INSTITUTO ENSINAR BRASIL

FACULDADES INTEGRADAS DE CARATINGA

FELIPE DA SILVEIRA DE JESUS

Comparativo da manutenibilidade em Web Services SOAP e REST conforme a ISO 25010

CARATINGA

2017

FELIPE DA SILVEIRA DE JESUS

FACULDADES INTEGRADAS DE CARATINGA

Comparativo da manutenibilidade em Web Services SOAP e REST conforme a ISO 25010

Monografia apresentada ao curso de Ciência da Computação das Faculdades Integradas de Caratinga como exigência parcial para obtenção título de Bacharel em Ciência da Computação, sob orientação do Professor MSc. Elias de Souza Gonçalves.

CARATINGA

2017

rede de ensino DOCTUM

FACULDADES INTEGRADAS DE CARATINGA

FOLHA DE APROVAÇÃO

O Trabalho de Conclusão de Curso intitulado: COMPARATIVO DA MANUTENIBILIDADE EM WEB SERVICES SOAP E REST CONFORME A ISO 25010, elaborado pelo aluno FELIPE DA SILVEIRA DE JESUS foi aprovado por todos os membros da Banca Examinadora e aceita pelo curso de Ciência da Computação das Faculdades Integradas de Caratinga, como requisito parcial da obtenção do título de

BACHAREL EM CIÊNCIA DA COMPUTAÇÃO

Caratinga, 13 de Dezembro 2017

Prof. MSc. Elias de Souza Gonçalves (Orientador)

Profa. MSc. Fabrícia Pires Souza (Debatedor 1)

Prof. Wanderson Miranda Nascimento (Debatedor 2)

AGRADECIMENTOS

Primeiramente agradeço a Deus pela vida, pela saúde, e por ter me dado forças para concluir mais uma etapa da minha vida, aos meus pais que me proporcionaram condições de estudar e sempre me apoiaram e incentivaram a prosseguir nesta jornada, aos meus colegas de classe por sempre estarem trocando conhecimento, que ao longo dos anos foi fundamental para o crescimento tanto quanto pessoal quanto intelectual, em especial aos que se tornaram amigos nessa caminhada.

Agradeço também a todos os professores, pelos valorosos ensinamentos nesses anos, em especial o meu orientador Elias de Souza Gonçalves por ter me guiado para que esse trabalho se realizasse, me apoiando durante toda a caminhada até este momento.

RESUMO

Em meados do século XX, surgiu a necessidade de se padronizar a comunicação entre diferentes plataformas (PC, Mainframe, Mac, Windows, Linux entre outros) e linguagens de programação (PHP, C#, Java entre outras), devido ao crescimento da comunicação por meio da internet e da chegada de dispositivos eletrônicos mais modernos.

Com isso surgiu o conceito de *web services*, que tem se apresentado como uma nova e importante tecnologia de computação. Todas as maiores empresas de software do mundo estão desde o momento da especificação desta tecnologia promovendo seu uso, crescimento e até mesmo a adoção destes em suas plataformas de desenvolvimento de softwares.

Um *web service* tem por objetivo integrar funcionalidades e reutilizar rotinas, tornar os aplicativos automatizados, distribuídos, gerenciáveis e escaláveis, atingindo assim de maneira certeira a estratégia de muitos gerentes e diretores da área de TI: a de utilizar uma plataforma já existente de aplicativos, mas provendo sua integração e distribuição de uma maneira simples, utilizando conjuntos de hardwares e softwares existentes.

Essencialmente, um *web service* faz com que os recursos da aplicação estejam acessíveis à rede de uma forma padronizada, acerca disto uma aplicação pode invocar outra para efetuar tarefas simples ou complexas mesmo que as aplicações tenham sido desenvolvidas em linguagens e/ou sistemas diferentes, fazendo com que a aplicação cliente possa analisar e obter os recursos disponibilizados pelo *web service*.

A proposta deste trabalho é analisar a arquitetura SOAP e a arquitetura REST, no intuito de descobrir qual técnica pode produzir melhores resultados tendo como base para análise a ISO 25010 no quesito da manutenibilidade. O questionário aplicado foi desenvolvido tendo como base a pesquisa bibliográfica realizada sobre as arquiteturas de *web service* SOAP e REST. A pesquisa contou com respostas de 60 profissionais da área de TI, que de maneira geral mostraram utilizar se não ambas as arquiteturas apresentadas no trabalho, pelo menos uma delas.

Palavras-chave: Arquitetura. *Web Service*. Manutenibilidade.

ABSTRACT

In the mid-twentieth century, there was a need to standardize communication between different platforms (PC, Mainframe, Mac, Windows, Linux and others) and programming languages (PHP, C #, Java among others) through the internet and the arrival of more modern electronic devices.

With this came the concept of web services, which has been presented as a new and important computing technology. All the major software companies in the world are from the moment of specifying this technology promoting their use, growth and even the adoption of these in their software development platforms.

A web service aims to integrate functionality and reuse routines, make the applications automated, distributed, manageable and scalable, thus accurately achieving the strategy of many IT managers and directors: to use an existing application platform, but providing its integration and distribution in a simple way using existing hardware and software sets.

Essentially, a web service makes the application's resources accessible to the network in a standardized way, about which one application can invoke another to perform simple or complex tasks even if the applications have been developed in different languages and / or systems, doing with which the client application can analyze and obtain the resources made available by the web service.

The purpose of this work is to analyze the SOAP architecture and the REST architecture, in order to discover which technique can produce better results based on ISO 25010 in the maintenance question. The applied questionnaire was developed based on the bibliographic research carried out on the SOAP and REST web service architectures. The research counted on the responses of 60 IT professionals, who in general have shown if not both the architectures presented in the work, at least one of them.

Keywords: Architecture. Web Service. Maintenance.

