**Pílula do conhecimento: Área de Performance**

O que define Performance, em pontos-chave, é o desempenho e o rendimento. Seria como extrair o potencial máximo de algo, desde a resolução dos problemas apresentados por ele até o polimento do mesmo. Nota-se a semelhança entre as palavras “Performance” e “Perfeição”, demonstrando que esse conceito é o mesmo que a busca por melhorias até atingir um outro patamar. Mas como esse conceito é aplicado em nossa área?

Em tecnologia, performance é uma área que aborda esses mesmos pontos como desempenho e melhorias, porém para sistemas, aplicativos, funcionalidades e entre outros temas tecnológicos. Para esses objetivos a área é responsável por práticas, metodologias e ferramentas para assegurar a estabilidade de uma aplicação, assim tornando crucial para qualquer sistema já que a confiabilidade na infraestrutura impacta a experiência do usuário.

Administrar um sistema de larga escala pode parecer uma tarefa difícil, porém com ferramentas, planejamentos e principalmente **testes de performance**, o trabalho de assegurar a estabilidade pode ser checado antes mesmo de chegar a um usuário. Os **testes de performance** não só a prática da área mais evidente, como é a principal forma de avaliar a velocidade, estabilidade e escalabilidade de um sistema, assim como identificar gargalos e limitações. Para realizar um teste de performance existem algumas etapas.

**Etapas de um Teste de Performance:**

**Requisitos Não-Funcionais:**

A primeira e mais crucial etapa na área de performance é o acordo dos **requisitos não-funcionais**. Esses requisitos são a respeito de exigências como desempenho, confiabilidade, manutenção, segurança e entre outros. Neles são definidas várias informações-chave, principalmente para a estratégia e o monitoramento de um **teste de performance**.

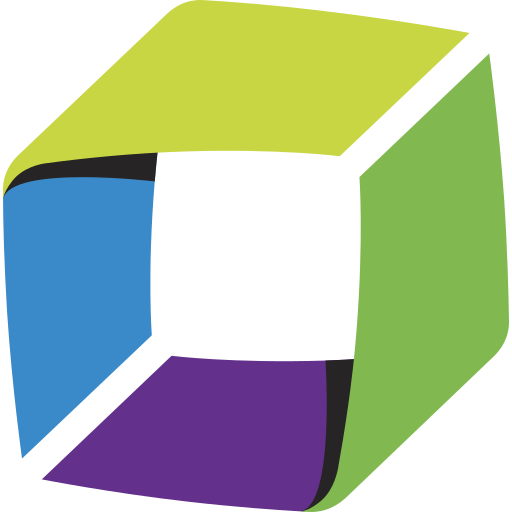
É importante ressaltar que esses mesmos requisitos são acertados no padrão de qualidade e nas necessidades do cliente, ou seja, exigências apontadas pela equipe que solicitou o **teste de performance**. A parte disso, cada equipe ou projeto podem possuir requisitos diferentes, porém há alguns que são extremamente frequente e essenciais.

Um desses requisitos principais é por exemplo é a compreensão dos objetivos de um sistema, seus problemas de performance, sua situação e o mais importante: seus **prazos de entrega**. Informações sobre a infraestrutura e ambiente também são essenciais para a construção de um script para o teste. Coleções ou documentações de ferramentas como *Postman* e *Swagger* são as principais nesse quesito

Antes do teste também é essencial o entendimento da **volumetria**. Dados como requisições por segundo do sistema, tempo de resposta, margem de erros e entre outros. Alguns testes necessitam de uma massa de dados em seus fluxos, como exemplo uma lista de CPFs cadastrados.

A disponibilidade de um sistema de monitoração para acompanhar a aplicação durante o teste pode auxiliar na análise, assim podendo observar como o sistema se comporta internamente com altas cargas de usuários simultâneos. Ferramentas como *Dynatrace* e *Datadog* são referências nesse aspecto.



**Automação**:

Para realizar um teste de performance é necessário a criação de um script para realizar o fluxo do sistema. Nesse sentido há muitas semelhanças com a área de qualidade, então qual seria a diferença desse tipo de script em QA e performance?

Começando pelos pontos em comum, ambas as áreas necessitam de um script funcional, executando perfeitamente o fluxo como um usuário real, porém esse o único ponto igual. Em qualidade, o que é checado é se o fluxo está correto e atendendo as expectativas da aplicação.

Em performance o script, mesmo inicialmente sendo funcional, será utilizado para “estressar” o fluxo do sistema. Esse “estresse” é realizado utilizando várias cargas de usuários diferentes, assim encontrando gargalos e, através de monitoramento e análises, apontar falhas e possibilidades de melhorias da aplicação.

Uma das modalidades para checar se um script está completamente funcional para os futuros testes de performance é o ***Smoke Test***. Trata-se de um pequeno teste com uma quantidade mínima de carga (usuários simulados), checando se o usuário virtual está realizando com perfeitamente o fluxo como solicitado. A carga pode variar de um a dez usuários, sendo uma quantia para não estressar a aplicação no processo de automação do script.

