principalmente os tempos de resposta e o *throughput* [Bondi 2014]. Os gargalos em um sistema podem estar localizados no próprio sistema (como uma *query* mal elaborada ou não-otimizada), na camada de software que sustenta esse sistema (sistemas operacionais, servidores de aplicação, *firewalls* mal configurados) ou no hardware, incluindo CPU, memória, disco e rede.

A avaliação de cada subcaracterística de desempenho, incluindo a identificação de gargalos, demanda a realização de diferentes tipos de testes de desempenho. A seção a seguir classifica e descreve os principais tipos de testes de desempenho.

5.3.3. Tipos de Testes de Desempenho

Um dos principais problemas da área de Teste de Software é a ausência de uma terminologia e de uma taxonomia consistentes e que sejam aceitas pela indústria. Quando se trata de Testes de Desempenho, em especial, surgem inúmeros termos (*performance*, carga, *stress*, etc.) que, em geral, não são compreendidos da mesma forma, até mesmo por profissionais da mesma organização e com muitos anos de experiência na área. A Figura 5.3 representa o "guarda-chuva" de termos relacionados a testes de desempenho.

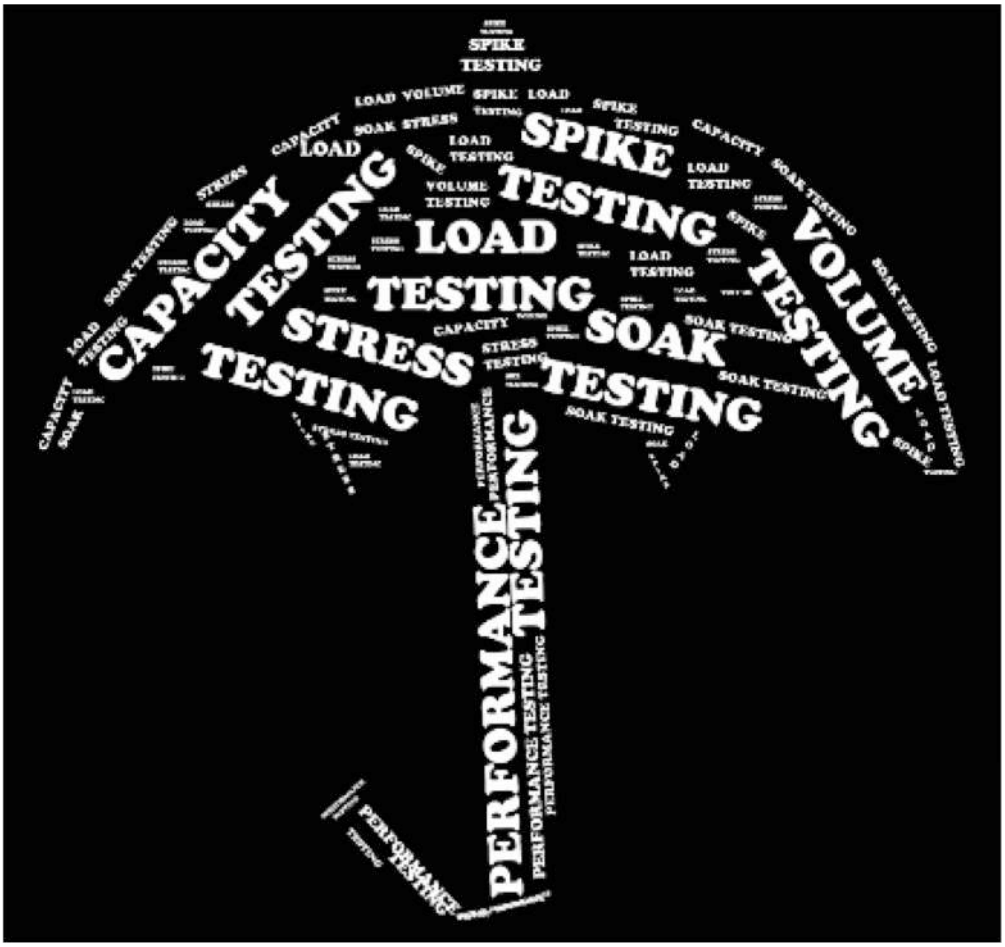


Figura 5.3: "Guarda-chuva" conceitual sobre testes de desempenho.

Essa falta de consenso pode levar tanto a ruídos simples de comunicação quanto a erros graves na cobrança pela prestação de serviços desta natureza. Além disso, o desconhecimento sobre Testes de Desempenho contribui para que os requisitos de desempenho não sejam elicitados e documentados de forma apropriada, podendo ocasionar diferentes impactos sobre o projeto.

Diante deste cenário, a norma ISO/IEC/IEEE 29119-1 sugere uma classificação e um conjunto de definições. Esta classificação considera, genericamente, "Teste de Desempenho" como um tipo de teste que possui cinco subtipos: 1) teste de carga; 2) teste de estresse; 3) teste de capacidade; 4) teste de resistência; e 5) teste de volume, definidos conforme a Tabela 5.2 [ISO/IEC/IEEE 2013]:

Tabela 5.2. Tipos de testes de desempenho conforme a norma ISO/IEC/IEEE 29119-1 [ISO/IEC/IEEE 2013].

Tipo de teste	Definição
Teste de carga	Avaliar o comportamento de um item de teste em condições **esperadas de carga** variável (uso baixo, típico e de pico).
Teste de estresse	Avaliar o comportamento de um item de teste em condições de **carga acima dos requisitos** de capacidade antecipados ou especificados ou da disponibilidade de recursos abaixo dos requisitos mínimos exigidos.
Teste de capacidade	Avaliar o nível no qual o **aumento de carga** (de usuários, transações, armazenamento de dados, etc.) **compromete a capacidade** de um item de teste para sustentar o desempenho requerido.
Teste de resistência	Avaliar se um item de teste pode sustentar uma **carga necessária continuamente** por um período de tempo determinado.
Teste de volume	Avaliar a capacidade do item de teste processar **volumes de dados** especificados (usualmente no ou **próximos ao limite máximo de capacidade** especificada) em termos de *throughput*, capacidade de armazenamento ou ambos.

Apesar de útil, essa classificação ainda se mostra superficial à medida em que não detalha as diferenças entre os cinco tipos de teste apresentados e desconsidera outros tipos de testes relacionados ao desempenho, tais como o *spike testing* e o *soak testing*.

Desta forma, foi realizado um trabalho de harmonização de conceitos, considerando o que já está definido na norma ISO/IEC/IEEE 29119-1 [ISO/IEC/IEEE 2013] e outros tipos de testes de desempenho que estão ausentes da norma. Com base nas definições encontradas, foram identificadas algumas relações de sinonímia e de hiponímia entre os termos, resultando no modelo conceitual representado na Figura 5.4.

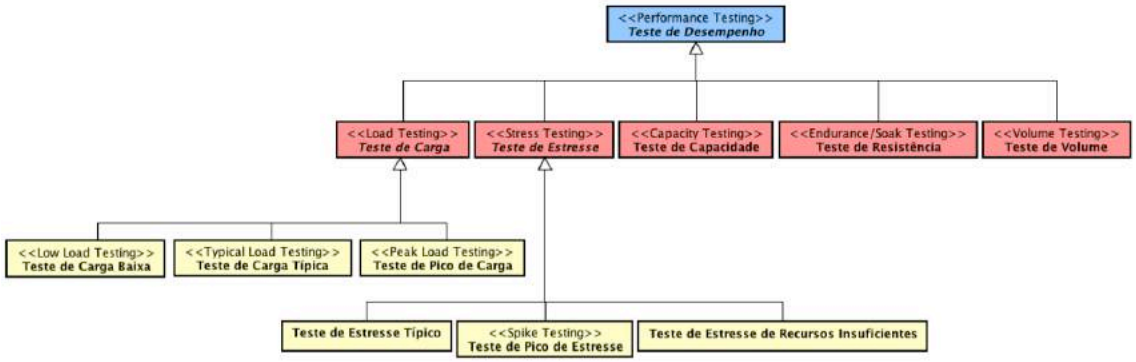


Figura 5.4: Tipos de testes de desempenho.

Para diferenciar os diversos tipos de testes de desempenho representados nesse modelo conceitual, deve-se levar em conta os seguintes parâmetros:

- **expectativa de utilização típica**: a quantidade de usuários/transações concorrentes e/ou de dados a serem processados, esperada para horários de utilização média (regular) do sistema (desconsiderando períodos de pico);
- **número de usuários/transações concorrentes**: a quantidade de usuários/transações concorrentes submetida ao sistema sob teste;

- **quantidade de dados a serem processados**: a quantidade (volume) de dados submetida ao sistema sob teste;
- **capacidade máxima projetada**: a medida da carga máxima suportada pelo sistema, incluindo seu ambiente de execução. Idealmente, a capacidade máxima projetada deve estar acima da expectativa de utilização típica, considerando picos de acesso e previsão de crescimento do sistema.

O **teste de carga baixa** é um subtipo do teste de carga que, semelhantemente a um teste fumaça, tem como objetivo principal avaliar se as funcionalidades críticas de um sistema estão funcionando como esperado. Desta forma, o sistema é submetido a um conjunto de casos de teste que exercitam diversas funcionalidades e possíveis integrações, sob uma carga reduzida, bem abaixo da expectativa de utilização típica, como representa a Figura 5.5.

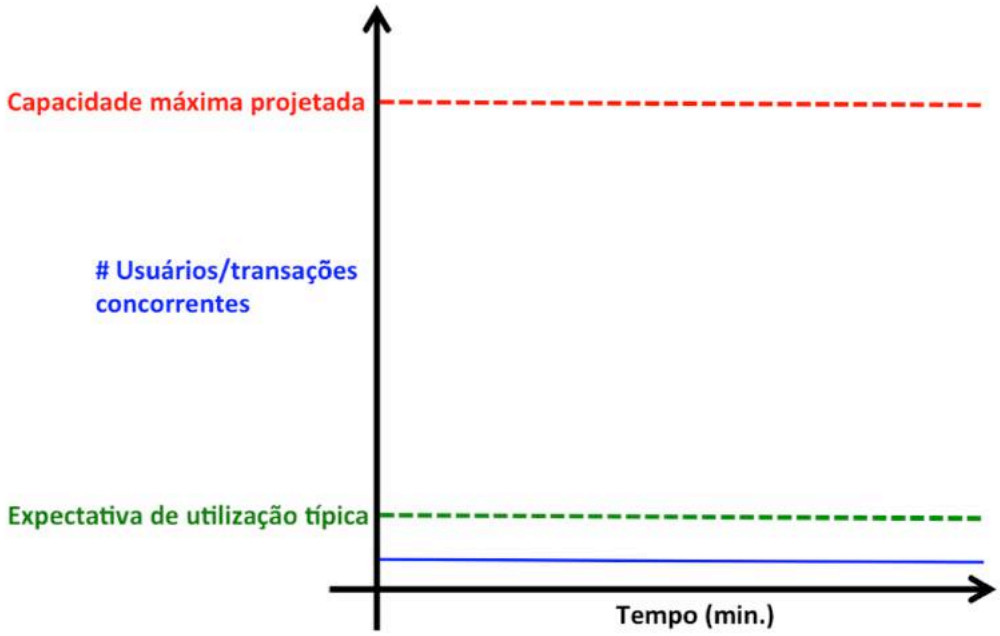


Figura 5.5. Teste de carga baixa.

O **teste de carga típica** é um subtipo do teste de carga que avalia o desempenho do sistema diante da submissão de uma carga correspondente à expectativa de utilização típica, conforme a Figura 5.6.

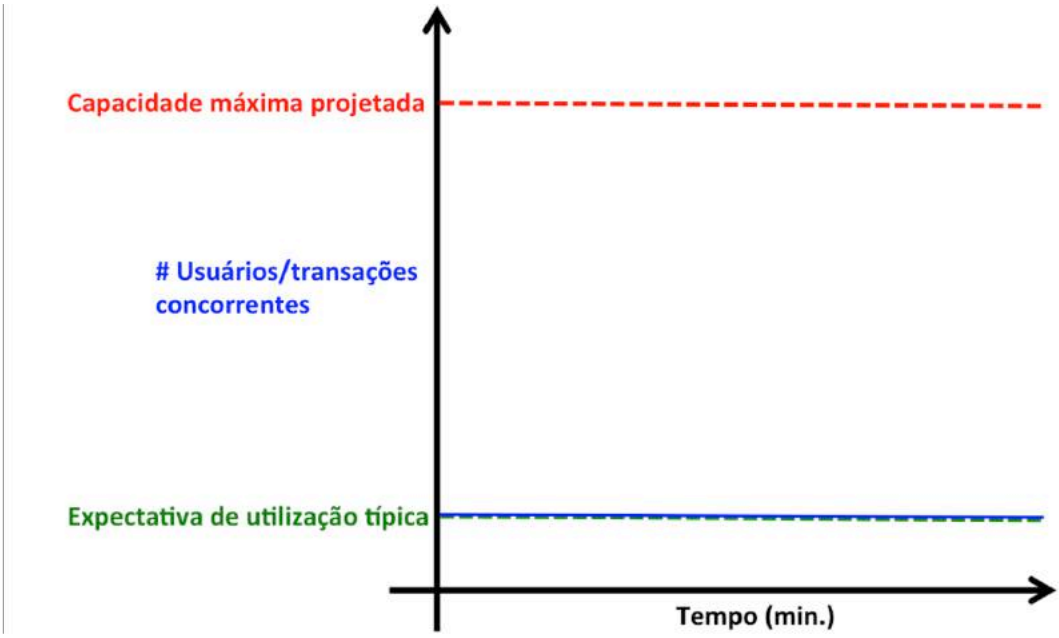


Figura 5.6. Teste de carga típica.