LISTA DE SIGLAS

HTTP – *Hypertext Transfer Protocol* (Protocolo de Transferência de Hipertexto)

IDL – *Interface Description Language* (Linguagem de Definição de Interface)

ISO – *International Organization for Standardization* (Organização Internacional para Padronizações)

JSON – *JavaScript Object Notation*(Notação de Objetos JavaScript)

OASIS – *Organization for the Advancement of Structured Information Standards* (Organização para o Avanço de Padrões em Informação Estruturada)

PHP – *Hypertext Preprocessor* (Pré-processador de hipertexto)

REST – *Representational State Transfer* (Transferência de Estado Representacional)

SOAP – *Simple Object Access Protocol* (Protocolo Simples de Acesso a Objetos)

TI – Tecnologia da Informação

UDDI – *Universal Description, Discovery and Integration* (Descrição Universal, Descoberta e Integração)

URI – *Uniform Resource Identifier* (Identificador Uniforme de Recursos)

W3C – *World Wide Web Consortium* (Consórcio World Wide Web)

WSDL – *Web Services Description Language* (Linguagem de Descrição de Serviços Web)

XML – *Extensible Markup Language* (Linguagem Extensível de Marcação Genérica)

LISTA DE ILUSTRAÇÕES

SUMÁRIO

INTRODUÇÃO

Em uma empresa é cada vez mais comum a necessidade da troca de informações entre diferentes sistemas. Uma forma prática e de baixo custo para solucionar a incompatibilidade de sistemas e garantir a sua comunicação são os *web services*.

Um *web service* permite que diferentes aplicações possam realizar a troca de informações entre si, ultrapassando barreiras como o tipo de plataforma ou as linguagens de programação utilizadas, proporcionando assim uma maior interoperabilidade entre sistemas distribuídos e facilitando os processos de negócios.

Outra característica de um *web service* que deve ser destacada é a reutilização dos componentes que pertencem aos sistemas integrados, onde cada componente pode representar um serviço distinto, podendo participar de múltiplos sistemas provendo maiores benefícios imediatos e aumento da agilidade do negócio. Como cada empresa ao elaborar um projeto busca a diminuição dos custos, aumento da produtividade e um maior lucro, cabe a esta empresa ao decidir pelo uso de um *web service* se certificar de que a arquitetura escolhida para desenvolvimento retorne o esperado.

Dentre as arquiteturas de *web services* disponíveis, temos o SOAP e o REST. SOAP é um protocolo de transferência de mensagens em formato XML para uso em ambientes distribuídos. O padrão SOAP funciona como um tipo de *framework* que permite a interoperabilidade entre diversas plataformas com mensagens personalizadas. REST é um protocolo de comunicação, baseado no protocolo de hipermídia HTTP. Porém ele não impõe restrições ao formato da mensagem, apenas no comportamento dos componentes envolvidos.

Visto que um *software* está sujeito a mudanças por uma série de fatores, como por exemplo, a correção de erros, adequação a novos requisitos, aumento da suportabilidade ou adequação a um ambiente novo, a manutenibilidade de *software* aplica um papel de suma importância na área da tecnologia da informação.

Tendo em vista as informações apresentadas anteriormente, este trabalho busca comparar as arquiteturas SOAP e REST destacando a facilidade de se aplicar

a manutenibilidade nas mesmas, levando-se em consideração as normas encontradas na ISO 25010.

Para isso, elaborou-se um questionário dividido em quatro grupos de questões: 1º) Perfil do respondente - buscava obter informações sobre o perfil do entrevistado. 2º) SOAP e REST – Conhecimento em *web service* baseado em SOAP e REST. 3º) SOAP – Conhecimento em *web service* baseado em SOAP. 4º) REST – Conhecimento em *web service* baseado em REST. A pesquisa contou com respostas de 60 profissionais da área de TI.

Além deste capítulo de introdução este trabalho está estruturado com um capítulo que irá descrever o referencial teórico, apresentando as principais características das arquiteturas a serem estudadas; com um capítulo para apresentar a metodologia utilizada para alcançar os resultados e com um capítulo onde serão mostrados os resultados obtidos com a aplicação da pesquisa aos profissionais de Tecnologia da Informação. Ao final será apresentado o fechamento da pesquisa realizada, relacionando os objetivos e os resultados e sugestão de tópicos a serem explorados como trabalhos futuros.

1. REFERENCIAL TEÓRICO

Nesta seção será descrito o referencial teórico deste trabalho, o qual aborda as principais características das arquiteturas SOAP e REST, além de abordar o funcionamento das mesmas. Visto que a regra de manutenibilidade da ISO 25010 é o meio utilizado para comparar as arquiteturas estudadas, esta seção também aborda assuntos referentes à ISO 25010. Permitindo assim uma melhor compreensão dos assuntos que são tratados na Metodologia (capítulo 2).

