Universidade Federal do Rio Grande do Norte
Instituto Metrópole Digital
Programa de Pós-graduação em Engenharia de Software
Mestrado Profissional em Engenharia de Software

Avaliação Arquitetural do Sistema SUAP: uma análise sistematizada sobre desempenho

Tarso Latorraca Casadei

Natal-RN
Dezembro de 2018

Tarso Latorraca Casadei

Avaliação Arquitetural do Sistema SUAP: uma análise sistematizada sobre desempenho

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Software da Universidade Federal do Rio Grande do Norte, como requisito parcial para a obtenção do grau de Mestre em Engenharia de Software.

Universidade Federal do Rio Grande do Norte – UFRN

Instituto Metrópole Digital – IMD

Programa de Pós-Graduação em Engenharia de Software

Orientador: Prof. Carlos Eduardo da Silva, PhD

Natal/RN

2018

Universidade Federal do Rio Grande do Norte - UFRN
Sistema de Bibliotecas - SISBI
Catalogação de Publicação na Fonte. UFRN - Biblioteca Central Zila Mamede

Casadei, Tarso Latorraca.
Avaliação arquitetural do Sistema SUAP: uma análise sistematizada sobre desempenho / Tarso Latorraca Casadei. - 2018.
81 f.: il.

Dissertação (mestrado) - Universidade Federal do Rio Grande do Norte, Instituto Metrópole Digital, Programa de Pós-Graduação em Engenharia de Software. Natal, RN, 2019.
Orientador: Prof. Dr. Carlos Eduardo da Silva.

1. Escalabilidade - Dissertação. 2. Arquitetura de software - Dissertação. 3. Avaliação arquitetural - Dissertação. 4. ATAM - Dissertação. 5. Análise de desempenho - Dissertação. 6. Testes de desempenho - Dissertação. I. Silva, Carlos Eduardo da. II. Título.

RN/UF/BCZM CDU 004.7

Elaborado por Ana Cristina Cavalcanti Tinôco - CRB-15/262

Tarso Latorraca Casadei

Avaliação Arquitetural do Sistema SUAP: uma análise sistematizada sobre desempenho

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Software da Universidade Federal do Rio Grande do Norte, como requisito parcial para a obtenção do grau de Mestre em Engenharia de Software.

Trabalho aprovado. Natal/RN, 07 de dezembro de 2018:

Prof. Carlos Eduardo da Silva, PhD
Orientador

Eiji Adachi Medeiros Barbosa, Dr.
Examinador Interno

André Gustavo Duarte de Almeida, Dr.
Examinador Externo

Natal/RN
2018

Este trabalho é dedicado à minha família e a todos que colaboraram com esta conquista. Aos que creem e perseveram.

Agradecimentos

Agradeço especialmente à minha família pelo estímulo e paciência durante toda essa jornada. Cada momento longe de vocês foi duro, mas necessário para o desenvolvimento deste trabalho. Meus filhos, espero que um dia possam entender e cumprir essa etapa na vida de vocês. Terão todo o meu amor e apoio para isso. Nêga, obrigado por cada momento de compreensão e estímulo que compartilhamos durante este cansativo período. Noites não dormidas, sábados na biblioteca, feriados de estudo... Tudo finalmente valeu a pena.

Mãe, Giba e Sofia, por entenderam minha dedicação a este projeto em um momento que vocês precisam de mais apoio do que eu. Às minhas irmãs Dani e Bibi pelo amor incondicional. Pai e Vô (em memória); e Vó, por desde sempre me incentivarem o estudo.

Kadu, por ter encarado um desafio que parecia perdido. A todas as críticas, compreensão e confiança que você dedicou nessa parceria. Obrigado por ter investido neste projeto, que fez de você não só um orientador, mas um amigo.

Aos meus amigos e colegas de IFRN por compartilharem desse momento especial. Em especial Welkson e Allyson que tanto contribuíram desde a concepção deste trabalho. Não tenho palavras para agradecer todo o apoio de vocês. Cadu (Egito) pelas discussões e participações ao longo deste trabalho.

Prof. André Gustavo, por tanto me incentivar a entrar no mestrado e pelas contribuições propostas desde o pré-projeto que submeti ao programa. Eiji e Uirá, pela cooperação ao longo de todo esse trajeto.

Por fim, a todos que contribuíram de alguma forma neste projeto e nesta fração da minha vida. Foram dois anos e meio de muito aprendizado...

"...man would not have achieved the possible unless time and again he had reached out for the impossible."

(Max Weber)

Resumo

Sistemas de informação têm se tornado cada vez mais robustos, trabalhando com altos volumes de dados, objetos e, consequentemente, processos. No contexto do Instituto Federal do Rio Grande do Norte (IFRN), foi desenvolvido um sistema próprio para atender a todas as atividades administrativas e acadêmicas da instituição: o SUAP, Sistema Unificado de Administração Pública. Ao longo de 11 anos, a aplicação evoluiu de apenas um módulo para mais de 40, além do crescimento vertiginoso de usuários – que hoje permeia o número de 35 mil. Esses aspectos muitas vezes impactam diretamente no desempenho de sistemas, levando os arquitetos de software a buscarem alternativas para melhor escalar suas aplicações. No caso do IFRN, mesmo após altos investimentos realizados em infraestrutura, os problemas de desempenho do SUAP persistem. Buscando um melhor aproveitamento da estrutura de hardware disponível, faz-se necessário compreender as razões destes problemas de desempenho, a fim de que estes recursos computacionais sejam utilizados de forma dinâmica e crescente, geridos de forma mais eficiente e capazes de suportar cargas excedentes sem prejuízo de desempenho da aplicação. Surge, assim, a necessidade de se avaliar aspectos mais abrangentes da arquitetura atual da aplicação, para que possam ser conhecidos elementos que prejudiquem seu desempenho. Este trabalho tem o objetivo de realizar uma análise sistematizada de desempenho do SUAP através da aplicação do método de avaliação arquitetural ATAM (*Architecture Trade-off Analysis Method*) e da execução de um conjunto de testes de desempenho que possibilitem a identificação de seus principais pontos de lentidão. Com isso, espera-se contribuir para a definição de uma estratégia para testes de desempenho que possam ser incorporados ao processo de desenvolvimento do SUAP.

Palavras-chave: escalabilidade; arquitetura de software; avaliação arquitetural; ATAM; análise de desempenho; testes de desempenho;

Architectural Evaluation of SUAP system: performance's systematized analysis.

Author: Tarso Latorraca Casadei
Supervisor: Carlos Eduardo da Silva, PhD.

Abstract

Information systems have become increasingly robust, working with high data volumes, objects and processes. In the context of the Federal Institute of Rio Grande do Norte (IFRN), a system was developed to attend all the administrative and academic activities of the institution: the SUAP, Unified System of Public Administration. Over the course of 11 years, the application has evolved from just one module to over 40, in addition to the dizzying growth of users – which today permeates the number of 35 thousand. These aspects often impact directly and negatively on systems performance, leading software architects to seek alternatives to scale better their applications. In the case of the IFRN, even after high investments in infrastructure, SUAP performance problems persist. In order to make better use of the available hardware structure, it is necessary to understand the reasons for these performance problems, so that these computational resources are used dynamically and incrementally, managed more efficiently and able to withstand surplus loads without prejudice to application performance. Thus, the need to evaluate more comprehensive aspects of the current architecture of the application is presented, so that elements that impair it's performance can be known. This work aims to perform a systemized SUAP's performance analysis through the application of the Architecture Trade-off Analysis Method (ATAM) and the execution of a set of performance tests that allow the identification of it's main points of slowness. With this, it is hoped to contribute to the definition of a strategy for performance tests that can be incorporated into the SUAP's development process.

Keywords: scalability; software architecture; architectural evaluation; ATAM; performance analysis; performance tests;

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AD	*Active Directory*
API	*Application Programming Interface*
ATAM	*Architecture Tradeoff Analysis Method*
COINRE	Coordenação de Infraestrutura e Redes
COSINF	Coordenação de Sistemas da Informação
DB	*Database*
DIGTI	Diretoria de Gestão de Tecnologia da Informação
HTTP	*Hypertext Transfer Protocol*
IFRN	Instituto Federal do Rio Grande do Norte
MEC	Ministério da Educação
PDI	Plano de Desenvolvimento Institucional
PROAD	Pró-Reitoria de Administração
SAAM	*Software Architecture Analysis Method*
SETEC	Secretaria de Educação Profissional e Tecnológica
SGBD	Sistema de Gerenciamento de Banco de Dados
SO	Sistema Operacional
SUAP	Sistema Unificado de Administração Pública
VM	*Virtual Machine* (ou máquina virtual)
VPN	*Virtual Private Network*

Sumário

1 Introdução

Em tecnologia da informação, tornou-se vital a capacidade de sistemas acomodarem um número cada vez maior de elementos ou objetos e processarem satisfatoriamente volumes crescentes de trabalho. Com isso, sistemas que não forem bem sustentados por sua infraestrutura podem apresentar problemas de desempenho e disponibilidade, dentre outros. Essas limitações podem deixar um produto sem competitividade comercial devido à sua ineficiência[1].

Para tentar dimensionar melhor os sistemas às crescentes demandas, uma das soluções mais comuns na literatura e no mercado é a de escalar as aplicações. Hill (1990) já demonstrava essa preocupação em seu estudo sobre o que seria escalabilidade, ainda no ano de 1990. Não escalar ou escalar mau um sistema pode resultar em perdas de desempenho, impactando direta e negativamente na qualidade do serviço que demandou tal software (BONDI, 2000). A partir de então, escalabilidade passou a ser considerado um atributo indispensável em um sistema, rede ou processo. Ao se criticar a escalabilidade de um produto de software, é necessário que avaliemos suas características e suas dificuldades, a fim de que possamos compreender todo o contexto daquela aplicação.

Tentando resolver problemas de escalabilidade, Bondi (2000) buscou entender a origem e os fatores que determinam a fraca capacidade de escalar de um software ainda em sua fase de concepção. Essa visão colaborou para o entendimento de que desempenho e escalabilidade estão fortemente ligados. Contudo, Duboc, Rosenblum e Wicks (2006) apresentaram uma visão de que escalabilidade era um termo amplamente utilizado em artigos científicos, revistas técnicas e descrições de software, contudo seu uso nos contextos mais variados contribuia para uma confusão geral sobre o seu real significado. Os autores ainda colaboraram com a ideia de que com a evolução e diminuição de custo dos hardwares, desenvolvedores passaram a não priorizar a eficiência algorítmica de suas aplicações, confiando em submeter seus projetos a máquinas cada vez mais potentes. Essa prática trouxe impactos diretos nas arquiteturas dos sistemas desde então.

Como consequência, o estudo da arquitetura de software surgiu como uma disciplina autônoma que exige conceitos, formalismos, métodos e ferramentas próprias (MUCCINI; BERTOLINO; INVERARDI, 2004). Para eles, esse estudo lida com a concepção de sistemas distribuídos complexos e abordam o problema de se conseguir escalar essas aplicações. Essa é tida como uma tarefa empírica, baseada na experiência do arquiteto de software, podendo ser feita mais facilmente através de uma classificação de observações do modelo

[1] Eficiência (rápido e ”enxuto”): é a capacidade do produto de software de prover desempenho apropriada, relativa ao conjunto de recursos usados quando o software está em uso dentro das condições especificadas, uma das métricas de qualidade de software, segundo a norma ISO/IEC:9126-1 (2003).

do software. Métodos semelhantes ao *Architecture Trade-off Analysis Method* (ATAM), no qual a informação arquitetônica é empiricamente capturada, podem colaborar nessa compreensão.

Desenvolvido por Kazman et al. (1998) e tido como uma maneira completa e abrangente de avaliar uma arquitetura de um software (BASS; CLEMENTS; KAZMAN, 2003), o ATAM analisa até que ponto a arquitetura do software satisfaz objetivos específicos de qualidade, como o de desempenho, por exemplo. Com sua aplicação, busca-se analisar sistematicamente a arquitetura de um software, analisando aspectos que possam gerar falhas de funcionamento, como mau desempenho.

No contexto do Instituto Federal do Rio Grande do Norte (IFRN), tem sido comum o Sistema Unificado de Administração Pública (SUAP) apresentar quedas de desempenho, sem uma origem conhecida dessas falhas. Com a recente expansão da rede federal de educação profissional e tecnológica, o IFRN, nos últimos dez anos, passou a contar com 19 novos *campi*, saltando de duas unidades de ensino (*campus* Natal-Central e *campus* Mossoró), para 21 – excluindo-se a Reitoria, que tem sede própria. Paralelamente ao crescimento físico, o IFRN teve uma extensa ampliação em seu quadro de servidores, o que demandou um sistema que atendesse às mais diversas tarefas administrativas da instituição. Com isso, a Diretoria de Gestão de Tecnologia da Informação (DIGTI) iniciou, no ano de 2007, o desenvolvimento de seu sistema próprio, o SUAP.

Na concepção do sistema, apenas um módulo inicial foi proposto: o de registro de ponto dos servidores da instituição. Ao longo desses 11 anos, o sistema sofreu uma evolução desse módulo único para mais de 40 – o SUAP hoje sustenta todos os principais processos e atividades do instituto, atendendo oito dimensões sistêmicas definidas: Administração, Assistência Estudantil, Ensino, Extensão, Gestão de Pessoas, Pesquisa, Planejamento e Tecnologia da Informação.

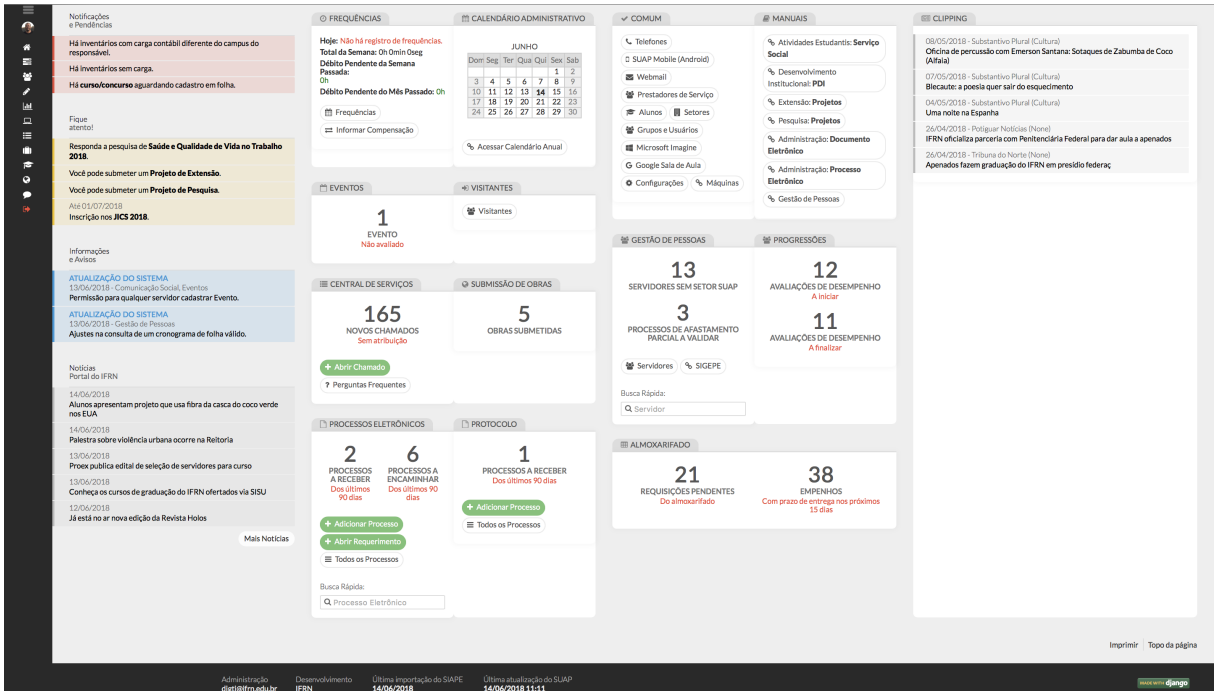
Visto isso, o SUAP hoje é um sistema vital ao IFRN, apoiando a realização de atividades administrativas, acadêmicas e financeiras, dentre outras. Como dito, o sistema também registrou crescimento substancial, tanto em questões de funcionalidades para atender novos requisitos de setores já existentes, bem como novas demandas decorrentes de novos setores do IFRN. Além disso, houve também um crescimento considerável no número de usuários do sistema, abrangendo as mais diversas atividades e setores do IFRN. Atualmente, esse sistema corporativo web atende aproximadamente 35 mil usuário apenas no âmbito do IFRN[2], contando com seus mais de 40 módulos.

[2] Além do IFRN, outros 24 Institutos Federais de ensino utilizam o SUAP atualmente.

1.1 Problemática e Motivação

No contexto do IFRN, quando o SUAP é submetido a uma grande quantidade de acessos, é percebida certa lentidão do sistema por usuários e desenvolvedores. Essa demora de resposta geralmente ocorre em períodos de pico de acesso ao sistema, como em lançamento de notas e fechamento dos diários dos professores, por exemplo. A própria página inicial do sistema (*index* da aplicação) tem apresentado problemas de desempenho, já que ela carrega consigo um grande número de *widgets*, que realizam consultas e requisições simultâneas a diversos módulos – a Figura 1 ilustra a página inicial do usuário, demonstrando o alto número de elementos que são carregados junto à requisição inicial.

Figura 1 – Página inicial do usuário



Fonte: Elaborada pelo autor.

Além disso, estima-se que a partir da disponibilização da API do SUAP[3], tenha havido um alto crescimento no número de requisições ao servidor. Sendo assim, além do crescente número de usuários do sistema, passou-se a ter de trabalhar com cargas de requisições oriundas de aplicações de terceiros, que geralmente são superior às que vêm direto de usuários.

A fim de melhorar o acesso dos usuários ao sistema, a administração optou por realizar altos investimentos na infraestrutura que suporta o sistema. Contudo, mesmo após a aplicação desses recursos, os problemas de desempenho não foram suprimidos. Dessa forma, a equipe da Coordenação de Infraestrutura e Redes (COINRE) estima que estes hardwares venham sendo subutilizados, sendo necessário um melhor aproveitamento dessa

3 <https://suap.ifrn.edu.br/api/docs/#/api>

estrutura, buscando compreender as origens dessas falhas de desempenho para permitir que tais recursos computacionais sejam utilizados de forma dinâmica e crescente, suportando cargas excedentes sem prejuízo de estabilidade e desempenho da aplicação.

Atualmente, não há uma clara identificação da origem dos problemas de desempenho do sistema, dificultando a busca por soluções reais para essas disfunções. Um das hipóteses consideradas é a de que o SUAP apresenta dificuldades de desempenho devido ao seu forte acoplamento, consequência da sua arquitetura monolítica atual. Esse estilo arquitetural traz consigo diversos impasses para as aplicações atuais, como permitir apenas a execução de um *deploy* único – isso resulta em períodos de *downtime*[4], dificultando a realização de testes ou atualizações na aplicação, por exemplo. Em tempo, são realizados apenas testes de conformidade em códigos que serão submetidos para produção que, por natureza, não avaliam o impacto das alterações sob a ótica de desempenho.

Surge, então, por parte da Diretoria de Gestão de Tecnologia da Informação (DIGTI), a demanda de analisar sistematicamente a arquitetura atual da aplicação, com um olhar crítico para investigar a causa das falhas de desempenho do SUAP. Para que se possa trabalhar essas demandas, é necessário identificar e avaliar pontos do sistema que atualmente possam prejudicar seu desempenho, a fim de propor uma estratégia eficaz para mitigar seus problemas de lentidão, oferecendo um monitoramento de desempenho da aplicação como apoio real para a equipe de desenvolvimento.

1.2 Objetivos

No contexto apresentado anteriormente, este trabalho tem como objetivo geral definir uma estratégia sistematizada para análise de desempenho do SUAP, identificando maneiras de diagnosticar os problema de desempenho do sistema.

Para tanto, os seguintes objetivos específicos serão considerados:

- Realizar uma avaliação arquitetural do SUAP, com foco em questões relacionadas ao desempenho; e
- Definir estratégias que possam ser utilizadas para avaliar e detectar situações em que o sistema apresente falhas de desempenho.

1.3 Metodologia

Para o desenvolvimento desta pesquisa, a dissertação seguiu as seguintes fases de execução:

[4] Período de tempo em que o serviço fica indisponível.

1. Levantamento bibliográfico sobre escalabilidade de sistemas web: pesquisa e embasamento teórico acerca do tema da pesquisa, conceituação e características sobre escalabilidade e trabalhos relacionados, dentre outros;

2. Avaliação da arquitetura atual da aplicação através do método ATAM: análise do modelo arquitetural adotado atualmente, identificando os pontos que prejudicam o desempenho da aplicação;

3. Realização de testes de software focados em desempenho; e

4. Proposta de um novo processo de desenvolvimento do SUAP, acrescentando testes de desempenho ao seu fluxo principal.

1.4 Estrutura do Documento

Na sequência desta dissertação, o Capítulo 2 apresentará os conceitos, contexto e técnicas que abordam escalabilidade em sistemas, métodos para avaliação arquitetural de software – com um detalhamento do método de avaliação elegido para analisar o sistema estudado, bem como uma abordagem sobre testes de software. No capítulo seguinte (Capítulo 3), será detalhado, com breve descrição do seu histórico, processo de desenvolvimento e arquitetura atual, o sistema SUAP. No Capítulo 4, será apresentada a avaliação arquitetural realizada, gerando os testes de carga focados em desempenho a que o sistema foi submetido (Capítulo 5). Na penúltima parte do documento, serão tratados os principais trabalhos relacionados estudados (Capítulo 6) e, por fim, o Capítulo 7 elucidará o resultado da pesquisa, além de abordar suas limitações, contribuições e indicativo de trabalhos futuros.

2 Referencial Teórico

Neste capítulo, serão apresentados e discutidos os principais conceitos considerados nestes trabalho. Iniciamos com uma breve discussão sobre escalabilidade, seguindo para o estudo sobre métodos de avaliação de arquitetura, em especial o *Architecture Trade-off Analysis Method* (ATAM). Por fim, serão apresentados conceitos sobre teste de software, com destaque para testes de desempenho.

2.1 Escalabilidade

Escalabilidade é um termo amplamente utilizado em artigos científicos, revistas técnicas e descrições de software. Contudo, seu uso nos contextos mais variados contribui para uma confusão geral sobre o que realmente o termo significa (DUBOC; ROSENBLUM; WICKS, 2006). A primeira tentativa de interpretar essa propriedade e definir formalmente o termo foi feita por Hill (1990), examinando aspectos da escalabilidade. Contudo, o autor apenas conseguiu indicar que escalabilidade se tornaria ”algo importante”, mas não seria útil para descrever uma arquitetura ou implementação de um software – afirmação contestada anos depois por Bondi (2000), ao afirmar que escalabilidade seria sim um atributo desejável em uma rede, sistema ou processo. Contudo, uma das grandes dificuldades para se qualificar escalabilidade é o critério subjetivo da avaliação. Brataas e Hughes (2004) definem que, idealmente, as considerações sobre a capacidade de escalabilidade de um software exigem uma visão geral e completa de seu sistema de hardware. Assim, é necessário ter uma compreensão profunda da arquitetura, analisando a relação entre sistemas, subsistemas e dispositivos, bem como a capacidade de replicação de tal software. Essa visão corrobora com a análise de Luke (1993), quando definiu escalabilidade como sendo a capacidade de um sistema e sua infraestrutura manterem boa relação custo-benefício à medida que a carga de trabalho cresce.

Não é possível falar de escalabilidade sem considerar aspectos como desempenho, manutenibilidade, usabilidade, confiabilidade, segurança, disponibilidade, etc (DUBOC; ROSENBLUM; WICKS, 2006). Na verdade, a ideia de escalabilidade está estreitamente ligada ao desempenho, apesar de não podermos considerar que os dois conceitos tenham o mesmo significado. Para Porto (2009), desempenho reflete o comportamento de um sistema em números, correlacionando capacidade de trabalho ao uso de recursos e tempo de execução. Já escalabilidade refere-se à capacidade desse sistema manter bom desempenho sob aumento de cargas de trabalho.