O **teste de pico de carga**, ou *peak load testing*, é o subtipo de teste de carga mais comum. Seu objetivo é avaliar o desempenho do sistema diante de cargas que correspondam aos picos de utilização previstos. Para simular um pico de utilização é necessário estabelecer o período de aceleração ou subida (*ramp-up period*) e, eventualmente, o período de desaceleração ou descida (*ramp-down period*), como mostra a Figura 5.7. O *ramp-up period* é o tempo dentro do qual todos os usuários virtuais (*threads*) devem ser iniciados, submetendo requisições ao sistema sob teste. Já o *ramp-down period* corresponde à faixa de tempo em que os usuários virtuais devem ser finalizados, encerrando o processamento de suas requisições.

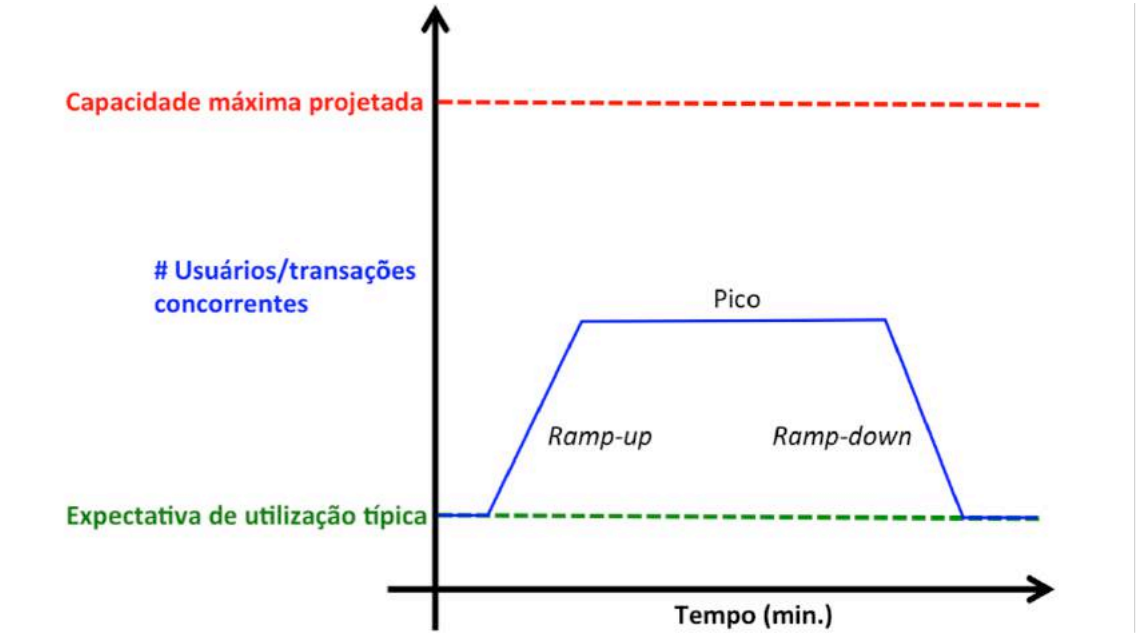


Figura 5.7. Teste de pico de carga.

De forma geral, testes de carga avaliam o desempenho do sistema dentro dos limites de capacidade máxima projetada. Já o testes de estresse extrapolam esses limites, com o objetivo de avaliar o comportamento do sistema diante de condições extremas de utilização. O resultado típico de um teste de estresse realizado sobre um sistema Web é a interrupção ("queda") do servidor HTTP. Há, ainda, casos de exposição de vulnerabilidades de segurança. No entanto, outros resultados podem ser observados, especialmente em plataformas não-Web. Portanto, é preciso se atentar a qualquer anomalia no comportamento do sistema quando submetido a um teste de estresse.

O **teste de estresse típico** é um subtipo do teste de estresse usado para avaliar como o sistema sob teste se comporta diante de cargas superiores à sua capacidade máxima projetada. Também pode ser usado para identificar o "ponto de quebra" (*breaking point*) do sistema, ou seja, o tamanho de carga que faz o sistema parar de funcionar. Esse tipo de teste envolve a submissão de uma carga que aumenta continuamente ao longo do tempo, conforme a Figura 5.8, sendo interrompido somente quando o sistema apresentar alguma anomalia.

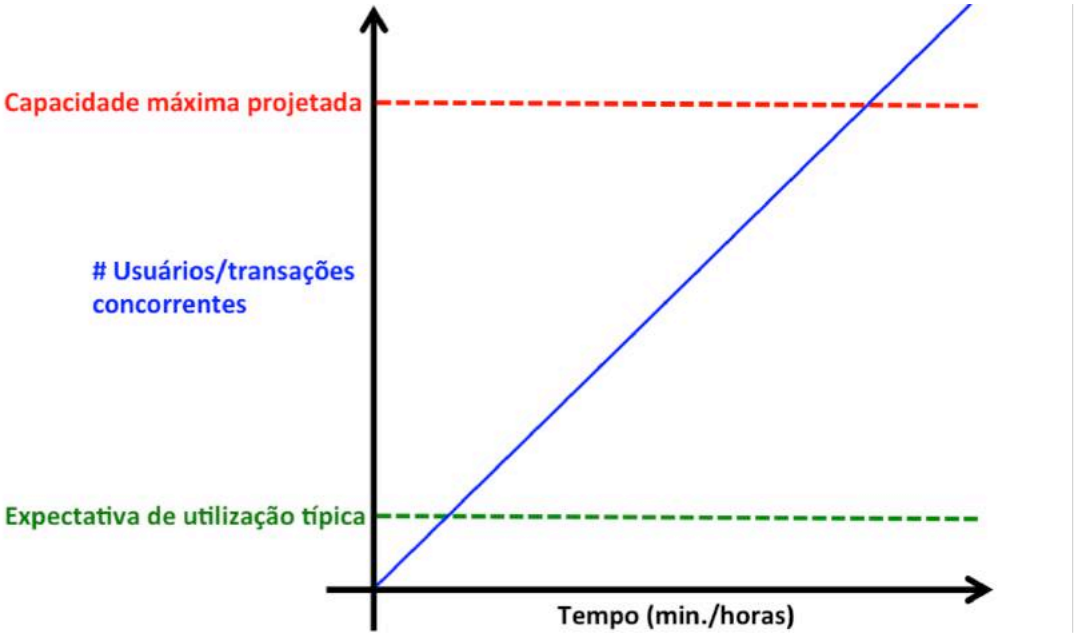


Figura 5.8. Teste de estresse típico.

O **teste de pico de estresse**, ou *spike testing*, é um subtipo do teste de estresse que submete o sistema a acréscimos e decréscimos extremos e repentinos de carga, como representa a Figura 5.9. Esse teste ajuda a determinar se o desempenho do sistema se deteriora quando há um aumento súbito da carga. Outro objetivo desse teste é determinar o tempo de recuperação do sistema. Entre dois picos consecutivos de carga do usuário, o sistema precisa de algum tempo para se estabilizar. Esse tempo de recuperação deve ser o mais baixo possível.

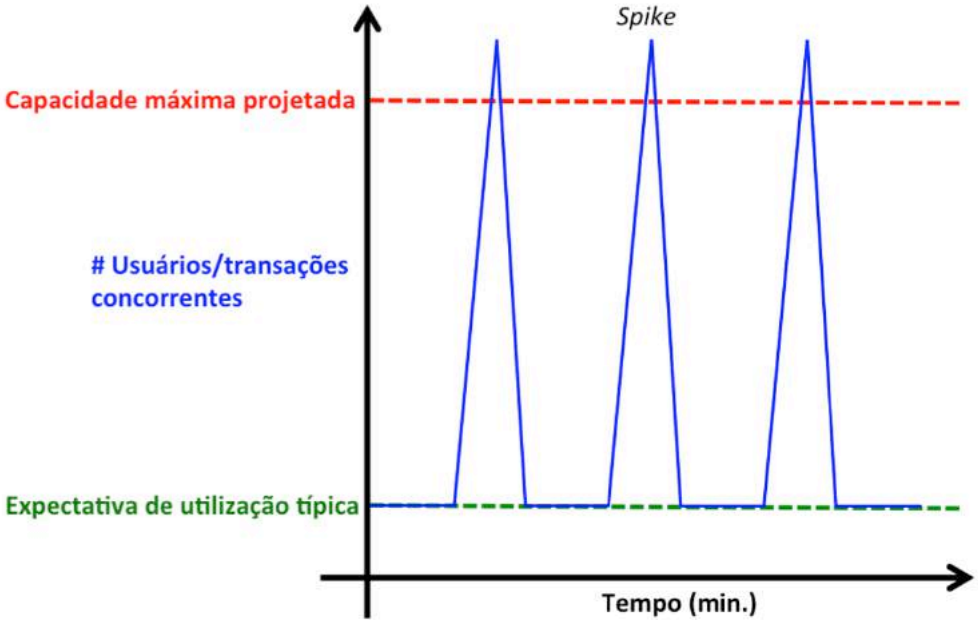


Figura 5.9. Teste de pico de estresse.

O **teste de estresse de recursos insuficientes** é um subtipo do teste de estresse no qual o comportamento do sistema é avaliado sob uma configuração de hardware e software muito aquém das configurações mínimas necessárias (os parâmetros de "expectativa de utilização típica" e "capacidade máxima projetada" são invertidos, como representado na Figura 5.10). Por exemplo, para executar um determinado programa *desktop* é necessário um computador com o sistema operacional Windows 10 e no mínimo uma CPU de dois núcleos, 8GB de RAM e 15GB de espaço em disco. Porém, o que poderia acontecer se esse programa fosse instalado em uma máquina que não atenda a esses requisitos mínimos? A resposta a essa pergunta pode ser obtida por meio de um teste de estresse de recursos insuficientes, em que, por exemplo, o programa fosse instalado em uma máquina com Windows 8 e demais requisitos abaixo do necessário.

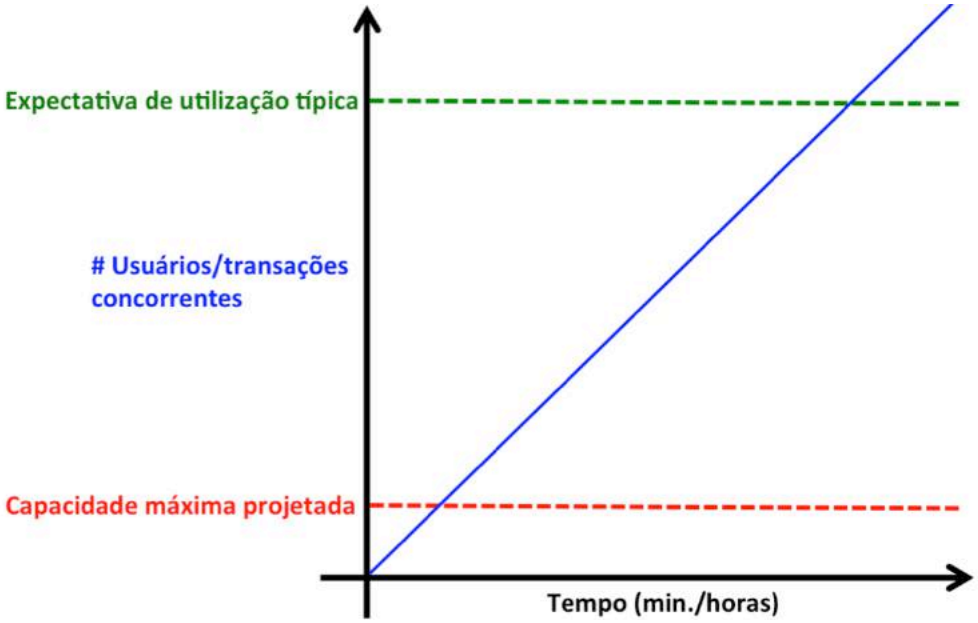


Figura 5.10. Teste de estresse de recursos insuficientes.