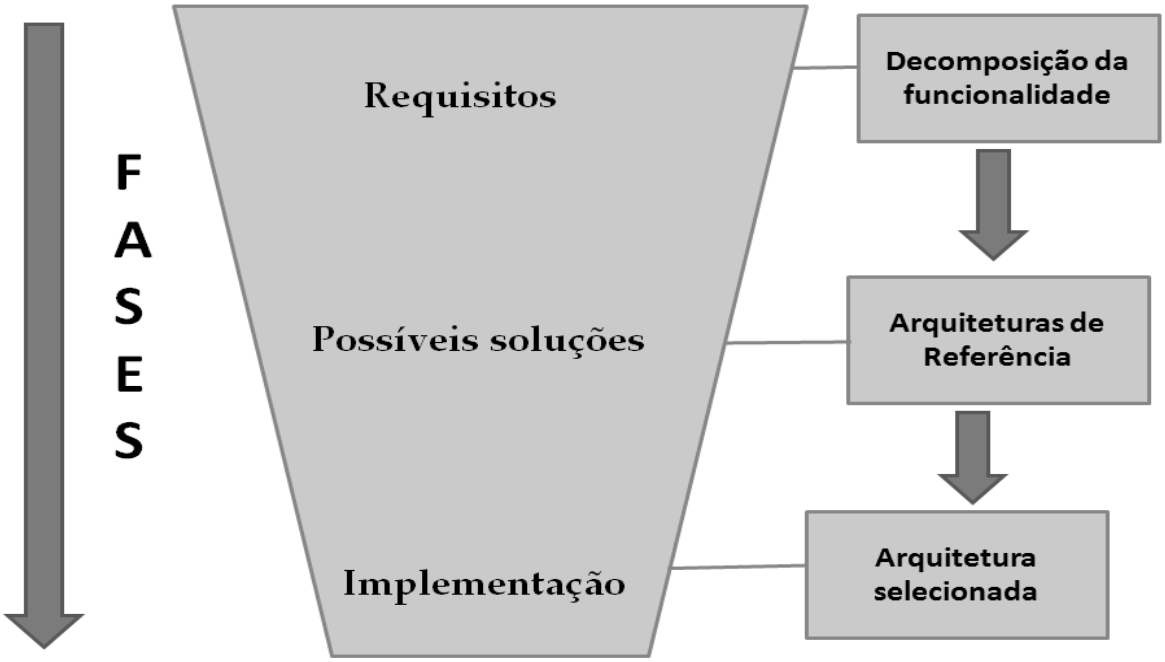
1.1 Arquitetura de software

Arquitetura de *software* é uma área que lida com a estrutura dos componentes de dentro de um *software*, onde se englobam os relacionamentos entre qualquer parte do sistema, seus princípios e diretrizes, assim conduzindo o desenvolvimento do projeto e evolução do mesmo.

A arquitetura de *software* é uma subárea da engenharia de *software*, que teve seu surgimento por volta da década de 1980, quando alguns pesquisadores e profissionais da época começaram a apontar a necessidade de se considerar o estado organizacional/arquitetural dos sistemas (SILVA, 2008). De acordo com Sommerville (2011, p.105), “O projeto de arquitetura é um processo criativo no qual você projeta uma organização de sistema para satisfazer aos requisitos funcionais de um sistema”.

A utilização de uma arquitetura no gerenciamento do processo de *software* pode fornecer suporte para a estimativa de custos na gerência do processo e principalmente pode atuar como a estrutura principal para atender aos requisitos do sistema.

Figura 1 - Níveis de descrição de um sistema



Fonte: Adaptada de Mendes (2003)

Para entender melhor o seu papel, a figura 1 ilustra o afunilamento nas descrições arquiteturais de um sistema através das fases do processo de desenvolvimento do *software*, que começa pela concepção do sistema, passa pela análise e aceitação dos requisitos e vai até sua efetiva implementação.

1.2 Web service

Web Service ou “serviço web” são aplicações que disponibilizam serviços e funcionam do lado do servidor, servem como uma solução para integração de diferentes aplicações, assim possibilitando uma comunicação entre elas.

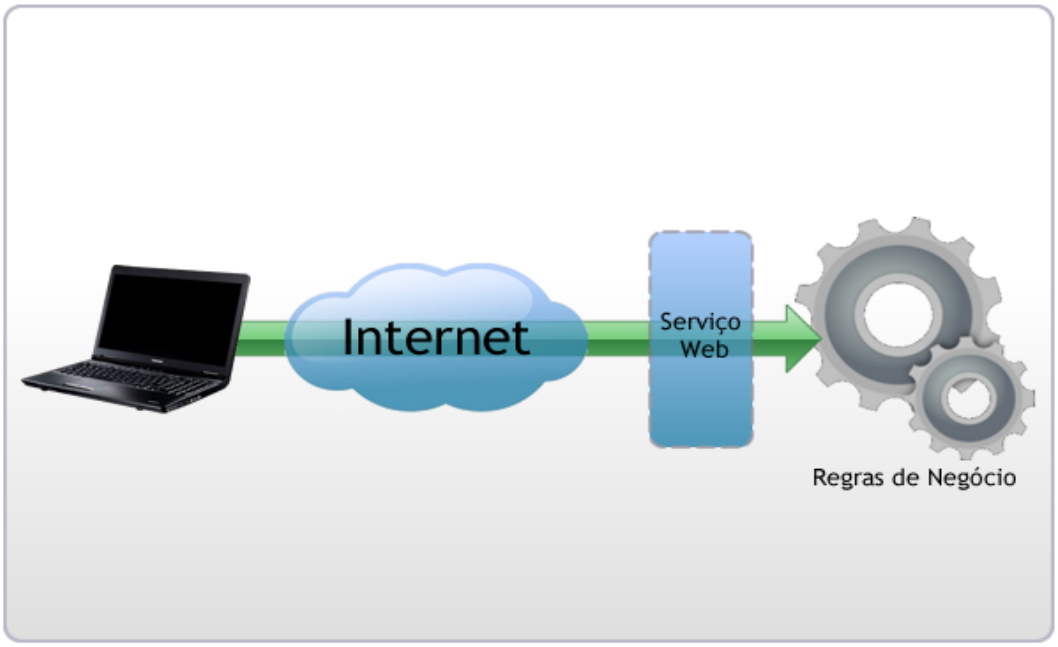
Com base na definição do W3C, *web services* são aplicações auto contidas, que possuem interface baseadas em XML e que descrevem uma coleção de operações acessíveis através de rede, independentemente da tecnologia usada na implementação do serviço (W3C, 2011).

Um *web service* pode ser construído em qualquer linguagem, assim como as aplicações que irão utilizar seus serviços, pois ao se comunicarem eles trocam suas informações em um formato universal, como, por exemplo, o XML e JSON.

Segundo Tidwell, Snell e Kulchenko (2001, p.8), o termo Serviço Web pode ser definido como “uma funcionalidade de uma aplicação acessível por meio de uma interface de rede construída utilizando tecnologias padrões de Internet”. Tal funcionalidade, justamente por ser baseada nos protocolos de internet, torna-se acessível para qualquer outra aplicação independente da linguagem ou da plataforma para a qual foi desenvolvida, desde que esta aplicação, esteja apta a se comunicar segundo um determinado protocolo.

Em uma abordagem apresentada por Gonsalves (2009, p.393), serviços web são definidos como “um tipo de lógica de negócio exposta por meio de uma interface de serviço para uma aplicação cliente”.