A propensão de um sistema em escalar bem vai muito além da capacidade de processamento das máquinas que o suportam. Gustavson (1994) aponta algumas propriedades

de hardware que podem ser analisadas ainda na etapa de projeto: desempenho, custo, tamanho físico, endereçamento, independência de software, capacidade de comunicação e independência de tecnologia.

Além do conceito, a subjetividade também se faz presente ao se avaliar a escalabilidade de um sistema. A viabilidade de execução desta tarefa esbarra em muitos dos problemas comuns de engenharia de software, principalmente na dificuldade de sistematização do processo de análise de escalabilidade, onde os autores afirmaram ainda existir alto grau de pessoalidade no método. São previstas, inclusive, a dependência de técnicas similares, como reutilização.

No geral, considerando as definições encontradas, podemos considerar escalabilidade como a capacidade do sistema se adaptar a diferentes cargas de trabalho a que é submetido, mantendo seu desempenho para cumprir seus requisitos funcionais e não funcionais.

Na literatura, duas propostas distintas de escalar sistemas são comumente descritas: escalabilidade estrutural e escalabilidade por distribuição de carga.

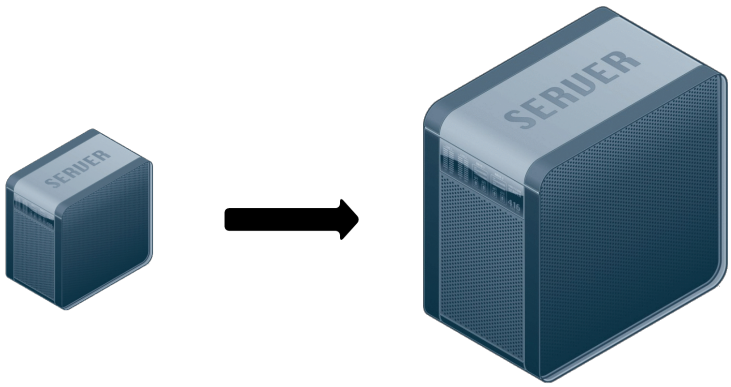
> **Escalabilidade** é uma medida de como o acréscimo de recursos (normalmente hardware) afeta o desempenho. Um sistema escalável é aquele que lhe permite adicionar hardware e obter uma melhora de desempenho proporcional, como dobrar o número de servidores disponíveis para dobrar o *throughput*. **Escalabilidade vertical**, ou *scale-up*, significa adicionar mais poder a um único servidor (por exemplo, acrescentar memória). **Escalabilidade Horizontal**, ou *scale-out*, significa adicionar mais servidores. (FOWLER, 2009)

A escalabilidade estrutural, também classificada como escalabilidade vertical (ou *scale-up*), refere-se à capacidade de um sistema expandir-se em uma dimensão pré-definida, sem a necessidade de grandes modificações arquiteturais (BONDI, 2000). Esta forma de escalar é considerada simples na literatura, pois busca melhorias de desempenho através da disponibilização de mais recursos computacionais ao software (LIU, 2011), conforme ilustrado na Figura 2. Formas de se escalar verticalmente um sistema vão desde adição de memória, disco rígido e/ou melhorias no processador, até o aumento do número de processos simultâneos suportados pelo sistema operacional.

Contudo, visto que toda máquina tem um limite para expansão, Porto (2009) alerta que esta prática pode se tornar ineficiente. O grande fator a ser considerado nesse modelo de escalar é o alto custo de componentes para essas máquinas, fato que acaba tornando o sistema não mais escalável, sendo necessárias mudanças arquiteturais na aplicação (LIU, 2011).

Propõe-se, então, a distribuição de carga, mais conhecida como Escalabilidade Horizontal – ou *scaled-out*. Bondi (2000) define essa metodologia como a capacidade de um sistema manter-se estável com relevantes aumentos de tráfego, replicando paralelismo.

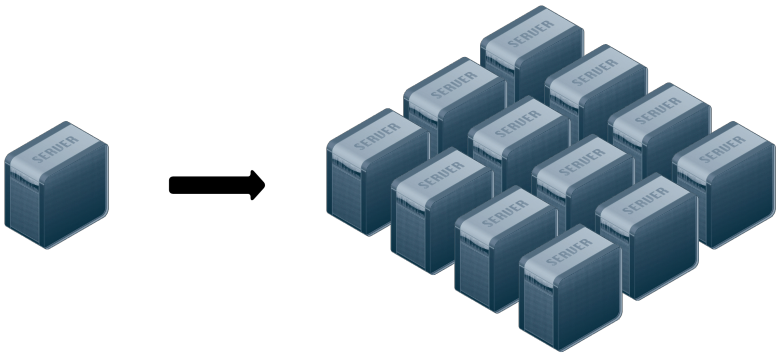
Figura 2 – Modelo de Escalabilidade Vertical



Fonte: Elaborada pelo autor.

Justamente por idealizar várias instâncias trabalhando paralelamente, esta técnica objetiva uma melhora de desempenho através da replicação de uma ou mais instâncias da mesma aplicação, conforme demonstra a Figura 3. Para isso, contudo, o sistema precisa ser arquiteturalmente projetado e construído para executar em vários nós de processamento. Com essa arquitetura é possível dimensionar apenas os serviços que precisam de dimensionamento, o que nos permite executar outras partes do sistema com hardware menor e menos potente (NEWMAN, 2015).

Figura 3 – Modelo de Escalabilidade Horizontal



Fonte: Elaborada pelo autor.

Com o recente barateamento do custo de componentes de hardware e de serviços de virtualização (BARUCHI, 2010), escalar horizontalmente aplicações tornou-se cada vez mais evidente.

Por sua vez, sistemas definidos como não escaláveis (ou *unscalable*), geralmente estão relacionados com um inadministrável custo computacional gerado por um determinado aumento de tráfego, não sendo praticável lidar com tal situação. Este custo pode ser

quantificado de várias maneiras, incluindo – mas não limitando – ao tempo de resposta, uso de processamento, espaço em disco, memória, ou até mesmo do custo financeiro (BONDI, 2000). Um sistema que não escala bem tem impactos diretos na qualidade do serviço que o demandou. Ele pode atrasar ou privar os usuários de oportunidades de receita, por exemplo, prejudicando sua competitividade. A escalabilidade de um sistema submisso a crescentes demandas é crucial para sua sobrevivência. Para tanto, como apoio para uma melhor solução em escalabilidade, é necessário compreender os elementos que envolvem a arquitetura da aplicação. Com isso, surge a necessidade de se avaliar essa arquitetura, a fim de que possam ser identificados problemas arquiteturais que denigram a escalabilidade do sistema.

2.2 Métodos de Avaliação de Arquitetura de Software

Ao se criticar a escalabilidade de um produto de software, é necessário que avaliemos suas características e seus obstáculos – esses pontos geralmente podem ser percebidos apenas analisando todo o contexto da aplicação (BONDI, 2000). Nesse sentido, metodologias aplicadas à avaliação arquitetural buscam permitir aos arquitetos de software refletir sobre os aspectos de qualidade dos sistemas ainda no início do processo de desenvolvimento (MATTSSON; GRAHN; MÅRTENSSON, 2006). Contudo, alguns desses métodos podem também serem utilizados para tal fim em sistemas já prontos. Deve-se considerar sempre que o custo para execução desses métodos são relativamente baixos frente ao ganho que eles podem proporcionar. Na literatura, as metodologias tidas como principais são SAAM (*Software Architecture Analysis Method*) e ATAM (*Architecture Trade-off Analysis Method*). Para Ionita, Hammer e Obbink (2002), todos os demais métodos reconhecidos são refinamentos deles.

Buscando avaliar essas metodologias, Mattsson, Grahn e Mårtensson (2006) inspecionaram 240 artigos que abordavam métodos de avaliação de arquitetura. Para os autores, muitos desses conceitos abordavam um único atributo de qualidade, com pouco deles sendo capazes de avaliar diversos atributos simultaneamente. Especificamente, apenas um método inclui análise de *trade-off*[1]. Ademais, os pesquisadores constataram que muitos métodos foram aplicados e validados apenas por seus próprios inventores. Das metodologias estudadas, apenas duas haviam sido utilizadas por outros pesquisadores que não os autores originais do método: exatamente SAAM e ATAM – neste contexto, foi considerado que este era uma indicação de maturidade de ambos.

SAAM foi o primeiro projeto de análise de arquitetura de software baseado em cenários como método. Tal metodologia foi criada objetivando avaliar a modificabilidade das arquiteturas, sendo que, se duas ou mais arquiteturas candidatas diferentes fornecerem

[1] Termo que define uma situação em que há conflito de escolha. Neste contexto, entende-se como a necessidade de se optar por um ganho em detrimento de perdas em um outro atributo.

a mesma funcionalidade e forem comparadas em relação à sua capacidade de transformação, o SAAM poderá produzir uma classificação relativa entre elas. A aplicação do método consiste em seis passos principais, que vão desde o entendimento sobre a definição do escopo da aplicação; passando pela definição da arquitetura; classificação e priorização de cenários e avaliação individual e da interação deles; e, por fim, a criação de uma avaliação geral.

Baseado no SAAM, o método ATAM, por sua vez, analisa até que ponto a arquitetura do software satisfaz objetivos específicos de qualidade, seguindo métricas especificadas na ISO/IEC:9126-1 (2003). Ele também fornece insights sobre as interdependências desses atributos de qualidade – analisando como eles se alternam, interagem e sofrem influência direta de um ou mais outros atributos. Kazman, Klein e Clements (2000) definem que o ATAM avalia características de apoio à modificabilidade, confiabilidade, desempenho e segurança, dentre outros atributos de qualidade. O método pode ser considerado, então, uma maneira completa e abrangente de avaliar uma arquitetura de um software (BASS; CLEMENTS; KAZMAN, 2003).

Orientado a cenários, o método ATAM foi desenvolvido por Kazman et al. (1998) e concentra-se em analisar um atributo de qualidade específico através da criação de um perfil de cenário, forçando uma descrição concreta desse requisito. Esses cenários são utilizados para percorrer toda a arquitetura da aplicação e documentar as consequências de sua análise. Com execução um pouco mais complexa que o SAAM – metodologia que originou esta, o *Architecture Trade-off Analysis Method* vem sendo utilizado e validado em diversos estudos.

Como insumos, Babar M.A.; Gorton (2004) descrevem que o ATAM utiliza o guia de negócio (*business drivers*) e a descrição da arquitetura – intrínseco a isso, também a especificação do software. Por sua vez, o método produz, como artefatos, uma definição das metas de negócio (*business goals*), possíveis abordagens arquiteturais para aquele produto, uma lista de cenários, os pontos de sensibilidade da aplicação, seus riscos e não riscos, bem como a descrição dos *trade-off points*. Os participantes previstos são os integrantes do time de avaliação da arquitetura, os tomadores de decisão do projeto e as partes interessadas na abordagem arquitetural do software.

Buscando realizar um comparativo entre os principais métodos de avaliação conhecidos, Oliveira e Nakagawa (2009) concluiram que os métodos tradicionais de avaliação arquitetural – como o SAAM e ATAM, são capazes de analisar e prever uma série de características de qualidade das arquiteturas, contudo são altamente dependentes dos *stakeholders* envolvidos. Sobre os participantes, Bass, Clements e Kazman (2003) definem que o ATAM requer a participação e cooperação mútua de três grupos principais:

1. Time de avaliação (*The evaluation team*): equipe externa ao projeto cuja arquite-

tura está sendo avaliada. Geralmente consiste de três a cinco pessoas, profissionais experientes e especializados em arquitetura de software. Cada membro desse time recebe um número de funções específicas para julgar durante a avaliação.

2. Tomadores de decisão do projeto (*Project decision makers*): grupo que tem autonomia para falar em nome do projeto ou tem autoridade para impor mudanças a ele. Geralmente é composto pelo gerente de projeto, pelo cliente, pelo arquiteto do projeto e pelo representante da equipe de avaliação.

3. Partes interessadas na arquitetura (*Architecture stakeholders*): grupo de doze a quinze profissionais cuja capacidade de realizar seus trabalhos depende da arquitetura, promovendo modificabilidade, segurança, alta confiabilidade ou algo parecido (inclui desenvolvedores, testadores, integradores, mantenedores, engenheiros de desempenho, usuários, etc). Seu trabalho durante uma avaliação é articular os objetivos específicos de atributos de qualidade que a arquitetura deve atender para que o sistema seja considerado um sucesso.

Adicionalmente, Paul, Kazman e Klein (2002) identificam outros papéis envolvidos na execução da avaliação, bem como suas atribuições e características desejáveis nos profissionais que exercerão tais papéis. Contudo, pelo contexto da pesquisa, esses papéis não foram considerados para desenvolvimento das atividades. Neste ponto, o próprio pesquisador exerceu os papéis de líder da avaliação, descritor de cenários e descritor de procedimentos.

- Líder da Avaliação: executa a avaliação; facilita a elicitação de cenários; administra o processo de seleção / priorização de cenários; e media a avaliação de cenários frente à arquitetura. Deve sentir-se confortável na frente do público; excelentes habilidades de mediação; boa compreensão de questões arquitetônicas; experiente em avaliar arquiteturas; capaz de avaliar quando uma discussão prolongada está levando a uma descoberta valiosa ou quando é inútil e deve ser redirecionada.

- Descritor de Cenários: escreve cenários em *flipchart*[2] ou quadro branco durante a elucidação do cenário; capta a concordância de cada cenário, interrompendo a discussão até que o texto exato seja capturado. Deve ter boa caligrafia; defensor de não se avançar antes que uma ideia (cenário) seja capturada; pode absorver e extrair a essência das discussões técnicas.

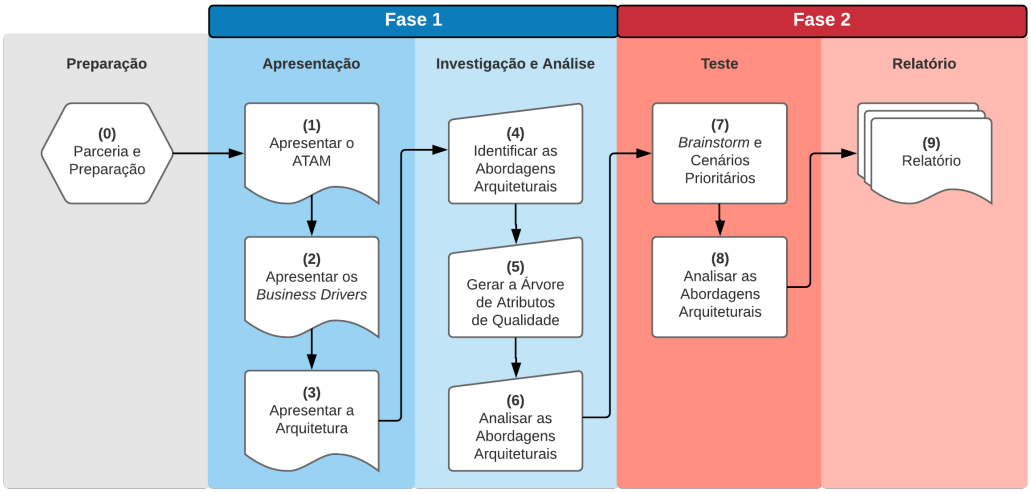
- Descritor de Procedimentos: captura eletronicamente procedimentos, cenários brutos e problema(s) que motivam cada cenário, bem como a resolução de cada cenário quando aplicado à(s) arquitetura(s); também gera uma lista impressa de cenários

[2] Conhecido no Brasil como tripé ou cavalete, é um tipo de quadro, usado geralmente para exposições didáticas ou apresentações, em que fica preso um bloco de papéis.

adotados (para distribuição a todos os participantes). Bom (e rápido) digitador; bem organizado (para recuperação rápida de informações); boa compreensão de questões arquitetônicas; capaz de assimilar rapidamente problemas técnicos; sem medo de interromper o fluxo de discussão (em momentos oportunos) a fim de testar a compreensão de um problema (para que informações apropriadas sejam capturadas) são características desejáveis para o papel.

O método ATAM é composto por nove atividades, agrupadas em quatro etapas e duas fases (ilustradas na Figura 4).

Figura 4 – Fases e etapas do ATAM



Fonte: Elaborada pelo autor.

Após a preparação de todos os envolvidos no processo, na fase 1, em suas etapas de apresentação, o ATAM descreve o processo de avaliação que será realizado, onde o líder da equipe de avaliação apresenta o ATAM aos *stakeholders* (atividade 1). Neste momento, é detalhado totalmente o processo que todos os envolvidos devem seguir, definindo o contexto e expectativa para as demais atividades. Todos os interessados aqui devem ser inteirados das informações que serão coletadas, como elas serão examinadas e a quem serão reportados esses dados. Os objetivos gerais desta atividade são a apresentação de uma breve visão sobre os demais passos do ATAM, técnicas que serão utilizadas para análise e saída esperada da avaliação, bem como a elucidação das dúvidas dos partícipes do processo.

Em seguida, na atividade 2, são apresentadas as diretivas de negócio e a relação delas com arquitetura de software. Aqui, o gerente do projeto apresenta uma visão geral da aplicação, inicialmente em uma abstração de alto nível. Sob uma perspectiva de negócio, nesta atividade são descritos itens como requisitos funcionais de maior relevância; restrições

gerais (técnicas, gerenciais e/ou econômicas, por exemplo); contexto e objetivo de negócio; principais *stakeholders*; e diretrizes arquiteturais.

Na atividade 3, é apresentado um modelo inicial, contudo detalhado, de arquitetura proposta para o software. Neste momento, devem ser identificadas e descritas possíveis restrições técnicas (como SO ou hardware, por exemplo); necessidade e nível de interação da aplicação com outros softwares; e abordagens arquiteturais utilizadas para obtenção dos atributos de qualidade.

Partindo para a etapa de investigação e análise (ainda na 1ª fase do ATAM), é executada a atividade de apresentação dos mecanismos arquiteturais que suportarão as metas de negócio da aplicação (atividade 4). Nesta atividade é substancial somente a identificação – e não a análise – de abordagens e estilos arquiteturais que possam ser aplicados ao software em questão. Espera-se neste momento a definição de como o sistema poderá crescer e responder a mudanças, resistir a ataques ou interagir com outros sistemas, etc.

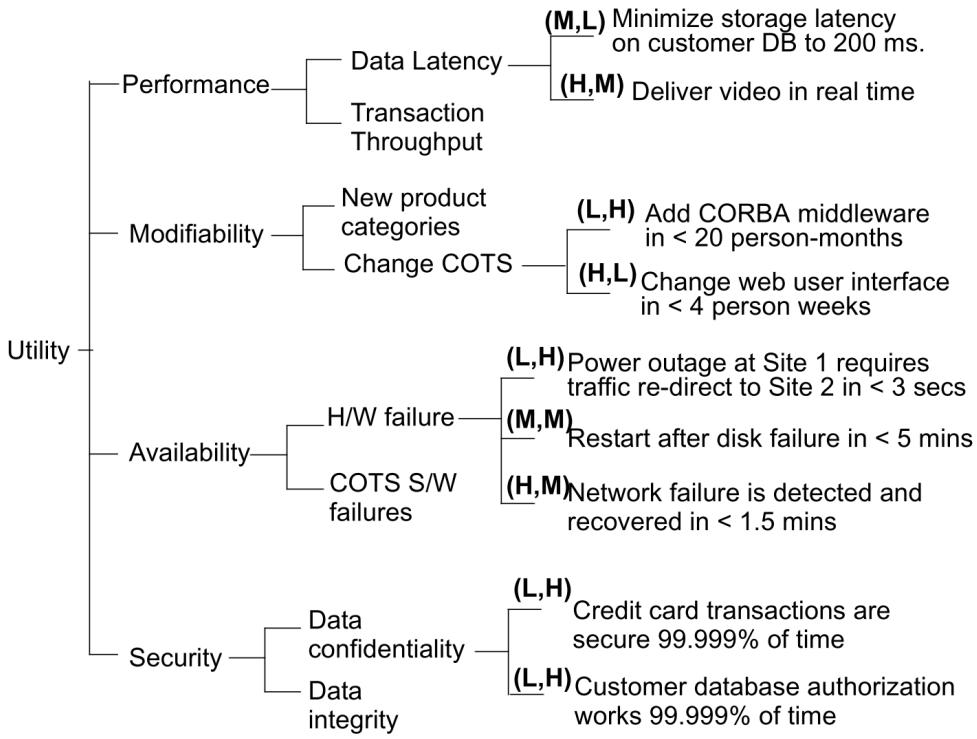
Sequencialmente, a 5ª atividade do processo busca identificar os atributos de qualidade e elaborar cenários para eles, através da composição da árvore de utilidades. Agora os *stakeholders* já devem ser capazes de identificar e priorizar atributos de qualidade importantes para a aplicação, elaborando um organograma semelhante ao ilustrado na Figura 5.

No primeiro nó da árvore, são considerados um ou mais atributos de qualidade que se deseja avaliar. Dentre esse atributos, os autores identificam como principais: modificabilidade, confiabilidade, desempenho e segurança. No segundo nível da árvore de atributos, deverão ser identificados os fatores que serão utilizados como foco para avaliar o atributo pai, como latência ou falha de componente de hardware, por exemplo. Já no terceiro (e último) nível, deverão ser descritas as métricas que cada fator deve atingir como um dos resultados da avaliação – por exemplo, o intervalo máximo aceitável de tempo de resposta da aplicação para determinada transação. Junto a cada métrica, ainda é necessário identificar, no contexto da arquitetura que será avaliada, os graus esperados de importância e dificuldade para se alcançar cada caracterização. Esses graus são classificados como L, M ou H: *Low*, ou baixo; *Medium*, ou médio; e *High*, ou alto, respectivamente.

Na última atividade desta fase, realiza-se a avaliação de cada cenário consolidado no passo anterior – descrevendo riscos, pontos de sensibilidade e *trade-offs*. Justamente por gerar esse conhecimento, esta análise deve ser plenamente documentada, a fim de identificar o atributo de qualidade a ser alcançado. O arquiteto e a equipe devem analisar cada abordagem possível, sendo capazes de elaborar um documento de cenário e análise arquitetural – documento este previsto por Clements, Kazman e Klein (2002).

Na fase seguinte, são executadas as etapas de teste (atividades 7 e 8) e é gerado

Figura 5 – Modelo de árvore de utilidades (*utility tree*)



Fonte: Clements, Kazman e Klein (2002).

um relatório final da aplicação da avaliação (atividade 9). Na etapa de teste, são complementados os cenários, com a descrição de priorização deles – atividade 7. Neste passo, são representados os cenários de caso de uso e o de mudança, demonstrando como o usuário final espera o uso do sistema. Já a perspectiva de mudança em um cenário é representada em como a aplicação reagirá a variações. Dessa forma, espera-se que, após os testes, arquiteto e equipe de avaliação possam apresentar e comparar suas priorizações com as apontadas pelos demais *stakeholders*. Toda diferença encontrada precisa ser analisada e adequada.

Na atividade 8 do processo, avalia-se cada cenário gerado na atividade 7, com riscos, pontos de sensibilidade e *trade-offs* dos novos cenários encontrados. Neste momento, o arquiteto deve iniciar o processo de mapeamento dos cenários – respeitando o ordenamento de classificação de relevância do atributo de qualidade. Complementar às atividades 6 e 7, esta atividade deve considerar todo o contexto já analisado.

Finalmente, na 9ª atividade são apresentados os resultados encontrados com a aplicação do ATAM, demonstrando todas as informações levantadas durante o processo de avaliação a seus envolvidos. Cada informação coletada durante o processo deve ser retornada a cada representante do projeto, descrevendo detalhadamente as etapas percorridas e como os artefatos foram gerados.

Para cada aplicação do ATAM são previstos diferentes contextos e cenários, contudo suas atividades gerais não se alteram. Em seu trabalho, Kazman, Klein e Clements (2000) descrevem um exemplo de uma agenda típica da ATAM, com um total de três dias de atividades. O autor descreve, em sua pesquisa, a execução de toda a fase 1 do método ainda no primeiro dia de avaliação. Após um intervalo de algumas semanas, o segundo dia de aplicação do método dá início às atividades da fase 2, que só é concluída no terceiro dia. Vale ressaltar que esses cronograma prevê apenas as atividades de preparação, investigação, análise e teste. A elaboração do relatório é uma atividade mais complexa, não estando prevista nesse momento da aplicação.