O **teste de capacidade**, representado na Figura 5.11, é um tipo de teste de desempenho cujo objetivo é avaliar como o sistema se comporta em termos de tempo de resposta e utilização de recursos, à medida em que se aumenta a carga. Trata-se de um tipo de teste com características parecidas com o teste de estresse típico. Porém, no teste de capacidade, a carga é aumentada de forma iterativa - para cada patamar, analisa-se o desempenho do sistema. Desta forma, testa-se a capacidade do sistema sob diferentes cargas até encontrar aquela cujo desempenho do sistema seja satisfatório para o cliente.

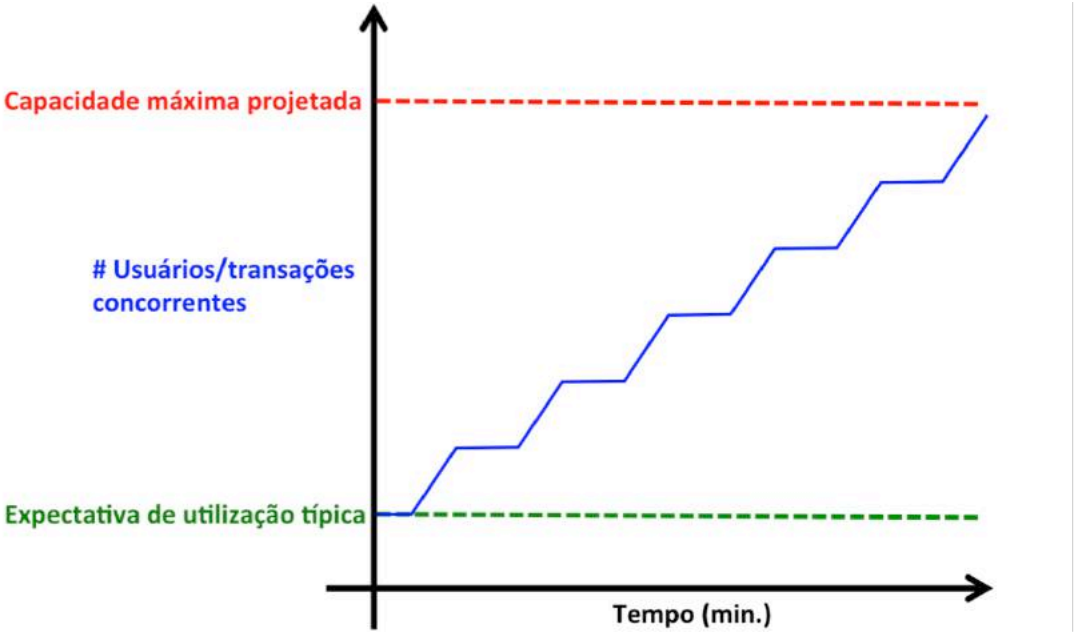


Figura 5.11. Teste de capacidade.

O **teste de resistência**, ou *soak testing*, é um tipo do teste de desempenho de longa duração usado para determinar o desempenho e/ou a estabilidade do sistema ao longo do tempo. Um sistema pode funcionar bem por uma ou duas horas e, em seguida, começar a ter problemas. Geralmente, um teste de resistência consiste em executar testes de carga continuamente (em *loop*) por cerca de 24h ininterruptas, com uma carga acima da expectativa de utilização típica, conforme a Figura 5.12. Esses testes são especialmente úteis para identificar vazamentos de memória (*memory leaks*).

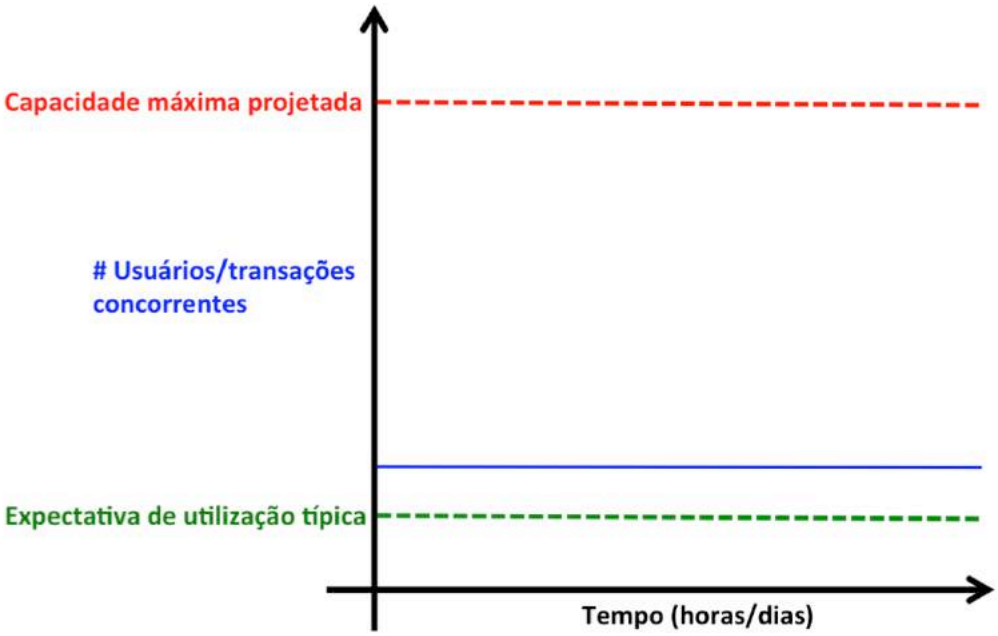


Figura 5.12. Teste de resistência.

De forma geral, testes de desempenho se referem à prática de simular vários usuários acessando o programa de modo concorrente (não necessariamente simultâneo). No entanto, há situações em que se deseja avaliar qual o volume de dados que o sistema é capaz de processar, independentemente da quantidade de usuários concorrentes. Por exemplo, um teste de carga pode envolver a execução de 100 usuários concorrentes realizando pedidos de compra típicos. Já um **teste de volume**, poderia envolver o processamento de uma única requisição com um pedido de compra que explorasse todos os limites de cardinalidade dos atributos, submetendo ao sistema uma grande quantidade de dados. O teste de volume, portanto, implica em submeter o sistema ao processamento da quantidade máxima de dados para a qual foi projetado, como ilustra a Figura 5.13.

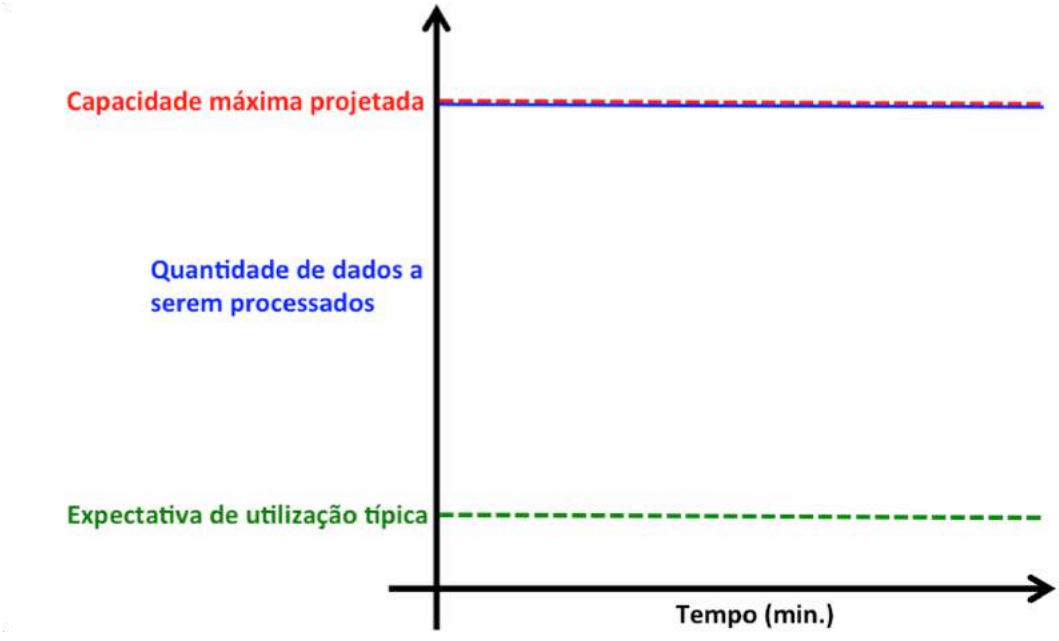


Figura 5.13. Teste de volume.

5.4. Apache JMeter

O Apache JMeter é uma ferramenta de código aberto desenvolvida em Java utilizada para realização de testes de desempenho. A primeira versão do JMeter foi lançada em 1998. Ele foi originalmente projetado para testar aplicações Web, mas desde então passou a cobrir diversas outras tecnologias de software cliente/servidor, como conexões de bancos de dados JDBC, serviços de mensageria JMS, serviços de diretórios LDAP, entre outros [Apache Foundation 2018].

O JMeter também pode ser utilizado para realização de testes funcionais, uma vez que possui recursos que possibilitam automatizar testes desse tipo, tais como asserções. No JMeter, uma asserção corresponde a uma condição que deve ser satisfeita pela resposta a uma requisição [Erinle 2017]. Há diversos tipos de asserções disponíveis, tais como as asserções de resposta, que possibilitam avaliar, entre outros itens, o conteúdo das respostas. Essa flexibilidade faz do JMeter uma das ferramentas de testes mais utilizadas na indústria [Erinle 2014].

Um *script* de teste no JMeter é organizado de forma hierárquica. O elemento mais ao topo dessa hierarquia é o Plano de Teste (*Test Plan*). Um Plano de Teste completo consistirá em um ou mais grupos de usuários (*threads*), controladores lógicos, requisições, ouvintes, temporizadores, asserções e elementos de configuração [Apache Foundation 2018]. A Figura 5.14 representa a interface gráfica do JMeter. À esquerda, é possível observar um exemplo de Plano de Teste com seus respectivos componentes, o que corresponde a um *script* de teste.

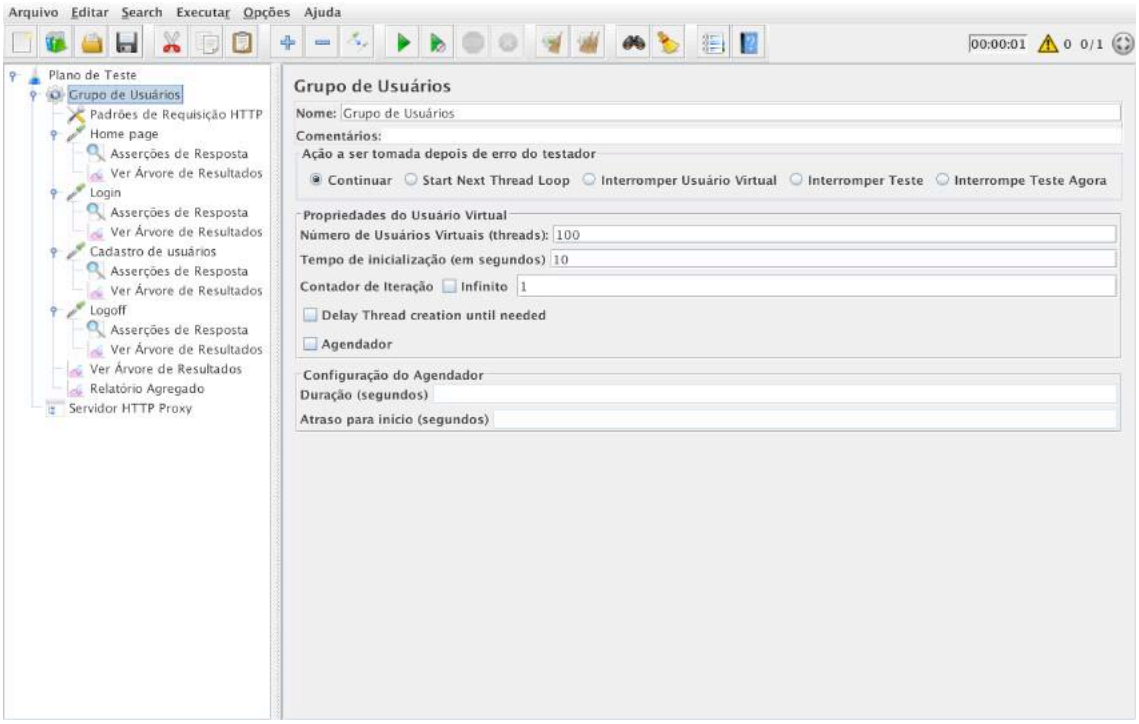


Figura 5.14. Interface gráfica do Apache JMeter.