Figura 2 - Acesso a regras de negócio de uma aplicação com uso de serviços web



Fonte: Adaptada de Tidwell, Snell e Kulchenko (2001)

Na figura 2 conseguimos observar este fluxo da informação onde uma máquina cliente ao ter acesso à internet utiliza-se de um serviço web para acessar as regras de negócio de outra aplicação.

1.3 Soap

SOAP iniciou em 1998 e em 1999 a Microsoft™ lançou a versão 1.0 do protocolo, época que ainda não havia sido criado nenhuma linguagem de esquema ou tipos de sistema para XML. Portanto, nesta época o foco maior do protocolo era definir tipos de sistema, derivados de tipos primitivos formando tipos de dados semelhantes a estruturas e composições acessadas por posição como em vetores. Somente depois de ter esses tipos representacionais definidos é que eram modelados de fato os tipos comportamentais, como métodos e operações (BOX, 2001).

Em 2000 a IBM™ começou a trabalhar no SOAP 1.1, e em 2001 o W3C lançou a linguagem WSDL. Ainda em 2000, UDDI foi desenvolvido pela OASIS (*Organization for the Advancement of Strutured Information Standards*) permitindo a publicação e descoberta de serviços web. Com suporte de grandes empresas estavam sendo desenvolvidas as principais tecnologias envolvidas em um Serviço Web que emprega o protocolo SOAP (GONSALVES, 2009).

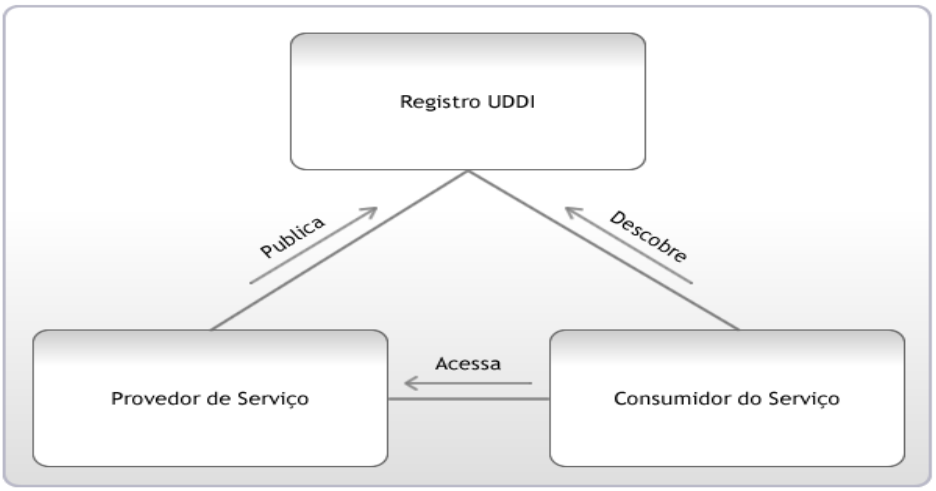
SOAP é um protocolo que não depende de plataforma e que utiliza o XML, para sua comunicação com outras aplicações sobre o protocolo HTTP. Cada mensagem, tanto enviada como recebida, é empacotada em uma mensagem SOAP com especificações definidas por um WSDL, que é um padrão que utiliza marcações em XML para definir a estrutura da mensagem. O protocolo utilizado é o HTTP, pois o *firewall*, que é uma parede de fogo ou barreira de segurança entre um computador e a rede, permite o tráfego HTTP com poucas limitações, o que possibilita enviar e receber mensagens SOAP por meio de conexões HTTP (DEITEL, 2010).

A solicitação e todos os parâmetros são empacotados em mensagem SOAP e são enviadas ao envelope SOAP. Quando esta mensagem chega ao servidor ela é analisada pelo WSDL, que é encarregado de realizar as devidas validações da mensagem, onde as mesmas são especificadas em um XML. Passando pela validação, o conteúdo da mensagem é processado. Com isso o *web service* chama o método desejado e passa seus respectivos parâmetros, enviando a mensagem de resposta em outra mensagem SOAP. Por fim, o cliente analisa a resposta para recuperar os dados da mensagem (DEITEL, 2010).

O princípio básico de funcionamento do SOAP é simples, consiste na

descoberta do serviço registrado em um registro (UDDI) e o acesso ao mesmo, como mostra a figura 3, que descreve um consumidor de serviço que pode ser considerado como o cliente, onde este acessa um provedor de serviço e o registro UDDI.

Figura 3 - A descoberta de um serviço mediante consulta a um registro UDDI



Fonte: Adaptada de Shomoyita e Ralph (2011)

Um envelope SOAP lembra muito a marcação da estrutura básica de uma página HTML, contendo um cabeçalho (opcional) e um corpo. No cabeçalho do enveloppe, são informados dados referentes a configurações de entrega da mensagem, autenticação ou regras de autorização ou contexto de transações. No corpo por sua vez, o conteúdo da mensagem a ser transmitida é informado.

Figura 4 - Estrutura de um envelope SOAP

Fonte: Adaptada de Tidwell, Snell e Kulchenko (2001)

Pode ser observado na figura 4, a estrutura de um envelope SOAP que é composto do cabeçalho, corpo do envelope e o corpo da mensagem.