Após o processo de avaliação, Bass, Clements e Kazman (2003) entendem que um relatório final escrito deve ser produzido. Ele deve conter o método, resumir os procedimentos aplicados e capturar os cenários e suas análises, bem como catalogar as descobertas. Visto que o ATAM não será seguido rigorosamente em todas as suas atividades neste estudo de caso, o relatório que será produzido ao final da avaliação do SUAP poderá suprimir alguns dos artefatos esperados na metodologia tradicional, que são:

- Apresentação concisa da arquitetura: geralmente, a documentação de arquitetura é considerada como o modelo de objeto, uma lista de interfaces e suas assinaturas (ou alguma outra lista volumosa). Mas um dos requisitos do ATAM é que a arquitetura seja apresentada em uma hora, o que leva a uma apresentação arquitetônica concisa e compreensível.

- Articulação dos objetivos de negócio: o gerente de projeto descreve quais metas de negócios estão motivando o esforço de desenvolvimento e, portanto, quais serão os principais impulsionadores de arquitetura (por exemplo, alta disponibilidade ou alta segurança). Comumente, as metas de negócios apresentadas no ATAM estão sendo vistas pela equipe de desenvolvimento pela primeira vez.

- Requisitos de qualidade em uma coleção de cenários: as metas de negócios levam a requisitos de qualidade. Alguns dos importantes requisitos de qualidade devem ser capturados em diferentes cenários.

- Mapeamento de decisões arquitetônicas sobre requisitos de qualidade: as decisões arquiteturais podem ser interpretadas em função dos parâmetros de qualidade que elas impactam. Para cada atributo de qualidade examinado durante um ATAM, são descritas as decisões de arquitetura que ajudam a alcançá-lo.

- Conjunto de pontos de sensibilidade e *trade-offs* identificados: essas são decisões de arquitetura que têm um efeito conhecido em um ou mais atributos de qualidade. A adoção de um banco de dados de backup, por exemplo, é claramente uma decisão arquitetônica, pois afeta a confiabilidade (positivamente) e, portanto, é um ponto de

sensibilidade em relação à confiabilidade. No entanto, manter um ciclo de backup consome recursos do sistema e afeta negativamente o desempenho. Portanto, é um ponto de equilíbrio entre confiabilidade e desempenho.

- Conjunto de riscos e não-riscos. Um risco é definido no ATAM como uma decisão arquitetural que pode levar a consequências indesejáveis sobre os atributos de qualidade declarados. Da mesma forma, um não-risco é uma decisão arquitetônica que, após análise, é considerada segura. Os riscos identificados podem formar a base para um plano arquitetural de mitigação de riscos.

- Conjunto de fatores de risco: quando a análise for concluída, a equipe de avaliação examinará o conjunto completo de riscos descobertos para procurar temas abrangentes que identifiquem os pontos fracos sistêmicos na arquitetura – ou até mesmo no processo e na equipe de arquitetura. Se não forem tratados, esses pontos de risco ameaçarão os objetivos de negócios do projeto.

Esse método será base da avaliação do sistema estudado. Contudo, devido ao contexto da pesquisa e por ser tratar da avaliação de um software em produção, a análise que será aplicada é apenas uma derivação do ATAM – toda a metodologia adotada é detalhada no Capítulo 4. Ainda assim, ao final da execução dos testes, um relatório será apresentado, tal qual na metodologia original.

Além do SAAM e do ATAM, outros métodos de avaliação baseado em cenários são conhecidos, como por exemplo o ALMA (*Architecture Level Modifiability Analysis*). Outras formas de avaliação também desatacadas são CBAM (*Cost-Benefit Analysis Method*), que baseia-se na avaliação de custo-benefício da arquitetura; e FAAM (*Family-Architecture Assessment Method*), que avalia dois aspectos de qualidade relacionados: interoperabilidade e extensibilidade da aplicação sob a ótica arquitetural. Como essas segmentações fogem do escopo desta pesquisa, elas não serão abordadas a fundo.

2.3 Teste de Software

Através da realização de testes, pode-se avaliar requisitos funcionais e não funcionais de um produto de software, além de possibilitar a validação de casos de uso da aplicação. Para Hörcher (1995), testar funcionalidades de um sistema em relação à especificação manterá suas funcionalidades e também suas especificações formais. Visto que um dos objetivos deste trabalho volta-se à avaliação de um sistema já concebido, será necessário executar testes de software durante essa análise e, posteriormente, para validar uma proposta. Considerando o foco da pesquisa no desempenho da aplicação, serão considerados apenas testes que analisem este atributo.

Definido como performance na literatura, o atributo de desempenho de um sistema condiz com o critério de qualidade descrito como eficiência pela ISO/IEC:9126-1 (2003). Ele é "o grau em que um sistema ou componente realiza suas funções designadas dentro de determinadas restrições, como velocidade, precisão ou uso de memória. Existem muitos aspectos do desempenho, como por exemplo, latência, taxa de transferência e capacidade" (MATTSSON; GRAHN; MÅRTENSSON, 2006). Desempenho é considerado um atributo substancial, pois todo sistema deve atender aos requisitos de funcionamento – caso contrário, ele será de uso limitado, ou não usado. O foco a longo prazo, com isso, força o sistema a ser sustentável e testável.

Para Myers (2004), o teste de desempenho tem por objetivo avaliar o comportamento do sistema submetido, quando submetido a uma determinada carga. A meta deste teste é determinar e assegurar que o sistema funcione adequadamente com uma carga de trabalho superior à carga máxima esperada. Além disso, o teste de carga avalia as características de desempenho (tempos de resposta, taxas de transação e outros aspectos sensíveis ao tempo). Obviamente, o autor indica que esses testes devam ser realizados em ambientes dedicados, com arquitetura e recursos semelhantes ao do ambiente real. Para ele, esse tipo de teste fornece indicadores de desempenho que podem ser utilizados como métricas para avaliar o quão um sistema (ou componente) é capaz de cumprir os requisitos de desempenho, como tempo de resposta ou *throughput*, por exemplo. Smith (2007) ainda corrobora que a compreensão desses fatores podem ser essenciais na identificação dos possíveis gargalos que possam estar degradando o desempenho do sistema.

O teste de desempenho normalmente é executado várias vezes, cada uma utilizando uma carga diferente no sistema. O teste inicial deve ser executado com uma carga nominal semelhante à carga normal observada (ou prevista) no sistema de destino. Gradativamente aumenta-se a quantidade de requisições, até atingir uma carga de pico.

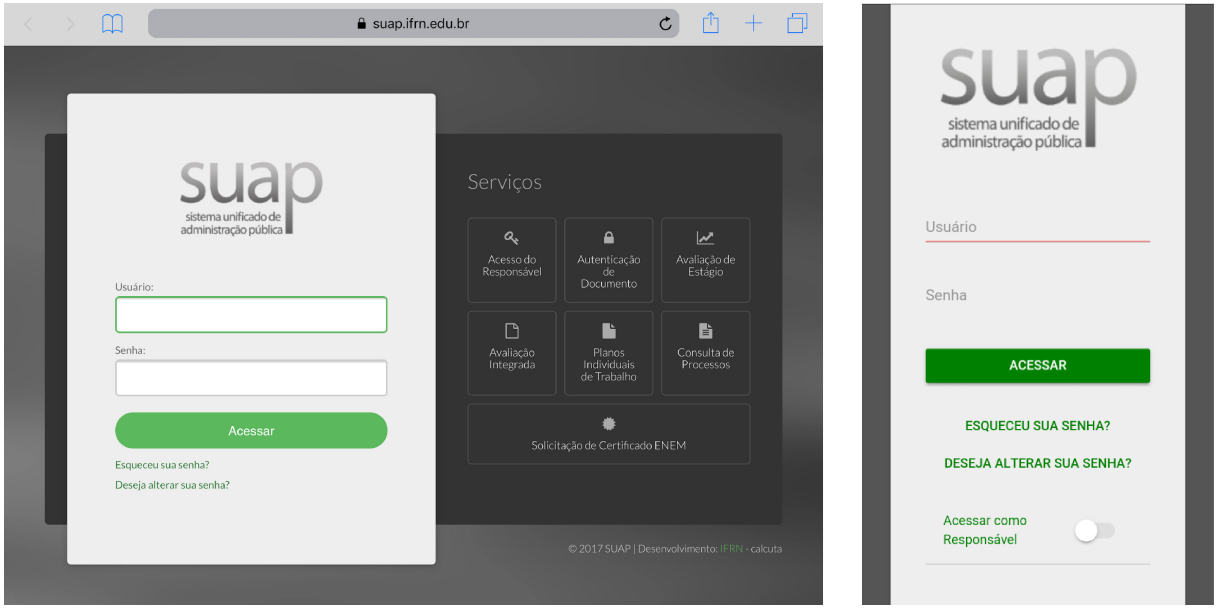
2.4 Considerações Finais

Neste capítulo, através da revisão da literatura, pôde-se arguir sobre escalabilidade em sistemas (e seus tipos), visto que os problemas do sistema estudado permeiam a dificuldade de se escalar bem uma aplicação. Na literatura, verificou-se a prática de avaliar a arquitetura de aplicações que sofrem problemas em escalar bem. Dessa forma, foi buscada base teórica acerca de métodos reconhecidos para avaliação arquitetural de sistemas. Dentre os cinco métodos tidos como principais, o ATAM foi o que se enquadrou nos objetivos deste estudo. Assim, esse método foi estudado mais a fundo e detalhado neste capítulo. Sua aplicação ao sistema estudo será descrita nos próximos capítulos.

3 SUAP

Concebido e com desenvolvimento iniciado em 2007, o Sistema Unificado de Administração Pública (SUAP) foi proposto pela Diretoria de TI do IFRN. Seu objetivo era gradualmente apoiar aos diversos setores, Pró-reitorias, Diretorias Sistêmicas e Direções Gerais dos *campi* em suas ações. Para tanto, ao longo desses anos, o aperfeiçoamento do sistema sustentou-se no Plano de Desenvolvimento Institucional (PDI) do IFRN, abrangendo as oito dimensões sistêmicas definidas: Administração, Assistência Estudantil, Ensino, Extensão, Gestão de Pessoas, Pesquisa, Planejamento e Tecnologia da Informação.

Figura 6 – Versões web e *mobile* do SUAP, respectivamente



Fonte: Elaborada pelo autor.

Em 2018, o sistema já conta com mais de 40 módulos, automatizando e gerenciando os processos de Gestão de Pessoas; Gestão Acadêmica; Controle Patrimonial e de Almoxarifado; Ponto Eletrônico; Planejamento Anual e Gestão Orçamentária; Gestão de Projetos de Pesquisa, Inovação e Extensão; Gestão de Protocolo de Documentos[1]; Controle de Acesso a chaves de ambientes; Gestão de Viagens Institucionais e Frota de Veículos; bem como Gestão de Contratos e Convênios – a descrição completa dos módulos pode ser vista no Apêndice A.

Atendendo a mais de 35 mil usuários apenas no âmbito do IFRN, estima-se que o SUAP atenda a outros quase meio milhão de usuários no país, visto que seu código é cedido a 30 órgãos conveniados através de Acordos de Cooperação Técnica. Desde a

[1] Desde 2017, todos os processos abertos através do sistema são totalmente eletrônicos.

expansão da Rede Federal de Educação, o Ministério da Educação (MEC), através da Secretaria de Educação Profissional e Tecnológica (SETEC), indicou ao IFRN o objetivo de tornar o SUAP o sistema de administração padrão para a Rede Federal de Ensino. No Apêndice A, a Tabela 10 lista todos os órgãos conveniados atualmente.

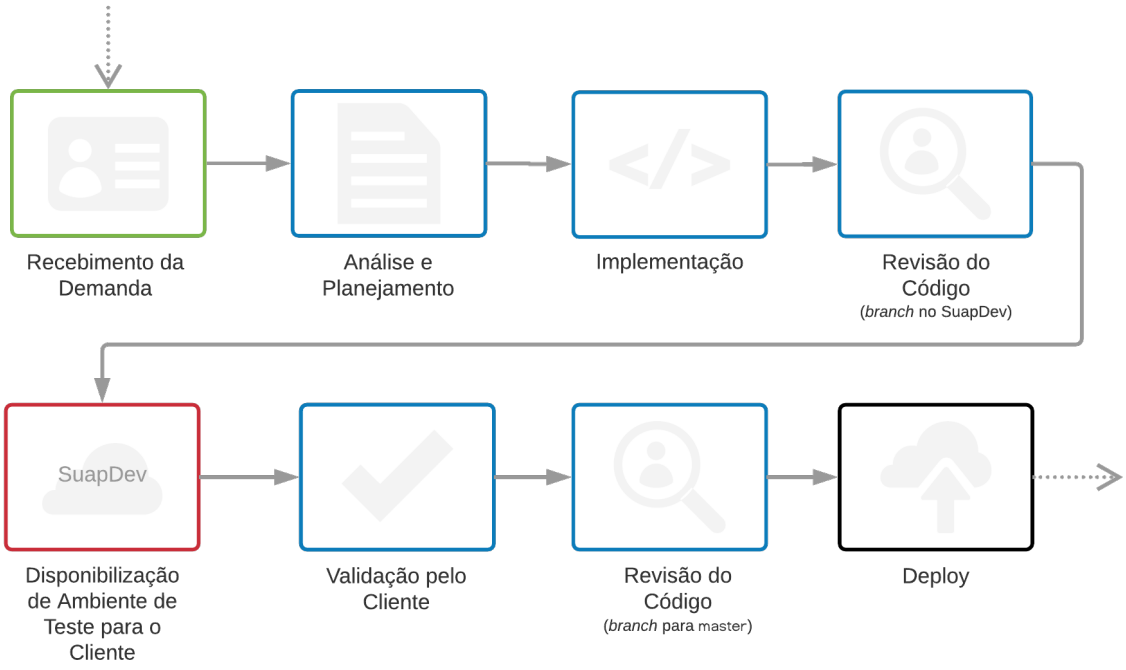
3.1 Processo de Desenvolvimento do SUAP

Por alguns fatores particulares, o SUAP não é guiado sob uma metodologia de processo de desenvolvimento de software tradicional. Inicialmente, a gestão optou pela adoção do *Scrum*, metodologia ágil de desenvolvimento de software, que explora a abordagem iterativa e incremental para aperfeiçoar a previsibilidade e o controle de riscos (SCHWABER; SUTHERLAND, 2013).

Contudo, visto a heterogeneidade dos clientes internos (setores) e pelo setor de desenvolvimento do IFRN não ter caraterísticas análogas a uma empresa privada de produção software, identificou-se e impossibilidade de seguir tal proposta. Segundo a equipe de desenvolvimento do IFRN, foi feita uma adaptação da metodologia tradicional do *Scrum* para o contexto da Instituição. O fluxo de processo adotado foi documentado pela equipe de desenvolvimento e é ilustrado no Figura 7[2].

[2] Esse processo atualmente é seguido para desenvolvimento de funcionalidades para o sistema web. No caso da versão *mobile*, como não há ainda grandes funcionalidades nem equipe direcionada para tal plataforma, não existe um processo definido por ora.

Figura 7 – Fluxo do processo de desenvolvimento de novas demanda para o SUAP



Fonte: Carvalho (2015), adaptada pelo autor.

O fluxo é iniciado ao receber a demanda de um setor, por exemplo. Neste momento, o projeto é analisado e inicia-se seu planejamento. Ao final dessa segunda etapa, é iniciada a implementação do módulo. Em seguida, a *branch* do projeto é incorporada ao SUAPDev (ambiente designado para testes de conformidade pelo cliente). Partindo para a fase de validação, esse ambiente é disponibilizado ao cliente, que realiza testes de conformidade. Com a validação do cliente, realiza-se uma última revisão do código, submetendo-o para *deploy* na etapa final.

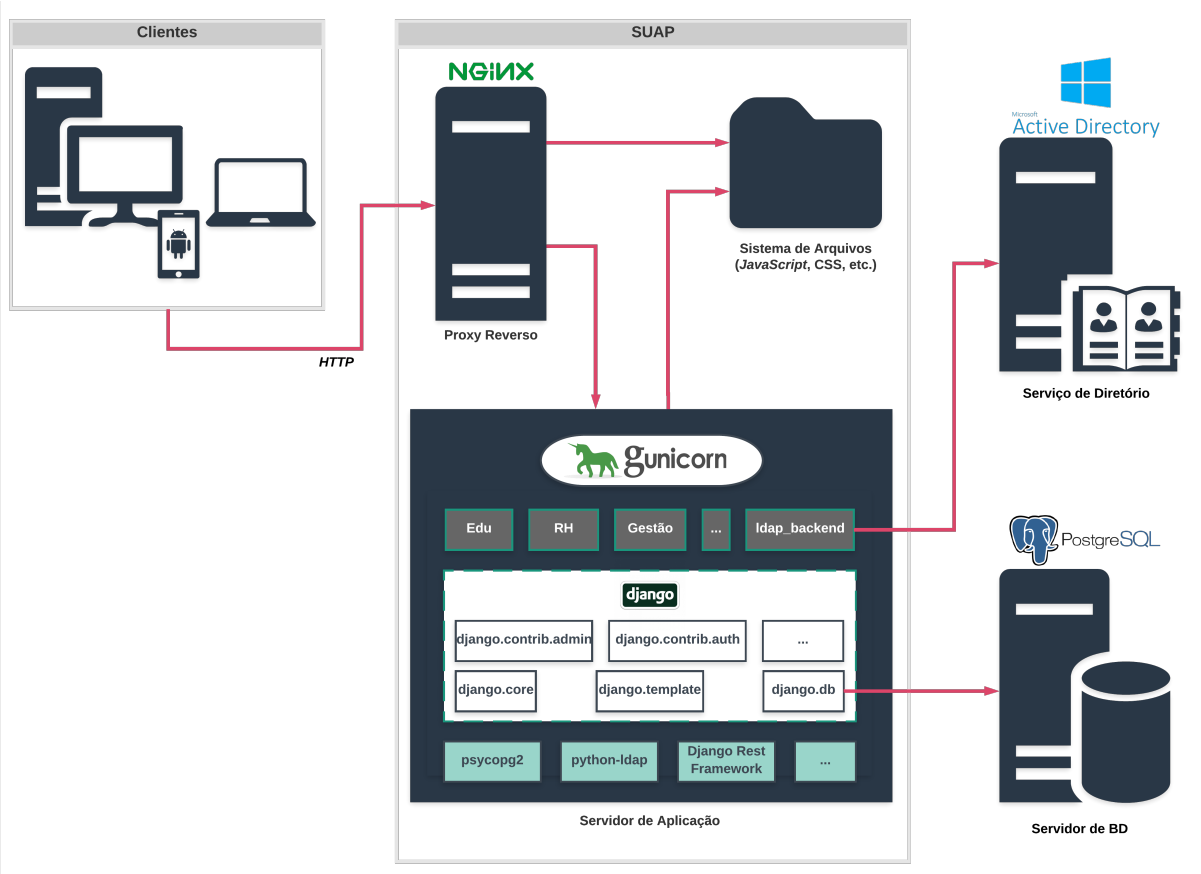
3.2 Arquitetura Atual do Sistema SUAP

Desde sua concepção, o SUAP foi projetado para ser desenvolvido com a linguagem de programação **Python**[3], sob o *framework* **Django**[4]. Dentre as demais tecnologias utilizadas para desenvolvimento do sistema, todas são de software livre. Como SGBD é utilizado o **PostgreSQL**[5], além dos serviços **NGINX**[6] e **Gunicorn**[7] – servidores HTTP

3 <https://www.python.org>
4 <https://www.djangoproject.com>
5 <https://www.postgresql.org>
6 <https://www.nginx.com>
7 <http://gunicorn.org>

e de aplicação, respectivamente. Na Figura 8 é apresentado o diagrama de arquitetura atual do projeto.

Figura 8 – Arquitetura Atual do SUAP



Fonte: Elaborada pelo autor.

No servidor de aplicação, são identificados alguns dos módulos desenvolvidos do sistema (caixas na cor cinza), dentre eles: Edu (gestão educacional); RH (recursos humanos); e Adm (administração). Já o módulo `ldap_backend` é responsável para que o SUAP sincronize toda sua base de usuários com o serviço de diretórios utilizado, o *Active Directory* (AD), da Microsoft. Através do AD, é possível gerenciar os usuários de forma centralizada, concentrando a autenticação dos usuários em um único serviço, além da possibilidade de controle de permissões através de grupos – como grupos que têm permissão de uso da VPN e grupos de impressão, por exemplo.

Na parte central da aplicação SUAP (ainda referente à Figura 8), estão representados componentes do projeto **Django**. Esses itens plugáveis e reutilizáveis são fornecidos por padrão pelo *framework*, para desempenhar, por exemplo, operações de inserção, exclusão, alteração e busca de registros (interface dinâmica `django.contrib.admin`); e autenticação, autorização e proteção (`django.contrib.auth`). Na parte inferior (módulos em verde), são apresentados alguns *frameworks* de terceiros que hoje estão conectados à aplicação. Dentre

outras funções, eles fornecem suporte ao banco de dados **PostgreSQL** (`psycopg2`[8]); e acesso a diretórios **LDAP** (`python-ldap`[9]).

Esses componentes são acessados pelos clientes através de requisições HTTP ao servidor de *proxy* reverso. Por sua vez, além de direcionar as solicitações ao servidor de aplicação (**Gunicorn**), o **NGINX** pode encaminhar a requisição diretamente ao sistema de arquivos – nele são armazenados arquivos estáticos, como **JavaScript**, **CSS**, **PNG**, etc. Para acessar o sistema, os usuários contam com a versão web[10] e um *app* para **Android**[11] (Figura 6).

Considerando esta arquitetura, atualmente há a prática de um *deploy* único para o lançamento de cada nova funcionalidade ao sistema. Isso acaba atrasando esse processo e o de correções de bugs, por exemplo, pois há um *gap* até o processo de integração com o sistema que está rodando em produção. Outros problemas desse tipo de *deployment* decaem sobre a dificuldade de implantação de ferramentas para automação dessa atividade, além de apresentar um alto custo para a execução de operação de *rollback*, quando necessário.

Para sustentar essa arquitetura, a Pró-Reitoria de Administração (PROAD) do IFRN disponibilizou recursos para a DIGTI adquirir e disponibilizar um servidor dedicado com 16 *cores* de processamento e 64 GB de memória RAM, além um de outro servidor virtual exclusivo para o banco de dados, com 16 *cores* de processamento e 24 GB de RAM. Contudo, mesmo com a disponibilização de todos esses recursos para a melhoria e ampliação da infraestrutura de TI da Instituição, o SUAP ainda apresenta momentos de lentidão. E mesmo após testes de práticas para balanceamento de carga e replicação, dentre outras, o SUAP continuou apresentando situações de instabilidade e mau desempenho.

3.3 Considerações Finais

Esta pesquisa entende que é necessário a realização de uma avaliação da arquitetura atual do sistema, já que os recursos disponibilizados são considerados, pelas equipes de desenvolvimento e infraestrutura, como superdimensionados. Assim, buscou-se na literatura trabalhos análogos que desenvolvessem essa análise, sendo que estes partiam da avaliação da arquitetura dos sistemas propostos.

Ao considerar os métodos apresentados na literatura para avaliar arquiteturas e consultar trabalhos relacionados, constatou-se que o ATAM seria o mais habilitado a colaborar com o objetivo geral desta pesquisa, visto que ele poderia ser aplicado em sistemas já concebidos, buscando identificar a origem de problemas de desempenho. Segundo os próprios autores do método, ele é capaz de avaliar, dentre outros atributos de qualidade,

[8] <http://initd.org/psycopg/>
[9] <https://www.python-ldap.org/>
[10] <https://suap.ifrn.edu.br>
[11] <https://play.google.com/store/apps/details?id=br.edu.ifrn.suap.mobile>

características de apoio para se obter desempenho, que é o principal foco deste estudo. O método é considerado uma maneira completa e abrangente de avaliar uma arquitetura de um software, percorrendo toda a arquitetura da aplicação e documentando as consequências de sua análise.

