UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

FELIPE THIAGO SILVA ARAGÃO

COMPARATIVO ENTRE ARQUITETURAS DE SERVIÇOS WEB

SÃO LUÍS-MA
2018

Felipe Thiago Silva Aragão

COMPARATIVO ENTRE ARQUITETURAS DE SERVIÇOS WEB

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Msc. Carlos Eduardo Portela Serra de Castro

São Luís-MA
2018

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Aragão, Felipe Thiago Silva Aragão.
Comparativo Entre Arquiteturas De Serviços Web / Felipe Thiago Silva Aragão Aragão. - 2018.
38 p.

Orientador(a): Carlos Eduardo Portela Serra de Castro Castro.
Monografia (Graduação) - Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís, 2018.

1. JSON. 2. REST. 3. Serviços Web. 4. SOAP. 5. XML. I. Castro, Carlos Eduardo Portela Serra de Castro. II. Título.

Felipe Thiago Silva Aragão

COMPARATIVO ENTRE ARQUITETURAS DE SERVIÇOS WEB

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Curso de Ciência da Computação da Universidade Federal do Maranhão - UFMA, Centro de Ciências Exatas e Tecnológicas.

Data de Aprovação: 23/01/2018

Banca Examinadora

Prof. Msc. Carlos Eduardo Portela Serra de Castro
Universidade Federal de Maranhão
Orientador

Prof. Dr. Samyr Beliche Vale
Universidade Federal de Maranhão

Prof. Dr. Mário Antonio Meireles Teixeira
Universidade Federal de Maranhão

A toda a minha família, em especial a minha esposa Orlenisce Passos Aragão e as minhas duas mães: Elizabeth Aragão e Nazaré Aragão, na qual eu dedico não só este trabalho, mais toda a minha vida.

AGRADECIMENTOS

Em primeiro lugar agradeço aquele que sempre me amou primeiro, que sempre esteve comigo, ao autor da vida, meu amado Jesus Cristo. A Universidade Federal do Maranhão pela oportunidade de concluir minha formação no curso de Ciência da Computação.

À minha família, pois ela é minha referência e foi a base formadora do meu caráter, orientando-me para tomar as decisões corretas, sobretudo as minhas mães que nos momentos mais difíceis sempre esteve ao meu lado, demostrando seu carinho e atenção. Sem vocês tudo seria quase impossível.

Ao meu amor, companheira e esposa Orlenisce Passos Aragão, obrigado, pela paciência, pelo companheirismo, por sempre, sempre está comigo, sem você nada disso seria possível, afinal você foi o meu incentivo para ter chegado até aqui. Te amo.

Agradeço em especial o professor Portela que sempre mostrou total disponibilidade, acompanhando e direcionando o desenvolvimento deste trabalho. Obrigado por se disponibilizar e por ser meu orientador.

Agradeço a todos os amigos do curso de Ciência da Computação, dos mais distantes aos mais próximos, pois através das experiências trocadas ao longo de toda vida acadêmica, ensinamo-nos uns aos outros grandes lições, e sem dúvida, estas serão lembradas por toda a vida.

Agradeço a todos os professores do Curso de Ciência da Computação da Universidade Federal do Maranhão, pois através dos seus ensinamentos eu pude chegar onde estou. Ao departamento de informática que continuamente visou oferecer um ensino de qualidade a todos os alunos. E a todos que direta ou indiretamente contribuíram para a realização desse trabalho. Muito obrigado.

"Quantas vezes eu estive cara a cara com a pior metade; A lembrança no espelho, a esperança na outra margem, Quantas vezes a gente sobrevive na falta de algo melhor, Nunca me faltou coragem."

Surfando Karmas e DNA
Engenheiros do Hawaii

RESUMO

Com a tecnologia em constante evolução, o desafio de sincronizar as informações de forma transparente em diferentes plataformas de software motivou o surgimento de soluções de desenvolvimento de sistemas baseados em serviços web. Neste conceito, se aponta a Arquitetura Orientada a Serviços, baseada no uso do padrão SOAP para troca de mensagens e WSDL para descrição de serviços através do XML, e a Arquitetura Orientada a Recursos, que é derivada do estilo arquitetural REST e se tem o JSON como padrão na transferência de dados. Este trabalho tem como objetivo analisar a performance dos serviços web SOAP e REST, avaliando o tempo de resposta e as taxas de transferências. Para isso, desenvolveu-se aplicações baseadas nos dois modelos utilizando a mesma regra de negócio, base de dados e framework. O teste de desempenho foi realizado com a ferramenta Apache jMeter com diferentes quantidades de amostras e 10 usuários virtuais no intervalo de 1 segundo. Em todos os testes realizados neste trabalho, apontam que o SOAP obteve um resultado inferior em relação a desempenho em comparação com o REST no estudo desenvolvido.

Palavras-chave: Serviços Web, SOAP, REST, XML, JSON

ABSTRACT

With the constant evolution of technology, syncing information in a transparent way in multiple platforms has motivated the creation of web applications based on web services. In this concept, we have two architectures that became standards: Service-oriented architecture, based on the SOAP pattern to exchange messages and WSDL to describe services through XML and Resource Oriented Architecture derived from the REST architecture that uses JSON as a pattern for transferring data. This work has as it's objective to do a performance analysis of web services using SOAP and REST, evaluating the response time and transfer cost for each of them. The tests were applied to applications developed with the same framework, accessing the same databases and with the same business rules. The performance tests were performed with the Apache JMeter, with samples of different sizes and 10 simultaneous virtual users accessing every 1 second. In every test case performed, the architectures using SOAP as a pattern for transferring data showed inferior performance when compared to those using REST.

Keywords:Web services, SOAP, REST, XML, JSON.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CRUD	Create, Read, Update e Delete
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JDK	Java Development Kit
JSON	JavaScript Object Notation
JVM	Design of Experiments
REST	Representational State Transfer
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
UDDI	Universal Description, Discovery and Integration
URI	UniformResourceIdentifier
URL	Uniform Resource Locator
WS	Web Service
XML	eXtensible Markup Language
WSDL	Web Services Description Language

SUMÁRIO

1 Introdução

Nas últimas décadas o estudo sobre mobilidade em sistemas distribuídos vem se expandindo de forma exponencial devido ao aumento de dispositivos eletrônicos portáteis e pela exploração de novas tecnologias de interconexão baseadas em comunicação sem fio. Esta nova forma de tecnologia é chamada de Computação Móvel (BARBOSA et al., 2007) Os dispositivos móveis são nos dias atuais ferramenta de comunicação mais importante e midiática (LEMOS, 2008).

A computação móvel supõe um mundo na qual as pessoas utilizam dispositivos que convivem a todo instante desempenhando funções chaves do seu dia a dia seja apenas para se comunicar ou para funções mais especificas como relacionadas a sua localização e contexto. A finalidade da comunicação está bem desenvolvida por meio de aplicativos de voz, e-mail e mensagens, porém quando se trata das funções de localização e contexto é essencial a aplicação de serviços para então fornecer informações (MCFADDIN et al., 2008). Neste contexto o termo "aplicação de serviços" refere-se de forma direta a uma tecnologia de Serviços da Web que por meio de dispositivos moveis, assim como dispositivo de outras plataformas se comunicam de forma clara e transparente (NASCIMENTO et al., 2013).

A propagação dos Serviços Web coincide com o aumento significativo da capacidade de processamento que os dispositivos móveis possuem (PILIOURA et al., 2007). Os serviços web fazem parte de uma Arquitetura Orientada a Serviços e tem como principais características um modelo fraco acoplado e transparente que assegura a interoperabilidade entre os serviços sem ter a necessidade de conhecer previamente quais tecnologias estão em cada lado da comunicação, possuem padrões que permitem a integralização de aplicações por meio de linguagens e protocolos bastante aceitos (MELLO et al., 2006).