1.4 Uddi

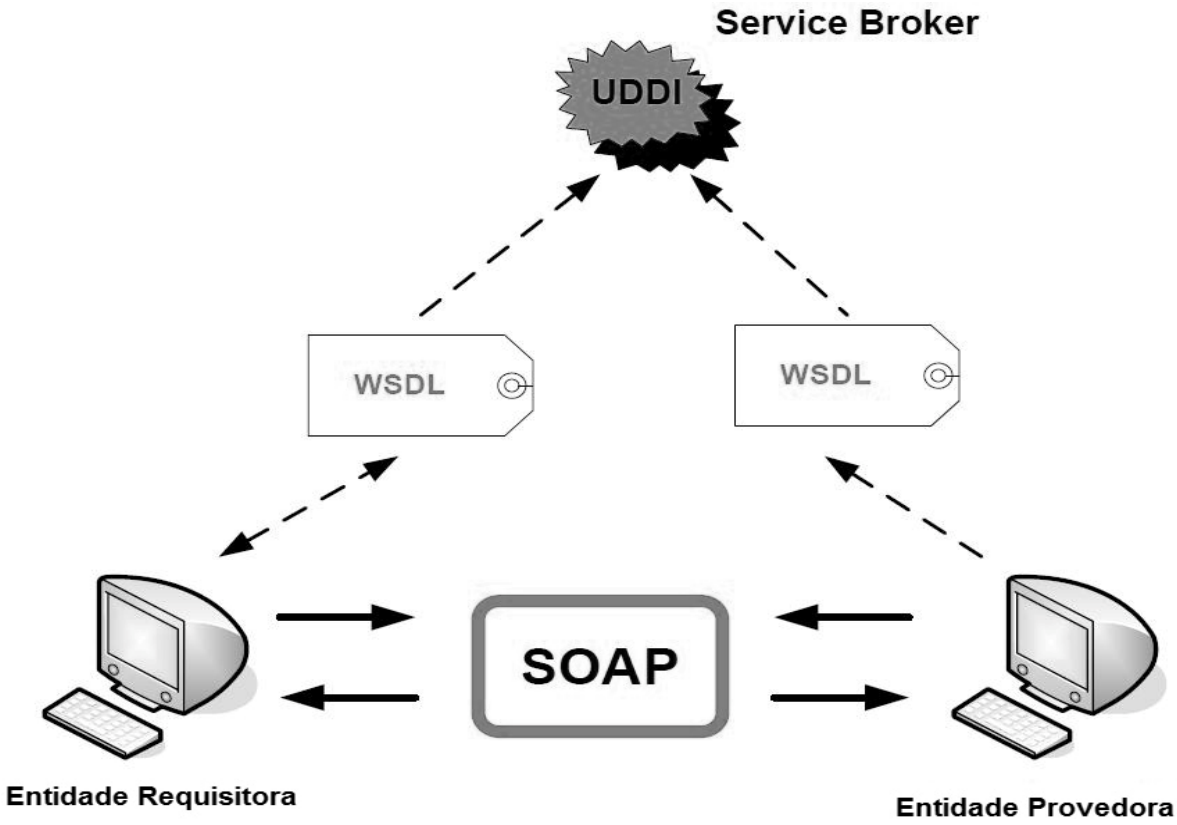
O padrão UDDI foi criado em agosto de 2000 pela organização OASIS e especifica um protocolo para obter e atualizar um diretório comum de informação dos serviços Web. O diretório inclui informações sobre *Service Providers*, os serviços que eles fornecem e os protocolos que são implementados por aquele serviço (MEDYK, 2006).

O padrão foi desenvolvido para ser acessado através de mensagens SOAP e fornecer acesso aos documentos WSDL descrevendo os protocolos e os formatos das mensagens necessários para a interação com determinado serviço listado no diretório.

A intenção de utilização do UDDI é permitir que aplicações que precisam comunicar-se umas com as outras por meio da *Web*, possam encontrar informações que viabilizem esta comunicação.

A especificação de UDDI para suporte a serviços web, consiste da implementação de bases de dados de serviços web on-line e distribuídas. Mais especificamente os registros UDDI suportam o gerenciamento de meta-informação, as quais descrevem serviços em particular. Normalmente estas meta-informações são representadas em linguagem WSDL. Estes registros possibilitam ainda a descoberta de forma automática ou semi-automática da composição de serviços, podendo desta forma ser visto como um diretório para serviços web (BLAKE, 2007).

Figura 5 - Interação entre as tecnologias *Web Services*



Fonte: Adaptada de Medyk (2006)

A figura 5 mostra a interação entre a arquitetura SOAP, UDDI e WSDL. Onde uma entidade requisitora e uma entidade provedora se comunicam através da arquitetura SOAP, sendo ambas podem acessar um UDDI tendo o WSDL como intermediador.

1.5 Rest

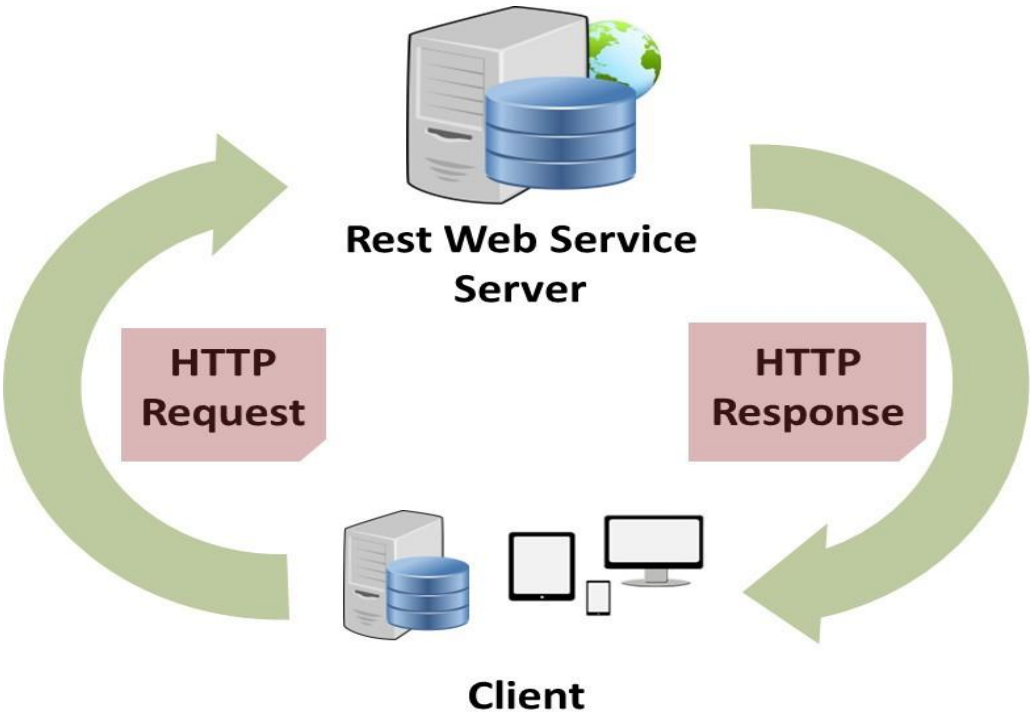
A arquitetura REST foi idealizada e definida por Roy Thomas Fielding em sua tese de PhD, muito embora não tenha sido inicialmente definido com o propósito para o qual é utilizado hoje. De fato o propósito inicial foi “designar uma arquitetura de design e desenvolvimento para a Web moderna...” (FIELDING, 2000, p.107). Porém, partindo desta apresentação, a arquitetura REST difundiu-se como uma forma simplificada para implementação de serviços web ou “*RESTful Web Services*” (RICHARDSON e RUBY, 2007).