4 Aplicação do ATAM no SUAP

Este capítulo apresenta a aplicação do ATAM no SUAP, bem como os dados que foram coletados na aplicação do método.

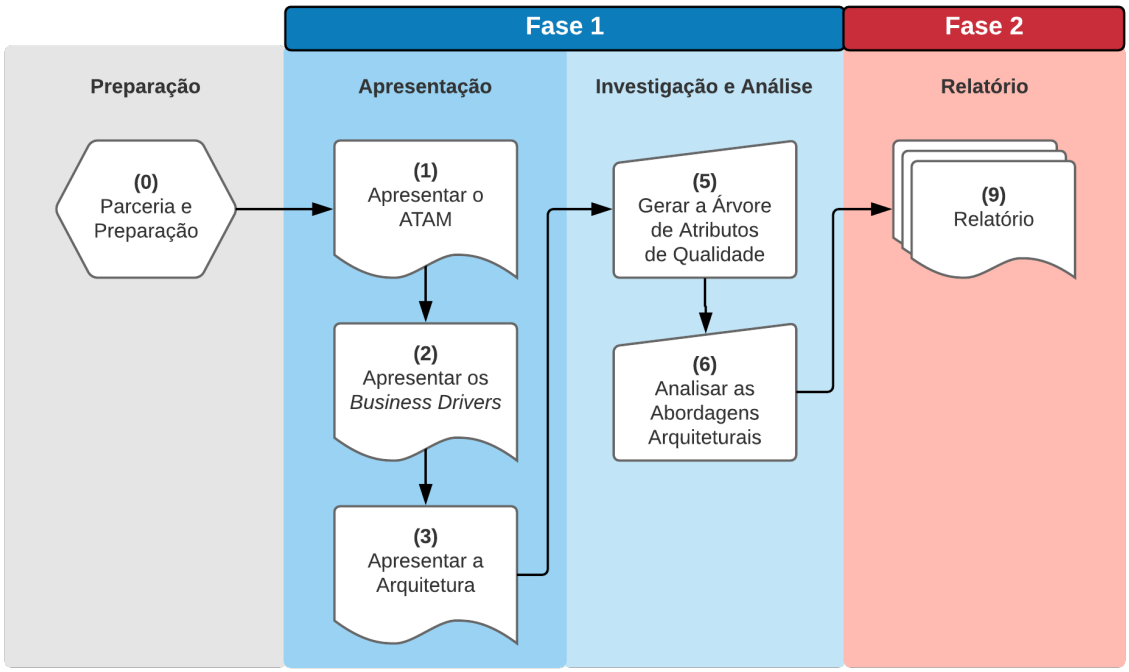
4.1 Avaliação da Arquitetura Atual

Feito o levantamento bibliográfico, identificou-se que o ATAM era o método que se adequava ao contexto do SUAP, no IFRN. Priorizou-se, então, a avaliação do atributo de qualidade de desempenho do sistema. Esse atributo será considerado por ser uma característica crucial para que o sistema dê o suporte esperado a seus usuários em suas atividades. Apesar de não formalizada, durante todo o processo houve a participação de membros das equipes de desenvolvimento e infra-estrutura do IFRN. Nessas reuniões foram indicados alguns módulos com possíveis problemas de desempenho, devido às reclamações recebidas dos usuários – além da real percepção de lentidão constatada por integrantes da própria equipe.

Devido à limitação de tempo da pesquisa e pela avaliação não ser aplicada a um sistema em concepção, optou-se por suprimir as seguintes etapas do processo tradicional:

- Identificação das abordagens arquiteturais – etapa 4 do processo formal: optou-se por suprimir esta etapa devido ao objetivo da avaliação. Neste estudo não buscava-se definir possíveis re-arquiteturas da aplicação, mas sim compreender se a arquitetura atual tem, de fato, impacto negativo no desempenho do SUAP;
- *Brainstorm* e cenários prioritários – etapa 7 do processo formal: esta etapa foi desconsiderada pois o estudo não busca apresentar uma proposta de mudança arquitetural. Na metodologia formal, aqui busca-se representar as maneiras pelas quais as partes interessadas esperam que o sistema seja usado e as maneiras pelas quais as espera-se que o sistema mude no futuro, o que não faz parte do escopo desta pesquisa.
- Análise das abordagens arquiteturais – etapa 8 do processo formal: esta é uma atividade complementar à anterior, que prevê a realização de testes com os cenários propostos. Visto que a etapa anterior não está contida no contexto da pesquisa, esta, consequentemente, também não está.

Figura 9 – Fases e etapas do ATAM aplicadas na análise do SUAP



Fonte: Elaborada pelo autor.

Dessa forma, a aplicação do método neste estudo resultou no fluxograma ilustrado na Figura 9, e considerou as seguintes atividades:

1. Apresentação do ATAM: como parte desta pesquisa, o conhecimento do processo de avaliação foi absorvido no estudo para elaboração do capítulo que descreve o método (Capítulo 2);

2. Apresentação dos guias de negócio (*business drivers*): como foco da avaliação, o objetivo aqui apresentado é garantir alto desempenho ao sistema sob demandas variáveis;

3. Apresentação da arquitetura: como já há uma arquitetura definida e aplicada ao sistema, esta execução buscou apenas compreender os elementos da arquitetura atual, servindo de base teórica para elaboração do diagrama exposto na Figura 8;

4. Geração da árvore de atributos de qualidade: etapa de grande relevância para o contexto da pesquisa, onde foi identificado o atributo de qualidade de maior importância para os testes realizados na subseção 4.2.1; e

5. Análise das abordagens arquiteturais: a partir do atributo identificado na etapa anterior, foi gerado um documento descrevendo cada cenário do atributo de desempenho.

6. Relatório: a explanação dos dados coletados e a própria seção 4.3 relatam a aplicação do ATAM, mesmo não havendo um relatório formal produzido.

4.2 Planejamento da Aplicação do ATAM

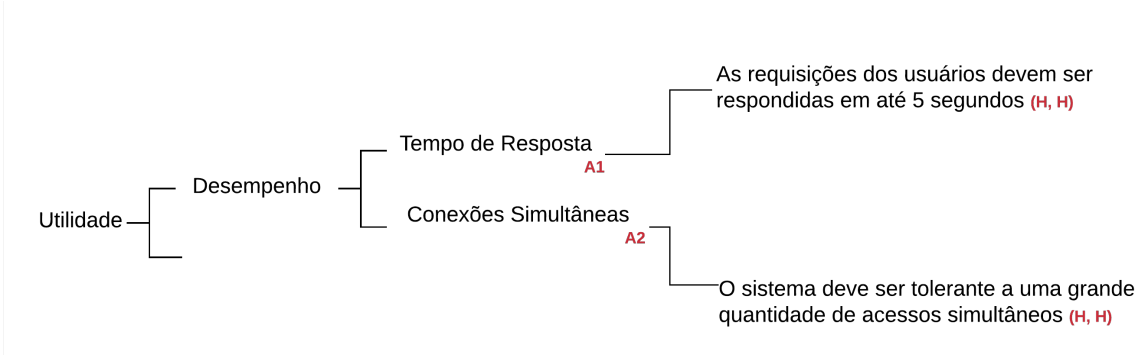
O fluxo tradicional de aplicação do ATAM prevê planejamento para sua aplicação. Neste plano, serão documentadas as atividades de geração da árvore de atributos de qualidade, bem como seus desdobramentos. Visto que as três primeiras etapas desta fase já foram executadas anteriormente (apresentação do ATAM e da arquitetura), a seguir serão detalhados os passos 4 (subseção 4.2.1) e 5 (subseção 4.2.2).

4.2.1 Geração da Árvore de Atributos de Qualidade

Feita a identificação do atributo, foi necessário um refinamento da análise, ilustrando a árvore de atributos de qualidade prevista para o SUAP (Figura 10). Para elaboração dessa árvore, foram considerados os objetivos do sistema, priorizando o grau de importância e dificuldade para se alcançar cada caracterização, respectivamente – como previsto na documentação do método.

Ao considerar o atributo de desempenho como o principal do sistema atualmente, foram identificadas, junto à equipe de desenvolvimento, as principais características que compõem esse parâmetro. Assim, foi considerado que o SUAP deveria apresentar resposta ao usuário em tempo satisfatório (desejável que ocorra em no máximo 5 segundos); e deveria suportar um número de conexões simultâneas no mínimo 2 vezes maior que a média real, considerando que eventos específicos podem disparar picos de acesso e buscando atender a constante expansão do sistema.

Figura 10 – Árvore de atributos de qualidade proposta para o SUAP



Fonte: Elaborada pelo autor.

4.2.2 Análise das Abordagens Arquiteturais

Após a elaboração da árvore de atributos, o ATAM prevê o detalhamento de cada atributo identificado. Na avaliação realizada, ao se optar por analisar o atributo de qualidade de desempenho, foram gerados os cenários A1 e A2, que serão detalhados ainda nesta seção. Para tanto, é necessário definir os fatores que serão considerados:

- Cenário: um cenário da árvore de utilidades gerada no *brainstorming* de cenários;
- Atributo: atributo de qualidade que será avaliado. Dentre os principais cenários sugeridos (desempenho, disponibilidade, modificabilidade e integridade, por exemplo), será considerado como alvo o atributo de desempenho;
- Ambiente: ambiente (desenvolvimento, testes ou produção, por exemplo) em que serão analisadas as abordagens arquiteturais, através da aplicação dos testes de desempenho detalhados no Capítulo 5.
- Estímulo: uma descrição precisa do estímulo ao atributo de qualidade (uma requisição, por exemplo);
- Resposta: detalhamento preciso da resposta do atributo de qualidade ao estímulo aplicado (tempo de resposta, baixa integridade, dentre outros);
- Decisão Arquitetural: lista de decisões arquiteturais que afetam a resposta do atributo de qualidade;
- Riscos/Não Riscos: descrição de variáveis conhecidas que podem (ou não) serem consideradas um risco para a decisão arquitetural;
- Sensibilidade: ponto de sensibilidade que corrobora a decisão arquitetural correlata;
- *Trade-off*: detalha, se houver, uma situação em que há conflito de escolha, onde uma decisão arquitetural acarreta um outro problema.

O método formal ainda prevê, na elaboração destas abordagens arquiteturais, um esboço de um diagrama de arquitetura. Como já dito, esta avaliação não objetiva uma avaliação focando em uma proposta arquitetural nova. Desta forma, optou-se por descrever o cenário da arquitetura atual neste espaço, como pode ser visto nos cenários descritos a seguir, na Tabela 1 e Tabela 2.

Tabela 1 – Análise do atributo de desempenho de tempo de resposta do sistema (A1)

Cenário:	A1 – Tempo de Resposta
Atributo:	Desempenho
Ambiente:	Teste
Estímulo:	Requisições ao servidor
Resposta:	O sistema deve apresentar uma resposta ao usuário em até 5 segundos

Decisão Arquitetural	**Risco / Não Risco**	**Sensibilidade**	***Trade-off***
Servidores de alto desempenho	R1	-	T1
Comunicação de alto desempenho na rede interna	R2	-	-
Comunicação de alto desempenho com o datacenter	R3	-	-
Monitoramento de temperatura do datacenter	-	S1	-

Raciocínio:	As decisões arquiteturais tomadas e listadas aqui deveriam garantir bom desempenho ao sistema.
Cenário Desejado:	É necessário, além das métricas atuais, monitorar processamento, uso de memória e de armazenamento do servidor, além de testar os tempos de resposta dos casos de teste propostos a cada nova funcionalidade implementada no SUAP.

- R1: não risco, pois o servidor para a aplicação conta com 8 *cores* de processamento e 24 GB de memória RAM; servidor virtual exclusivo para o banco de dados, com 16 *cores* de processamento e 24 GB de RAM – e ainda há capacidade para escalar verticalmente.
- T1: atualmente há uma infraestrutura que suporta ser escalada verticalmente. Contudo, futuramente pode ser necessário aplicar mais recursos para ampliar ainda mais as capacidades de hardware do data center.
- R2: não risco, já que a rede atual do IFRN dispõe de uma velocidade mínima de 100 Mbps para garantia do cumprimento dos requisitos do sistema.
- R3: não risco, visto que há um link de alta velocidade para conexão com o DB disponível.
- S1: monitoramento de temperatura dos servidores é realizada em tempo real.

Tabela 2 – Análise do atributo de desempenho sob altas demandas ao sistema (A2)

Cenário:	A2 – Conexões Simultâneas
Atributo:	Desempenho
Ambiente:	Teste
Estímulo:	Várias *threads* concorrentes acessando o mesmo recurso
Resposta:	O sistema não deve retornar uma quantidade de erros maior do que as requisições com sucesso

Decisão Arquitetural	**Risco / Não Risco**	**Sensibilidade**	***Trade-off***
Servidores de alto desempenho	R4	-	-
Comunicação de alto desempenho com o datacenter	R5	-	-

Raciocínio:	As decisões arquiteturais tomadas e listadas aqui não são suficientes para garantir bom desempenho ao sistema.
Cenário Desejado:	Solução para distribuição de carga entre servidores ou adoção de uma arquitetura elástica.

- R4: apesar de atualmente a infraestrutura dispor de hardware suficiente para uma ampliação emergencial, o monitoramento dessa necessidade não é realizado e esse incremento de hardware é feito manualmente.
- R5: a comunicação com o data center terá que ser incrementada para sustentar com a crescente demanda.

4.3 Considerações Finais

Neste capítulo foram detalhadas as etapas e atividades previstas na aplicação do ATAM. Em seu desenvolvimento, foram identificadas as etapas que superam o escopo e o contexto da pesquisa, optando-se assim por suprimi-las. Considerando isso, foi elaborada uma listagem com as atividades previstas para esta seção do trabalho. Em seguida, foi detalhado todo o planejamento da aplicação dessa primeira fase do método, gerando a árvores de atributos de qualidade e seus desdobramentos.

Considerando a árvore de atributos elaborada (Figura 5) e os cenários descritos na Tabela 1 e na Tabela 2, buscou-se na literatura formas de se medir o tempo de resposta e a capacidade de suportar requisições simultâneas de uma aplicação. Dessa forma, foi constatada a necessidade de submeter o SUAP a testes de desempenho.

O próximo capítulo trará, assim, o detalhamento dos testes de desempenho realizados. Considerando a necessidade iminente deste trabalho e do IFRN submeterem o SUAP a esses testes, esta pesquisa recebeu uma grande ênfase em seu planejamento e execução.

5 Testes de Desempenho

Após a aplicação do ATAM e coletados seus resultados, constatou-se a necessidade da aplicação de testes de desempenho para analisar tempo de resposta e capacidade máxima de usuários que o SUAP suporta. Deste modo, este capítulo apresenta os testes realizados, descrevendo a metodologia adotada, e discutindo os resultados alcançados.

5.1 Introdução

Este capítulo busca seguir a estrutura de plano de teste proposta por Naik e Tripathy (2008). Segundo os próprios autores, suas propostas concentram-se no conteúdo, não na forma. O conteúdo de um plano de teste é essencialmente como os testes foram criados, por que os testes foram criados e como eles serão executados. Esse plano é baseado no padrão ANSI/IEEE 829-1983, composto por Introdução; Casos de Uso considerados para os testes; Ambiente de testes; Estrutura dos testes; Estratégia de execução de teste; Resultados; e, por fim, Discussão e Considerações Finais.

Na sequência definimos o escopo dos testes realizados, apresentando seus objetivos, resultados esperados e critérios de conclusão.

Objetivos do Teste: coletar o tempo de resposta do sistema para funções de negócios sob um crescente volume de requisições. Como fruto da aplicação do ATAM, determinou-se que seriam necessários avaliar o tempo de resposta e a quantidade de erros registrados em cada faixa da escala de número de usuários ativos, em diferentes módulos do sistema. Com os dados resultantes, estima-se identificar os limites de desempenho do SUAP.

Resultados Esperados: obter dados suficientes para compreender os problemas de desempenho do SUAP; obter a média de tempos de resposta em cada caso de teste; obter o limite de usuários no qual o SUAP atende aos requisitos estabelecidos de 5 segundos de tempo de resposta;

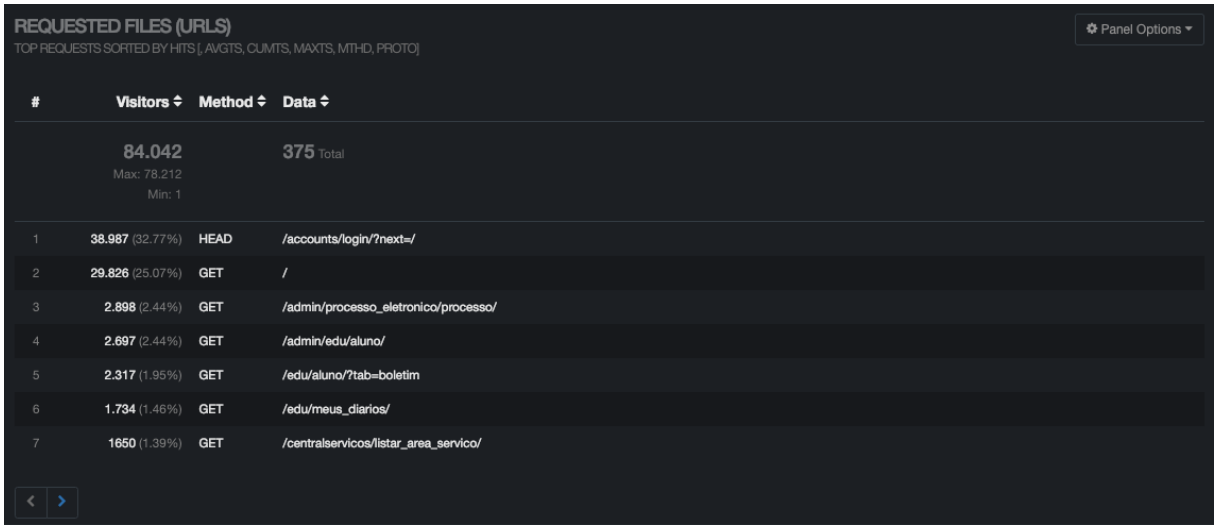
Critérios de Conclusão: o teste será encerrado ao se atingir o acesso de 300 usuários (*threads*) em cada uma das 10 execuções previstas para cada caso de teste. Em protótipos desenvolvidos para os testes de desempenho, observou-se que o número de 300 usuários seria suficiente para a análise desejada, já que a partir desse número todas as requisições passaram a ser negadas.

5.2 Casos de Uso considerados para os testes

Na sequência, serão descritos os casos de uso que foram escolhidos como alvos de teste. Esse levantamento foi feito baseado em reclamações dos usuários do sistema SUAP. Adicionalmente, realizamos uma análise do *log* do sistema identificando os casos de uso mais acessados de cada módulo. Serão testadas funcionalidades do sistema pertencentes aos módulos **adm** e **edu**, além da página inicial do usuário.

Para identificar os casos de uso que seriam relevantes para a pesquisa e que têm demanda real de usuários, foram analisados *logs* da aplicação referentes a um período de 9 meses (setembro de 2017 a junho de 2018), onde foram consultadas as páginas com mais acesso dos usuários e média de usuários simultâneos, dentre outros dados. A ferramenta utilizada para auxiliar no tratamento e visualização dessas informações foi o GoAccess[1] – a Figura 11 ilustra o resultado obtido com tal análise.

Figura 11 – Resultados obtidos com a análise de *log*



Fonte: Elaborada pelo autor.

Baseado na análise de *logs* foi possível identificar as URLs mais acessadas do SUAP – e assim sendo, potenciais focos de falhas de desempenho. Como a ferramenta exibe apenas sete resultados por página a o objetivo seria detalhar apenas cinco páginas mais acessadas, a figura ilustra apenas a primeira página da coleta. Com isso, pôde-se descrever as URLs, módulos e funções a serem consideradas no teste. Contudo, o primeiro resultado encontrado é, obviamente, a página de login do sistema. Como esta não traz consigo grandes elementos, optou-se por não incluí-la nos testes. Os módulos identificados foram, então: página inicial do usuário logado; módulo adm, acesso a processos eletrônicos;

[1] Analisador e visualizador interativo de *log* de código aberto. Disponível em <https://goaccess.io>, fornece estatísticas HTTP para administradores de sistema que exigem um relatório visual.

módulo edu, acesso a listagem de alunos; módulo edu, acesso a boletim do aluno; e modulo edu, acesso aos diários do professor.

Entretanto, analisou-se junto à equipe de desenvolvimento, a necessidade de se avaliar o desempenho da página que exibe um processo eletrônico específico. Esses processos geralmente contêm muitos documentos vinculados e são fontes constante de reclamação dos usuários sobre o desempenho. Além disso, essas páginas têm em sua própria URL o número do processo, o que não contribui para que sejam identificadas como funções mais acessadas do sistema, já que cada processo possui um número diferente.

A seguir, serão detalhados cada caso de uso nas tabelas 3 a 8.

Tabela 3 – Detalhamento do Caso de Teste CT001

Caso de Teste (*id*):	• CT001 – Realizar acesso ao *index* do SUAP.
Objetivo do Teste:	• Verificar o tempo de resposta da página inicial do sistema no primeiro acesso do usuário (assim desconsidera-se o uso de cache da aplicação).
Passos:	1. Elaborar script no JMeter para acessar o endereço <https://suap.ifrn.edu.br> utilizando matrículas ativas de servidores 2. Executar o script de teste seguindo os parâmetros definidos.
Critérios de Êxito:	• O script executar as 300 *threads* programadas.

Tabela 4 – Detalhamento do Caso de Teste CT002

Caso de Teste (*id*):	• CT002 – Visualizar Alunos (módulo **edu**).
Objetivo do Teste:	• Verificar o tempo de resposta para acesso à página de busca/listagem de todos os alunos cadastrados na instituição, subsistema (módulo) Ensino, função de visualizar todos os alunos.
Passos:	1. Elaborar script no JMeter para acessar o SUAP utilizando matrículas de docentes 2. Executar o script, que acessa a página <https://suap.ifrn.edu.br/admin/edu/aluno/> 3. Executar o script de teste seguindo os parâmetros definidos.
Critérios de Êxito:	• O script executar as 300 *threads* programadas.

Tabela 5 – Detalhamento do Caso de Teste CT003

Caso de Teste (*id*):	• CT003 – Visualizar Meus Diários (módulo **edu**).
Objetivo do Teste:	• Verificar o tempo de resposta da página de consulta de diários pelo professor, subsistema (módulo) Ensino, visualizar Meus Diários.
Passos:	1. Elaborar script no JMeter para acessar o SUAP utilizando matrículas de docentes 2. Executar o script, que acessa a página <https://suap.ifrn.edu.br/edu/meus_diarios/> 3. Executar o script de teste seguindo os parâmetros definidos.
Critérios de Êxito:	• O script executar as 300 *threads* programadas.

Tabela 6 – Detalhamento do Caso de Teste CT004

Caso de Teste (*id*):	• CT004 – Visualizar Todos os Processos (módulo **adm**).
Objetivo do Teste:	• Verificar o tempo de resposta da página para listagem de todos os processos eletrônicos cadastrados na instituição, subsistema (módulo) Administração, função de visualizar todos os processos.
Passos:	1. Elaborar script no JMeter para acessar o SUAP utilizando matrículas de docentes 2. Executar o script, que acessa a página <https://suap.ifrn.edu.br/admin/processo_eletronico/processo/?tab=tab_any_data> 3. Executar o script de teste seguindo os parâmetros definidos.
Critérios de Êxito:	• O script executar as 300 *threads* programadas.

Tabela 7 – Detalhamento do Caso de Teste CT005

Caso de Teste (*id*):	• CT005 – Consultar Processo (módulo **adm**).
Objetivo do Teste:	• Verificar o tempo de resposta da página para exibição completa de um determinado processo eletrônico cadastrados na instituição, subsistema (módulo) Administração, função de visualizar um processo.
Passos:	1. Elaborar script no JMeter para acessar o SUAP utilizando matrículas de docentes 2. Executar o script, que acessa a página <https://suap.ifrn.edu.br/processo_eletronico/imprimir_processo/12791/> 3. Executar o script de teste seguindo os parâmetros definidos.
Critérios de Êxito:	• O script executar as 300 *threads* programadas.