1.1 MOTIVAÇÃO

Com dispositivos de alta tecnologia compostos de softwares que possuem várias informações nas quais as pessoas desejam compartilhar, o grande desafio é sincronizar as informações de forma transparente que possam ser usadas em diferentes plataformas de softwares e linguagens de programação. Segundo Menascé e Almeida (2003), a disponibilidade, qualidade e desempenho dos serviços disponibilizados na web, precisam ser constantemente avaliados para que a necessidade do usuário final seja garantida. Diante disso, surge a necessidade de avaliar o desempenho de arquitetura SOAP e REST a fim de classificar qual arquitetura apresenta o melhor

desempenho.

1.2 OBJETIVO

1.2.1 Objetivo Geral

Este trabalho tem como objetivo apresentar um comparativo do desempenho de duas das principais abordagens de Arquitetura de Serviços Web que possibilite através de seus resultados um embasamento na escolha de futuras implementações de integração entre sistemas heterogêneos.

1.2.2 Objetivo Específico

- Identificar as vantagens e desvantagens dos Serviços Web em relação as abordagens tradicionais;
- Identificar as vantagens e desvantagens dos Serviços Web em relação a abordagem REST;
- Desenvolver um conjunto de serviços web baseados nas arquiteturas SOAP e REST para realização de testes de avaliação de desempenho;

1.3 ORGANIZAÇÃO DO TRABALHO

O presente documento está organizado da seguinte forma: O Capítulo 2 apresenta uma visão geral da solução baseada em serviços web, incluindo algumas definições, a abordagem orientada a serviços, sua arquitetura e principais padrões web envolvidos. Este capítulo também introduz conceitos sobre o protocolo SOAP e o estilo arquitetural REST.

O Capítulo 3 apresenta as tecnologias usadas para o desenvolvimento das aplicações SOAP e REST e além de exibir as ferramentas de testes. Também explica a metodologia para o desenvolvimento dos serviços web.

O estudo de caso deste trabalho será apresentado no capítulo 4, onde foi realizada uma avaliação de desempenho no protocolo REST e no SOAP e logo após foram feitas comparações entre eles.

Finalmente, o Capítulo 5 destaca os principais resultados obtidos e propõe trabalhos futuros relacionados.

2 Fundamentação Teórica

Neste capítulo são introduzidos os conceitos básicos e as tecnologias adotadas neste trabalho. Inicialmente, uma explicação sobre o que são serviços web. Em seguida, são introduzidos os protocolos de comunicação SOAP e REST. Finalmente, é apresentado princípios sobre testes de software.

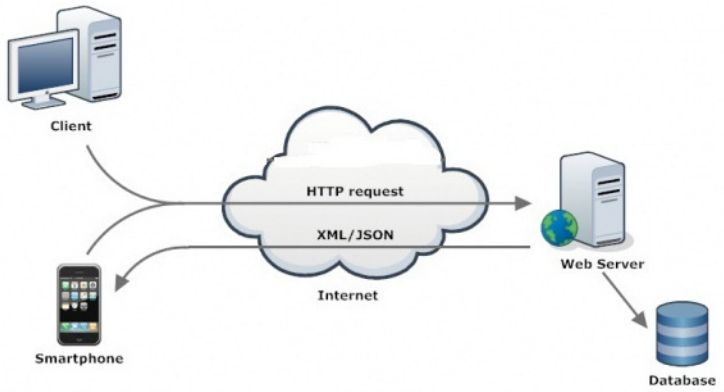
2.1 SERVIÇOS WEB

O ambiente corporativo tem se tornado mais dinâmico, competitivo e globalizado. Empresas crescem, diversificam seus negócios, se unem, em geral, estão cada vez mais se relacionando umas com as outras (GUIMARÃES, 2000). Com os avanços tecnológicos ocorridos nas últimas décadas, gerou um aumento significativo de informações que precisam ser compartilhadas entre si. Além disso, cada corporação procura desenvolver seus sistemas de forma que se encaixe melhor nas suas regras de negócio, deixando assim, uma mistura de linguagens de programação, protocolos e padrões de projetos muito específicos. Neste cenário que os serviços web tem um grande destaque (MARINS, 2012).

E nesse contexto surge a Arquitetura Orientada a Serviços (SOA) e tem por objetivo a neutralidade de tecnologia independente da plataforma, que fornece os aspectos de reutilização, agilidade e interoperabilidade com a ajuda de um conjunto de serviços. A principal vantagem dos serviços é o fornecimento de uma forma comum de interação. A tecnologia que melhor se adaptou as características do SOA foram os Serviços Web ou Web Services (KUMARI; RATH, 2015).

Um Serviço Web, é definido pelo W3C em uma abordagem tipicamente SOAP, como sendo um sistema de software projetado para suportar interações interoperáveis de máquina-a-máquina através de uma rede(W3C, 2004). Outros sistemas interagem com o serviço da web de uma maneira prescrita por sua descrição usando mensagens SOAP, normalmente transmitidas usando HTTP com uma serialização XML em conjunto com outros padrões relacionados à Web (W3C, 2004). Outra definição interessante é apresentada por Benharref et al. (2011), que define os serviços web como “aplicativo que expõe sua funcionalidade através de uma interface e disponibiliza para uso por outros programas”.

Figura 1 – Funcionamento do Serviço Web

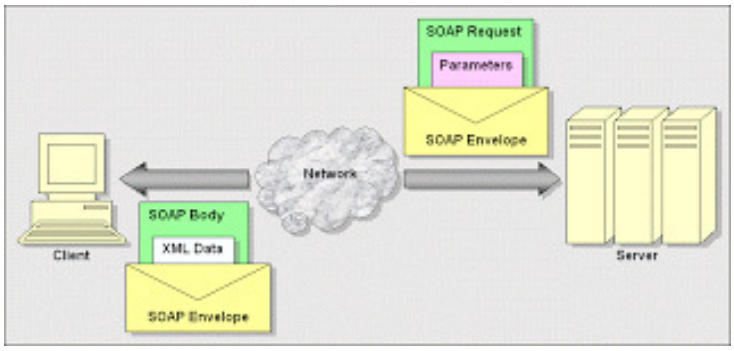


Os serviços web possibilita que aplicações em diferentes linguagens de programação, padrões de projetos e configuração hardware envie e recebe informações entre si, permitindo uma interação de forma distribuída (GONÇALVES; CASTRO, 2016). Aliado a isso, diversas tecnologias têm surgido para sua implementação, sendo que os principais delas são o protocolo SOAP e a arquitetura REST.

2.2 SOAP

O SOAP (Simple Object Access Protocol) é um protocolo de comunicação baseado em XML que proporciona a comunicação de mensagens entre aplicações por meio de HTTP (Hypertext Transfer Protocol) (W3SCHOOLS, 2017). Segundo Snell et al. (2001), SOAP se põe no topo da pilha de tecnologias para serviços web, pois se trata de um meio robusto e seguro de desenvolvimento orientado a serviços, além de ser regulamentado pelo W3C (World Wide Web Consortium).

Figura 2 – Requisição SOAP no Serviço Web



As comunicações entre as aplicações de Serviço Web no padrão SOAP fazem uso de algumas tecnologias para o encapsulamento da requisição e resposta entre um servidor

e um cliente. Estes elementos são: UDDI, WSDL, XML, Protocolo de Transporte e SOAP (W3SCHOOLS, 2017).

2.2.1 UDDI

O UDDI (Universal Description, Discovery and Integration) consiste em um serviço estruturado na forma de repositórios para nomeação e localização dos serviços web (W3SCHOOLS, 2017). Tem como principal função representar a localização central onde o provedor de serviços pode relacionar os seus serviços web, possibilitando assim a pesquisa e a descoberta desses serviços (RODRIGUES, 2009). O UDDI é um padrão e foi desenvolvido para ser conectado através de mensagens SOAP e oferecer acesso aos documentos WSDL descrevendo os protocolos e os formatos das mensagens necessários para a comunicação com determinado serviço (RODRIGUES, 2009).

2.2.2 WSDL

A Web Services Description Language (WSDL) é uma linguagem que tem como objetivo estabelecer um contrato entre o servidor e o cliente. A partir do contrato são especificados quais são os serviços, a identificação dos métodos que serão chamados nos clientes para execução destes, os tipos de dados que os serviços poderão utilizar ou retornar. Esse contrato é disponibilizado no servidor para a visualização do cliente, possibilitando o entendimento dos métodos disponíveis no servidor e como chamá-los(XAVIER, 2011).