Segundo Fielding (2000), o REST é um estilo de arquitetura para sistemas de hipermídia distribuídos, ou seja, arquitetura criada para se trabalhar com protocolos HTTP. REST é uma referência para um estilo arquitetônico de se desenvolver *web service*, onde o conjunto dos serviços criados é chamado de *web service* ou “serviço web”.

Os serviços implementados seguem o padrão *web*, por exemplo: todos os métodos implementados seguindo esta arquitetura são identificados por um URI único. Sempre que o servidor recebe uma solicitação o mesmo já sabe exatamente qual serviço deve executar. Além disso, as chamadas aos serviços podem ser realizadas tanto em um aplicativo quanto por um navegador de internet. Após a execução destes serviços eles podem ser armazenados no cache dos navegadores, então, uma solicitação do tipo POST pode ter seus resultados armazenados e posteriormente reutilizados, deixando operações subsequentes mais rápidas, pois poderá carregar os dados direto do cache (DEITEL, 2010).

REST é um estilo de arquitetura direcionado para sistemas de hipermídia distribuídos. Basicamente esse estilo é composto por dois papéis: Cliente e Servidor.

Figura 6 - Estilo Arquitetural Cliente-Servidor



Fonte: Adaptada de Fielding (2000)

Na figura 6 podemos observar esta interação de uma máquina cliente e um servidor por meio da arquitetura REST utilizando o protocolo HTTP.

1.6 Xml

O padrão XML hoje é largamente utilizado, por facilitar a compreensão por humanos e até certo ponto por máquinas, do conteúdo que descrevem. Este padrão de marcação teve seu início em 1998 e evoluiu juntamente com o SOAP. Porém, o mesmo não está tão fortemente atrelado ao SOAP, quanto o SOAP a ele, visto que a base de funcionamento do protocolo SOAP é fundamentada na troca de mensagens serializadas e envelopadas em documentos XML. Outros padrões como o REST também podem empregar o XML como meio de serialização de objetos.

XML é uma linguagem utilizada para gerar linguagens de marcação para necessidades especiais capaz de descrever diversos tipos de dados. Tem como objetivo principal a facilidade de compartilhamento de informações através da internet.

De acordo com Silberschatz, Korth e Sudarshan (2006, p.264):

> Ao contrário da HTML, XML não prescreve o conjunto de marcações permitidas, e o conjunto pode ser escolhido conforme a necessidade por cada aplicação. Esse recurso é a chave para a principal função do XML na representação e troca de dados, enquanto HTML é usada principalmente para a formatação de documentos.

Um documento XML nada mais é de que um arquivo texto com dados representados em um formato padrão no qual é codificada toda a informação contida no documento.

1.7 Wsdl

WSDL é uma linguagem de marcação utilizada para descrever um *web service*. Desenvolvida pela IBM e Microsoft, o documento WSDL descreve qual o serviço que o *web service* oferece como ele se comunica e onde ele pode ser

encontrado (RIBEIRO, BRAGA e SOUSA, 2011).

Na arquitetura do protocolo SOAP o registro UDDI deve apontar para um arquivo WSDL público, disponível na internet para potenciais consumidores do serviço web. O arquivo WSDL é escrito em linguagem de definição de interface IDL, e define a interface do serviço web, informando os tipos de mensagens suportadas, porta, protocolo de comunicação, operações suportadas, localização, e demais informações que o desenvolvedor do serviço web julgar necessária informar para quem deseje consumir o serviço. Para assegurar a interoperabilidade, um serviço web padrão é preciso que o consumidor e o produtor compartilhem e entendam as mensagens. Sendo assim, este é o foco do WSDL (ORT, 2005).

O WSDL disponibiliza um mecanismo estruturado para descrever as operações que um *web service* pode executar, além do formato das mensagens que pode processar, os protocolos que suporta e o ponto de acesso de uma instância de um *web service*.