Tabela 8 – Detalhamento do Caso de Teste CT006

Caso de Teste (*id*):	• CT006 – Visualizar Histórico (módulo **edu**).
Objetivo do Teste:	• Verificar o tempo de resposta da página de consulta de histórico pelo aluno, subsistema (módulo) Ensino, visualizar Boletins.
Passos:	1. Elaborar script no JMeter para acessar o SUAP utilizando matrículas de discentes 2. Executar o script, que acessa a página <https://suap.ifrn.edu.br/edu/aluno/?tab=historico> 3. Executar o script de teste seguindo os parâmetros definidos.
Critérios de Êxito:	• O script executar as 300 *threads* programadas.

Dessa forma, todos os testes foram elaborados privilegiando aspectos de desempenho, com foco na análise do tempo de resposta e quantidade de erro nas requisições. Para tanto, será considerada a resposta do sistema em cada caso descrito quando estiver carregado com até 300 usuários com log on efetuado – 1 usuário inicial, com aumento gradativo. Esse limite de 3 centenas de usuários foi adotado por, em um primeiro protótipo de testes, ter sido considerado o número de até 500 usuários, mas acima de 300 todas as requisições passaram a ser negadas. Ao investigar a causa dessa limitação, foi identificado que tratava-se de uma configuração no servidor de aplicação, que delimitava a infraestrutura de software a esse número.

5.3 Ambiente de testes

Como ambiente para os testes, foi feito um clone do sistema e do banco de dados da aplicação em máquinas virtuais (VM's), buscando reproduzir o ambiente de produção o mais fidedignamente possível. Esta versão de testes do SUAP reproduz toda a arquitetura ilustrada na Figura 8.

Na Tabela 9 são descritos os *hostnames* definidos para cada VM, bem como os recursos alocados e a função de cada máquina. Na última coluna, quando for o caso, ainda são listados os principais softwares instalados e suas versões.

Tabela 9 – Recursos alocados para a infraestrutura de testes

Nome da VM	**Recursos**	**Função**
mpes-suap01	Cores: 16 (2.2 GHz) Memória: 32 GB Disco: 40GB	SUAP (clone da aplicação)
mpes-db	Cores: 32 (2.2 GHz) Memória: 32 GB Disco: 160GB	Banco de Dados (PostgreSQL 9.6)
mpes-load	Cores: 32 (2.2 GHz) Memória: 32 GB Disco: 40GB	Testes de Desempenho (JMeter 4.0)

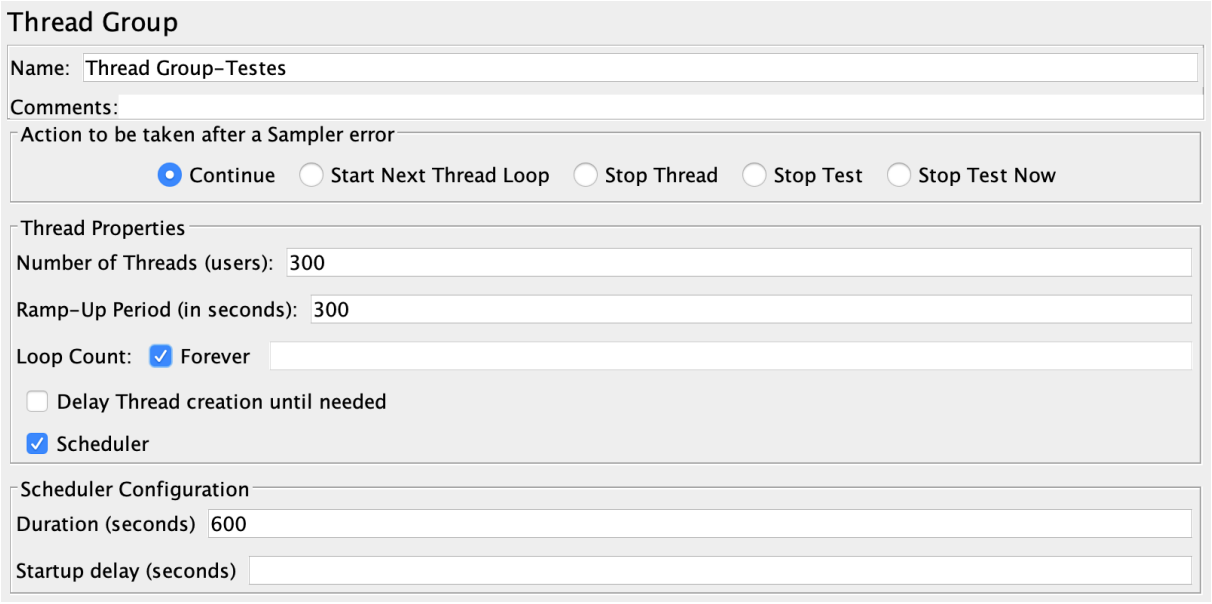
Detalhando cada máquina, a VM mpes-suap01 possui os mesmos recursos do SUAP oficial de produção. Na VM mpes-db está instalado um banco de dados PostgreSQL[2] 4, com os mesmos ajustes (*tunnings*) utilizados em produção e com uma cópia[3] do banco de dados utilizado pela equipe de desenvolvimento do SUAP no IFRN (dados mascarados). Por fim, a VM mpes-load é a responsável por executar os scripts de teste através de uma instância do JMeter instalada.

5.4 Estrutura dos testes

Buscando facilitar a escrita dos casos de testes, o ambiente gráfico do JMeter foi utilizado. As configurações gerais são comuns aos seis casos de teste elaborados, com exceção apenas para a URL acessada e o perfil dos usuários do teste (servidor ou aluno da instituição, indicado pelo arquivo .csv atribuído para cada teste). Na figura que ilustra a configuração do *Thread Group* (Figura 12), pode-se observar cada usuário a ser incrementado no teste, tratado como *thread* pelo JMeter. Além das 300 *threads* (usuários), é necessário indicar o tempo de *Ramp-Up*. Essa propriedade indica que 1 novo acesso de usuário será disparado a cada segundo (300 *Ramp-up* divido pelo número de *threads* = 1 *thread* por segundo, até o limite de 300 usuários).

[2] <https://www.postgresql.org>
[3] Cópia atualizada em 31 de julho de 2018.

Figura 12 – Configurações do *Thread Group* do JMeter

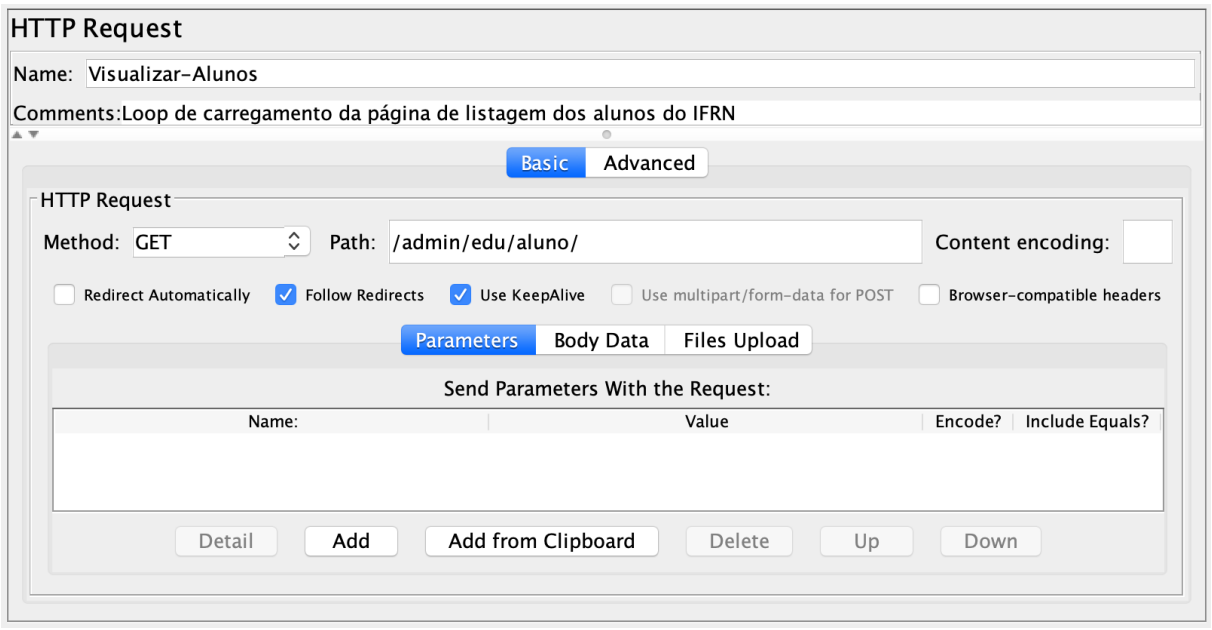


Fonte: Elaborada pelo autor.

Já a propriedade *scheduler* possibilita indicar o tempo total do teste. Como os casos desenvolvidos por este trabalho consideram 300 segundos para a criação de todas as *threads* em cada caso, o tempo total atribuído para cada teste foi de 600 segundos, suficiente para todos os usuários ficarem ativos e executem as ações previstas. Outra opção que foi assinalada foi a de *Loop Count*. Com este atributo ativo, as *threads* mantêm sua execução, necessário para a carga real ser mantida.

Na parte de configuração das requisições, cada caso de teste teve apenas o método assinalado (em todos os casos como *GET*) e o *Path* configurado (que é relativo às URL's que serão testadas). A figura 13 demonstra tais configurações.

Figura 13 – Configurações do *HTTP Request* do JMeter



Fonte: Elaborada pelo autor.

Após a validação dos casos de teste, o arquivo do plano gerado pela aplicação (extensão .jmx) foi copiado para o servidor mpes-load para serem executados em um ambiente não-gráfico. Seguindo as boas práticas recomendadas pelo JMeter 6 e objetivando obter resultados mais puros, procedimentos como a desativação dos *listeners* ao executar o teste no servidor de carga foram realizadas. Isso diminui as chances do teste ser comprometido com processamentos desnecessários, como ambiente gráfico, geração de estatísticas, e latência de rede (teste realizado localmente, mas apontando para um servidor remoto no datacenter, por exemplo). Os testes realizados através da ferramenta geram um arquivo no formato .csv, que, ao finalizar o teste, são processados para gerar uma página html com estatísticas, gráficos, entre outras informações.

Como ferramenta complementar para a elaboração dos scripts de teste, foi utilizado o BlazeMeter. Este *plug-in* para o Google Chrome registra as ações do navegador, convertendo-o em um script JMeter[4]. Em sua versão 4.0, essa ferramenta de testes possibilita a simulação de n acessos a uma url, por exemplo.

Dessa forma, pode-se considerar que o ambiente de testes disponibilizado é uma representação adequada do ambiente de produção.

[4] Aplicação **Java** de código aberto, projetada para carregar o comportamento funcional do teste e medir o desempenho. Originalmente projetado para testar aplicativos da web, vem se expandindo para executar outras funções de teste. Disponível em <https://jmeter.apache.org>

5.5 Estratégia de execução de teste

Para se obter um valor final do tempo de resposta e de quantidade de erros dos testes, cada iteração foi executada dez vezes. Ao final dos registros, foram calculadas média, desvio padrão e mediana dos tempos de resposta e do número de erros registrados, obtendo-se o valor final em cada escala de requisições. Apesar do incremento e do registro de uma *thread* a cada segundo, para esta análise serão coletados apenas os resultados em escalas de 25 usuários, buscando promover uma visualização mais fácil desses dados.

Para início dos testes foi escrito um script no servidor, executando o arquivo JMeter (.jmx) programado. Cada arquivo é referente a um caso de teste, sendo rodado, sequencialmente, dez vezes através desse comando. Para simular o acesso a diversos módulos da aplicação, foi programado log in no sistema com matrículas ativas de usuários diversos. Nos cinco primeiros casos, os testes foram implementados utilizando um arquivo .csv, contendo 1276 matrículas distintas de usuários ativos do IFRN. Considerado que seriam realizados testes em funcionalidades exclusivas das atividades docentes da instituição – e que não trariam prejuízo aos demais casos de teste, todas as matrículas utilizadas foram de professores. Já para o último Casos de Teste, por ser tratar de uma funcionalidade exclusiva para alunos, utilizou-se uma base atual com todos os alunos que possuem matrícula ativa na instituição. Por questões de privacidade e segurança da informação, os dados das bases de usuários utilizados nos testes foram mascarados e todos receberam uma senha aleatória única, redefinindo a respectiva coluna das bases .csv.

Foram simuladas cargas a partir de um número de 1 cliente, até o limite de 300. Apesar de o IFRN possuir mais de 35 mil usuários atualmente, ao analisar os *logs* da aplicação, foi identificada uma média de 100 usuários simultâneos.

Após o término da execução de cada caso de teste, foi feito um tratamento com os dados coletados com auxílio das ferramentas Anaconda Navigator[5] e Jupyter Notebook[6]. Dessa forma, pôde-se filtrar apenas os dados registrados na escala interessada de usuários, coletando os tempos de resposta e quantidade de erros nas requisições.

5.6 Resultados

A seguir, será apresentado resultados em cada caso de teste executado. Os resultados completos podem ser consultados em <https://github.com/tarsolatorraca/suap-load-tests>. Os resultados finais de cada caso de teste serão apresentados na próxima seção, em duas formas complementares: tabela contendo média, desvio padrão e mediana das 10 amostras coletadas; e em forma gráfica, onde serão representados, pela média, a evolução do tempo de resposta e do número de erros coletados com o crescimento de *threads* ativas – ao

[5] Disponível em <https://anaconda.org/anaconda/anaconda-navigator>
[6] Disponível em <http://jupyter.org/>

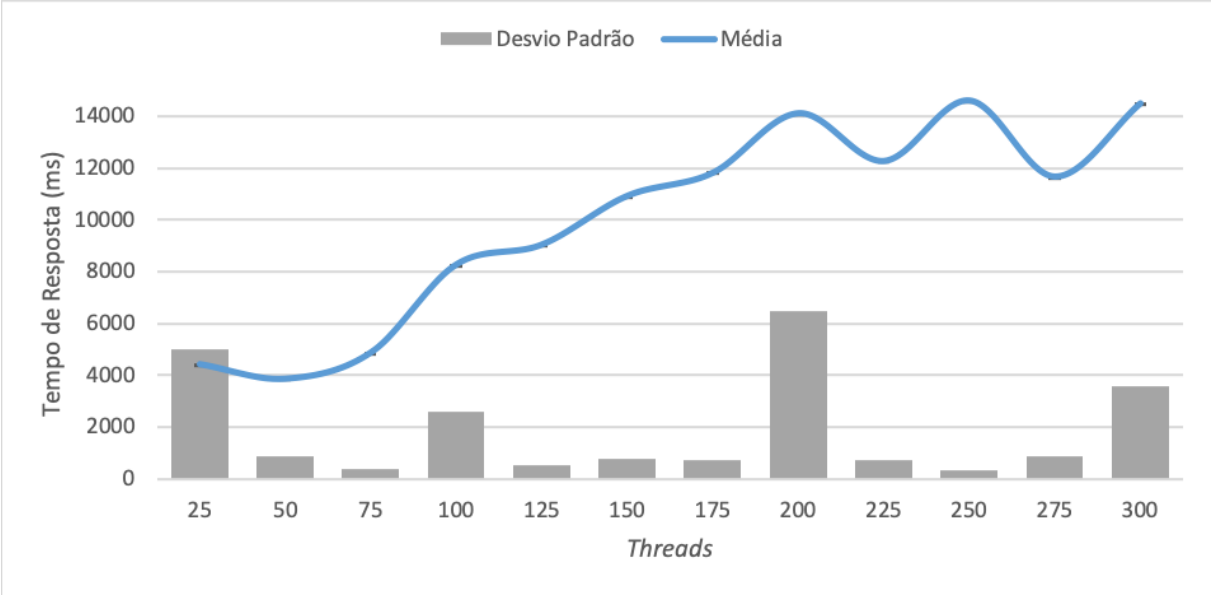
representar os resultados através da média, também faz-se necessário expor o desvio padrão registrado.

5.6.1 CT001: Acesso ao *index*

Como já tratado anteriormente, o acesso à página inicial do usuário é considerada como uma das funcionalidades mais lentas do sistema. Para carregar todas as suas informações, são realizadas diversas consultas simultâneas, o que estima-se que venha prejudicando seu desempenho.

Os resultados são apresentados na Tabela 11 e nas Figuras 14 (tempo de resposta variando número de usuários simultâneos) e 15 (quantidade de erros por número de requisições).

Figura 14 – CT001: *threads* x tempo de resposta



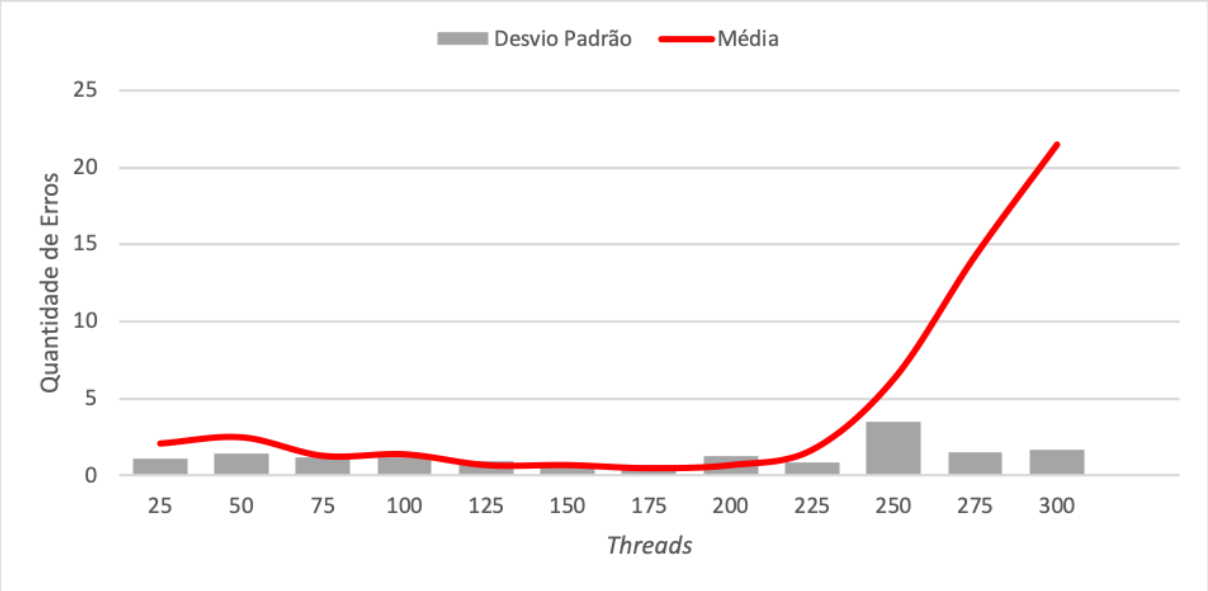
Fonte: Elaborada pelo autor.

Com relação ao tempo de resposta, este caso apresentou respostas em até 5 segundos quando submetidos a até 75 usuários (*thread*). Acima desse número, o tempo de resposta, quase que duplicou – a média de resposta para 75 usuários foi de 4873,51 milissegundos (ms), enquanto que para 100 usuários chegou à média final de 8271,42 ms, por exemplo. Visto a média real atual de 100 usuários simultâneos, isso já pode ser considerado um problema, dada a resposta esperada do cenário A1, que considera ideal até 5 segundos (vide Tabela 1).

A partir de 100 e até 200 usuários, o comportamento da aplicação também apresentou piora significativa, quase que duplicando o tempo de resposta dentro dessa faixa. Contudo, ao atingir 200 usuários, o tempo de resposta estabiliza entre 12 e 14 segundos,

reflexo do aumento da quantidade de requisições negadas, que pode ser observado no gráfico registrado pela Figura 15.

Figura 15 – CT001: *threads* x erros



Fonte: Elaborada pelo autor.

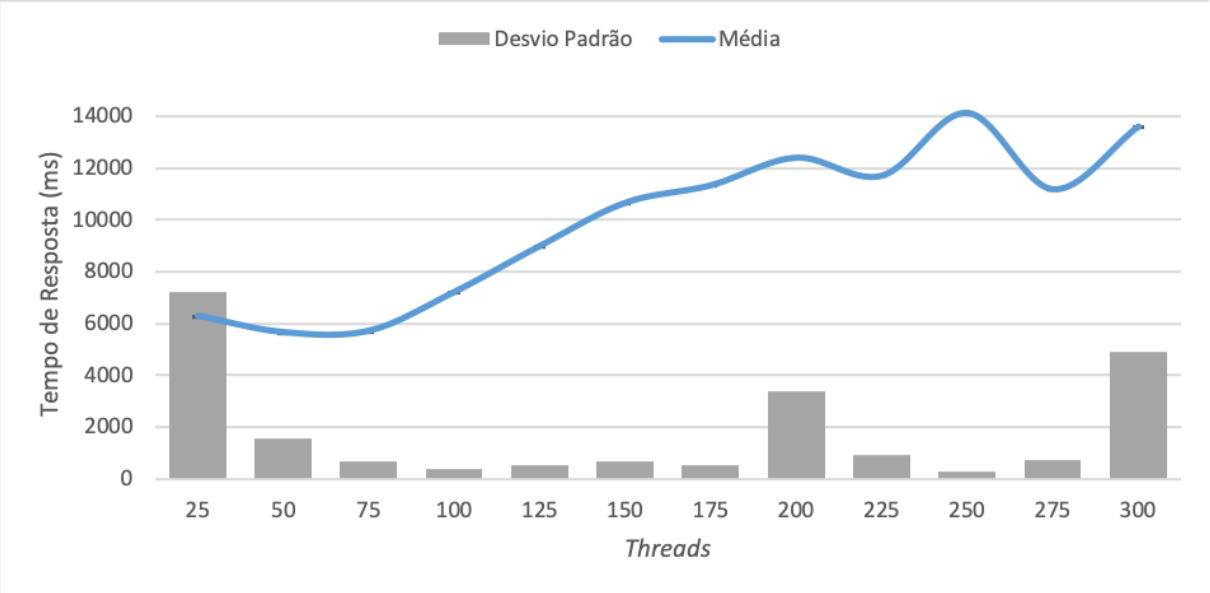
Com essa análise, percebe-se que o problema de desempenho se manifesta bem antes da quantidade de erros crescer. Os tempos de resposta superam os 5 segundos definidos pela aplicação do ATAM com apenas 75 usuários simultâneos. Pelo limite definido na infraestrutura de software ser de 300 conexões simultâneas, ao atingir a faixa de 250 usuários, o sistema começa a recusar as solicitações. Estes resultados indicam que o sistema apresenta problemas de desempenho independente dos limites impostos pelos recursos disponíveis por sua infraestrutura.

5.6.2 CT002: Visualizar Alunos

Este caso de teste buscou verificar a página de listagem de todos os alunos cadastrados no SUAP. Esta funcionalidade tem grande uso pelas secretarias acadêmicas e pelos próprios professores ao buscar informações gerais de um determinado discente.