A seção a seguir descreve os principais recursos do JMeter por meio de um exemplo prático. Para tal, utilizou-se a versão mais recente da ferramenta no momento da escrita desse texto, a versão 5.0.

5.5. Exemplo Prático

Para demonstrar os conceitos apresentados e o uso da ferramenta JMeter, esta seção apresenta um exemplo prático de testes de desempenho sobre uma aplicação Web. Levando-se em conta que o JMeter é uma ferramenta com enorme variedade de recursos, as possíveis soluções (*scripts*) para um mesmo problema tendem ao infinito. Portanto, não se pretende cobrir todos os recursos do JMeter, muito menos apresentar uma solução como a única possível.

Este exemplo prático é uma adaptação livre de exemplos encontrados na obra de Matam & Jain (2017). Trata-se do sistema fictício de comércio eletrônico "Digital Toys", disponível em https://github.com/jmeterbyexample, juntamente com diversos *scripts* de teste do JMeter. A Figura 5.15 apresenta a *home page* do sistema Digital Toys.

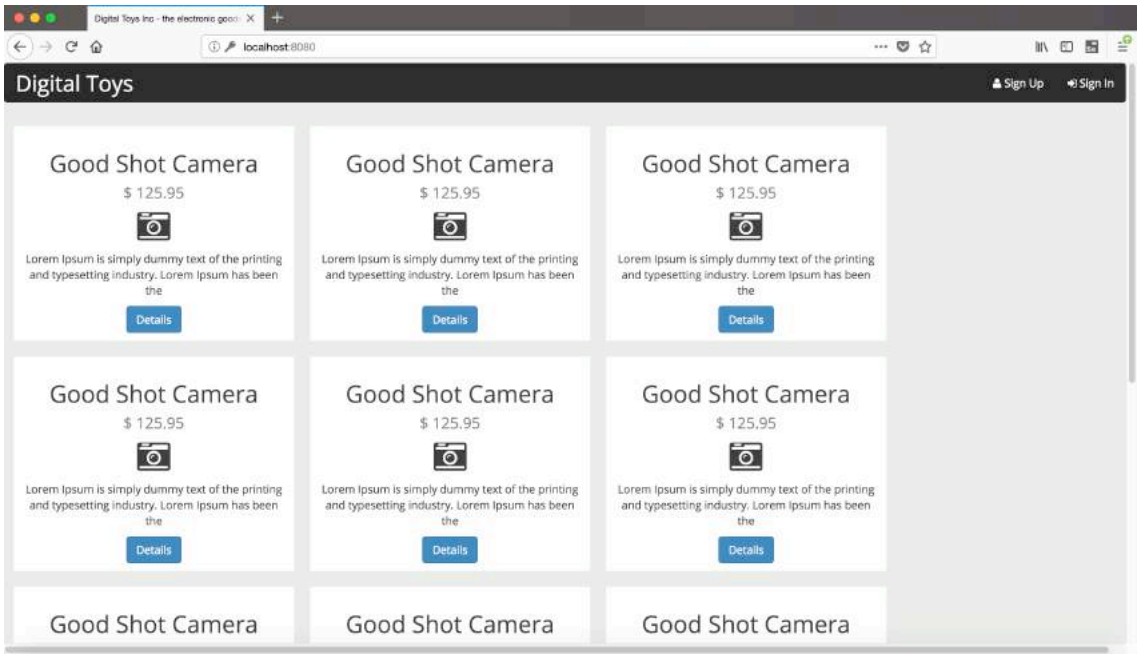


Figura 5.15. *Home page* do Digital Toys.

Como um sistema típico de comércio eletrônico, o Digital Toys possui funcionalidades que possibilitam a compra de produtos, tais como eletroeletrônicos e livros. Dentre as funcionalidades mais importantes do sistema estão as seguintes:

- Realizar compra: funcionalidade mais importante do sistema, tanto do ponto-de-vista arquitetural quanto de negócio. Por meio desta funcionalidade, os clientes do Digital Toys realizam a compra de produtos comercializados pelo sistema.
- Manter produtos: API REST que possibilita ao administrador do sistema Digital Toys realizar o cadastro, a consulta, a alteração e a remoção de produtos.
- Cadastrar cliente: funcionalidade que permite ao usuário interessado em realizar compras no Digital Toy se cadastrar no sistema.
- Autenticar usuário ("*Sign Up*"): recurso que possibilita aos usuários realizarem o *login* no sistema.

- Converter moeda: trata-se de um *web service* SOAP que é consumido sempre que um cliente opta por realizar o pagamento em uma moeda diferente do dólar americano.

Considerando as funcionalidades apresentadas, o testador definiu o modelo de carga representado na Tabela 5.3. Como se pode observar, estima-se que a funcionalidade "Listar produtos" (representada pelo cenário de teste "Listar 1.000 livros"), que compõe o caso de uso "Manter produtos", é a funcionalidade que representa a maior parte (40%) da carga do sistema em momentos de pico. É importante notar que esse cenário de teste indica a quantidade de registros que devem ser retornados pelo sistema, tendo em vista que não seria justo comparar tempos de resposta de duas ou mais execuções com quantidades diferentes de registros retornados.

Tabela 5.3. Modelo de carga do Digital Toys.

#CT	Cenário de teste	% da carga total	Usuários	Rampa	Agentes	Tempo de resposta tolerável
CT1	Cadastrar cliente	10%	100	10s	1	0,5s
CT2	Realizar compra	30%	300	10s	2	3s
CT3	Converter moeda	5%	50	5s	1	2s
CT4	Cadastrar livro	15%	150	10s	1	1s
CT5	Listar 1.000 livros	40%	400	20s	2	8s

Outro aspecto importante de se observar na Tabela 5.3 é que, dada a quantidade de usuários concorrentes, optou-se por executar os cenários de teste CT2 e CT5 a partir de dois agentes. Um **agente** é cada máquina que compõe o ambiente de ativação. O **ambiente de ativação** é o conjunto de máquinas responsáveis por disparar requisições concorrentes sobre o ambiente-alvo. O **ambiente-alvo** é o servidor para o qual são direcionadas as requisições provenientes do ambiente de ativação (trata-se do servidor que hospeda o sistema sob teste).

Sendo assim, a título de exemplo, são descritas a seguir as principais etapas para implementação do *script* de teste referente ao cenário de teste CT1 ("Cadastrar cliente"). Esse cenário é composto pelos seguintes passos, que devem ser reproduzidos pelo *script* do JMeter:

1. Acessar *home page*;
2. Escolher opção para cadastrar cliente ("*Sign Up*");
3. Preencher formulário de cadastro e confirmar ("*Submit*");
4. Sair do sistema ("*Sign Out*").

Definido o conjunto de passos do cenário a ser testado, deve-se proceder a implementação do *script* correspondente no JMeter. Quando o JMeter é aberto, a estrutura do *script* já se apresenta com um Plano de Teste vazio. Todos os elementos do *script* devem ser incluídos hierarquicamente abaixo desse Plano de Teste. A Figura 5.16 representa a interface gráfica do JMeter com um Plano de Teste.

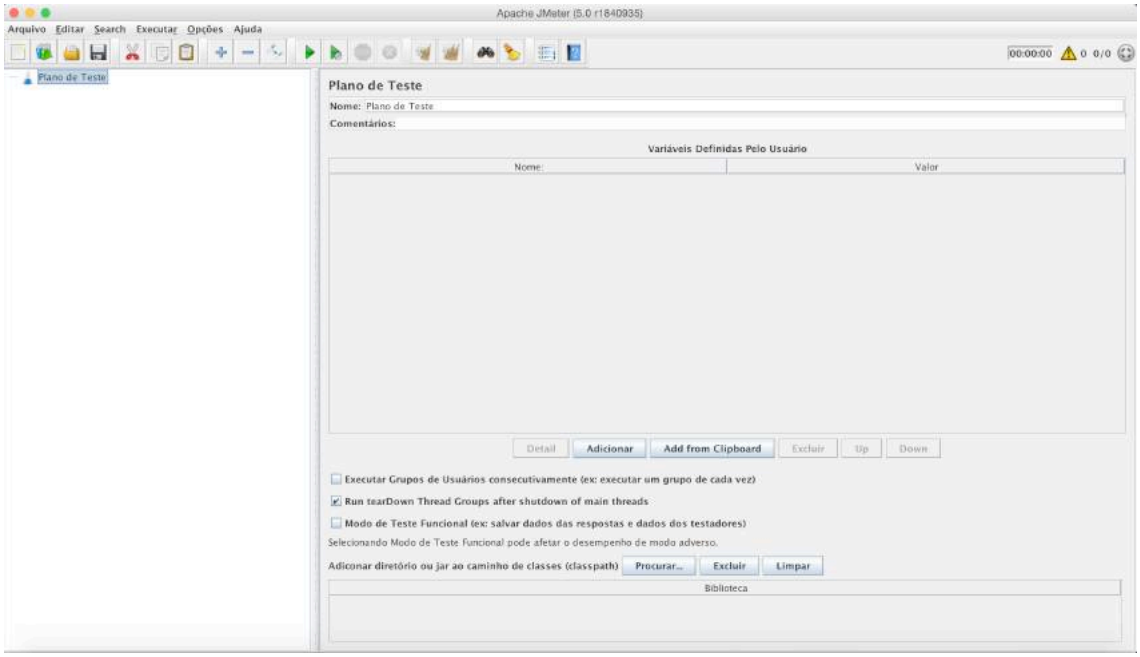


Figura 5.16. Plano de teste do JMeter.

Em seguida, deve-se incluir um Grupo de Usuários (*Thread Group*) em um nível abaixo do Plano de Teste, conforme a Figura 5.17. Nesse componente são definidos, entre outros, o número de usuários concorrentes e a rampa de subida, respectivamente nos campos "Número de Usuários Virtuais (*threads*)" e "Tempo de inicialização (em segundos)". Inicialmente, deve-se mantê-lo com os valores-padrão (1). Após a implementação do *script*, esses campos devem ser atualizados com os respectivos valores, de acordo com o modelo de carga.

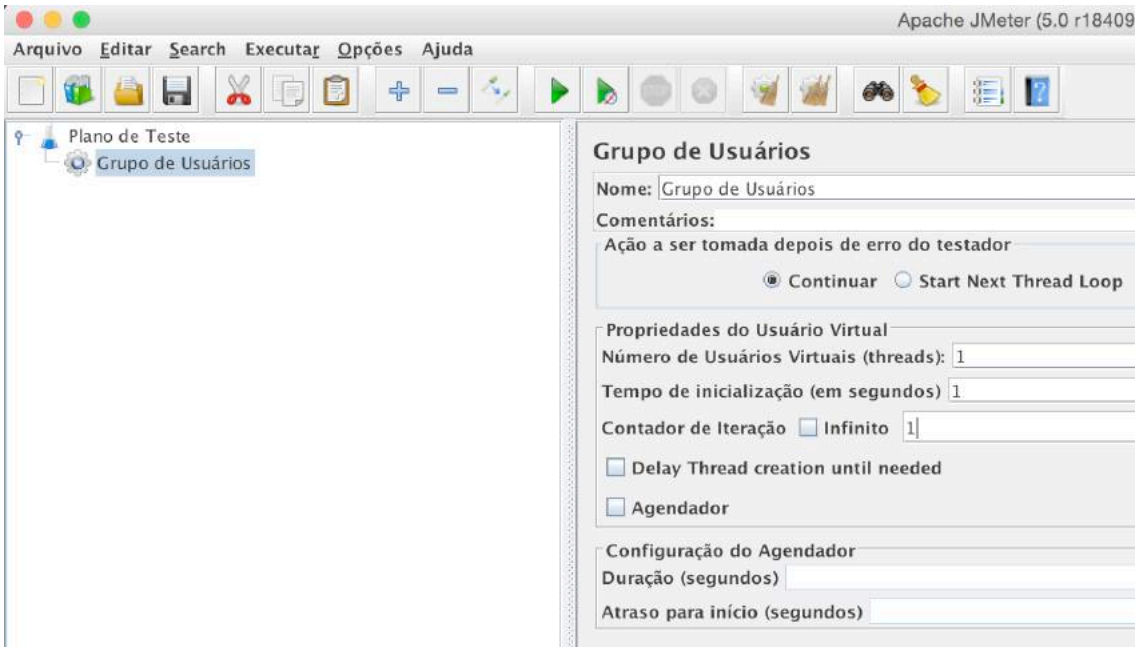


Figura 5.17. Inclusão do Grupo de Usuários.