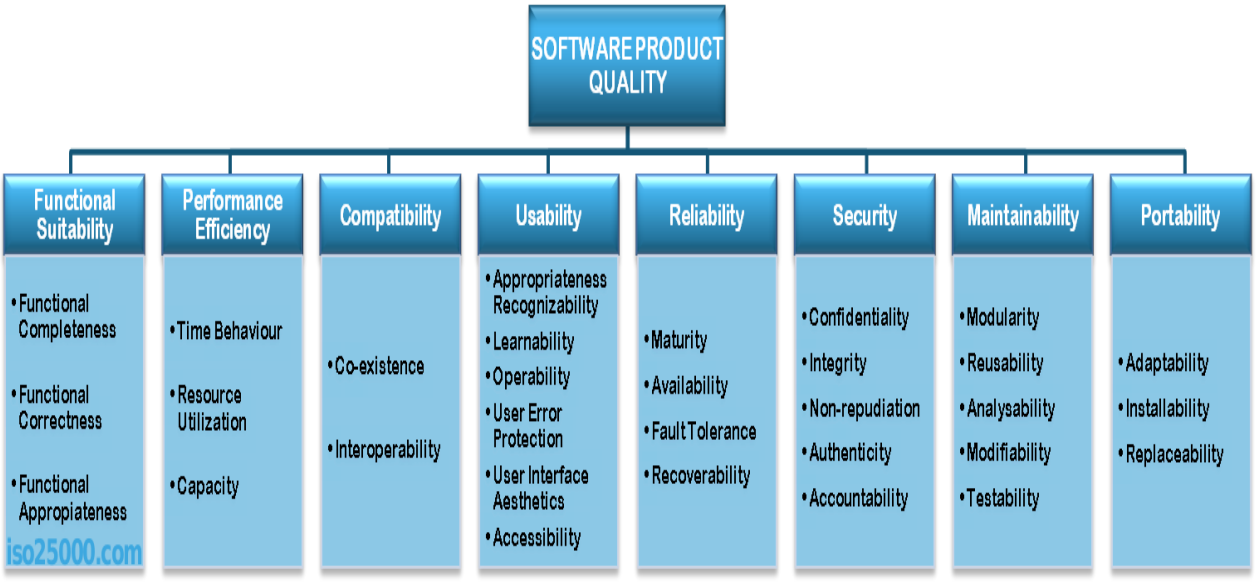
1.8 ISO 25010

A ISO/IEC 25010 é uma norma ISO para qualidade de produto de *software*, ela define um conjunto de parâmetros com o objetivo de padronizar a avaliação da qualidade de *software*. A norma 25010 se enquadra no modelo de qualidade das normas da família 9000.

A norma ISO/IEC 25010 (ISO/IEC, 2011) apresenta o modelo de qualidade para *software*, com o nome *Product Quality Model*, atualizando o modelo definido na norma ISO/IEC 9126. Esse modelo auxilia a avaliação por vários atores de um projeto de *software* como desenvolvedores, gerentes, compradores e integradores de componentes e o usuário.

A norma 25010 tem como foco a qualidade do *software*, propondo atributos de qualidade, distribuída em seis características principais, como funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade, sendo cada uma delas divididas em subcaracterísticas, como podem ser observadas na figura 7.

Figura 7 - Atributos de Qualidade



Fonte: ISO 25000 (2017)

Visto que um software esta sujeito a sofre mudanças, por uma série de fatores, como correção de erros, adequação a novos requisitos ou adequação a um novo ambiente a manutenibilidade de software tem um papel de suma importância na área da tecnologia da informação, sendo assim a manutenibilidade é a principal característica da ISO 25010 a ser considerada neste trabalho.

1.9 Manutenibilidade

A manutenibilidade corresponde ao grau de eficácia e eficiência com que um produto ou sistema pode ser modificado para melhorá-lo, corrigi-lo ou adaptá-lo às mudanças no ambiente e nos requisitos. De acordo com o portal ISO 25000 (2017), essa característica é composta das seguintes subcaracterísticas:

- Modularidade – Atributo de *software* em que o *software* é composto de componentes discretos, de modo que alterações em um componente tenham um impacto mínimo em outros componentes.

- Reutilização – Atributo de *software* em que um bem pode ser usado em mais de um sistema, ou na construção de outros sistemas.

- Analisabilidade – Atributo de *software* que identifica a facilidade em se diagnosticar problemas e identificar as causas das falhas.

- Modificabilidade – Atributo de *software* que caracteriza a facilidade com que o *software* pode ser modificado.

- Testabilidade – Atributo de *software* que representa a capacidade de se testar o *software* modificado, tanto quanto as novas funcionalidades quanto a funcionalidades não modificadas.

Essa seção apresentou as bases para o entendimento das arquiteturas de serviços web SOAP e REST, essenciais para a sequência do presente estudo. Na próxima seção serão apresentados os passos que foram seguidos para a elaboração desta análise de manutenibilidade em *web services*.

2 METODOLOGIA

O presente trabalho tem por objetivo comparar as arquiteturas de serviços web SOAP e REST, no intuito de avaliar qual arquitetura atende melhor à ISO 25010 no que se refere à manutenibilidade, para isto a princípio foram estudados todos os conceitos que possam fornecer o conhecimento necessário sobre arquitetura de *software*, a fim de saber a estrutura dos componentes de dentro de um *software*, que serão os objetos de estudo desta área.

Após a abordagem dos conhecimentos necessários referentes à arquitetura de *software*, foram estudados de forma detalhada os conceitos de *web service* com enfoque nas arquiteturas SOAP e REST, a fim de que estes possam ser utilizados para estruturação do objeto de estudo. Posteriormente, realizou-se um estudo referente às normas da ISO 25010 com enfoque nas diretrizes sobre manutenibilidade.

Tendo em vista todos os conceitos necessários, foi realizada uma pesquisa via internet para profissionais da área de tecnologia da informação sobre a facilidade de manutenção em *software* que utilizam a arquitetura SOAP e REST, através de um questionário que será detalhado nas subseções seguintes. Sendo que o questionário não aborda questões específicas para a manutenção da arquitetura no cliente ou no servidor, mas sim da arquitetura como um todo.

2.1 Público alvo do questionário

O questionário foi direcionado para profissionais da área de Tecnologia da Informação (TI) que possuem certo conhecimento de *web services* com enfoque nas arquiteturas SOAP e REST, com o intuito de identificar o atual perfil destes e entender o ponto de vista dos mesmos em relação ao tema em questão.

Os entrevistados foram incentivados a participar do questionário que foi divulgado por *e-mails* aos profissionais de TI, alunos e ex-alunos do curso de Ciência da Computação das Faculdades Integradas de Caratinga, contendo uma

mensagem explicando o objetivo da pesquisa e um *link* que possibilitou o acesso ao questionário. A pesquisa também foi divulgada em grupos de discussão e nas redes sociais.

2.2 Elaboração do questionário

As questões que compõem o questionário sobre o comparativo da manutenibilidade em *web services* foram elaboradas com base no conhecimento obtido através das leituras realizadas.

Foram feitas na pesquisa quarenta e três perguntas que foram organizadas em quatro grupos. A primeira página contém uma mensagem explicando o objetivo da pesquisa, informando que os dados são confidenciais e agradecendo pela colaboração. As questões foram numeradas e a maioria delas são obrigatórias, assim o entrevistado só conseguirá enviar o formulário após responder todas as perguntas. Para proporcionar um fluxo de leitura agradável, lógico e menos cansativo, o formulário de pesquisa foi dividido em apenas duas páginas, de forma que as páginas estavam condicionadas às respostas das questões da página anterior, ou seja, os entrevistados só respondem as questões conforme seu perfil.