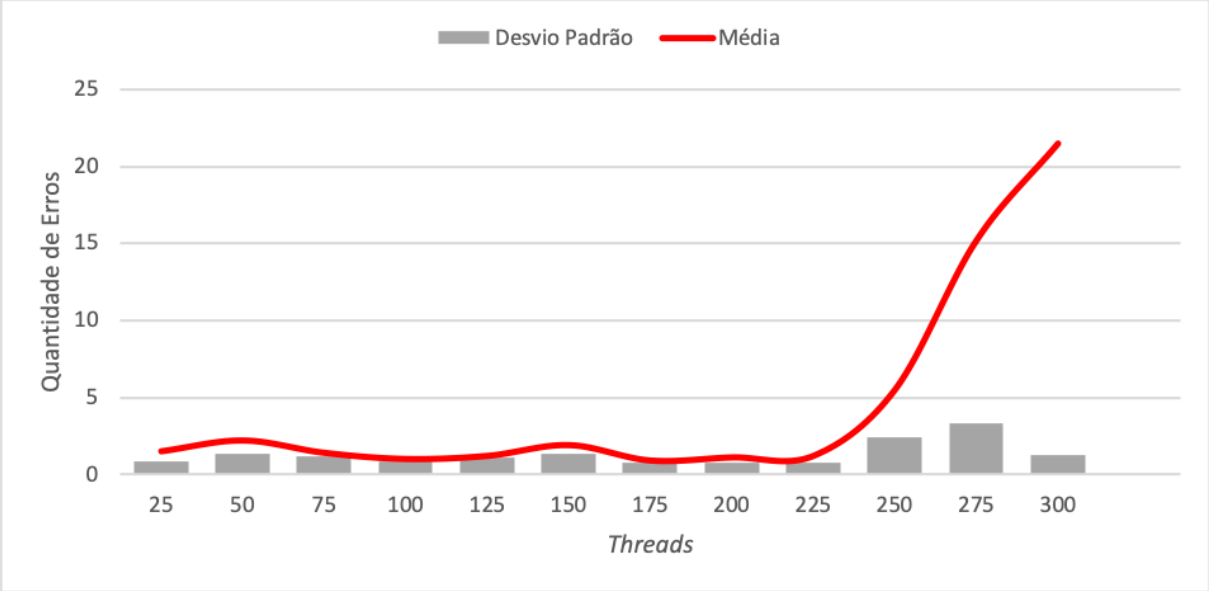
Os gráficos a seguir ilustram os resultados médios e medianos do CT002, bem como a curva de desvio padrão calculada.

Figura 16 – CT002: *threads* x tempo de resposta



Fonte: Elaborada pelo autor.

Figura 17 – CT002: *threads* x erros



Fonte: Elaborada pelo autor.

Diferentemente do caso anterior, o CT002 apresentou problemas de desempenho com um número menor de *threads*. Com apenas cinquenta usuários, a média de tempos de resposta foi superior aos desejáveis 5 segundos. Com 250 usuários houve uma média de 14 segundos para resposta, o que configura um desempenho ruim considerando a baixa complexidade desta operação.

Já as requisições com erro seguiram o mesmo padrão observado no caso anterior, com uma média de 15,20 requisições com erro a partir de 251 usuários, atingindo 21,50 erros médios até o número de 300 usuários.

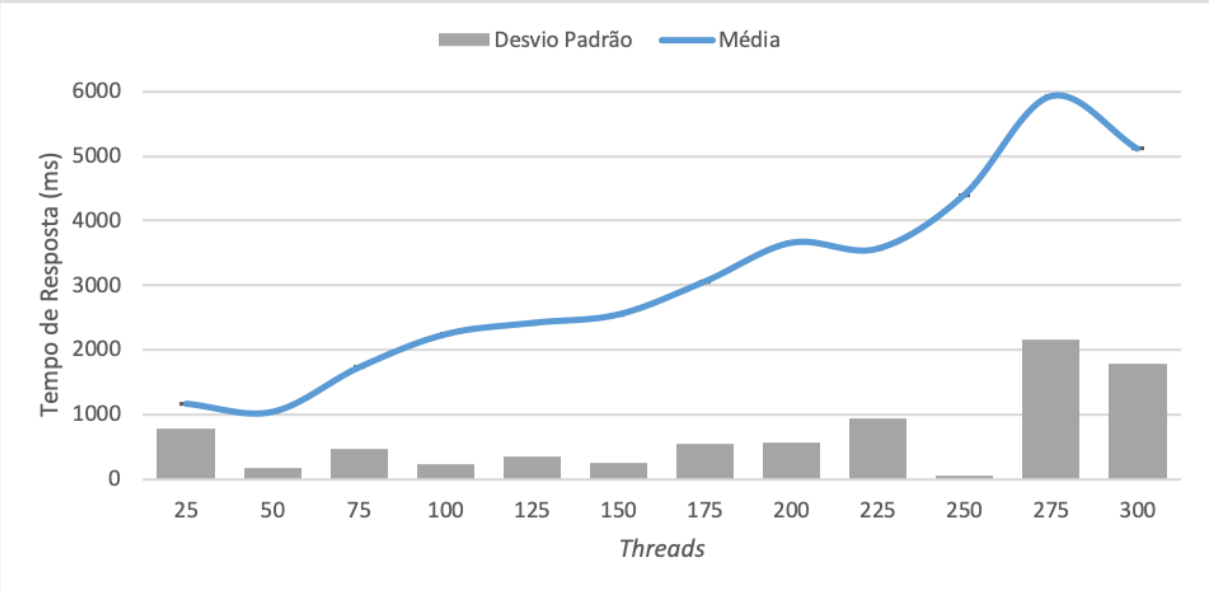
Seguindo o caso de teste anterior, novamente percebe-se que o problema de desempenho se manifesta bem antes da quantidade de erros crescer. Em todas as escalas do números de *threads* ativas os tempos de resposta superam os 5 segundos e, ao atingir a faixa de 250 usuários, o sistema começa a recusar as solicitações. Outra vez, observa-se que o impacto da limitação da infraestrutura de software, indicando que o sistema apresenta problemas de desempenho independente dos limites impostos pelos recursos disponíveis por sua infraestrutura.

5.6.3 CT003: Visualizar Meus Diários

Esta funcionalidade foi a única testada que é de acesso exclusivo dos docentes. Nela são executadas algumas consultas a outros objetos, como informações de alunos, por exemplo.

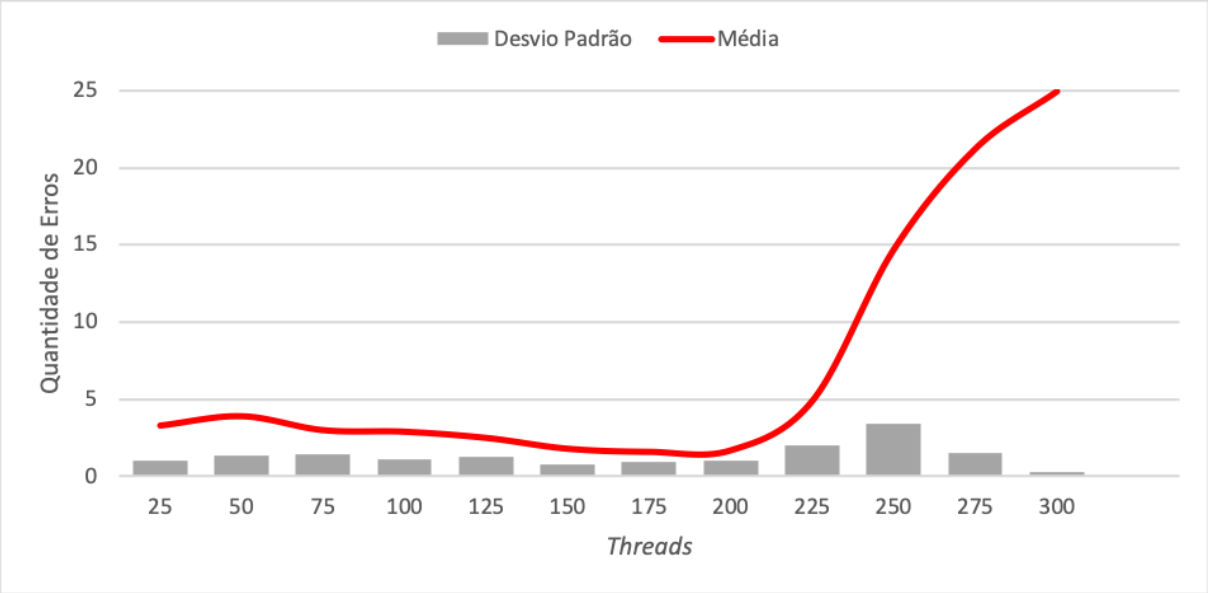
A tabela e figuras abaixo apresentam os resultados registrados, seguindo o padrão dos casos de teste anteriores.

Figura 18 – CT003: *threads* x tempo de resposta



Fonte: Elaborada pelo autor.

Figura 19 – CT003: *threads* x erros



Fonte: Elaborada pelo autor.

Dentre os casos de teste propostos, este foi o que apresentou desempenho consideravelmente superior. Apesar de o número de erros ter se manifestado seguindo o mesmo padrão observado nos casos anteriores, os tempos de resposta apresentaram expressiva melhora, se mantendo abaixo de 5 segundos até cerca de 250 usuários simultâneos. Neste caso, tempos de resposta superiores foram observados apenas na faixa de 275 *threads*, registrando 5931,24 ms de média neste quadrante.

Já no outro quesito avaliado, o de erros por quantidade de requisições, o desempenho mostrou-se marginalmente inferior aos calculados até aqui. Entre 226 e 250 usuários, foi registrado uma média de 14,70 requisições com erro, indicando mais uma vez a limitação que a infraestrutura de software exerce no comportamento do SUAP, quando submetido a um alto número de conexões simultâneas.

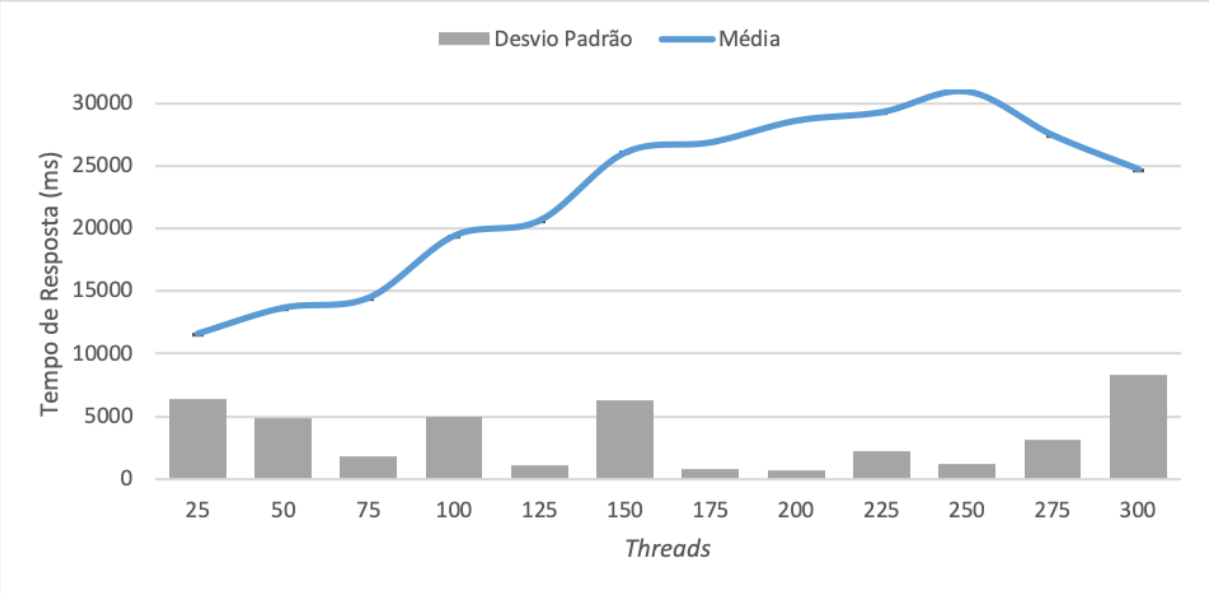
Estes resultados indicam que a funcionalidade exercitada por este caso de uso apresenta desempenho adequado à aplicação do ATAM, sendo impactada majoritariamente por restrições de infraestrutura.

5.6.4 CT004: Visualizar Todos os Processos

Uma das novas principais funcionalidades do SUAP, o módulo de processo eletrônico teve seu desempenho testado neste caso de teste. Através desse módulo, pode-se ter acesso a todos os processos que tramitam na Instituição, que atualmente são todos abertos e documentados eletronicamente.

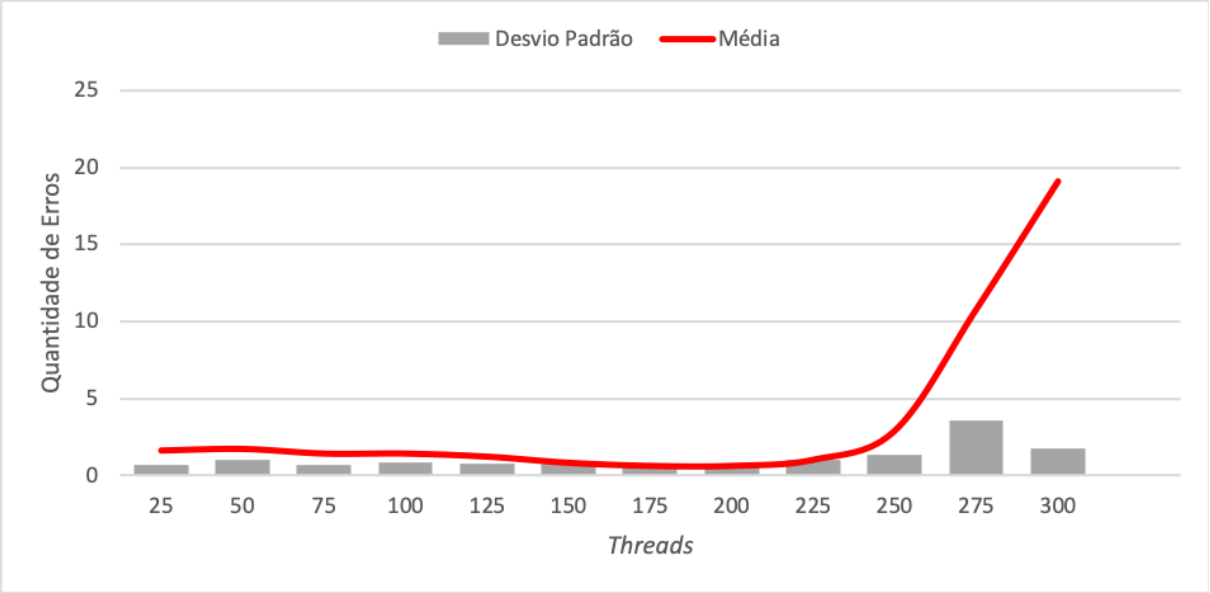
Os dados coletados são disponibilizados a seguir, na Tabela 14 e Figuras 20 e 21.

Figura 20 – CT004: *threads* x tempo de resposta



Fonte: Elaborada pelo autor.

Figura 21 – CT004: *threads* x erros



Fonte: Elaborada pelo autor.

Assim como no CT002, este caso de teste não atingiu o desempenho satisfatório de respostas em até 5 segundos em nenhum dos registros. Com apenas 25 *threads* já foi registrada uma média de 11579,94 ms, atingindo o pico de 30928,56 ms com 300 usuários ativos.

O desempenho relativo aos erros continuou constante, no entanto apresentando os melhores resultados dentre todos os casos de teste. O número de requisições com erro

superou as com sucesso apenas no última escala do teste, compreendida entre 276 e 300 *threads* – neste quadrante foi registrada média de 19,10 erros, indicando novamente que o problema provém da limitação da infraestrutura de software do aplicação.

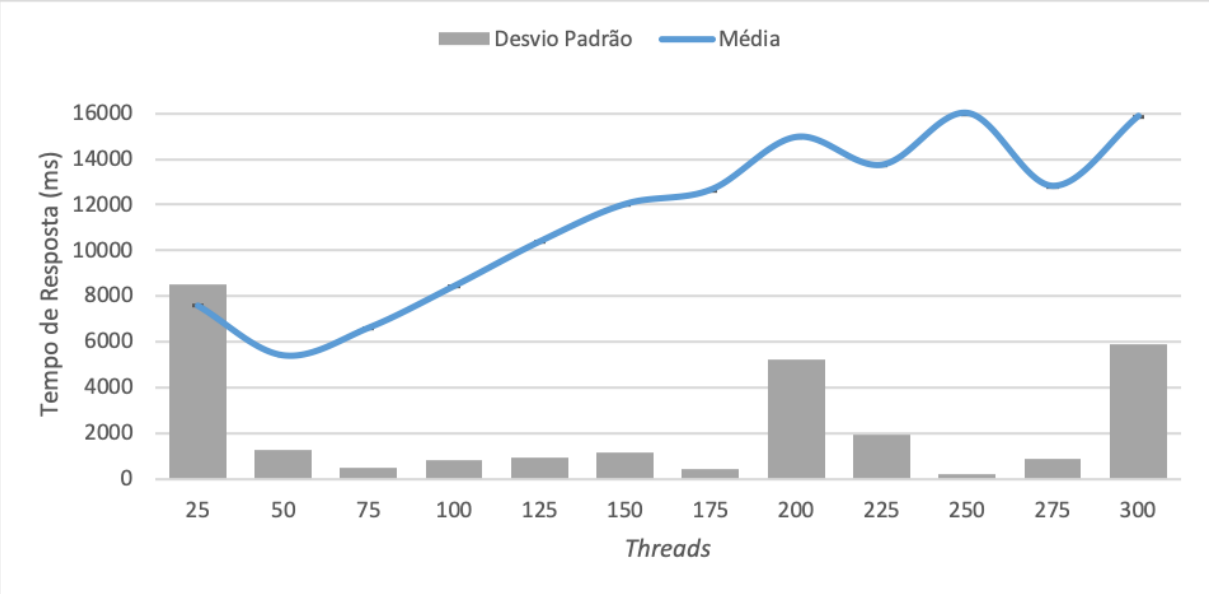
Visto que a quantidade de erros cresce apenas próximo ao número limitado pelo software, pode-se considerar que este caso segue os padrões observados no CT002. Os erros retornados manifestam-se tardiamente, mas o tempo de resposta não atinge a métrica definida desde a escala inicial de *threads*.

5.6.5 CT005: Consultar Processo

Este caso executa o acesso a um determinado processo eletrônico. Para o teste, considerou-se um processo específico, fazendo com que todos os scripts criados acessassem esse mesmo processo. Visto que um processo pode ter vários documentos vinculados (memorandos, despachos, etc), é esperado que seu desempenho seja inferior aos já testados.

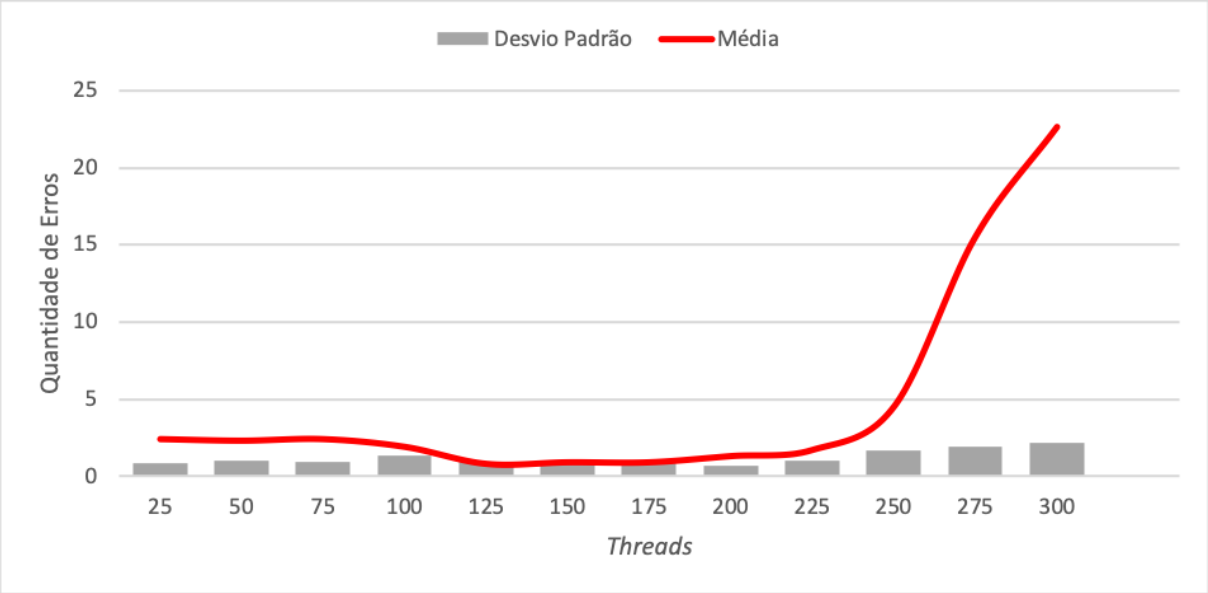
Tabela 15, Figura 22 e Figura 23 apresentam os resultados coletados.

Figura 22 – CT005: *threads* x tempo de resposta



Fonte: Elaborada pelo autor.

Figura 23 – CT005: *threads* x erros



Fonte: Elaborada pelo autor.

Este é mais um caso de teste que não atingiu a métrica esperada pela Tabela 1. Todas as médias registradas aqui superaram os 5 segundos esperados.

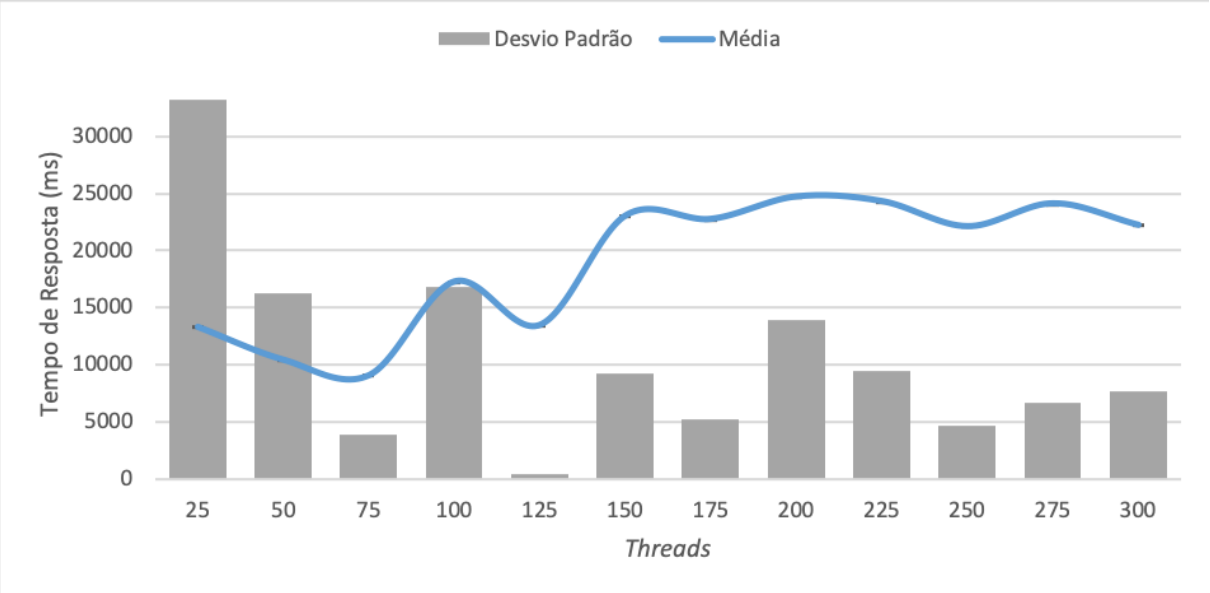
Ainda é necessário considerar que o processo considerado para o teste possui poucos documentos vinculados, o que não é um padrão na instituição. Processos abertos via SUAP possuem tamanhos e quantidade de documentos anexados altamente heterogêneos, o que nos leva à conclusão de que este tempo pode piorar bastante em processos maiores.

Os erros registrados, mais uma vez, seguem a média observada nos casos CT002 e CT004, com crescimento vertiginoso apenas a partir do antepenúltimo quadrante dos registros. Esse registro novamente indica que a limitação da infraestrutura de software atua diretamente na quantidade de erros que o sistema apresenta sob altas demandas.