Um arquivo WSDL define um XML Schema (XSD) para representar um serviço Web. Dispõe de uma parte abstrata, que descreve a interface de um serviço e as operações suportadas e uma parte concreta, que especifica os formatos de dados e protocolos específicos que serão utilizados. A figura 1 mostra a estrutura de um documento WSDL.

Figura 3 – Estrutura de um documento WSDL

Fonte: Rodrigues (2009)

2.2.3 XML

XML é uma linguagem de marcação criada pelo W3C a partir do SGML (Standard Generalized Markup Language). Por oferecer padronização, flexibilidade e a possibilidade de descrever classes com diversos tipos de dados, o uso de XML amenizou drasticamente os custos de desenvolvimento. Estas classes de dados são documentos XML (W3C, 2004). A linguagem tem como objetivo facilitar o compartilhamento de informações pela Web, podendo ser usado também por serviços Web.

2.2.4 Protocolo de Transporte

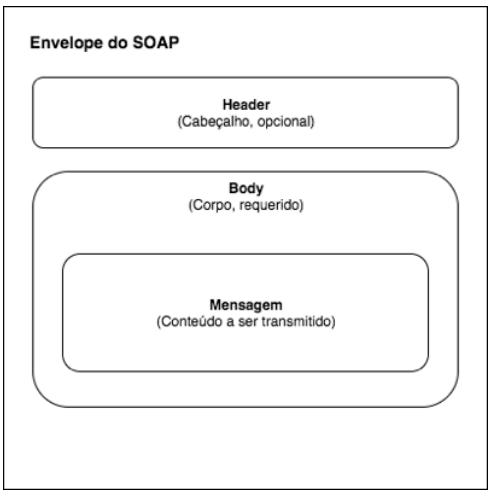
O protocolo mais comum da Internet é o HTTP, embora para configurações que exijam maior segurança no tráfego das informações o uso de HTTPS. Não tanto convencional quanto o HTTP, qualquer outro protocolo de transporte como o TCP/IP (Transfer Control Protocol/Internet Protocol), SMTP (Simple Mail Transfer Protocol) ou FTP (File Transfer Protocol), podem ser usados para transferência de mensagens (GONCALVES, 2009).

2.2.5 SOAP

SOAP é uma aplicação da especificação XML, onde são implementados envelopes baseado em XML para transporte de informações que tem conjunto de regras para traduzir tipos de aplicações e plataformas específicas para representação XML (SNELL et al., 2001).

Um envelope SOAP tem semelhanças com a estrutura básica de uma página HTML (Hyper Text Markup Language), com header sendo opcional e um body. No header do envelope, são dados referentes a configurações de entrega da mensagem, autenticação ou regras de autorização ou contexto de transações. No body por sua vez, o conteúdo da mensagem a ser transmitida é informado.

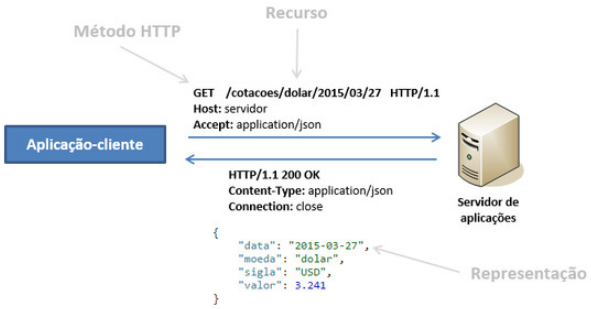
Figura 4 – Estrutura de um envelope SOAP



2.3 REST

REST (Representational State Transfer) foi criada no ano 2000 por Roy Fielding na sua dissertação de pós-doutorado (RICHARDSON; RUBY, 2007). É um estilo híbrido, derivado de vários estilos arquiteturais que são fortemente baseadas nos fundamentos de funcionamento da web, em especial, do protocolo HTTP utilizado na requisição (GET, POST, PUT, DELETE). Segundo Fielding e Taylor (2000), um estilo arquitetural é um conjunto de regras que limita as funcionalidades dos elementos arquiteturais e o grau de relacionamento entre esses elementos nos sistemas que adotam esse estilo, permitindo nomear um conjunto de padrões, técnicas e regras de forma que seja fácil identificá-los, discutir melhoramentos e aplicá-los de forma efetiva no desenvolvimento aplicações.

Figura 5 – Funcionamento do REST



A arquitetura REST difundiu-se como uma forma simplificada para desenvolvimento de serviços web ou “RESTful Web Services”, ou seja, Sistemas construídos segundo os princípios REST são chamados aplicativos RESTful (RICHARDSON; RUBY, 2007). Na prática, o REST

estabelece procedimentos para a construção de serviços Web baseados no padrão cliente/servidor, sem estado, cache, código sob demanda e interface uniforme (RICHARDSON; RUBY, 2007).

- **Cliente/servidor:** O estilo cliente/servidor define a separação das entidades em dois papéis distintos: o servidor, responsável por oferecer um conjunto de serviços, e o cliente, que busca e faz uso os serviços disponibilizados pelo servidor.

- **Sem estado:** toda comunicação deve ser sem estado, ou seja, qualquer informação necessária para atender uma requisição deve estar contida na própria requisição. Isso implica que toda informação de contexto e de estado da sessão está contida exclusivamente no cliente (FIELDING; TAYLOR, 2000).

- **Cache:** a arquitetura cache é usada para a eliminação parcial ou total de algumas comunicações entre cliente e servidor, o que melhora a eficiência, escalabilidade e performance, já que o servidor recebe uma carga menor de requisições evitando desperdício de banda, pois impede que dados que já tenham sido enviados anteriormente ao cliente sejam reenviados (FIELDING; TAYLOR, 2000).

- **Código sob demanda:** através da restrição opcional código sob demanda, REST permite que funcionalidades do cliente sejam estendidas através do download e execução de código na forma de applets ou scripts. Embora essa característica possa simplificar o cliente, reduzindo o número de recursos a serem pré-implementados, também pode acarretar em perda de visibilidade (FIELDING; TAYLOR, 2000).

- **Interface uniforme:** em virtude da restrição arquitetural, o conjunto de métodos são conhecido e padronizado, sendo que a cada execução de um método, sua semântica é visível. Isto é, a interface uniforme define que o servidor é capacitado para determinar o que deve ser feito ao receber uma requisição HTTP em uma URI apenas pela observação do método presente nessa requisição (FIELDING; TAYLOR, 2000).

2.3.1 Arquitetura Orientada a Recursos

A W3C definiu HTTP como o protocolo padrão para a transmissão de mensagens na Web. Sendo assim, ao interligar o REST ao HTTP foi possível a criação do ROA (Resource- Oriented Architecture), uma arquitetura que segue exatamente o estilo arquitetural REST, ao mesmo tempo em que utiliza o HTTP para preencher as principais lacunas teóricas com realizações concretas(RICHARDSON; RUBY, 2007). Funciona igual a outros serviços web, onde se recebe

uma requisição detalhada com uma ação a ser executada e retorna uma resposta detalhando o resultado. Outro fato muito importante para o ROA é que tanto a ação como seu escopo de execução são discriminados na requisição (FILHO, 2011).

A seguir serão detalhados os cinco componentes principais da arquitetura ROA: Recurso, Representação, Identificador Uniforme, Interface Unificada e Escopo de Execução.

2.3.1.1 Recursos

Um recurso é qualquer coisa que seja importante o suficiente para ser referenciado como algo em si (RICHARDSON; RUBY, 2007). Em uma aplicação RESTful um recurso pode ser um registro em um banco de dados, um arquivo em disco, um serviço, uma coleção de outros recursos.

Em ROA, todos os recursos devem ser identificados unicamente através de URIs. É importante que elas sigam uma estrutura intuitiva e que variem de forma previsível, pois devem refletir os relacionamentos entre os recursos que identificam. Outra função das URIs é endereçar adequadamente os recursos na Web.

2.3.1.2 Representação

Todo recurso possui uma representação, que nada mais é que o formato das mensagens trocadas entre o servidor e o cliente através dos métodos HTTP. Representações são usadas para capturar o estado atual ou previsto dos recursos (FIELDING; TAYLOR, 2000).