A próxima etapa consiste na definição das requisições HTTP correspondentes a cada passo do cenário de teste. Esta etapa pode ser realizada de duas formas: manual ou automática. A forma manual consiste de se criar cada requisição HTTP no JMeter, imediatamente abaixo do Grupo de Usuários. No entanto, dependendo da quantidade e da complexidade das requisições necessárias, pode ser mais interessante mapeá-las de forma automática, utilizando um *proxy* definido no próprio JMeter. Para tal, deve-se incluir um "Servidor HTTP Proxy" no nível abaixo do Plano de Teste. O "Servidor HTTP Proxy" deve ser configurado com a indicação da porta que irá receber as requisições durante a execução dos testes, qual o controlador-alvo do *script* que irá receber as requisições mapeadas, entre outras informações. Neste exemplo, conforme a Figura 5.18, foi definida a porta 8888 (que deve ser configurada também no navegador) e as requisições mapeadas serão direcionadas para o controlador "Cadastrar cliente" (controlador simples).

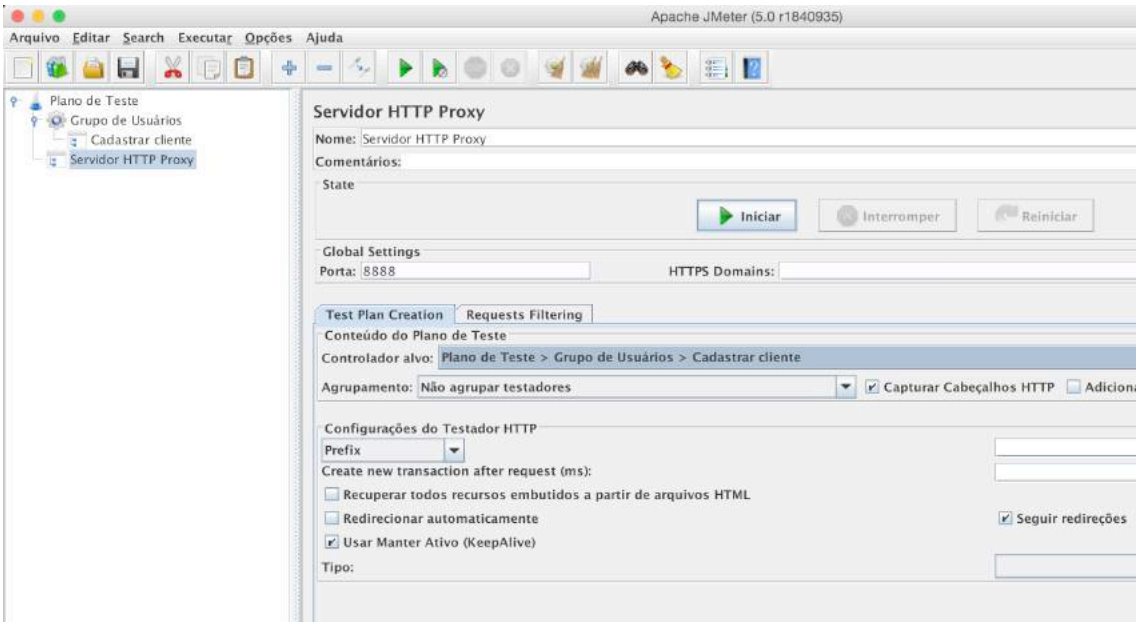


Figura 5.18. Inclusão de um Servidor HTTP Proxy.

Além disso, como o *proxy* é capaz de capturar todas as requisições feitas para quaisquer recursos HTTP, recomenda-se aplicar um filtro de modo a excluir requisições sobre recursos como arquivos CSS ou JS, tornando o *script* mais compreensível. Esse filtro pode ser aplicado na aba "*Requests Filtering*", seção "Padrões de URL a serem excluídos", clicando-se no botão "Adicionar" e informando as extensões de arquivos a serem excluídos. Pode-se, também, clicar no botão "*Add suggested Excludes*", que insere uma expressão regular informando várias extensões de arquivos que são comumente excluídos. A Figura 5.19 representa essa opção.

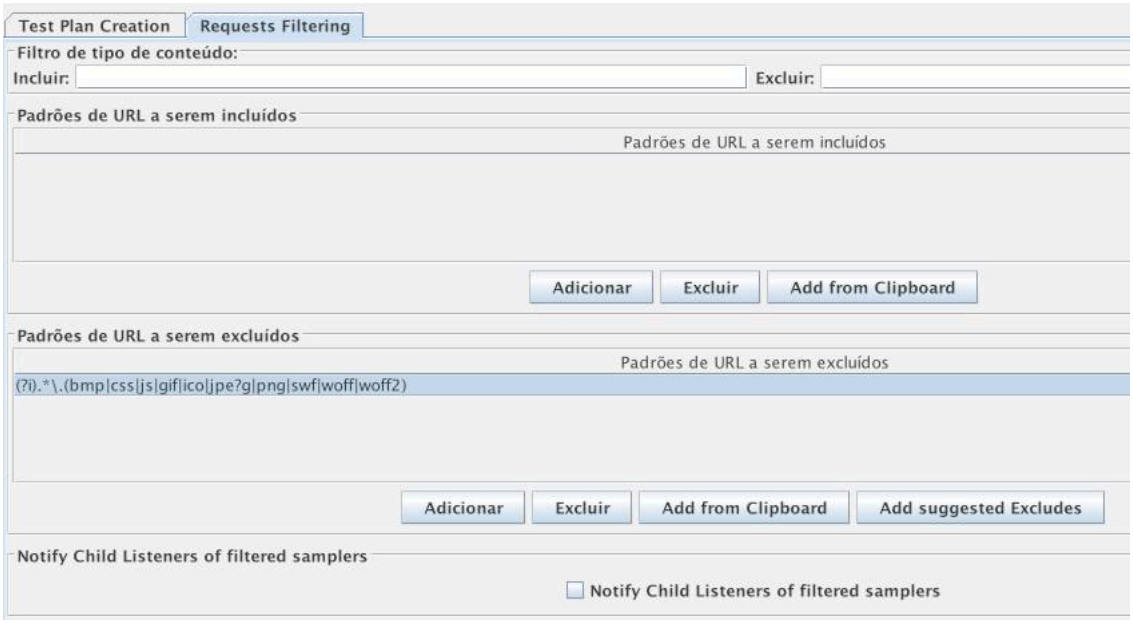


Figura 5.19. Definição de padrões a serem excluídos da gravação.

Após a configuração do "Servidor HTTP Proxy", deve-se clicar no botão "Iniciar" deste próprio componente. Em seguida, deve-se acessar a aplicação sob teste. A partir desse ponto já é possível observar que o JMeter está capturando as requisições realizadas pelo testador. Portanto, o primeiro passo consiste de acessar a *home page* do sistema Digital Toys, representado anteriormente na Figura 5.15. Em seguida, o testador deve clicar na opção "*Sign Up*", que representa o segundo passo do cenário CT1. Com isso, um formulário de cadastro de cliente será exibido, conforme a Figura 5.20. Esse formulário deve ser preenchido e submetido.

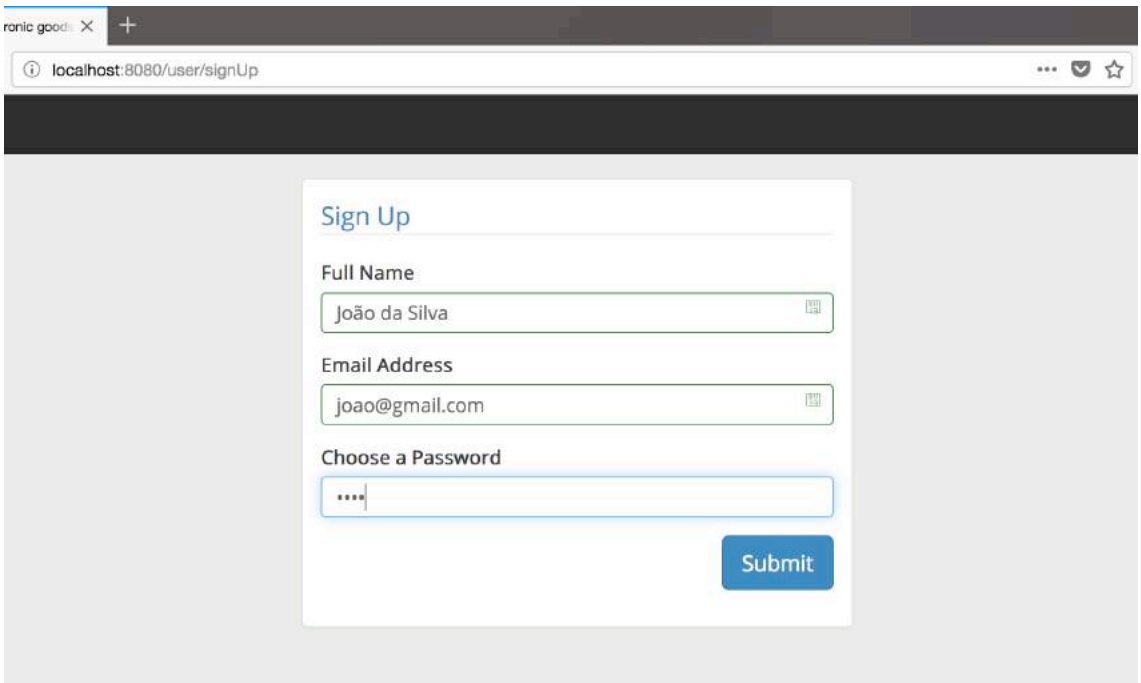


Figura 5.20. Formulário de cadastro de novo cliente.

Após a submissão do formulário de cadastro, o usuário é redirecionado para a *home page* do sistema, porém, com a indicação de que está autenticado. A Figura 5.21 destaca que o usuário recém-criado "João da Silva" está autenticado. Por fim, o testador deve clicar na opção "*Sign Out*" (sair), também representada na Figura 5.21, que corresponde ao último passo do cenário de teste.

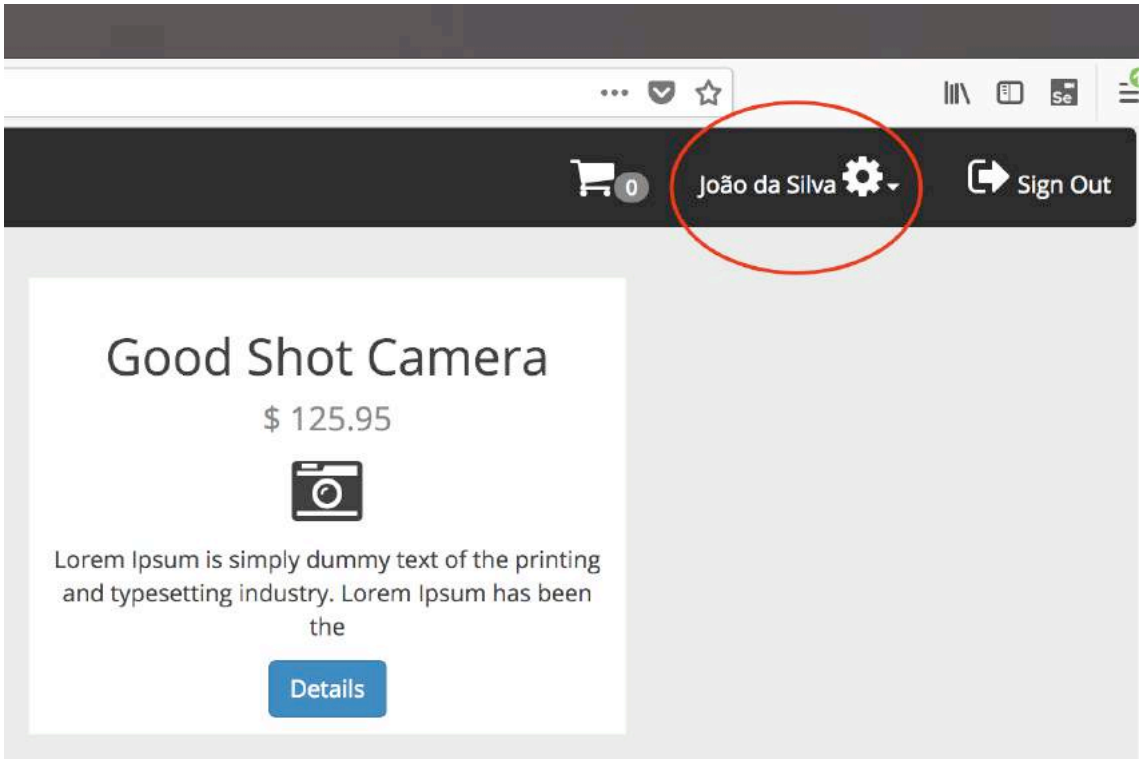


Figura 5.21. Indicação de usuário autenticado no sistema.