5.6.6 CT006: Visualizar Histórico

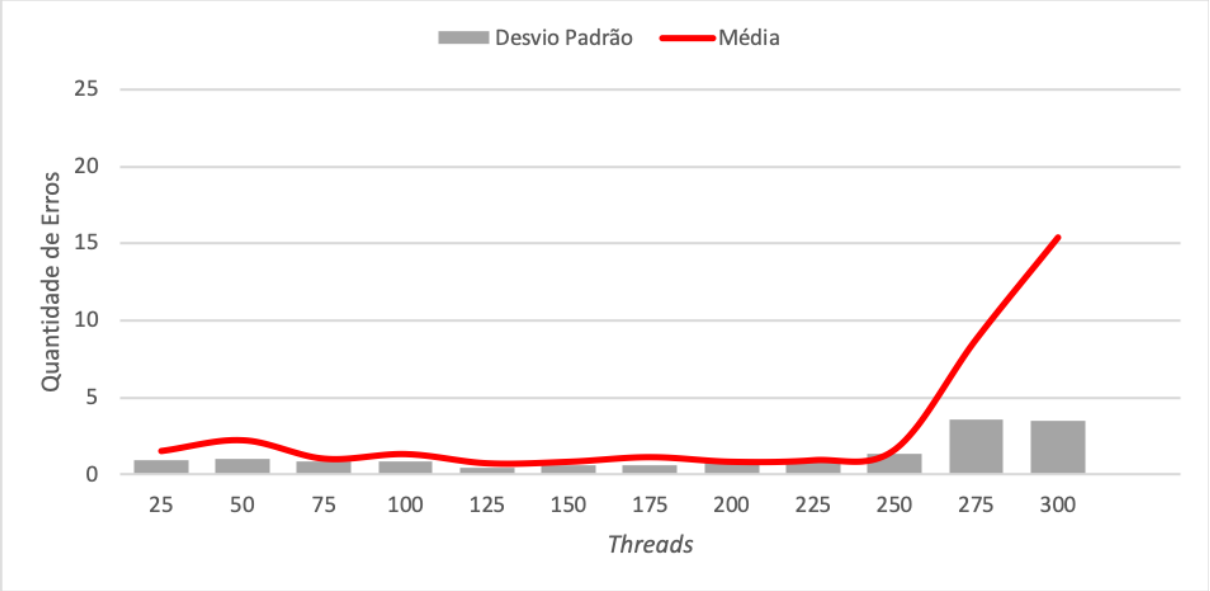
Este é considerado o principal caso de uso do sistema para os alunos, onde eles podem visualizar também seu boletim. Essa funcionalidade possibilita ao discente o acesso a todo seu histórico acadêmico, dados de disciplinas e trufas, por exemplo. Na sequência, pode-se verificar o desempenho deste caso de teste.

Figura 24 – CT006: *threads* x tempo de resposta



Fonte: Elaborada pelo autor.

Figura 25 – CT006: *threads* x erros



Fonte: Elaborada pelo autor.

Mais uma vez não foi atingida a métrica de resposta em até 5 segundos para nenhuma das escalas do teste, seguindo o que foi observado nos casos CT002, CT004 e CT005. Contudo, apesar de não desempenhar bem, este caso de teste foi o que suportou maior quantidade de acessos simultâneos, visto que ele foi o que retornou menos erros nas faixas finais da escala. Mesmo assim, percebe-se a limitação do software disparando erros com números de acessos próximos ao limite de 300 *threads*.

Apesar de nas faixas iniciais este teste ter apresentado resultados semelhantes aos já observados nos casos CT002, CT004 e CT005, a curva do gráfico da Figura 24 foi a única a não seguir um padrão linear crescente, apresentando certa irregularidade (ondulação). Os resultados obtidos também apresentaram alto desvio padrão, aparentando constância apenas nas faixas de 75 e 125 *threads*, que tiveram tempo de resposta médio de 9120,62 ms e 13512,66 ms, respectivamente.

5.7 Discussão e Considerações Finais

Através dos casos de teste propostos, buscou-se mensurar a capacidade da aplicação em sustentar um alto grau de operações simultâneas, proporcionando o entendimento da capacidade atual do SUAP.

Para tanto, foram testadas funcionalidades do sistema pertencentes aos módulos **adm** e **edu**, além da página inicial do usuário. Com foco em desempenho e considerando a análise conjunta com a equipe de desenvolvimento, foi considerado que o SUAP deveria apresentar resposta ao usuário em até 5 segundos e deveria suportar um número de pelo menos 200 conexões simultâneas. Dessa forma, foram desenvolvidos e aplicados os seis casos de testes tratados neste capítulo.

Os teste previstos neste plano limitaram-se a registrar o tempo de resposta e requisições negadas de cada caso de teste até a execução de 300 *threads*. Não foram consideradas possibilidades de mais acessos simultâneos ou registro de outras métricas, como consumo de processamento, memória, armazenamento ou energia, por exemplo. Os testes também foram limitados a analisar apenas seis casos de uso do SUAP, de 2 módulos distintos. Por fim, todos os testes foram planejados e executados em ambiente controlado e conhecido, representando o ambiente de produção. Por tratar-se de testes de desempenho, que impactam diretamente o funcionamento da aplicação, não é indicado que sejam implantados em ambiente de produção.

Ao coletar e refinar os resultados, pôde-se observar que os tempos de resposta apresentam um crescimento linear quando há uma concorrência de acessos, ultrapassando o limite de cinco segundos quando passamos, na média total, de apenas 25 usuários simultâneos. Em alguns casos, o sistema chega a um tempo de resposta de mais de 30 segundos com 250 usuários, com uma quantidade baixa de erro (vide subseção 5.6.4). A principal exceção, foi o caso CT003 (módulo **edu**).

Ao analisar mais detalhadamente os resultados coletados, foi observado que nos seis casos de teste desenvolvidos, 83,33% não apresentaram média de tempo de reposta abaixo de 5 segundos ao atingir 100 usuários[7] – foram os casos dos módulos de acesso ao *index*, visualizar alunos, visualizar todos os processos eletrônicos, consultar um processo específico

[7] Número de referência por ser o mais próximo da média real de utilização do sistema.

e visualizar histórico do aluno. Em 2 casos específicos, (CT004 e CT006), pertencentes aos módulos **adm** e **edu**, respectivamente, esse número foi mais de 3 vezes superior ao esperado, registrando tempos de resposta em 19,3 e 17,2 segundos, respectivamente.

No outro extremo, o caso CT003 (módulo **edu**) registrou tempo de resposta significantemente abaixo dos padrões registrados. Este caso de teste apresentou retorno médio ao usuário em apenas 2,2 segundos, sendo o único a atender a resposta esperada no cenário A2 (Tabela 2). Sendo assim, apenas uma funcionalidade testada atingiu totalmente os resultados esperados, comprovadando o quão é necessário sugerir melhorias voltadas para a eficiência e estabilidade da aplicação sob altas demandas.

Com relação à quantidade de erros registrados, a aplicação apresentou maior quantidade de requisições negadas (erro 502) frente às requisições com sucesso de resposta ao atingir uma média de 250 usuários, identificando assim o valor limite atual de usuários simultâneos suportados. Ou seja, considerando os mais de 35 mil usuários ativos registrados, o SUAP não suportaria 1% dos seus usuários totais consumindo o mesmo recurso.

6 Trabalhos Relacionados

Este capítulo apresenta uma breve discussão sobre alguns trabalhos relacionados aos temas explorados nesta dissertação. Considerando que os trabalhos identificados na literatura não seguiam o mesmo objetivo deste, este capítulo busca apontar pesquisas desenvolvidas sobre avaliação arquitetural de aplicações, bem como trabalhos que têm por objetivo avaliar ou testar o desempenho de sistemas de software.

A avaliação arquitetural de uma aplicação busca, de forma geral, prever o comportamento de sistemas (ou arquiteturas) propostos através da elaboração de cenários. Ressaltando que a arquitetura desempenha um papel fundamental na determinação da qualidade do produto de software, Oliveira e Nakagawa (2009) considera a avaliação estrutural como fundamental para obtenção dessa métrica. Para eles, métodos de avaliação de arquiteturas de software não devem ser diretamente aplicados a todos os tipos de arquitetura, como em arquiteturas de referência. Os autores corroboram a visão do nosso modelo, onde a utilização desses métodos permite analisar os riscos relacionados à evolução de um software, identificando potenciais estratégias para lidar com essas ameaças – diferentemente do método formal, que visa apenas a avaliação de sistemas em concepção. Por fim, eles discutiram possíveis extensões dos métodos tradicionais de avaliação para aplicação em tipos específicos de arquitetura ou em sistemas já concebidos.

Desde a introdução dos tradicionais e mais difundidos métodos de avaliação arquitetural, a indústria de softwares sofreu inúmeras evoluções. Sistemas passaram a não ser apenas concebidos ”do zero”, mas sim a crescer e evoluir em um ciclo natural. Dessa forma, Santos (2013) desenvolveu sua pesquisa adaptando o ATAM para ser aplicado nesse contexto. Ele traz o ATAM como método capaz de avaliar a evolução de um produto de software, reduzindo a quantidade de papéis envolvidos na avaliação em comparação à metodologia formal, além de suprimir as etapas 7 e 8 do ATAM, referentes à fase 2 do método. Santos (2013) constatou ainda que, embora pouco difundida, a aplicação adaptada do ATAM em sistemas legados é viável, tendo sido registrada diminuição de tempo e o esforço despendidos em manutenções imediatas, implantações do sistema e retrabalhos resultantes de falhas arquiteturais. Para o autor, a falta de uma equipe de avaliação não se mostrou como um impedimento no alcance dos objetivos do processo de avaliação. Ele ainda corroborou com a dificuldade discutida na literatura de se disseminar a importância da aplicação de um processo de avaliação.

Na nossa análise do SUAP, houve também uma adaptação do ATAM, alinhada à essa forma ”ágil” do ATAM proposta por Santos (2013). As mesmas etapas foram suprimidas e não houve formalmente uma equipe responsável pela avaliação.

Já Silva (2016), desenvolveu seu estudo objetivando compreender quais as principais estratégias utilizadas na definição e avaliação de uma arquitetura, principalmente para a obtenção e cumprimento dos requisitos não-funcionais, como desempenho, por exemplo. Seu trabalho levantou a tese de que a avaliação de arquitetura é um processo custoso, que demanda significativos recursos organizacionais. Aplicando um questionário com os arquitetos de software participantes da pesquisa, o autor identificou que desempenho é o requisito não-funcional mais importante a ser avaliado, e que o tempo de resposta é a métrica mais relevante para tal. Como resultados, ele identificou que a principal estratégia dos arquitetos para definição de uma arquitetura é a realização de testes automatizados, através de testes de carga elaborados sem qualquer parametrização ou análise sistematizada da arquitetura da aplicação fim.

Seguindo esse princípio e com abordagem baseada em cenários, Pinto, Kulesza e Treude (2015) desenvolveram um modelo de implementação que utiliza análise dinâmica e técnicas de mineração de repositório para fornecer uma maneira automatizada de revelar possíveis fontes de degradação de desempenho entre versões de um sistema de software. A abordagem define quatro fases: (i) preparação – escolha dos cenários e preparação dos lançamentos-alvo; (ii) análise dinâmica – determinando o desempenho de cenários e métodos calculando seu tempo de execução; (iii) análise de degradação – processando e comparando os resultados da análise dinâmica para diferentes lançamentos; e (iv) mineração de repositórios – problemas de desenvolvimento e comprometimento associados com o desvio de desempenho. Com um total de 3 softwares e 57 cenários analisados, a ferramenta auxiliou na identificação de 14 cenários com desvio de desempenho, quase em sua totalidade podendo ser atribuídos a alterações no código-fonte.

Contudo, diferentemente da nossa proposta, o autor optou por não realizar avaliação arquitetural, além de analizar os resultados apenas sob a métrica de tempo de execução (resposta) – não considerando qualquer outra medida que indique falha de desempenho ou quantidade de erros por requisições, por exemplo. A execução comum à nossa proposta foi a de aplicar múltiplas execuções de cenários para aumentar a confiabilidade das amostras – cada conjunto de testes foi executado dez vezes, como adotamos em nossa análise.

Koziolek et al. (2012) desenvolveram um novo método de predição, capaz de realizar análise prévia de desempenho de softwares em evolução. Tal produto busca permitir aos arquitetos quantificar possíveis ganhos ao planejar alterações arquiteturais, considerando atributos de qualidade – como desempenho, confiabilidade e capacidade de manutenção. O objetivo dos pesquisadores foi propor um modelo que apoie a análise preditiva de comportamento, não se baseando apenas em protótipos ou experiências anteriores para avaliar alternativas de projeto. Porém, para refletir as características de desempenho, esse modelo precisa receber propriedades dinâmicas de difícil inferência, como controle e fluxo de dados através da arquitetura, ou demandas de recursos, por exemplo. Em nossa

proposta, os dados podem ser obtidos de forma mais simplificada, através da aplicação do ATAM sintetizado.

Contudo, para criação desses modelos, não há uma previsão do esforço a ser empenhado, que pode ser potencialmente alto. Outro problema da adoção dos modelo de Koziolek et al. (2012) é que, segundo os próprios autores, existem apenas um número limitado de estudos de casos documentados e de relatórios de experiência industrial para abordagens preditivas baseadas em modelos. Em seu próprio estudo, registrou-se uma ocorrência de até 30% de amostras inválidas, o que pode representar um baixo benefício em sua implantação, visto o alto esforço que ele demanda. Com isso, frequentemente a precisão dessas análises preditivas são contestadas.

Conforme apresentado, há na literatura diversas pesquisas sobre a aplicação de métodos de avaliação arquitetural, bem como trabalhos de análise de desempenho. Contudo, essas pesquisas ou aplicam a metodologia de avaliação de arquitetura, ou buscam analisar desempenho. Nos estudos sobre os métodos mais disseminados de avaliação de arquiteturas de software, destacou-se o argumento comum de que esses métodos são custosos, demandam muito esforço e dificilmente são aplicados por completo. Observou-se também que as principais aplicações de testes de desempenho não trazem vasto embasamento teórico, não demonstrando parametrizações e sistematizações de tais aplicações.

Comparado com os trabalhos mencionados, realizamos uma análise sistematizada para geração dos casos de teste, fundamentada na avaliação da arquitetura como análise inicial. Além disso, propomos que sejam registrados também o limite de requisições da aplicação. Este trabalho buscou adaptar o ATAM para ser aplicado de forma mais simplificada que o processo tradicional, contudo utilizando de parametrizações e sistematizando este modelo mais ágil, seguido da execução de testes de software. Buscou-se enfatizar, então, a atividade de testes de desempenho subsequente à análise arquitetural realizada. Esse foco superior na execução dos testes deu-se pela priorização de um único atributo de qualidade considerado na aplicação do ATAM. Dessa forma, foi necessário minuciar este atributo, considerando ainda a necessidade fundamentada da equipe de desenvolvimento em compreender o limite atual do SUAP. Somado a isso, entendeu-se que a execução estruturada desses testes poderia colaborar com a identificação da origem dos problemas de desempenho da aplicação.

7 Considerações Finais

Este capítulo apresenta as considerações finais relacionadas a proposta apresentada nesta dissertação. A seção 7.1 discorre sobre as principais contribuições obtidas no trabalho, em seguida na seção 7.2 são listadas as limitações, e por na seção 7.3 são indicados alguns trabalhos futuros.

7.1 Contribuições

Este trabalho apresentou uma avaliação arquitetural do SUAP, com uma abordagem focada em desempenho. Foi realizada a aplicação do ATAM de forma simplificada, servindo de base para posterior desenvolvimento de testes de desempenho para os principais casos de uso do sistema trabalhado. Este capítulo pode ser considerado como parte do resultado final da aplicação do método de avaliação arquitetural, correspondendo à etapa 9 do ATAM (apresentação dos resultados).

Ao submeter o SUAP ao método avaliativo ATAM, o objetivo foi investigar os problemas de desempenho da aplicação. Por se tratar de uma arquitetura monolítica, acreditava-se que ela seria a principal responsável por essas falhas. Através do ATAM, pôde-se analisar o critério de qualidade que seria avaliado: desempenho. Ao elaborar a árvore de atributos de qualidade do SUAP (Figura 5), foram definidos dois cenários distintos como base para os testes executados posteriormente, análise do tempo de resposta (A1) e de capacidade em suportar crescentes conexões simultâneas (A2). Ao consultar a literatura, viu-se que ambos os cenários poderiam ser analisados através dos resultados obtidos em testes de desempenho.

Baseado nesses cenários, foram elaborados testes de desempenho para o SUAP. Durante este estudo, identificamos os casos de uso mais utilizados de cada módulo da aplicação, que serviram de base para o desenvolvimento de casos de teste. Após a execução dos testes, ficou diagnosticado que o SUAP apresentou problemas de desempenho em funções básicas do sistema quando ocorre uma alta simultaneidade de requisições a uma mesma funcionalidade. Considerando que o SUAP tem uma base de usuários de mais de 35.000 usuários ativos, identificamos que a aplicação não chega a suportar 1% dos usuários ativos totais acessando a mesma funcionalidade. Sendo assim, fica identificado que os problemas de desempenho não são exclusivos de um módulo ou funcionalidade específica. Os resultados coletados indicam que a infraestrutura de software limita o número de acessos paralelos, indicando a necessidade de uma análise mais profunda e conclusiva na arquitetura do SUAP.

Durante a execução deste trabalho, através de relatos da equipe de desenvolvimento, foram percebidos problemas causados pela falta de adoção de uma política de testes de desempenho no fluxo de desenvolvimento do SUAP, visto que os testes de conformidade atuais não analisam o desempenho de novas implementações. Esta etapa a mais no processo de desenvolvimento poderia ajudar a identificar alterações no código que tenham impactado negativamente o desempenho do sistema, considerando que durante a pesquisa foram presenciadas situações análogas. Uma alteração em uma consulta SQL pode atender a um requisito de conformidade, porém denegrir o desempenho da aplicação, por exemplo. Dessa forma, os testes de carga poderiam identificar versionamentos em que alterações de código atendam os testes de conformidade, mas impactem a performance da aplicação.

Surge, então, a necessidade de buscar uma solução que vise monitorar o desempenho desses principais casos de uso ainda em ambiente de teste, antes de serem incorporados ao ambiente de produção. Sendo assim, recomenda-se eminentemente a adoção de testes de desempenho para novas *merges* do SUAP. Para tanto, disponibiliza-se o ambiente de testes criado para este estudo, bem como os scripts e ferramentas necessárias.

A seguir, serão apontadas as limitações levantadas com o desenvolvimento deste estudo, bem como indicativos de trabalhos futuros.

7.2 Limitações do Trabalho

Esta análise de desempenho do SUAP limitou-se a aplicar o ATAM de uma forma simplificada, não considerando algumas etapas do processo tradicional. Embora exista um indicativo de que os problemas de desempenho do SUAP sejam causados por sua arquitetura, este trabalho não conseguiu reunir indícios suficientes para se chegar a esta conclusão. Para comprovar essa hipótese, seria necessário analisar detalhadamente a infraestrutura de software do sistema, alterando as configurações de limites do servidor de aplicação e analisar o impacto dessa mudança nos outros componentes da arquitetura, bem como se a infraestrutura de hardware suportaria tal demanda. Como essas atividades não são previstas pelo ATAM, esse análise não foi considerada no escopo deste trabalho. Dessa forma, também não foram avaliadas arquiteturas candidatas para o SUAP.

Ainda como limitador, considerando a complexidade do planejamento e análise de cada caso de teste proposto, esta pesquisa limitou-se a analisar seis casos de uso principais da aplicação.

7.3 Trabalhos Futuros

Considerando que SUAP não suportou os cenários de altas demandas simulados, pode-se sugerir que trabalhos futuros avaliem uma re-arquitetura do sistema. Para tanto,

sugere-se a aplicação formal do ATAM nas possíveis arquiteturas propostas para o sistema. Entende-se que essa avaliação possa trazer grande colaboração na definição de uma arquitetura candidata.

Sugere-se também a automação dos testes de desempenho proposto para novas funcionalidades do SUAP. Antes de serem incorporadas ao ambiente de produção, poderia-se submeter a última versão *master* à realização dos testes de desempenho de forma autônoma, executando, de forma independente, os casos de testes desenvolvidos nesta pesquisa. A metodologia dessa ferramenta seria verificar a última versão do sistema a ter sido submetida aos testes e comparar com a versão atual em *master*. Caso sejam detectadas atualizações, um script automatizado é executado para atualizar a versão do sistemas no ambiente de teste. Ao final da execução dos casos de teste, um novo comando automatizado deve ser iniciado para calcular as médias de cada escala n nos testes realizados, armazenando esses resultados em um banco de dados, a fim de possibilitar um comparativo aos dados anteriormente registrados.

Por fim, entende-se como necessário implementar novos casos de teste para as demais funcionalidades do sistema. Como já considerado na seção 7.2, este trabalho considerou apenas seis dos principais casos de uso do sistema. A aplicação de novos casos de teste podem colaborar com o mapeamento de um padrão de desempenho em um módulo específico ou até o desempenho de atributos envolvidos.

Referências

BABAR M.A.; GORTON, I. Comparison of Scenario-Based Software Architecture Evaluation Methods. *Proceedings of the 11th Asia-Pacific Software Engineering Conference*, p. 600–607, 2004. ISSN 1530-1362. Disponível em: <http://www.mendeley.com/import/>. Citado na página 23.

BARUCHI, A. *Memory Dispatcher: Uma Contribuição para a Gerência de Recursos em Ambientes Virtualizados*. Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo, 2010. Citado na página 21.

BASS, L.; CLEMENTS, P.; KAZMAN, R. *Software Architectura in Practice*. [S.l.]: Pearson, 2003. 560 p. ISBN 0-321-15495-9. Citado 3 vezes nas páginas 15, 23 e 28.

BONDI, A. B. Characteristics of scalability and their impact on performance. *Proceedings of the second international workshop on Software and performance - WOSP '00*, p. 195–203, 2000. ISSN 158113195X. Disponível em: <http://portal.acm.org/citation.cfm?doid=350391.350432>. Citado 4 vezes nas páginas 14, 19, 20 e 22.

BRATAAS, G.; HUGHES, P. Exploring Architectural Scalability. In: *4th international workshop on Software and performance*. Redwood Shores: [s.n.], 2004. p. 125–129. ISBN 1581136730. Citado na página 19.

CARVALHO, T. de P. M. Iii encontro técnico do suap. 2015. Disponível em: <http://portal.ifrn.edu.br/tec-da-informacao/digti/iii-encontro-tecnico-do-suap/apresentacao-de-metodologia-de-desenvolvimento>. Citado na página 33.

CLEMENTS, P.; KAZMAN, R.; KLEIN, M. *Evaluating Software Architectures: Methods and Case Studies*. [S.l.]: Addison-Wesley, 2002. (SEI series in software engineering). ISBN 9780201704822. Citado 2 vezes nas páginas 26 e 27.

DUBOC, L.; ROSENBLUM, D. S.; WICKS, T. A framework for modelling and analysis of software systems scalability. In: ACM. *Proceedings of the 28th international conference on Software engineering*. [S.l.], 2006. p. 949–952. Citado 2 vezes nas páginas 14 e 19.

FOWLER, M. *Padrões de Arquitetura de Aplicações Corporativas*. [S.l.]: Bookman, 2009. ISBN 9788577800643. Citado na página 20.

GUSTAVSON, D. B. The Many Dimensions of Scalability. In: *Compcon Spring*. San Francisco: [s.n.], 1994. Citado na página 19.

HILL, M. D. What is scalability? *ACM SIGARCH Computer Architecture News*, v. 18, n. 4, p. 18–21, 1990. ISSN 01635964. Disponível em: <ftp://ftp.cs.wisc.edu/markhill/Papers/can90{_}scalability.> Citado 2 vezes nas páginas 14 e 19.

HÖRCHER, H.-M. Improving software tests using z specifications. In: BOWEN, J. P.; HINCHEY, M. G. (Ed.). *ZUM 95: The Z Formal Specification Notation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995. p. 152 – 166. ISBN 978-3-540-44782-5. Citado na página 29.

IONITA, M. T.; HAMMER, D. K.; OBBINK, H. Scenario-based software architecture evaluation methods: An overview. *Workshop on Methods and Techniques for Software Architecture Review and Assessment at the International Conference on Software Engineering.*, p. 1–12, 2002. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.8382{&}rep=rep1{&}ty>. Citado na página 22.

ISO/IEC:9126-1. *ISO/IEC 9126: Qualidade de produto-Parte 1: Modelo de qualidade.* Rio de Janeiro, 2003. 1–21 p. Disponível em: <http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:Engenharia+de+software+-+Qualidade+de+produto+Parte+1+:+Modelo+de+quali>. Citado 3 vezes nas páginas 14, 23 e 30.

KAZMAN, R. et al. The architecture tradeoff analysis method. *Proceedings. Fourth IEEE International Conference on Engineering of Complex Computer Systems (Cat. No.98EX193)*, p. 68–78, 1998. Disponível em: <http://ieeexplore.ieee.org/document/706657/>. Citado 2 vezes nas páginas 15 e 23.