Um recurso pode possuir várias representações, como texto simples, XML, HTML (HyperText Markup Language) ou JSON (Java Script Object Notation), por exemplo. A escolha da melhor representação depende da necessidade do cliente. A negociação de conteúdo pode ser efetuada através de meta-dados enviados nos cabeçalhos das requisições (SILVESTRE; POLÔNIA, 2008).

2.3.1.3 Vinculação de recursos

A dinâmica de sistemas RESTful é conseguida através da navegação entre recursos, que, por sua vez, é alcançada com a utilização de enlaces embutidos nas representações.

Quando o servidor envia uma representação com enlaces, ele está enviando para o cliente uma série de URIs com estados possíveis da aplicação. Esta propriedade é perfeitamente representada através de grafos, onde os nós são os recursos e as arestas são os enlaces ligando-os (SILVESTRE; POLÔNIA, 2008).

O princípio da vinculação de recursos está relacionado com o uso da abordagem HATEOAS (Hypermedia as Engine Of Application State). Nessa abordagem, as aplicações são consideradas como uma máquina de estado onde cada página representa um estado e os links representam todas as possíveis transições de estado a partir do estado corrente.

2.3.1.4 Interface Uniforme

A interface uniforme é definida como o conjunto de operações possíveis para cada recurso. Em ROA, os métodos das requisições HTTP (GET, POST, PUT ou DELETE) são utilizados para indicar ao provedor do recurso a ação que deve ser realizada (FRANÇA et al., 2011).

- GET: Obtêm a representação de um recurso, assim como os meta-dados associados;
- DELETE: Remove um recurso;
- PUT: Cria ou atualiza um recurso na URI especificada;
- POST: Cria um recurso subordinado ou anexa informações a um recurso, respondendo com a URI do novo recurso;
- OPTIONS: Lista quais os métodos que podem ser executados no recurso;
- HEAD: Idêntico ao GET, mas omitindo a representação;

2.4 TESTE DE SOFTWARE

Com o aumento da utilização de sistemas no cotidiano das pessoas, se faz necessário uma busca contínua por mais qualidade de software, menos riscos e melhores resultados, adicionar o teste nos processos de desenvolvimento do software se tornou indispensável.

O teste de software é definido como o processo de executar um software de uma maneira controlada com o objetivo de avaliar se o mesmo se comporta conforme o especificado (CRESPO et al., 2004). Para Pressman (1995) é possível identificar algumas fases durante o processo de teste, tais como teste de unidade, teste de integração, teste de validação e teste de sistema.

- **Teste de unidade:** podemos dizer que é a menor parte que pode ser testada de um programa, sendo ela independente das demais unidades que o compõem, priorizando os pequenos trechos de códigos, métodos, sub-rotinas entre outros pontos (SOFTWARE, 2016).

- **Teste de integração:** tem por objetivo verificar a compatibilidade entre os módulos, por exemplo, cada módulo pode ser validado no teste de unidade sem apresentar erros, porém, isso não assegura que quando esses módulos passarem a se comunicar não haverá problemas relacionados à conectividade entre eles (DELAMARO et al., 2017).

- **Teste de validação:** Nesta fase é executado um conjunto de testes visando avaliar as funcionalidades do software, onde é possivel verificar se o sistema está de acordo com que foi descrito durante a etapa de levantamento de requisitos (COSTA, 2012).

- **Teste de sistema:** Nesta fase o objetivo é testar o sistema do ponto de vista do usuário, onde as condições e ambiente de teste necessita ser o mais próximo possível da realidade do usuário ou até mesmo idêntico (COSTA, 2012).

2.4.1 Teste de Desempenho

O desempenho faz parte da fase de teste de sistema, onde é responsável pela avaliação também de outros requisitos não funcionais (como segurança e confiabilidade) (COSTA, 2012). Um sistema tem bom desempenho quando apresenta um tempo de resposta adequado e aceitável, mesmo se submetendo a um volume de processamento próximo de situações reais ou de pico (BASTOS et al., 2007).

A Engenharia de Desempenho de Software (EDS) é a área responsável por estudos relacionados a desempenho. De acordo com Woodside et al. (2007), a EDS pode ser definida como a coleção de atividades vindas da Engenharia de Software direcionadas para os requisitos de desempenho. Para estes autores, existe duas abordagens possíveis dentro da ED: baseada em medição e a outra baseada em modelos. A abordagem em medição diz respeito as atividades realizadas apenas no final do ciclo de desenvolvimento, dado que o sistema real precisa ser executado e medido(FREITAS, 2013). A outra abordagem baseada em modelos, tem foco resultados quantitativos dos modelos que são usados para ajustar a arquitetura e o projeto com o objetivo de atingir os requisitos de performance (FREITAS, 2013). Tendo conceituado essa área de estudo, podemos explicar melhor a definição de teste de desempenho dada a seguir.

O Teste de Desempenho de Software (TDS) é um tipo de teste de software em quem nele são realizadas análises de desempenho para descobrir pontos no programa que contenham problemas, como por exemplo gargalos no software, ou para extrair algumas métricas de desempenho e até mesmo tempo de resposta (MEIER et al., 2007).

Um teste de desempenho atinge seus objetivos se o resultado obtido não for compatível

com o que foi especificado na construção do sistema, ou seja, se foi detectada uma especificação de desempenho do sistema que não foi cumprida. Ele pode ser usado para determinar ou validar a velocidade, escalabilidade e/ou características de estabilidade do produto em teste (FREITAS, 2013).Apesar da divergência entre autores quanto à definição dos tipos de teste de desempenho, os mais conhecidos são:

- **Teste de carga:** tem por objetivo determinar ou validar o comportamento de um sistema sob condições normais de carga. Busca verificar se o sistema cumpre os requisitos de desempenho especificados;
- **Teste de estresse:** tem por objetivo determinar o comportamento de um sistema quando ele é submetido além das condições normais de carga. Busca revelar erros e determinar os pontos de defeito do sistema.

3 Metodologia

Neste capítulo é apresentado de forma detalhada as tecnologias utilizadas para o desenvolvimento deste trabalho. Por fim, explica o desenvolvimento de um sistema de empréstimo de livros nos padrões de serviços web SOAP e REST.

3.1 TECNOLOGIAS

Para o desenvolvimento deste trabalho, está sendo usadas as seguintes tecnologias: Spring Framework, MySQL, Apache Tomcat, SoapUI e jMeter.

3.1.1 Spring Framework

A Spring Framework facilita a criação de aplicativos fornecendo uma série de opções na linguagem Java em um ambiente corporativo e com a flexibilidade para criação de muitos tipos de arquiteturas, dependendo das necessidades (MOREIRA, 2012).

Por ser um framework completo, o Spring é dividido em módulos e sub-módulos. Esses módulos são agrupados em Core Container, Data Access / Integration, Web, AOP (Aspect Oriented Programming), Instrumentation, Messaging e Test.

O módulo da Web é um dos mais utilizados pois oferece funções básicas de integração orientadas para a web, como a funcionalidade de upload de arquivos multipart e a inicialização do contêiner do IoC usando os ouvintes do Servlet e um contexto de aplicação orientado para a web. Também contém um cliente HTTP e as partes relacionadas à web do suporte remoting da Spring (JOHNSON JUERGEN HOELLER, 2016).

3.1.2 MySQL

O MySQL é um gerenciador de banco de dados relacional de código aberto (SGBD) com licença GPL, empregados em numerosas aplicações gratuitas para gerenciamento dos dados. Esse gerenciador utiliza a linguagem SQL (Structure Query Language – Linguagem de Consulta Estruturada) que é usada para acessar, inserir e gerenciar o conteúdo de um banco de dados. Nesta aplicação, o SQL é utilizado em conjunto com o Spring Framework para acessar os dados contidos no MySQL (PISA, 2012).

3.1.3 Apache Tomcat

O servidor Apache Tomcat é um container Web open source baseado em Java que surgiu para executar aplicações Web que utilizam tecnologias Servlets e JSPs. Além disso, é um servidor bem seguro e com todas as características que um container comercial de aplicações web dispõe (MEDEIROS, 2011).