Uma vez executados todos os passos do cenário de teste, deve-se finalizar o proxy do JMeter clicando no botão correspondente. Feito isso, deve-se observar as requisições que foram mapeadas pelo *proxy*. A Figura 5.22 representa as quatro requisições HTTP mapeadas dentro do controlador "Cadastrar cliente" (todas foram renomeadas posteriormente). Ainda na Figura 5.22 pode-se observar em detalhes a requisição "Submeter formulário de cadastro". Trata-se de um requisição do tipo POST que passa três parâmetros: "*fullName*", "*email*" e "*password*".

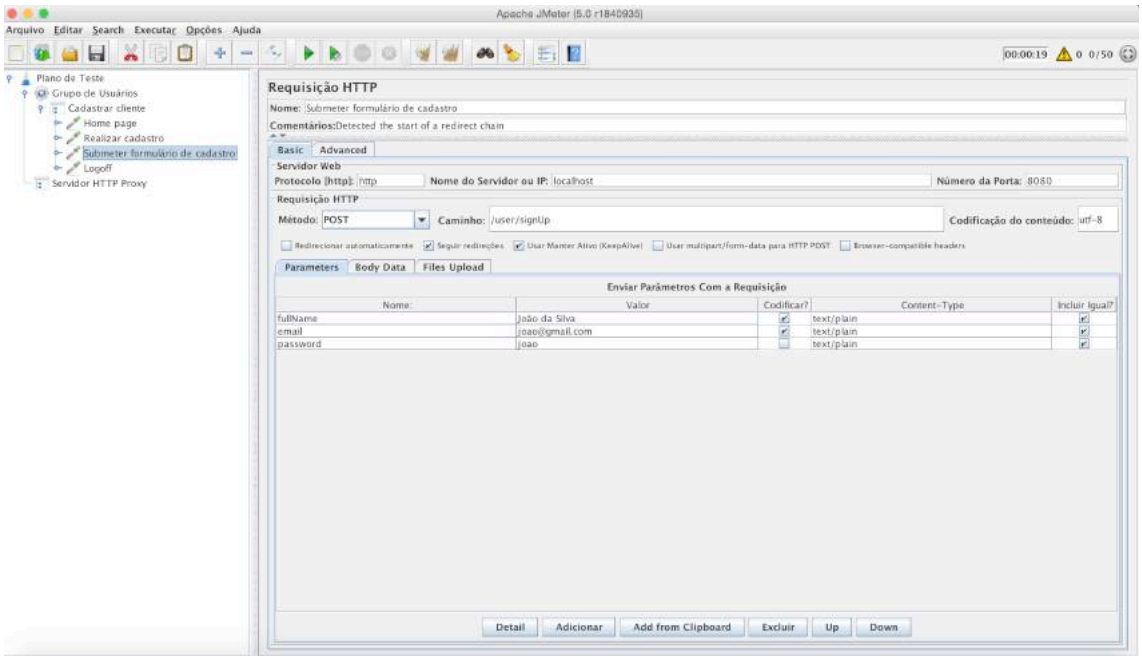


Figura 5.22. Requisições HTTP mapeadas pelo proxy.

Outra etapa importante é a inclusão de ouvintes no *script*. Um ouvinte (ou "*listener*") é uma espécie de relatório que representa os resultados das execuções das *threads* (usuários virtuais). Há diversos tipos de ouvintes para os mais variados gostos. Neste caso, foram incluídos os ouvintes "Ver Árvore de Resultados" e "Relatório Agregado", como mostra a Figura 5.23. Como eles foram adicionados no nível abaixo de Grupo de Usuários, as respostas de todas as requisições serão exibidas neles. Também é possível incluir ouvintes específicos por requisição, bastando que sejam incluídos no nível hierárquico abaixo da requisição desejada.

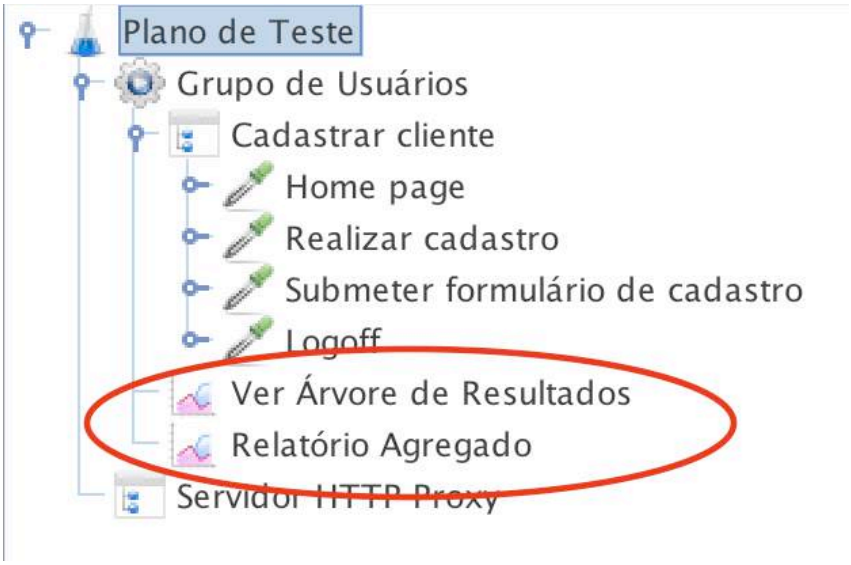


Figura 5.23. Inclusão de ouvintes (*listeners*).

Portanto, após definidos os componentes principais do *script*, deve-se retornar ao Grupo de Usuários para definir a quantidade de usuários concorrentes e a rampa de subida, como representado na Figura 5.24. Desta forma, a cada segundo, serão disparadas dez requisições a partir do ambiente de ativação.

Figura 5.24. Definição da quantidade de usuários concorrentes e da rampa de subida.

Após a configuração do Grupo de Usuários, o testador deve clicar no botão "Iniciar" na barra de ferramentas. A partir de então, as requisições serão disparadas ao ambiente-alvo. Na mesma barra, no canto direito, um contador indica ao longo da execução do teste a quantidade de requisições que ainda faltam ser processadas.

Após o término da execução, o testador deve observar os resultados por meio dos ouvintes. No ouvinte "Ver Árvore de Resultados", representado na Figura 5.25, pode-se observar quais requisições foram processadas com sucesso e quais falharam, quais cabeçalhos e qual o conteúdo de cada resposta, entre outros recursos. A grande vantagem desse ouvinte é a possibilidade de se realizar uma análise individualizada sobre requisições e suas respectivas respostas. No entanto, ele não fornece informações suficientes para determinar se o requisito de desempenho definido para o cenário de teste foi satisfeito ou não.

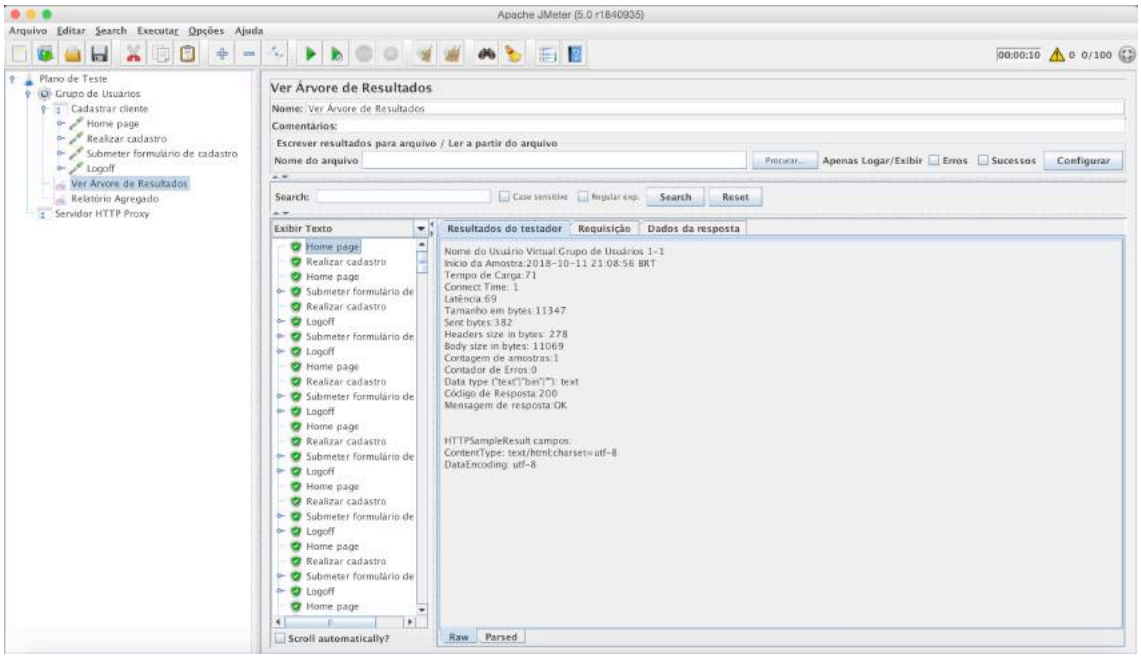
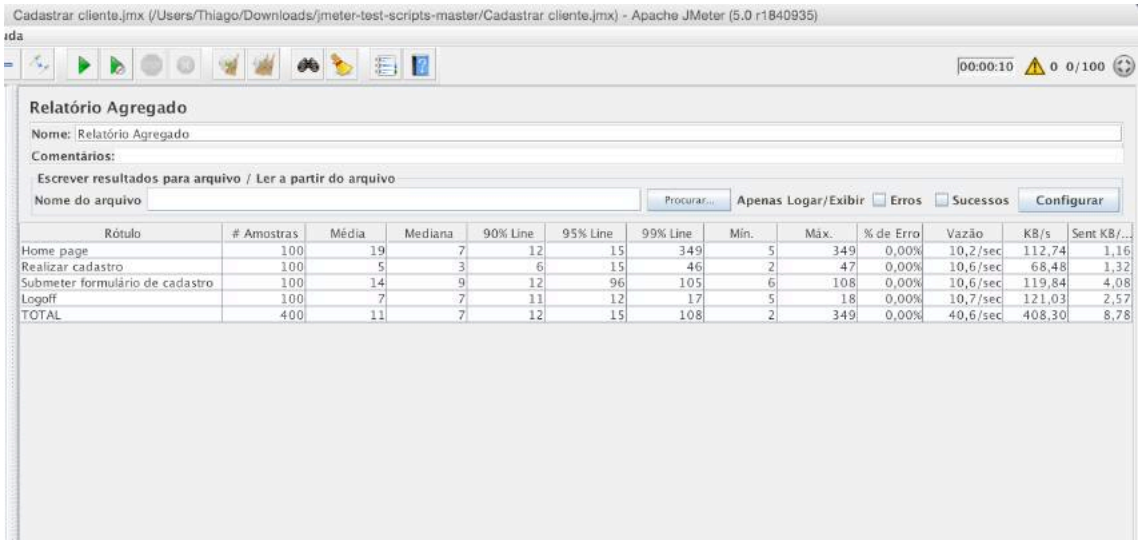


Figura 5.25. Resultados da execução no ouvinte "Ver Árvore de Resultados".

Já o ouvinte "Relatório Agregado", representado na Figura 5.26, traz informações suficientes para determinar se o cenário de teste CT1 passou ou falhou. Neste cenário, o tempo de resposta tolerado para uma carga de 100 usuários em uma rampa de 10s é de 0,5s. Recomenda-se, não utilizar a média como critério para aceitação de requisito de desempenho, tendo em vista que se trata de uma medida estatística muito sensível a valores extremos (*outliers*). Dependendo do nível de criticidade do sistema, recomenda-se utilizar o tempo de resposta nas linhas de 90%, 95% ou 99%. Para um sistema típico de comércio eletrônico a linha de 90% parece ser a mais adequada. Essa linha indica o tempo de resposta no 90º. percentil, isto é, organizadas as 100 respostas em função do tempo de resposta, a linha de 90% corresponde ao tempo de resposta da 90ª. resposta mais lenta.



Rótulo	# Amostras	Média	Mediana	90% Line	95% Line	99% Line	Mín.	Máx.	% de Erro	Vazão	KB/s	Sent KB/...
Home page	100	19	7	12	15	349	5	349	0,00%	10,2/sec	112,74	1,16
Realizar cadastro	100	5	3	6	15	46	2	47	0,00%	10,6/sec	68,48	1,32
Submeter formulário de cadastro	100	14	9	12	96	105	6	108	0,00%	10,6/sec	119,84	4,08
Logoff	100	7	7	11	12	17	5	18	0,00%	10,7/sec	121,03	2,57
TOTAL	400	11	7	12	15	108	2	349	0,00%	40,6/sec	408,30	8,78

Figura 5.26. Resultados da execução no ouvinte "Relatório Agregado".

Portanto, considerando que o tempo total na linha de 90% indicado no Relatório Agregado foi de 12ms, o requisito de desempenho foi satisfeito. No entanto, vale lembrar que neste exemplo o sistema Digital Toys está hospedado na mesma máquina que dispara os testes, o que certamente afeta os resultados, já que o ambiente real seria composto por redes de computadores e inúmeros dispositivos que contribuiriam para aumentar os tempos de resposta.