KAZMAN, R.; KLEIN, M.; CLEMENTS, P. ATAM : Method for Architecture Evaluation. *Cmusei*, v. 4, n. August, p. 83, 2000. ISSN CMU/SEI-2000-TR-004. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.118.6014{&}rep=rep1{&}ty>. Citado 2 vezes nas páginas 23 e 28.

KOZIOLEK, H. et al. Performance and reliability prediction for evolving service-oriented software systems. *Empirical Software Engineering*, v. 18, 2012. Citado 2 vezes nas páginas 65 e 66.

LIU, H. H. *Software Performance and Scalability: A Quantitative Approach.* [S.l.]: Wiley, 2011. v. 7. (Quantitative Software Engineering Series, v. 7). ISBN 9781118211311. Citado na página 20.

LUKE, E. A. Defining and Measuring Scalability. In: *Scalable Parallel Libraries Conference.* Mississippi State: [s.n.], 1993. p. 183–186. ISBN 0818649801. Citado na página 19.

MATTSSON, M.; GRAHN, H.; MÅRTENSSON, F. Software Architecture Evaluation Methods for Performance, Maintainability, Testability, and Portability. *2nd International Conference on the Quality of Software Architectures*, 2006. Citado 2 vezes nas páginas 22 e 30.

MUCCINI, H.; BERTOLINO, A.; INVERARDI, P. Using software architecture for code testing. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, v. 30, n. 3, p. 160 – 171, 2004. Citado na página 14.

MYERS, G. J. *The Art of Software Testing.* New Jersey: John Wiley & Sons, Inc, 2004. Citado na página 30.

NAIK, K.; TRIPATHY, P. *Software Testing and Quality Assurance: Theory and Practice.* New Jersey: Wiley-Spektrum, 2008. Citado na página 43.

NEWMAN, S. *Building Microservices: Designing Fine-Grained Systems.* [S.l.]: O'Reilly Media, 2015. ISBN 9781491950333. Citado na página 21.

OLIVEIRA, L. B. R. de; NAKAGAWA, E. Y. Um levantamento de métodos de avaliação de arquiteturas de software específicas. *III Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software - SBCARS*, p. 52 – 65, 2009. Citado 2 vezes nas páginas 23 e 64.

ADDISON-WESLEY (Ed.). *Evaluating Software Architectures: Methods and Case Studies*. [S.l.: s.n.], 2002. Citado na página 24.

PINTO, F.; KULESZA, U.; TREUDE, C. Automating the performance deviation analysis for multiple system releases: An evolutionary study. *IEEE 15th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, p. 201 – 210, 2015. Citado na página 65.

PORTO, I. O. *Padrões e diretrizes arquiteturais para escalabilidade de sistemas*. Dissertação (Mestrado) — Universidade Federal de Uberlândia, 2009. Citado 2 vezes nas páginas 19 e 20.

SANTOS, T. da C. Aplicação do atam na evolução arquitetural de um sistema corporativo. *10th International Conference on Information Systems and Technology Management – CONTECSI*, p. 974 – 989, 2013. Citado na página 64.

SCHWABER, K.; SUTHERLAND, J. Guia do SCRUM. *Harvard Business Review, Boston*, IV, p. 163–179, 2013. Disponível em: <https://www.scrum.org/Portals/0/Documents/ScrumGuides/Scrum{_}Guide.> Citado na página 32.

SILVA, J. C. Leoncio da. *Um Estudo de Avaliação e Documentação de Arquiteturas de Software na Indústria*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2016. Citado na página 65.

SMITH, C. U. Introduction to software performance engineering: origins and outstanding problems. *Proceedings of the 7th international conference on Formal methods for performance evaluation*, p. 395–428, 2007. Citado na página 30.

Apêndices

APÊNDICE A – Detalhamento do SUAP

A.1 Órgãos Conveniados

Através de Acordos de Cooperação técnica, hoje o IFRN compartilha o projeto SUAP com diversos órgãos do país. A seguir (Tabela 10) são listados os órgãos utilizadores do SUAP[1].

[1] Listagem atualizada até junho/2018

Tabela 10 – Órgãos que atualmente utilizam o SUAP

ID	**Sigla**	**Instituição**	**UF**
1	IFPB	Instituto Federal da Paraíba	PB
2	IFSertão	Instituto Federal Sertão Pernambucano	PE
3	IFPE	Instituto Federal de Pernambuco	PE
4	FUNDAJ	Fundação Joaquim Nabuco	PE
5	IFCE	Instituto Federal do Ceará	CE
6	IFPI	Instituto Federal do Piauí	PI
7	IFMA	Instituto Federal do Maranhão	MA
8	IFBA	Instituto Federal da Bahia	BA
9	IFBaiano	Instituto Federal Baiano	BA
10	IFTO	Instituto Federal do Tocantins	TO
11	IFG	Instituto Federal de Goiás	GO
12	IFGoiano	Instituto Federal Goiano	GO
13	IFMT	Instituto Federal do Mato Grosso	MT
14	IFMS	Instituto Federal do Mato Grosso do Sul	MS
15	IFNMG	Instituto Federal do Norte de Minas Gerais	MG
16	IFRR	Instituto Federal de Roraima	RR
17	IFRO	Instituto Federal de Rondônia	RO
18	IFAP	Instituto Federal do Amapá	AP
19	IFB	Instituto Federal de Brasília	DF
20	IFF	Instituto Federal Fluminense	RJ
21	CP2	Colégio Pedro II	RJ
22	IFSP	Instituto Federal de São Paulo	SP
23	IFRS	Instituto Federal do Rio Grande do Sul	RS
24	IFSul	Instituto Federal Sul-rio-grandense	RS
25	IFSULDEMINAS	Instituto Federal Sul de Minas Gerais	MG
26	ENAP	Escola Nacional de Administração Pública	DF
27	CEFET/RJ	Centro Federal de Educação Tecnológica Celso Suckow da Fonseca	RJ
28	UENP	Universidade Estadual do Norte do Paraná	PR
29	MINC	Ministério da Cultura	DF
30	Fiocruz	Fundação Oswaldo Cruz	RJ
31	IFMG	Instituto Federal de Minas Gerais	MG
32	Sudeco	Superintendência do Desenvolvimento do Centro-Oeste	DF

Fonte: Elaborada pelo autor.

A.2 Listagem dos Módulos do SUAP

- **INÍCIO**

- **ADMINISTRAÇÃO**
  - **Cadastros**
    - ✓ Pessoas Físicas
    - ✓ Pessoas Jurídicas
    - ✓ Prestadores de Serviço
    - ✓ Prédios
    - ✓ Salas
  - **Agendamento de Salas**
    - ✓ Requisições
    - ✓ Acompanhamento
  - **Almoxarifado**
    - ✓ Entradas
    - ✓ Requisições
      - ➢ Saída de Material para Consumo
      - ➢ Transferência de Material entre *Campi*
      - ➢ Ver Pendentes
      - ➢ Requisições
    - ✓ Pré-Carga
    - ✓ Categorias de Material de Consumo
    - ✓ Categorias de Material Permanente
    - ✓ Empenhos
    - ✓ Materiais de Consumo
    - ✓ Unidades de Medida
    - ✓ Relatórios
      - ➢ Saídas por Setor
      - ➢ Notas de Fornecimento
      - ➢ Balancete
        - ❖ Elemento de Despesa
        - ❖ Material de Consumo
        - ❖ Elemento de Despesa Detalhado
      - ➢ Total por ED Permanente
    - ✓ Estoque
      - ➢ Configuração
      - ➢ Controle
  - **Patrimônio**
    - ✓ Movimentação
      - ➢ Carga
      - ➢ Transferência no *Campus*
      - ➢ Transferência
      - ➢ Doação
    - ✓ Requisição
      - ➢ Requisições
    - ✓ Baixas
    - ✓ Cautelas
    - ✓ Entradas
    - ✓ Elementos de Despesa
    - ✓ Inventários
    - ✓ Rótulos
    - ✓ Relatórios
      - ➢ Termos
      - ➢ Totalizações
      - ➢ Totalização por *Campus*
    - ✓ Servidores com Carga
  - **Protocolo**
    - ✓ Caixa de Entrada e Saída
    - ✓ Caixa de Tramitação Externa
    - ✓ Processos
  - **Frota**
    - ✓ Cargos de Motorista
    - ✓ Motoristas
    - ✓ Viaturas
    - ✓ Modelos de Veículos
      - ➢ Agendamento
      - ➢ Adicionar
      - ➢ Agendamentos
      - ➢ Meus Agendamentos
      - ➢ Agendamentos Futuros no *Campus*
      - ➢ Avaliar
    - ✓ Viagem
      - ➢ Viagens
      - ➢ Cadastro Retroativo
      - ➢ Registrar Saída
      - ➢ Registrar Chegada
    - ✓ Ordens de Abastecimento
    - ✓ Trocas de Óleo
    - ✓ Relatórios
      - ➢ Frota Atual
      - ➢ Motoristas Temporários
      - ➢ Viagens por Solicitante
      - ➢ Deslocamento / Consumo
      - ➢ Ordens de Abastecimento
      - ➢ Viagem por Viatura
    - ✓ Estatísticas
      - ➢ Viaturas
      - ➢ Deslocamento
  - **Chaves**
  - **Estacionamento**
    - ✓ Modelos de Veículos
    - ✓ Veículos
  - **Contratos**
    - ✓ Busca Avançada
    - ✓ Contratos
    - ✓ Notificar Pendências
    - ✓ Relatórios
      - ➢ Contratos a Serem Aditivados
      - ➢ Contratos a Serem Licitados
      - ➢ Situação dos Contratos
      - ➢ Pendências
    - ✓ Fiscais
      - ➢ Rol de Responsabilidades
  - **Convênios**
    - ✓ Buscar Convênios
    - ✓ Convênios
    - ✓ Convênios a Vencer
    - ✓ Tipos de Convênios
    - ✓ Tipos de Anexos
    - ✓ Situações de Convênios
    - ✓ Instituições
    - ✓ Estatísticas
  - **Orçamento**
    - ✓ Cadastros básicos
      - ➢ Eventos
        - ❖ Desatualizados
        - ❖ Utilizados
        - ❖ Eventos
      - ➢ Unidades de Medida
      - ➢ Unidades Gestoras
    - ✓ Execução Orçamentária
    - ✓ Notas
      - ➢ Notas de Crédito
        - ❖ Buscar
        - ❖ Notas de Crédito
      - ➢ Notas de Dotação
    - ✓ Notas de Empenho
  - **Materiais**
    - ✓ Categorias
    - ✓ Tag de Materiais
    - ✓ Materiais
    - ✓ Unidade Medida
    - ✓ Relatórios
      - ➢ Materiais sem cotação
  - Compras
    - ➢ Processos de compra

- **RECURSOS HUMANOS**
  - **Servidores**
  - **Setores**
  - ***Campus***
  - **Ponto**
    - ✓ Frequência
      - ➢ Frequências por Funcionário
      - ➢ Frequências Noturnas Extras
      - ➢ Frequências de Estagiários
      - ➢ Frequências de Bolsistas
      - ➢ Frequências de Terceirizados
      - ➢ Frequências por Cargo
      - ➢ Frequências por Setor
      - ➢ Frequências Inconsistentes por Setor
    - ✓ Observações
    - ✓ Liberações
    - ✓ Afastamentos
    - ✓ Tipos de Afastamento
    - ✓ Máquinas
  - **Indicadores**
  - **Controle de Documento**
  - **Cargos de Emprego**
  - **Relatórios**
    - ✓ Buscar servidores
    - ✓ Recursos Humanos
    - ✓ Servidores com Função
    - ✓ Aniversariantes
    - ✓ Endereços
      - ➢ Agrupar
      - ➢ Fora do estado
    - ✓ Recadastramento
      - ➢ Aposentado
      - ➢ Pensionista
    - ✓ Dependentes Irregulares
  - **Financeiro**
    - ✓ Ressarcimento
      - ➢ Planos de Saúde
    - ✓ DGP
    - ✓ Indicadores
    - ✓ Rubricas
    - ✓ Bruto Anual
    - ✓ Titulações
  - **Histórico de Setores**
  - **Cursos e Concursos**
    - ✓ Horas trabalhadas
    - ✓ Atividades
    - ✓ Cota Anual de Servidor
    - ✓ Cursos e Concursos
    - ✓ Horas Permitidas
    - ✓ Inscrição Fiscal
  - **Remanejamento**
    - ✓ Editais
    - ✓ Disciplinas
    - ✓ Inscrições
    - ✓ Recursos ao Edital

- **ENSINO**
  - Edu
    - ✓ Cadastros Gerais
      - ➢ Naturezas de Participação
      - ➢ Turnos
      - ➢ Formas de Ingresso
      - ➢ Níveis de Ensino
      - ➢ Modalidades
      - ➢ Áreas dos Cursos
      - ➢ Eixos Tecnológicos
      - ➢ Convênios
      - ➢ Núcleos
      - ➢ Tipo de Professor em Diário
      - ➢ Situação de Matrícula
      - ➢ Situação de Matrícula no Período
      - ➢ Diretorias Acadêmicas
    - ✓ Procedimentos de Apoio
      - ➢ Horário do *Campus*
      - ➢ Calendários Acadêmicos
      - ➢ Matrícula Direta
      - ➢ Solicitações dos Usuários
      - ➢ Fechar Período
      - ➢ Abrir Período
    - ✓ Alunos e Professores
      - ➢ Professores
      - ➢ Alunos
    - ✓ Cursos, Matrizes e Componentes
      - ➢ Estrutura de Curso
      - ➢ Componentes
      - ➢ Matrizes Curriculares
      - ➢ Cursos
    - ✓ Horários, Turmas e Diários
      - ➢ Turmas
      - ➢ Diários
    - ✓ Relatórios e Estatísticas
      - ➢ Listagem de Alunos

- **DESENVOLVIMENTO INSTITUCIONAL**
  - **Gestão**
    - ✓ Período de Referência
    - ✓ Variáveis
      - ➢ Acadêmico
      - ➢ Recursos Humanos
      - ➢ Financeiro
      - ➢ Extensão
      - ➢ Pesquisa
      - ➢ Processo Seletivo
      - ➢ Socioeconômico
      - ➢ Tecnologia da Informação
      - ➢ Cadastrar Variável
    - ✓ Indicadores
      - ➢ Cadastrar
      - ➢ Visualizar
  - **Planejamento**
    - ✓ Cadastros básicos
      - ➢ Dimensões
      - ➢ Unidades de Medida
      - ➢ Naturezas de Despesa
    - ✓ Configuração
      - ➢ Períodos
      - ➢ Origens de Recurso
      - ➢ Unidades Administrativas
    - ✓ Macro Projeto Institucional
    - ✓ Metas
    - ✓ Metas de Unidades Administrativas
    - ✓ Ações
      - ➢ Buscar
      - ➢ Ações
      - ➢ Avaliar
      - ➢ Extra Teto
    - ✓ Acompanhamento
      - ➢ Execução
      - ➢ Execução
    - ✓ Relatórios
      - ➢ Avaliação de Ações
      - ➢ Origem de Recurso
      - ➢ Detalhamento dos *Campi*
      - ➢ Despesas das Dimensões por Origem de Recurso
      - ➢ Detalhamento de Gastos por Natureza de Despesa
      - ➢ Detalhamento de Despesas por Natureza de Despesas
      - ➢ Plano de Ações

- **EXTENSÃO**
  - **Projetos**
    - ✓ Editais
    - ✓ Projetos
    - ✓ Pré-avaliar Projetos
    - ✓ Avaliar Projetos
    - ✓ Monitoramento
    - ✓ Cadastros
      - ➢ Foco Tecnológico
      - ➢ Área do Conhecimento
      - ➢ Área Temática
      - ➢ Tema

- **ATIVIDADES ESTUDANTIS**
  - **Alunos**
  - **Atendimentos**
  - **Serviço Social**
    - ✓ Caracterização Socioeconômica
    - ✓ Programas
    - ✓ Inscrições
    - ✓ Períodos de Inscrição
    - ✓ Ofertas
      - ➢ Alimentação
      - ➢ Turmas de Idiomas
      - ➢ Bolsa de Iniciação Profissional
  - **Coordenação**
    - ✓ Bolsas
    - ✓ Financeiro
      - ➢ Valor da Refeição
      - ➢ Valor Auxílio-Transporte
      - ➢ Valor de Bolsa de Iniciação Profissional
    - ✓ Atendimentos do Setor
  - **Relatórios**
    - ✓ Caracterização Socioeconômica
    - ✓ Programas
    - ✓ Alunos por Programa
    - ✓ Atendimentos Individuais
    - ✓ Atendimentos do Setor
    - ✓ Bolsas
  - **Cadastros**
    - ✓ Tipos de Atendimentos Individuais
    - ✓ Tipos de Atendimentos do Setor
    - ✓ Categorias de Alunos
    - ✓ Categorias de Bolsas

- **PESQUISA**
  - CNPQ

APÊNDICE B – Resultados dos Testes

Este Apêndice apresenta o refinamento dos resultados coletados em cada caso de teste. As tabelas 11 a 16 apresentam uma escala de número de usuários (a cada 25 novas *threads*), os tempos de respostas obtidos (média, desvio padrão e mediana), além da quantidade de erros apresentados (também em números médios, de desvio padrão e medianos).

A tabela a seguir exibe os resultados coletados com a execução dos testes da subseção 5.6.1.

Tabela 11 – CT001: Resultados

Threads	**Tempo de Resposta** (em milissegundos)			**Erros** (em números totais)		
	Média	Desvio Padrão	Mediana	Média	Desvio Padrão	Mediana
25	4421,23	5006,15	3006,63	2,10	1,10	2,00
50	3860,85	898,67	4131,64	2,50	1,43	3,00
75	4873,51	378,91	4914,29	1,30	1,16	1,00
100	8271,42	2573,53	7772,62	1,40	1,17	1,50
125	9041,66	528,54	8915,47	0,70	0,95	0,00
150	10923,58	795,06	10748,06	0,70	0,67	1,00
175	11816,81	735,55	11655,35	0,50	0,71	0,00
200	14115,02	6475,33	12259,04	0,70	1,25	0,00
225	12269,29	714,19	12316,17	1,70	0,82	1,50
250	14606,90	337,68	14477,06	6,30	3,47	5,50
275	11667,69	850,45	11428,55	14,30	1,49	15,00
300	14495,48	3598,42	13160,82	21,50	1,72	21,50

Na sequência, a Tabela 12 demonstra os resultados coletados com a execução dos testes da subseção 5.6.2.

Tabela 12 – CT002: Resultados

Threads	**Tempo de Resposta** (em milissegundos)			**Erros** (em números totais)		
	Média	Desvio Padrão	Mediana	Média	Desvio Padrão	Mediana
25	6295,55	7217,28	4228,63	1,50	0,85	1,50
50	5662,54	1565,56	5283,39	2,20	1,32	2,00
75	5716,26	674,44	5785,69	1,40	1,17	1,00
100	7220,68	377,62	7380,81	1,00	0,82	1,00
125	9002,26	515,09	9062,34	1,20	1,14	1,00
150	10675,02	658,71	10508,87	1,90	1,37	1,50
175	11356,40	521,64	11517,20	0,90	0,74	1,00
200	12421,95	3377,32	11469,06	1,10	0,74	1,00
225	11727,61	937,07	11470,47	1,20	0,79	1,00
250	14161,63	278,10	14155,12	5,50	2,46	6,00
275	11200,88	716,44	11251,70	15,20	3,33	15,00
300	13633,19	4923,74	11592,98	21,50	1,27	22,00

Já os resultados do teste com melhor desempenho (CT003 – subseção 5.6.3) são exibidos na Tabela 13.

Tabela 13 – CT003: Resultados

Threads	**Tempo de Resposta** (em milissegundos)			**Erros** (em números totais)		
	Média	Desvio Padrão	Mediana	Média	Desvio Padrão	Mediana
25	1171,49	787,15	903,39	3,30	1,06	3,50
50	1047,97	172,05	990,83	3,90	1,37	4,00
75	1737,27	461,66	1661,79	3,00	1,41	3,50
100	2247,86	234,54	2243,76	2,90	1,10	3,00
125	2422,15	353,97	2338,26	2,50	1,27	2,50
150	2555,53	252,73	2527,43	1,80	0,79	2,00
175	3065,41	555,27	2950,07	1,60	0,97	1,50
200	3664,1	558,07	3626,17	1,70	1,06	1,50
225	3572,98	935,61	3451,05	4,90	1,97	5,00
250	4405,93	57,31	4430,88	14,70	3,40	15,50
275	5931,24	2154,24	5748,05	21,20	1,48	21,00
300	5121,11	1791,19	4431,39	24,90	0,32	25,00

Seguindo para o módulo **adm**, o CT004 (subseção 5.6.4) tem seus resultados

detalhados a seguir.

Tabela 14 – CT004: Resultados

Threads	**Tempo de Resposta** (em milissegundos)			**Erros** (em números totais)		
	Média	Desvio Padrão	Mediana	Média	Desvio Padrão	Mediana
25	11579,94	6426,93	11647,25	1,60	0,70	1,50
50	13635,73	4898,26	13925,64	1,70	1,06	1,00
75	14454,19	1806,49	14724,84	1,40	0,70	1,00
100	19398,18	4940,42	18675,50	1,40	0,84	1,00
125	20633,29	1050,11	20729,90	1,20	0,79	1,00
150	26049,46	6301,09	24395,97	0,80	0,79	1,00
175	26871,35	785,56	26943,97	0,60	0,70	0,50
200	28600,60	685,93	28706,00	0,60	0,70	0,50
225	29291,36	2257,53	29835,95	1,00	1,05	1,00
250	30928,56	1189,03	31221,29	2,90	1,37	3,00
275	27439,00	3157,32	28908,09	10,80	3,55	11,00
300	24723,91	8264,64	24701,47	19,10	1,73	19,00

Ainda no modulo voltado à administração do IFRN, o CT005 (subseção 5.6.5) obteve os seguintes resultados:

Tabela 15 – CT005: Resultados

Threads	**Tempo de Resposta** (em milissegundos)			**Erros** (em números totais)		
	Média	Desvio Padrão	Mediana	Média	Desvio Padrão	Mediana
25	7602,06	8494,36	4454,51	2,40	0,84	2,00
50	5437,77	1273,62	5632,55	2,30	1,06	2,50
75	6643,99	472,92	6612,38	2,40	0,97	2,00
100	8465,36	815,13	8200,59	1,90	1,37	2,00
125	10406,75	946,77	10287,89	0,80	0,92	0,50
150	12038,38	1175,58	11690,60	0,90	0,99	1,00
175	12666,24	450,75	12695,06	0,90	0,88	1,00
200	14963,61	5234,01	13413,77	1,30	0,67	1,00
225	13750,07	1952,76	13371,33	1,70	1,06	2,00
250	16010,20	213,85	16036,55	4,50	1,72	5,00
275	12828,91	857,40	12635,59	15,50	1,90	14,50
300	15879,01	5915,91	12790,47	22,60	2,17	23,00

Por fim, a Tabela 16 detalha o último caso de teste (subseção 5.6.6).

Tabela 16 – CT006: Resultados

Threads	**Tempo de Resposta** (em milissegundos)			**Erros** (em números totais)		
	Média	Desvio Padrão	Mediana	Média	Desvio Padrão	Mediana
25	13335,87	33182,72	2911,78	1,50	0,97	2,00
50	10476,40	16266,19	5367,07	2,20	1,03	2,00
75	9120,62	3833,70	7984,06	1,00	0,82	1,00
100	17296,79	16800,54	10745,09	1,30	0,82	1,00
125	13512,66	402,97	13389,15	0,70	0,48	1,00
150	23039,74	9266,94	16852,99	0,80	0,63	1,00
175	22754,90	5169,54	21044,59	1,10	0,57	1,00
200	24712,23	13940,70	20281,79	0,80	0,79	1,00
225	24306,44	9477,26	21794,92	0,90	0,88	1,00
250	22121,58	4616,03	20206,50	1,60	1,35	1,00
275	24109,25	6684,35	19851,72	8,80	3,55	9,00
300	22234,56	7661,52	19100,93	15,40	3,50	16,00