3.1.4 SoapUI

SoapUI é uma ferramenta de teste API que é livre, *open source*, com uma interface gráfica fácil de usar e recursos de classe empresarial, permite que você crie e execute de forma fácil e rápida funções, regressões, conformidade e carga automáticas para testes. SoapUI não é a penas uma ferramenta de teste de API funcional, mas também nos permite realizar testes não funcionais, como teste de desempenho e segurança (KUMAR, 2015).

3.1.5 Apache JMeter

O Apache JMeter é usada para testar o desempenho em recursos estáticos e dinâmicos. Ele pode ser usado para simular um volume grande de dados em um servidor, grupo de servidores, rede ou objeto para testar sua força ou para analisar o desempenho geral sob diferentes tipos de carga. JMeter tem como característica a criação de loop e grupo de threads. O loop simula pedidos sequenciais ao servidor com um tempo predefinido. O grupo de threads foi projetado para simular uma carga simultânea. Além disso, fornece uma interface de usuário. Ele também fornece uma API que permite executar testes baseados em JMeter a partir de um aplicativo Java (HALILI, 2008). Os recursos do Apache JMeter incluem:

- Capacidade de carga e teste de desempenho de diferentes tipos de aplicativos/ servidor/protocolo (HTTP, web services SOAP/REST e FTP);
- Capacidade de extrair dados dos formatos de resposta mais populares, HTML, JSON, XML ou entre outros;
- O quadro multi-threading completo permite a amostragem simultânea por muitos threads e a amostragem simultânea de diferentes funções por grupos de threads separados;

Por ser uma ferramenta totalmente escrita em Java, o jMeter é compatível com qualquer ambiente capaz de suportar a máquina virtual Java. A cada solicitação realizada com a ferramenta, são somadas as informações de resposta e apresentado no relatório de sumário o número de

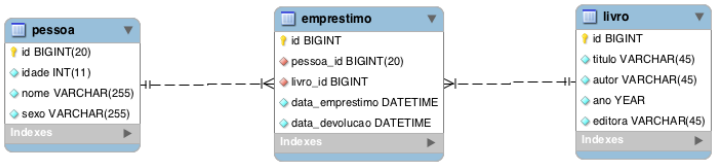
amostras, a média, o mínimo, o máximo, o desvio padrão, o percentual de erro, a vazão e os kilobytes por segundo de transferência (JMETER, 2017).

- Amostras: O número de amostras com o mesmo serviço web;
- Média: tempo médio, em milissegundos, decorrido de um conjunto de resultados;
- Mínimo: menor tempo, em milissegundos, decorrido para as amostras com o mesmo serviço web;
- Máximo: maior tempo, em milissegundos, decorrido para as amostras com o mesmo serviço web;
- Desvio padrão: apresenta os casos em que amostras se afastam do comportamento médio das demais amostras utilizadas em razão do tempo de resposta. Quanto menor for este valor, mais consistente é o padrão é o padrão de tempo das amostras coletadas;
- Erros: quantidade de amostras pedidas com erros;
- Vazão: expressa a quantidade máxima de dados que pode ser transportada de uma origem até o seu respectivo destino;
- KB/s: fluxo medido em kilobytes por segundo;

3.2 DESENVOLVIMENTO DO SISTEMA

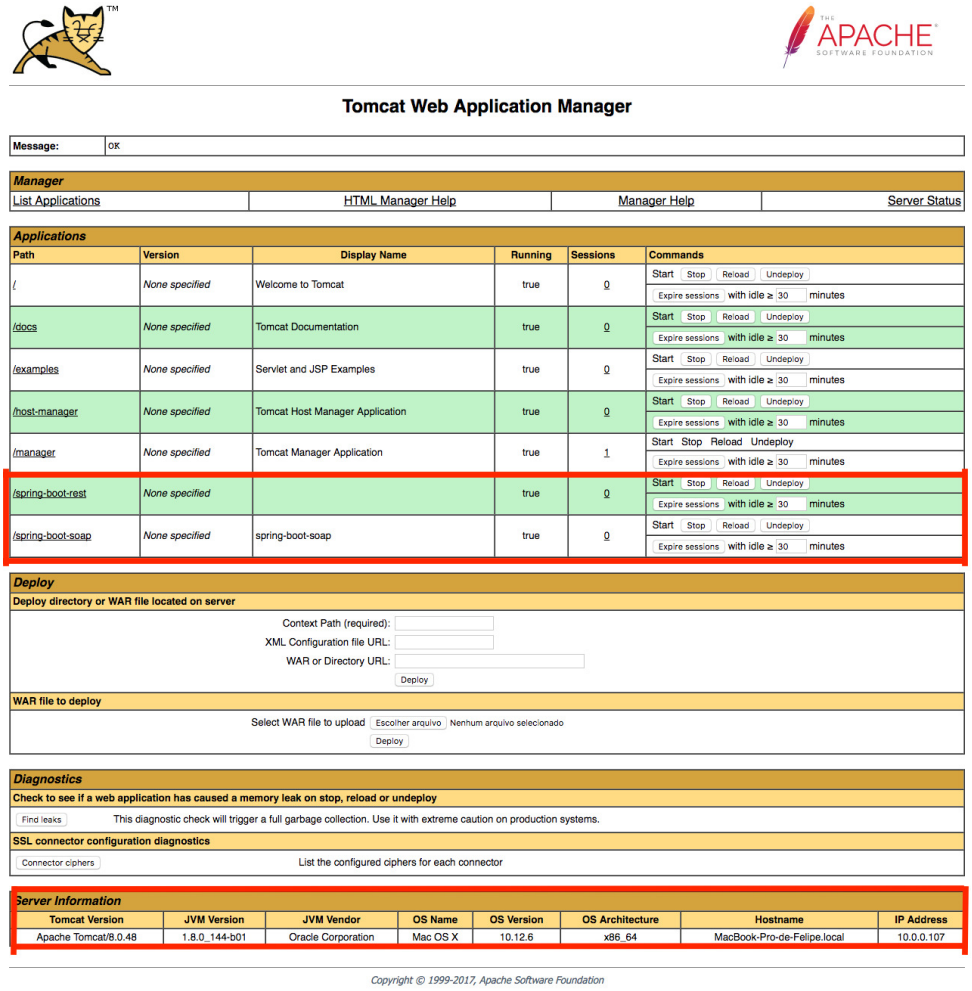
Para o desenvolvimento dos serviços web, foi criado um sistema de empréstimo de livros, onde se utilizou o Spring Framework para a criação dos serviços, permitindo a construção dos Serviços Web em SOAP e REST. O sistema permite gerenciar empréstimos de livros de uma biblioteca, possibilitando o cadastro e atualização de pessoas e livros, além de ter o controle dos livros emprestados. A Figura 6 apresenta um diagrama de entidade-relacionamento de uma base de dados feita em MySQL para armazenar as informações salvas na aplicação.

Figura 6 – Diagrama de Entidade-Relacionamento do Sistema de Empréstimo de Livros



Assim que concluídas o desenvolvimento dos serviços web em SOAP e REST, foi configurado um servidor Apache Tomcat para hospedar os serviços. As configuracões do servidor e os serviços estão sendo exibidos em destaque na Figura 7.

Figura 7 – Servidor utilizado para fazer deploy das aplicações



Tomcat Web Application Manager

Message:	OK

Manager

List Applications	HTML Manager Help	Manager Help	Server Status

Applications

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/spring-boot-rest	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/spring-boot-soap	None specified	spring-boot-soap	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy

Deploy directory or WAR file located on server

Context Path (required):
XML Configuration file URL:
WAR or Directory URL:
Deploy

WAR file to deploy

Select WAR file to upload Escolher arquivo Nenhum arquivo selecionado
Deploy

Diagnostics

Check to see if a web application has caused a memory leak on stop, reload or undeploy

Find leaks This diagnostic check will trigger a full garbage collection. Use it with extreme caution on production systems.