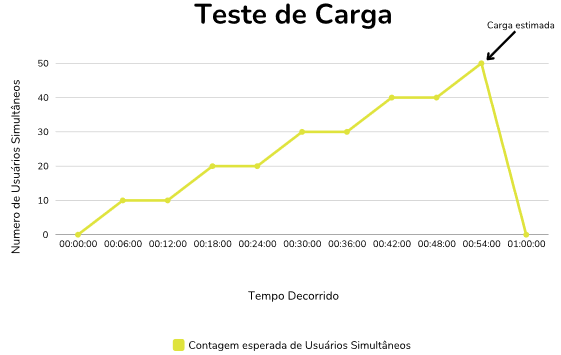
Os scripts para os testes são frequentemente criados utilizando a ferramenta: JMeter



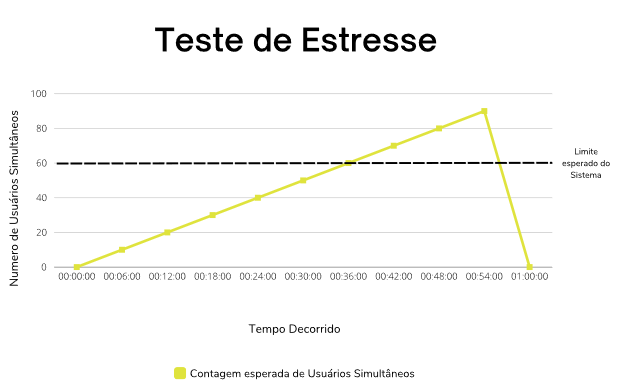
**Testes de performance**:

Uma vez com o script funcional e também checado pelo ***Smoke Test***, os testes de performance podem ser realizados. Nesses tipos de teste será feito várias cargas de usuários, a variação dessas cargas depende da estratégia planejado, ou seja, existem vários tipos de teste de performance.

Por exemplo um dos mais frequentes em projetos é o **teste de carga**, onde o sistema é exposto e observado sobre um aumento gradual de cargas, assim conseguindo analisar a capacidade com o tráfego simultâneo, além de gargalos e limitações como aumento no tempo de resposta e quantidade de erros, diminuição de Vazão e entre outros.



Um outro tipo de teste de performance muito requisitado é o **teste de estresse**, onde o objetivo é observar como o sistema se comporta contra picos de cargas, chegando ou ultrapassando dos limites de cenários reais. Nesse teste também é analisado como o sistema lida com disponibilidade reduzida de recursos, memória e processamento e entre outros



**Monitoramento**:

Como mencionado anteriormente o uso de ferramentas de monitoramento é muito importante para análise do teste. Mas já tendo tecnologias de performance como o *JMeter*,porquê dessa importância em plataformas como *Dynatrace* e *Datadog*?

Um dos motivos principais são os “lados” em um teste. Pelo *JMeter* será simulado cargas de usuários virtuais, onde além de estressarem o sistema vão coletar informações como o tempo de resposta, vazão, erros e entre outros. Porém essas informações são do lado de um cliente, logo não há uma forma de ver informações e processos internos da aplicação.

Com o uso de plataformas de monitoramento, além da coleta de informações pelo *JMeter* há como a aplicação se comporta nos mínimos detalhes durante o teste. Assim, vários dados da infraestrutura se tornam fontes ricas de dados para a construção de relatórios e análises. Mas quais seriam essas informações?

Existem métricas fundamentais, uma delas é o tempo resposta, onde é medido o tempo que um sistema leva para responder a solicitação de usuários. Um ponto crucial para checar a qualidade de uma aplicação, já que além de refletir a eficiência é um dos fatores que mais impacta a experiência de um usuário.

Outras métricas importantes, além de conceitos básicos da área como vazão e percentis, são a análise dos erros. Com o *JMeter*, assim como a experiência do usuário é possível só ver a resposta da requisição quando o erro ocorre. Porém com monitoramento, o fluxo da requisição e o erro são expostos para poderem ser investigados, aprimorando a infraestrutura do sistema.

**Relatórios e Otimização**:

Pós testes e análises há a criação de um relatório. Além de um documento também pode ter uma apresentação sobre o material criado. Nesse relatório há uma explicação detalhada de como a aplicação se comportou durante os testes.

Seria apresentada informações como vazão, tempo de respostas, porcentagens de erros e entre outros. Porém com um diferencial, tendo uma linha do tempo durante a execução do teste. Assim é possível por exemplo ver como o sistema se comportou em cargas e horários específicos.

Dessa forma há uma clareza em entender os gargalos e os períodos que eles ocorrem, assim como os cenários e ambientes onde eles apresentam suas limitações. Quando um erro ocorre, com o auxilio de uma plataforma de monitoramento, ele pode ser detalhado no relatório desde da sua requisição a nível de código onde se comprometeu.

Um dos pontos positivos dos relatórios é conseguir também observar como os recursos estão sendo gerenciados em certos períodos, por exemplo em um gargalo, qual componente está em seu limite? Seria a memória ou CPU? Informações como essas podem ser expressas no documento, inclusive com que carga e qual cenário elas ocorrem.

O oposto nesse caso também é real, em situações de alta carga existe algum componente que não está sendo completamente usado? E se sim o que fazer? O relatório pode não expor melhorias para aperfeiçoar o uso completo de um componente, como pode relatar que sua alocação é muito maior que o necessário, assim podendo impactar positivamente no orçamento de um sistema.