5.6. Recomendações Práticas

Esta seção apresenta um conjunto de recomendações que servem para orientar a realização de testes de desempenho. Esse conjunto de recomendações reflete a experiência do autor com esse tipo de teste, não representando, portanto, uma lista exaustiva de recomendações.

5.6.1. Conheça os conceitos relacionados a testes de desempenho

Conhecer os conceitos relacionados a testes de desempenho é fundamental para a realização de um bom trabalho. Além disso, a ausência de uma terminologia comum pode afetar negativamente a comunicação da equipe do projeto.

5.6.2. Conheça os requisitos de desempenho do sistema a ser testado

Leia o documento de requisitos não-funcionais (RNF) do projeto e converse com a equipe responsável. Procure identificar e compreender os requisitos de desempenho de modo que se tenham todas as informações necessárias para a realização do teste. Se for necessário, aplique o questionário a seguir, que pode ser complementado com outras perguntas que forem pertinentes.

1) Quais são os requisitos de desempenho descritos no documento de RNF?
2) Resumidamente, como é a arquitetura da aplicação a ser testada?
3) Quais são os casos de uso arquiteturalmente significativos?
4) Para cada caso de uso arquiteturalmente significativo, quais são suas expectativas de:
    a) quantidade de usuários concorrentes em horários de pico?
    b) tempo de resposta aceitável, considerando o pico de usuários concorrentes?
5) Considerando as respostas da pergunta 4, quais cenários de teste se mostram relevantes?
6) Qual a massa de teste necessária para cada cenário de teste escolhido? Como ela pode ser gerada?
7) Há integrações com outros sistemas nos cenários de teste escolhidos? Quais sistemas?
8) Que ambiente será utilizado nos testes de desempenho? Caso não seja o ambiente de produção, o ambiente escolhido é semelhante ao de produção em termos de hardware e software?

9) Há restrições de horários de execução dos testes, considerando o ambiente de testes escolhido?

10) Há testes funcionais automatizados?

5.6.3. Planeje e documente o projeto de testes de desempenho

Testes de desempenho costumam demandar muitos recursos e podem impactar diversos sistemas. Sendo assim, é fundamental que esses testes sejam planejados, de modo a otimizar o uso dos recursos e a evitar o retrabalho ("queimar cartuchos"). Além disso, tratam-se de testes que serão executados diversas vezes, eventualmente, por profissionais diferentes. É preciso garantir o mesmo estado do sistema e de variáveis de contexto a cada execução. Desta forma, é necessário que todos os procedimentos de execução sejam documentados para garantir a repetibilidade dos testes e, consequentemente, a consistência dos resultados. Portanto, documente as informações referentes ao planejamento dos testes de desempenho em um Plano de Testes.

5.6.4. Defina um modelo de carga

No contexto de testes de desempenho, o modelo de carga (*workload model*) é um dos itens mais importantes do Plano de Testes. Um modelo de carga descreve como um sistema será usado no ambiente de produção em termos de distribuição da carga de trabalho [Molyneaux 2014]. Para obter resultados confiáveis, o teste de desempenho deve se basear em um modelo de carga que retrate como o sistema será utilizado, levando-se em conta a quantidade de acessos concorrentes, o volume de dados transmitidos, os tempos de resposta toleráveis, a expectativa de consumo de recursos, etc. A Tabela 5.3, apresentada anteriormente, representa um exemplo de modelo de carga de um sistema que deve suportar 1.000 usuários concorrentes realizando diferentes transações.

5.6.5. Represente a arquitetura física do sistema

Observe o sistema sob teste como um encanamento. A bitola dos canos é equivalente à largura de banda da rede, enquanto que a vazão da água é equivalente ao *throughput* do sistema. A representação do sistema mais próxima de um encanamento é o modelo da arquitetura física, como ilustra a Figura 5.27, onde estão representados os servidores, bases de dados, *links* e equipamentos de rede, entre outros elementos que formam a infraestrutura do ambiente do sistema. Esse tipo de representação pode orientar o planejamento de casos de teste mais significativos. Além disso, esse modelo pode apoiar a descoberta de gargalos durante a execução de testes de desempenho.

Modelos como esse costumam estar presentes em documentos de arquitetura do software. Caso não haja, recomenda-se que sejam representados, com o apoio dos arquitetos do projeto, e incluídos no Plano de Testes.

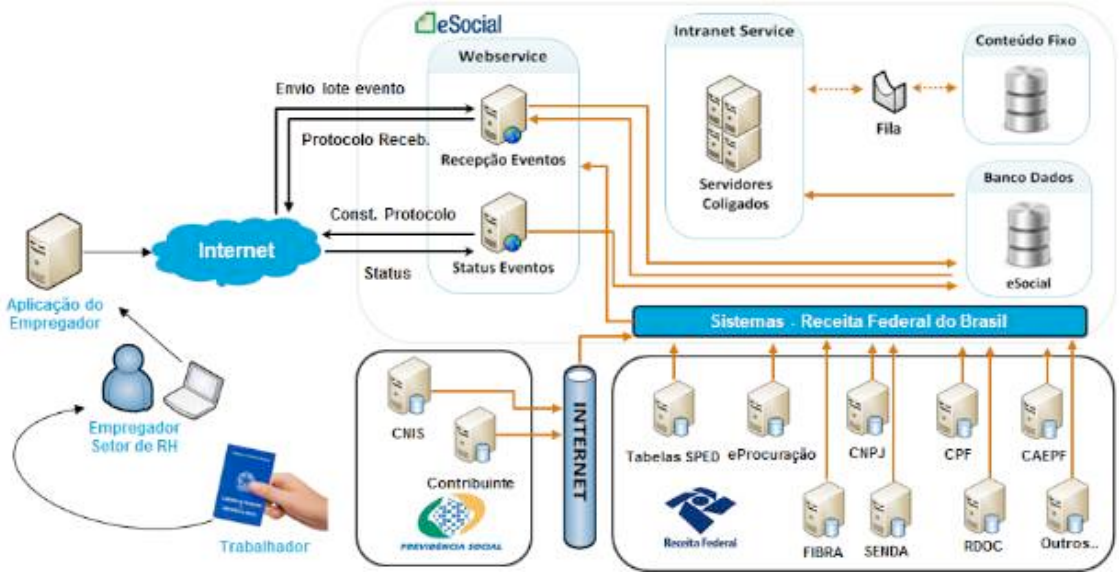


Figura 5.27. Exemplo de arquitetura física de um sistema [Brasil 2018].

5.6.6. Projete testes realistas

Os resultados do teste de desempenho serão mais precisos se for possível simular os cenários de uso do sistema de forma realista, preferencialmente em ambiente de produção. Testes não-realistas irão produzir resultados inúteis e que poderão levar a conclusões equivocadas.

Há pelo menos três elementos determinantes para a definição de casos de teste de desempenho realistas: 1) a quantidade de usuários/transações concorrentes, 2) a relação rampa de subida/número de agentes e 3) o *think time*. Veja os exemplos a seguir:

- Em um sistema com 500 usuários cadastrados e pico de acesso em torno de 150 usuários concorrentes, não seria realista um teste com mil usuários concorrentes. Portanto, a quantidade de usuários cadastrados, ou que se espera ter cadastrados, pode ser usada como parâmetro para delimitar a quantidade de usuários realizando acessos concorrentes em um cenário de teste.
- Um teste com 1.000 *threads* e uma rampa de subida de 100 segundos, quando disparado por apenas um agente, resulta em 10 requisições submetidas sequencialmente ao ambiente-alvo a cada segundo. Mas se o mesmo teste for executado a partir de vinte agentes (50 *threads* em cada), teoricamente, o ambiente-alvo poderá ficar 2s inteiros sem receber requisições, bem como, receber 20 requisições na mesma fração de segundo, provocando um cenário irreal de uso de recursos. Sendo assim, é necessário equilibrar adequadamente a relação entre a rampa de subida e o número de agentes, mesmo que para isso seja necessário executar diversos testes a título de calibração, até encontrar a medida ideal.
- Em cenários de teste onde o usuário navega por diversas páginas e, consequentemente, submete várias requisições ao servidor, é necessário simular o tempo em que ele passa interagindo com a aplicação. Esse tempo é conhecido como *think time*, o qual representa o tempo em que o usuário pensa e toma uma

decisão de interação com o sistema (p. ex., confirmar ou cancelar uma operação). Quando não se considera o *think time*, a execução dos testes de desempenho se torna não-realista, uma vez que as requisições de cada usuário virtual serão executadas imediatamente uma após a outra, o que não aconteceria em um cenário real.

5.6.7. Mantenha os *scripts* e demais artefatos sob controle de versão

Assim como qualquer outro artefato do projeto, os *scripts* e demais artefatos referentes a testes de desempenho devem ser mantidos sob controle de versão, tendo em vista que eles podem ser implementados de forma iterativa e colaborativa. Além disso, pode ser necessário reexecutar um cenário de teste de desempenho que foi executado há anos, para avaliar o impacto de uma manutenção recente em um sistema.

5.6.8. Execute os testes sobre um ambiente-alvo específico para testes de desempenho

Em um cenário ideal, os testes de desempenho seriam realizados sobre o ambiente de produção, mesmo que durante a noite ou fins de semana, pelo fato de proporcionar os resultados mais realistas possíveis. No entanto, são raros os casos em que esse cenário é viável. Também não se recomenda utilizar outros ambientes, como o de desenvolvimento, de testes funcionais ou de homologação, para realizar testes de desempenho por conta dos seguintes motivos:

- Geralmente, os ambientes de desenvolvimento, testes e homologação não refletem completamente as características de hardware e de software do ambiente de produção. É provável que processadores, memória, discos, *links* de rede, *firewalls*, a configuração de programas e o tamanho do bancos de dados sejam diferentes do ambiente de produção. Logo, não se pode concluir que um sistema apresentará tempos de resposta iguais em ambientes tão distintos.
- Quando se realizam testes de desempenho em ambientes compartilhados, o resultado do teste pode sofrer interferências das transações concorrentes que estavam sendo realizadas pelos desenvolvedores, testadores e usuários comuns. Da mesma forma, os testes de desempenho podem afetar o trabalho desses outros usuários, fornecendo respostas mais lentas ou, até mesmo, provocando a interrupção do sistema.

Portanto, recomenda-se o estabelecimento de um ambiente específico para testes de desempenho, criado à semelhança do ambiente de produção.

5.6.9. Execute os testes a partir de um ambiente de ativação específico para testes de desempenho

Em qualquer projeto de teste de software recomenda-se ter um ambiente específico de hardware e software para realização dos testes. Em testes de desempenho, particularmente, recomenda-se que a execução dos testes seja realizada com apoio de um ambiente de ativação dedicado a testes de desempenho, tendo em vista que um ambiente de ativação com hardware e *link* de rede compartilhados poderia afetar os resultados dos testes (por exemplo, provocando maior latência).

A Figura 5.28, que ilustra um cenário composto por três ambientes distintos: 1) um ambiente de implementação, no qual o profissional de teste desenvolve os *scripts* de teste de desempenho; 2) um ambiente de ativação, composto por diversos agentes, configurado para disparar *jobs* da ferramenta Jenkins que executam *scripts* do JMeter; 3) por fim, um ambiente-alvo, composto por duas camadas físicas (*tiers*), representando o sistema que recebe as requisições provenientes do ambiente de ativação.