SSL connector configuration diagnostics

Connector ciphers List the configured ciphers for each connector

Server Information

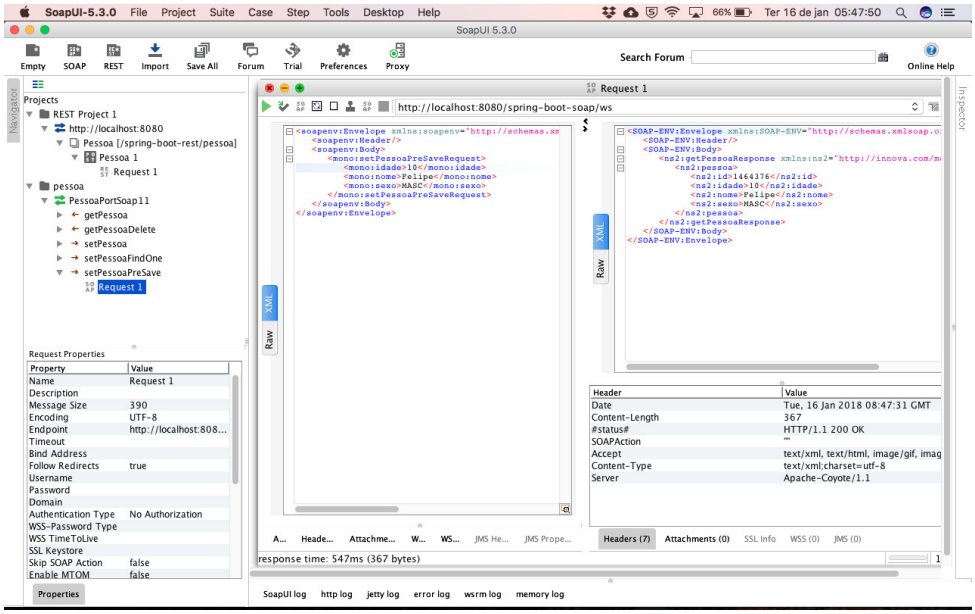
Tomcat Version	JVM Version	JVM Vendor	OS Name	OS Version	OS Architecture	Hostname	IP Address
Apache Tomcat/8.0.48	1.8.0_144-b01	Oracle Corporation	Mac OS X	10.12.6	x86_64	MacBook-Pro-de-Felipe.local	10.0.0.107

Copyright © 1999-2017, Apache Software Foundation

Com o auxilio da ferramenta SoapUI, foram realizadas requisições aos servidores REST e SOAP afim de validar o funcionando dos sistemas, então, o objetivo deste teste é assegurar que durante os testes de performance os serviços web estão respondendo as requisições solicitadas sem apresentar erros. Será utilizado a requisição POST na ação de cadastro de uma pessoa no sistema.

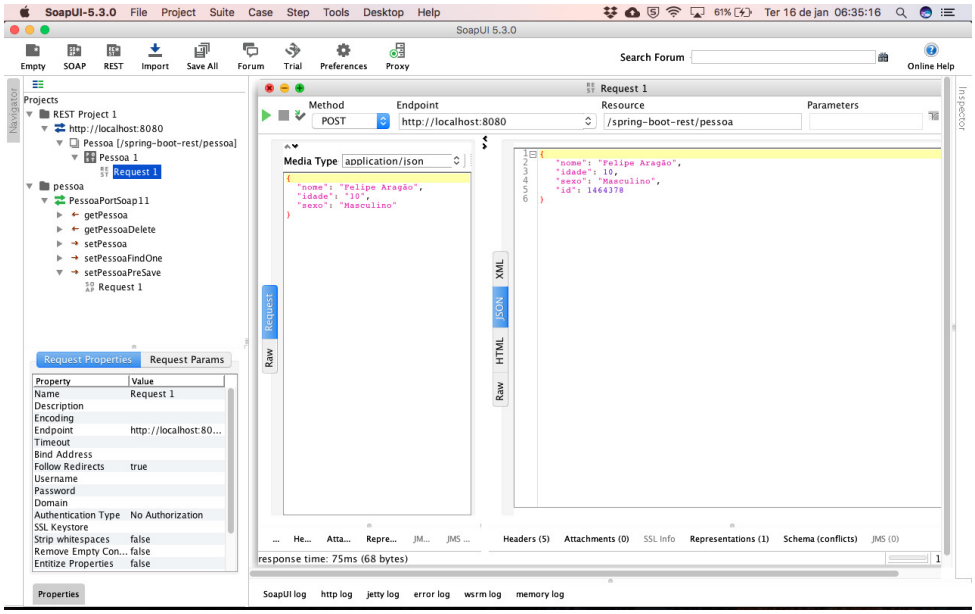
O serviço web em SOAP, que é demonstrado na Figura 8, como pode ser observado que tanto a requisição quanto a resposta são todas em XML e que levou 547 ms o tempo de resposta.

Figura 8 – Teste da Serviço Web SOAP utilizando o SoapUI



Na Figura 9, é obtido o retorno de uma chamada HTTP realizada ao Web service REST com o auxílio da ferramenta SoapUI. O a requisição e o retorno do Web service estão no formato JSON. O tempo de resposta dessa requisição foi de 75 ms.

Figura 9 – Teste da Serviço Web REST utilizando o SoapUI



A partir dos serviços web respondendo as requisições e retornando os registros em cada formato, de XML e JSON, será possível realizar testes de desempenho com a ferramenta Apache jMeter no estudo de caso.

4 Estudo de Caso

Este capítulo se destina a apresentar um estudo de caso com o intuito de comparar o desempenho de dois protocolos de comunicação de serviços Web, SOAP e REST, a fim de confrontar os seus resultados.

4.1 MÉTODO DE AVALIAÇÃO

Os testes de desempenho foram realizados com a ferramenta Apache jMeter versão 3.3, com amostras de 1 mil, 10 mil, 50 mil, 75 mil, 100 mil e 250 mil em cada serviço web. Além disso, usou-se dez usuários virtuais no intervalo de um segundo e contador de interação de acordo com a quantidade de amostras. Os testes foram realizados de forma isolada em cada serviço web para que não houvesse interferência de desempenho. A configuração da máquina utilizada para o teste de desempenho está descrita na Tabela 1.

Tabela 1 – Configuração da máquina local

Modelo	MacBook Pro
Nome do Processador	Intel Core i5
Velocidade do Processador	2,5 GHz
Número de Processadores	1
Memória	4 GB

Os serviços web baseados em SOAP e REST foram construídos para responder as requisições POST buscando salvar uma nova pessoa do Sistema de Empréstimo de Livros, a fim de gerar um fator de medição de desempenho da execução dos testes. A Figura 10 exibe a requisição HTTP para o serviço em SOAP onde é enviado as informações no formato XML. A Figura 11 mostra a requisição HTTP para o serviço web em REST e com as informações em JSON.

Figura 10 – Exibir o XML para requisição HTTP no serviços Web em SOAP

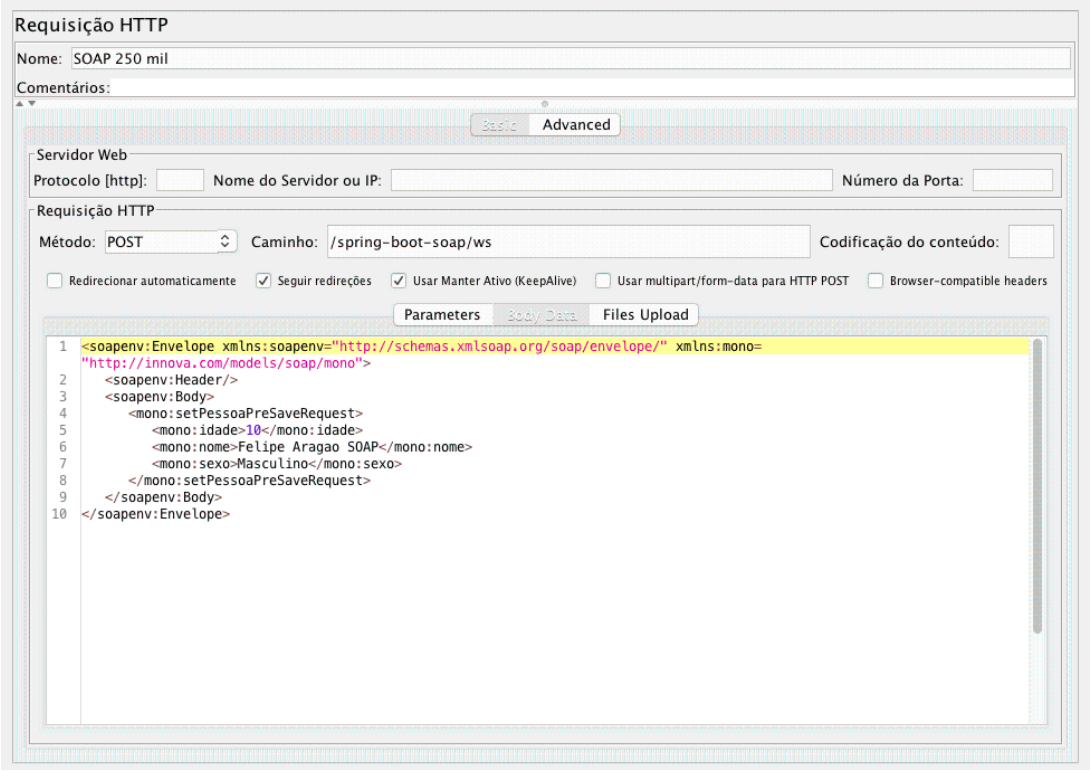
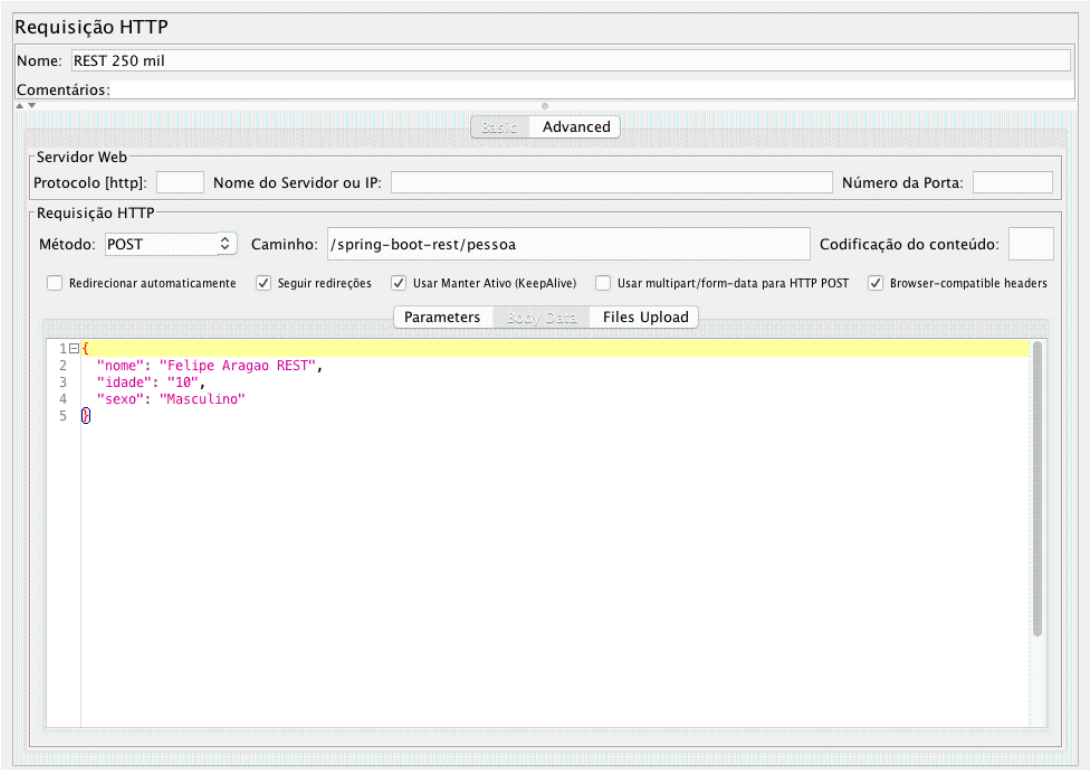


Figura 11 – Exibir o JSON para requisição HTTP no serviços Web em REST



4.2 AVALIAÇÃO DOS RESULTADOS

Para demonstrar os resultados neste trabalho, o jMeter possibilitou a exportação de duas tabelas com as informações dos testes realizados nas aplicações desenvolvidas.

Tabela 2 – Dados do relatório agregado do web service REST

Amostras	**Média**	**Mínimo**	**Máximo**	**Desvio Padrão**	**Vazão**	**KB/s**
1000	12 ms	2 ms	416 ms	26,02	502,01 /seg	119,22
10000	7 ms	1 ms	169 ms	13,26	1191,61 /seg	283,00
50000	5 ms	1 ms	189 ms	7,74	1639,40 /seg	389,34
75000	6 ms	1 ms	333 ms	10,58	1595,74 /seg	380,18
100000	6 ms	1 ms	241 ms	8,56	1542,26 /seg	367,78
250000	6 ms	1 ms	379 ms	9,84	1565,52 /seg	373,33

Tabela 3 – Dados do relatório agregado do web service SOAP

Amostras	**Média**	**Mínimo**	**Máximo**	**Desvio Padrão**	**Vazão**	**KB/s**
1000	28 ms	4 ms	507 ms	53,46	279,72 /seg	167,15
10000	14 ms	3 ms	444 ms	21,02	665,82 /seg	398,04
50000	12 ms	3 ms	300 ms	10,85	790,24 /seg	472,43
75000	11 ms	3 ms	404 ms	11,26	819,97 /seg	490,74
100000	13 ms	3 ms	829 ms	17,81	748,18 /seg	448,01
250000	12 ms	3 ms	1784 ms	20,09	800,27 /seg	479,21

Os dados da Tabela 2 apresenta a mesma quantidade de amostras da Tabela 3. Para a análise de cada métrica, odotou-se os resultados obtidos com os testes realizados com 250 mil amostras.

Na tabela 2 a Média apresentou um menor tempo de resposta de 6 ms, enquanto que a Tabela 3 apresentou o tempo maior para o mesmo parâmetro de 12 ms. A diferença representa uma variação percentual superior em 100% do sistema REST em relação ao SOAP.

O tempo Mínimo foi menor em relação aos testes realizados com 1 ms para o REST e 3 ms para o SOAP. Isso representa uma variação percentual superior em 200% no tempo Mínimo.

O REST se mostrou mais eficiente no tempo Máximo de resposta, onde levou 416 ms, enquanto que o SOAP durou 1784 ms para responder a requisição, criando assim uma variação percentual de 371% entre as duas.

O desvio padrão do SOAP de 20,09, foi superior ao apresentado pelo REST que foi de 9,84, ou seja, possui uma variação percentual de 104% entre eles. Nenhuma amostra apresentou erro durante a execução dos testes.

Na métrica de volume de transferência de dados por segundo, coluna KB/s o REST foi superior em aproximadamente 20%, tendo o menor tempo 373,33, contra 479,21 do SOAP para processar as requisições.

5 Conclusão

A implementação de serviços web em REST provou ser mais rápida e eficaz nas métricas de menor desvio padrão por requisição, menor tempo médio, mínimo e máximo por reposta de requisição, volume de transferência de dados por segundo para processar requisições e menor tamanho de pacote por solicitação, causando assim uma melhor performance.

Um dos principais fatores para o sucesso dos serviços web em REST é o fato de se basearem em JSON, que é um formato leve e basicamente texto puro e os serviços web em SOAP faz uso exclusivo de XML, um formato pesado e que exibe mais recursos como mémoria e processamento do servidor onde está hospedada o sistema. Sendo assim, aplicações distribuídas com um volume grande de requisições e fluxo intenso de troca de mensagens com muitas informações podem ter ganho expressivo de desempenho com a escolha de uso de serviço web REST.

Em trabalhos futuros, podem ser considerados outros critérios de testes, com banda larga de internet e tempo de resposta entre servidores tendo como foco outros métodos de requisição como o GET, PUT, DELETE e entre outros.