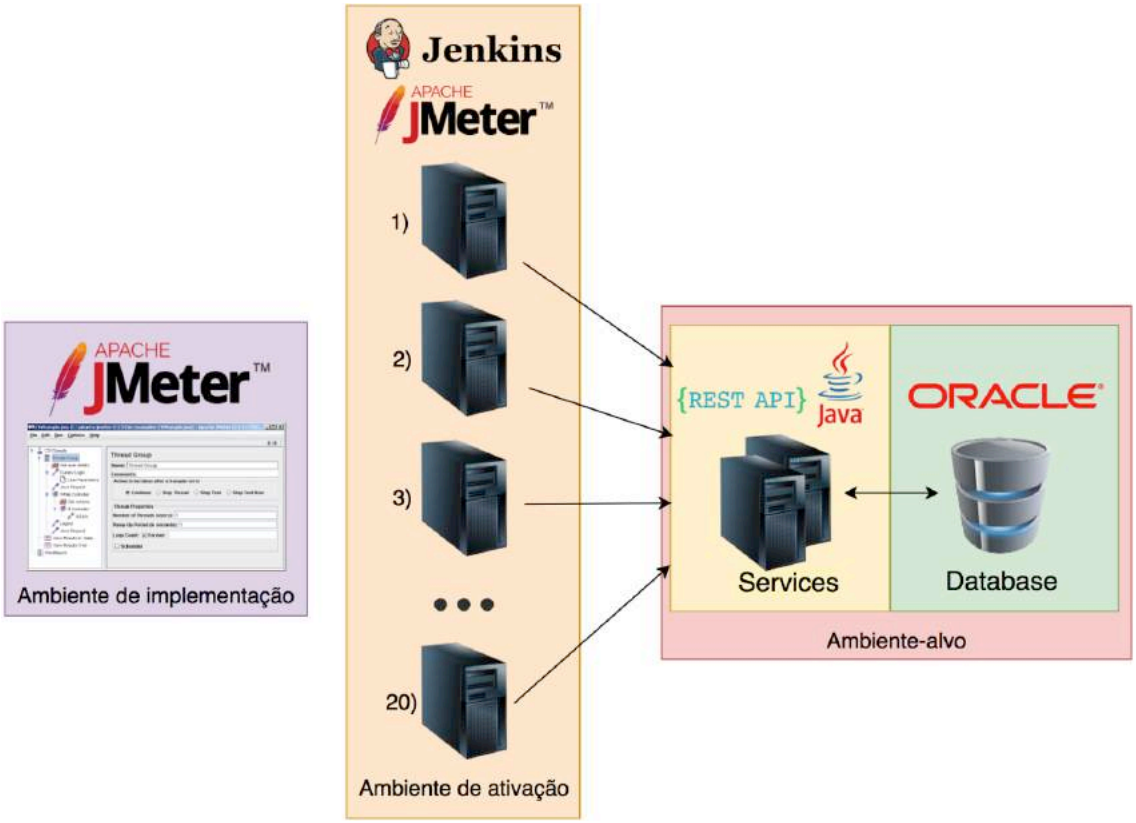


Figura 5.28. Diferentes ambientes utilizados em testes de desempenho.

5.6.10. Não utilize *listeners* no ambiente de ativação

O JMeter possui um conjunto de *listeners* (“ouvintes”) que exibem dados das requisições e das respostas. Em geral, esses *listeners* consomem uma quantidade significativa de memória da máquina que dispara os testes (especialmente os *listeners* “Ver Árvore de Resultados” e “Gráfico de Resultados”. Os *listeners* são muito úteis em tempo de implementação dos *scripts*, porém devem ser evitados nos *scripts* que serão executados pelo ambiente de ativação.

5.6.11. Execute os testes várias vezes e em diferentes horários

Uma única execução de um *script* de teste de desempenho pode não ser conclusiva. A cada execução, os testes podem apresentar tempos de resposta diferentes. Eventualmente, algumas execuções do mesmo cenário de teste apresentam tempos de resposta satisfatórios e outras não. Isso se deve a inúmeros fatores, tais como a latência da rede, esgotamento temporário de recursos, entre outros. Sendo assim, recomenda-se que um mesmo cenário de teste seja executado ao menos três vezes e em diferentes

horários para evitar que fatores momentâneos interfiram nos resultados de todas as execuções do teste.

5.6.12. Acompanhe a execução dos testes com uma ferramenta de monitoramento

O valor agregado ao projeto pelos testes de desempenho é potencializado quando o time realiza a monitoração dos servidores do ambiente-alvo, com o objetivo de identificar os gargalos. Portanto, recomenda-se que toda execução de testes de desempenho seja monitorada via ferramentas como o *Application Performance Management* (APM) [CA Technologies 2018], representado na Figura 5.29.

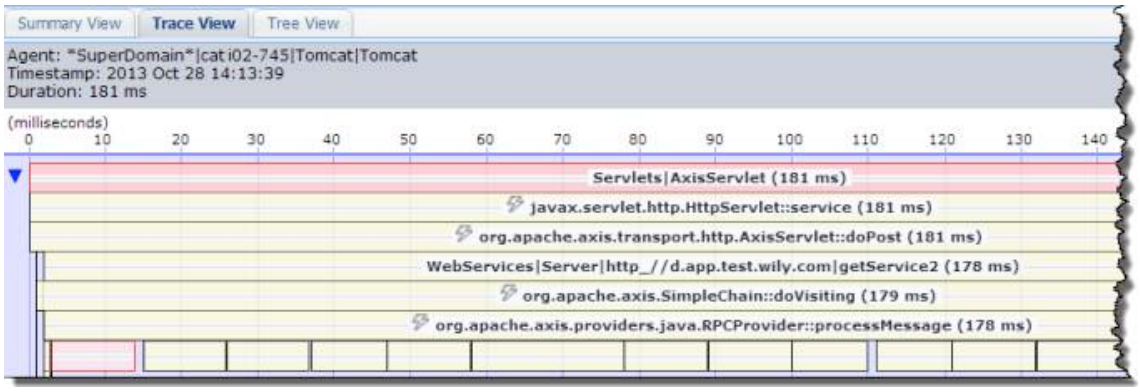


Figura 5.29. Visão da ferramenta APM.

5.6.13. Execute os testes cedo e com frequência

Tradicionalmente, os testes de desempenho são realizados quando o sistema está próximo de ser homologado ou colocado em produção (abordagem reativa). Isso se deve ao fato de que em um cenário ideal esse tipo de teste deva ser realizado em um produto completo e livre de defeitos (funcionais). No entanto, considerando a frequência de entregas nos novos projetos e a decomposição dos sistemas em partes cada vez menores (microsserviços), adotar uma abordagem proativa, antecipando esse tipo de teste, pode trazer uma série de benefícios, mesmo que não se tenha uma percepção definitiva do desempenho do sistema. Entre os benefícios observados, temos os seguintes:

- definição antecipada dos critérios de aceitação de desempenho;
- identificação dos riscos associados ao desempenho;
- descoberta antecipada de gargalos.

5.6.14. Comunique os resultados visualmente

Os resultados dos testes de desempenho são de interesse de toda a equipe. Portanto, comunique seus resultados a todos os interessados, de preferência, de forma visual. Uma sugestão é utilizar a ferramenta Grafana [Grafana Labs 2018], que possibilita a criação de *dashboards*, como representa a Figura 5.30.

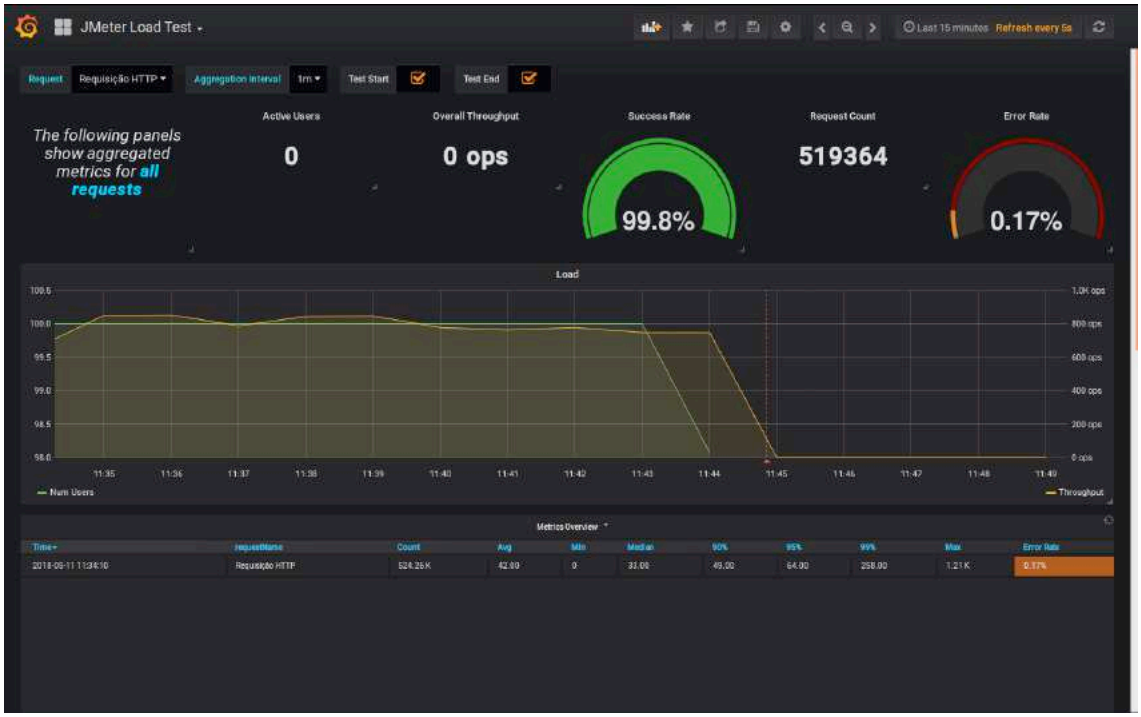


Figura 5.30. *Dashboard* do Grafana.

1.7. Conclusão

Este trabalho apresentou aspectos teóricos e práticos a respeito de testes de desempenho de software. Quanto aos aspectos teóricos, mereceu destaque uma classificação que apresenta nove tipos de teste de desempenho tipicamente praticados na indústria. Essa classificação tem especial importância pelo fato de que não há consenso estabelecido na área a esse respeito, podendo levar a falhas de comunicação em projetos e a consequências sérias.

Já os aspectos práticos foram apresentados por meio de um exemplo prático de uso da ferramenta Apache JMeter e de um conjunto de recomendações práticas sobre testes de desempenho. O exemplo prático, apesar de simples, pode ser suficiente para um testador sair da "estaca zero" de conhecimento sobre a ferramenta JMeter. As recomendações prática apresentadas representam a experiência do autor com testes dessa natureza ao longo dos últimos três anos em projetos Web de larga escala. Provavelmente, outras recomendações podem ser adicionadas ao conjunto apresentado, especialmente em outros contextos de projeto de software.

Em relação a trabalhos futuros, especialmente para alunos de pós-graduação em Computação, sugere-se a criação de um método para apoiar a definição de modelos de carga, em especial, a definição da relação entre usuários concorrentes, rampa de subida e número de agentes. Geralmente, essa tarefa é realizada por tentativa e erro, como uma espécie de calibração. Um método que indicasse a quantidade ideal de cada um desses três parâmetros para cada cenário de teste seria útil para a maioria das organizações que realizam esse tipo de teste.

Referências Bibliográficas

Apache Foundation (2018) *Apache JMeter 5*. Disponível em: <https://jmeter.apache.org>. Acesso em: agosto, 2018.

Brasil (2018) *Sistema eSocial: Manual de Orientação do Desenvolvedor*. Disponível em: <http://portal.esocial.gov.br/manuais/manualorientacaodesenvolvedoresocialv1-7.pdf>. Acesso em: setembro, 2018.

Bondi, A. B. (2014). *Foundations of software and system performance engineering: process, performance modeling, requirements, testing, scalability, and practice*. Pearson Education.

CA Technologies (2018) *Application Performance Management (APM)*. Disponível em: <https://www.ca.com/br/products/ca-application-performance-management.html>. Acesso em: setembro, 2018.

Erinle, B. (2017). *Performance Testing with JMeter 3*. Packt Publishing Ltd.

Erinle, B. (2014). *JMeter Cookbook*. Packt Publishing Ltd.

Grafana Labs (2018) *Grafana 5.1.0*. Disponível em: <https://grafana.com>. Acesso em: agosto, 2018.

ISO/IEC. (2011) *ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models*. International Organization for Standardization. Geneva.

ISO/IEC/IEEE. (2013) *ISO/IEC/IEEE 29119-1 - Software and systems engineering - Software testing - Part 1: Concepts and definitions*. International Organization for Standardization. Geneva.

ISO/IEC/IEEE. (2013) *ISO/IEC/IEEE 29119-2 - Software and systems engineering - Software testing - Part 2: Test process*. International Organization for Standardization. Geneva.

Matam, S. and Jain, J. (2017). *Pro Apache JMeter: Web Application Performance Testing*. Apress.

Molyneaux, I. (2014). *The Art of Application Performance Testing: From Strategy to Tools*. O'Reilly Media, Inc.

Autor

Thiago Silva de Souza é Doutor em Engenharia de Sistemas e Computação pela COPPE/UFRJ, Mestre em Informática pelo PPGI/UFRJ, Especialista em Gerência e Desenvolvimento de Sistemas Distribuídos pelo NCE/UFRJ e Bacharel em Sistemas de Informação pela Universidade do Grande Rio (UNIGRANRIO). Possui diversas certificações internacionais, especialmente nas áreas de Teste de Software e Métodos Ágeis. Atualmente é Analista de Sistemas no Serviço Federal de Processamento de Dados (SERPRO) e Professor Adjunto na UNIGRANRIO. Seus interesses de pesquisa incluem Engenharia de Software Experimental, Teste de Software, Tecnologia de Web Services e Internet das Coisas (IoT). Seu e-mail para contato é profthiagodesouza@gmail.com.