REFERÊNCIAS

BARBOSA, J.; HAHN, R.; RABELLO, S.; PINTO, S. C. C.; BARBOSA, D. N. F. Computação móvel e ubíqua no contexto de uma graduação de referência. **Brazilian Journal of Computers in Education**, v. 15, n. 3, 2007.

BASTOS, A.; RIOS, E.; CRISTALLI, R.; MOREIRA, T. et al. Base de conhecimento em teste de software. **São Paulo**, 2007.

BENHARREF, A.; SERHANI, M. A.; BOUKTIF, S.; BENTAHAR, J. Online monitoring for sustainable communities of web services. In: **12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops**. [S.l.]: IEEE, 2011.

COSTA, L. T. **Conjunto de características para teste de desempenho: uma visão a partir de ferramentas**. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio Grande do Sul, 2012.

CRESPO, A. N.; SILVA, O. J.; BORGES, C. A.; SALVIANO, C. F.; ARGOLLO, M.; JINO, M. Uma metodologia para teste de software no contexto da melhoria de processo. **Simpósio Brasileiro de Qualidade de Software**, p. 271–285, 2004.

DELAMARO, M.; JINO, M.; MALDONADO, J. **Introdução ao teste de software**. [S.l.]: Elsevier Brasil, 2017.

FIELDING, R. T.; TAYLOR, R. N. **Architectural styles and the design of network-based software architectures**. [S.l.]: University of California, Irvine Doctoral dissertation, 2000.

FILHO, O. F. F. **Serviços semânticos: uma abordagem RESTful.** Tese (Doutorado) — Universidade de São Paulo, 2011.

FRANÇA, T. C. de; PIRES, P. F.; PIRMEZ, L.; DELICATO, F. C.; FARIAS, C. **Web das coisas: conectando dispositivos físicos ao mundo digital**. 2011.

FREITAS, A. L. S. d. C. **Ontologia para teste de desempenho de software**. 2013.

GONCALVES, A. **Beginning Java EE 6 Platform with GlassFish 3: from novice to professional**. [S.l.]: Apress, 2009.

GONÇALVES, L. P.; CASTRO, R. de O. Análise de desempenho entre serviços web soap e restful utilizando a ferramenta apache jmeter. **Revista TIS**, v. 4, n. 1, 2016.

GUIMARÃES, A. S. Estratégias competitivas adotadas por empresas de tecnologia da informação. 2000.

HALILI, E. H. **Apache JMeter: A practical beginner's guide to automated testing and performance measurement for your websites**. [S.l.]: Packt Publishing Ltd, 2008.

JMETER, A. **The apache software foundation**. 2017. Acesso em: 10 janeiro 2018. Disponível em: <http://jmeter.apache.org/>.

JOHNSON JUERGEN HOELLER, K. D. C. S. R. H. T. R. A. A. D. D. D. K. M. P. T. T. E. V. P. T. B. H. A. C. J. L. C. L. M. F. S. B. R. L. A. P. C. B. T. A. A. C. D. S. O. G. R. S. P. W. R. W. B. C. S. N. S. D. R. **Spring Framework Reference Documentation**. 2016. Disponível em: <https://docs.spring.io/spring/docs/4.3.12.RELEASE/spring-framework-reference/htmlsingle/>.

KUMAR, Y. Comparative study of automated testing tools: Selenium, soapui, hp unified functional testing and test complete. **Journal of Emerging Technologies and Innovative Research**, v. 2, n. 9, p. 42–48, 2015.

KUMARI, S.; RATH, S. K. Performance comparison of soap and rest based web services for enterprise application integration. In: **2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)**. [S.l.: s.n.], 2015. p. 1656–1660.

LEMOS, A. Comunicação e práticas sociais no espaço urbano: as características dos dispositivos híbridos móveis de conexão multirredes (dhmcm). **Comunicação, mídia e consumo**, v. 4, n. 10, p. 23–40, 2008.

MARINS, C. E. F. **Desenvolvimento de um guia eletrônico de programação baseado em serviços web utilizando a abordagem REST**. [S.l.]: Monografia (Bacharelado Em Ciência da Computação), 2012.

MCFADDIN, S.; COFFMAN, D.; HAN, J.; JANG, H.; KIM, J.; LEE, J.; LEE, M.; MOON, Y.; NARAYANASWAMI, C.; PAIK, Y. et al. Modeling and managing mobile commerce spaces using restful data services. In: IEEE. **Mobile Data Management, 2008. MDM'08. 9th International Conference on**. [S.l.], 2008. p. 81–89.

MEDEIROS, H. **Introduzindo o servidor de aplicação Apache Tomcat**. 2011. Acessado em 01 de janeiro de 2018. Disponível em: <https://www.devmedia.com.br/introduzindo-o-servidor-de-aplicacao-apache-tomcat/27939>.

MEIER, J.; FARRE, C.; BANSODE, P.; BARBER, S.; REA, D. **Performance Testing Guidance for Web Applications: Patterns & Practices**. Redmond, WA, USA: Microsoft Press, 2007. ISBN 9780735625709.

MELLO, E. R. de; WANGHAM, M. S.; FRAGA, J. da S.; CAMARGO, E. Segurança em serviços web. **Livro de Minicursos do VI Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais. Santos: SBC**, p. 1–48, 2006.

MENASCÉ, D. A.; ALMEIDA, V. A. Planejamento de capacidade para serviços na web: Métricas, modelos e métodos. **Rio de Janeiro: Campus**, 2003.

MOREIRA, W. C. **Uma arquitetura de software para sistemas de pesquisa das pneumonias na infância**. Dissertação (Mestrado) — Universidade Federal de Goiás, 2012.

NASCIMENTO, M. E. H. et al. Uma arquitetura de serviços web como meio de intercâmbio de dados entre sistemas heterôgeneos. **TCC (graduação) - Universidade Federal de Santa Catarina, Campus Araranguá, Curso de Tecnologias da Informação e Comunicação.**, 2013.

PILIOURA, T.; HADJIEFTHYMIADES, S.; TSALGATIDOU, A.; SPANOUDAKIS, M. Using web services for supporting the users of wireless devices. **Decision Support Systems**, Elsevier, v. 43, n. 1, p. 77–94, 2007.

PISA, P. **O que é e como usar o MySQL?** 2012. Disponível em: <http://www.techtudo.com.br/artigos/noticia/2012/04/o-que-e-e-como-usar-o-mysql.html>.

PRESSMAN, R. S. **Engenharia de software**. [S.l.]: Makron books Sao Paulo, 1995. v. 6.

RICHARDSON, L.; RUBY, S. **RESTful Web Services-Web services for the Real World. 2007**. [S.l.]: O'Reilly Media, 2007.

RODRIGUES, L. C. R. Arquitetura rest. Available at http://monografias.ice.ufjf.br/tcc-web/tcc?id=17. 2009.

SILVESTRE, E.; POLÔNIA, P. V. Uma aplicação da arquitetura orientada a recursos. **TCC (graduação) - Universidade Federal de Santa Catarina (UFSC)**, 2008.

SNELL, J.; TIDWELL, D.; KULCHENKO, P. **Programming Web Services with Soap**. O'REILLY & ASSOC INC, 2001. ISBN 0596000952. Disponível em: <http://www.ebook.de/de/product/3246387/james_snell_doug_tidwell_pavel_kulchenko_programming_web_services_with_soap.html>.

SOFTWARE, T. de. 2016. Disponível em: <http://testesdesoftware.com/teste-de-unidade/>.

W3C. **Web Services Architecture**. 2004. Disponível em: <https://www.w3.org/TR/ws-arch/#whatis>.

W3SCHOOLS. **XML Soap**. 2017. Disponível em: <https://www.w3schools.com/xml/xml_soap.asp>.

WOODSIDE, M.; FRANKS, G.; PETRIU, D. C. The future of software performance engineering. In: IEEE. **Future of Software Engineering, 2007. FOSE'07**. [S.l.], 2007. p. 171–187.

XAVIER, O. C. **Serviços web semânticos baseados em restful: Um estudo de caso em redes sociais online. 2011**. Dissertação (Mestrado) — Universidade Federal de Goiás, Brasil, 2011.