2.3 O Questionário

As questões do questionário (Anexo A) foram organizadas em quatro grupos, cada um deles buscando coletar dados sobre diferentes aspectos.

O primeiro grupo engloba o perfil do respondente, com questões como *e-mail*, cidade, formação acadêmica e cargo que ocupa profissionalmente, para saber o meio em que ele trabalha.

O segundo grupo é direcionado aos respondentes que possuem algum conhecimento tanto da arquitetura SOAP como da arquitetura REST, são dezessete questões onde os entrevistados são indagados sobre características destas arquiteturas e comparações entre as mesmas.

O terceiro e o quarto grupo de questões foram direcionados aos respondentes que possuem, respectivamente, conhecimento apenas sobre SOAP ou apenas sobre REST. Cada grupo é composto por dez questões direcionadas aos entrevistados que conhecem ou trabalham com uma das arquiteturas. Assim, é possível conhecer como o respondente classifica a arquitetura e a frequência com que utiliza a mesma.

2.4 Coleta de dados

O questionário foi elaborado mediante um formulário criado por meio da ferramenta *Google Docs*. Essa ferramenta possibilita a criação, compartilhamento e realização da pesquisa *online*, sendo possível acessar o questionário em qualquer local e em qualquer computador.

A coleta dos dados da pesquisa durou quarenta e um dias. O questionário foi disponibilizado no dia 13 de setembro 2017 às 00h00min, ficando disponível até o dia 23 de outubro de 2017 às 23h59m, neste período foram coletadas 60 respostas.

2.5 Tratamento de dados

As respostas coletadas pelo formulário foram armazenadas automaticamente em uma planilha criada pelo *Google Docs,* o que auxiliou a tabulação dos dados. Através da planilha é possível visualizar cada reposta de cada participante separadamente. Para a manipulação dos dados, a planilha foi exportada para edição no *Microsoft Excel*, onde os dados foram organizados de acordo com o perfil do respondente, possibilitando a análise das respostas e a exibição dos resultados através de gráficos com a porcentagem das respostas, para proporcionar uma melhor compreensão dos resultados que serão apresentados na seção seguinte.

3 RESULTADOS

Nesta seção serão apresentados, analisados e discutidos os dados obtidos por meio das respostas dos 60 entrevistados que responderam ao questionário. Inicialmente, buscou-se caracterizar o perfil dos respondentes da pesquisa, avaliando informações como cidade e formação acadêmica; além de informações referentes a área de trabalho bem como tempo de atuação no mercado; e em seguida, buscou-se saber os conhecimentos do respondente referentes as arquiteturas SOAP e REST.

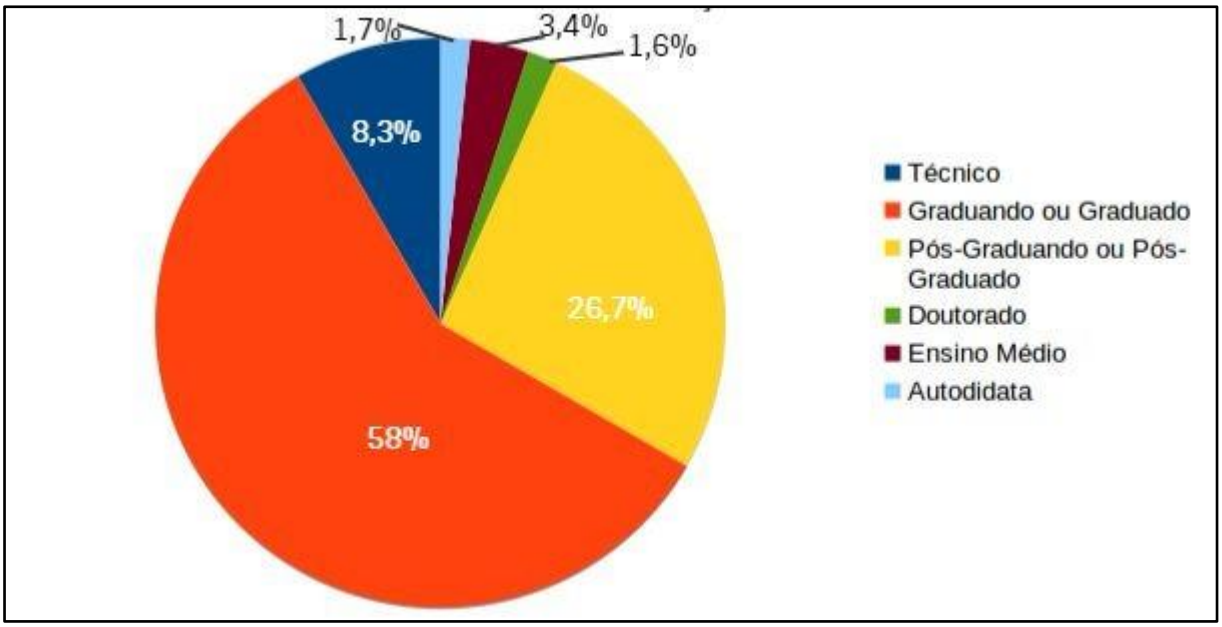
As respostas exibidas a seguir serão organizadas pelos grupos do questionário, cada um em uma seção, sendo: Perfil do Entrevistado, Conhecimento ou Experiência com ambas as arquiteturas (se enquadrando aqui as pessoas que responderam sobre ambas as arquiteturas estudadas) e Conhecimento ou Experiência com uma das arquiteturas (aqui já se encontram as pessoas que responderam somente sobre uma das arquiteturas).

3.1 Perfil do entrevistado

A pesquisa contou com as respostas de 60 pessoas, de 12 estados brasileiros, um país norte americano (Canadá), um país europeu (Portugal) e um país asiático (Vietnã), sendo que aproximadamente 33,33% afirmaram residir em Minas Gerais na cidade de Caratinga.

Quando perguntados sobre a formação acadêmica, 35 pessoas (aproximadamente 58,3%) declararam ter formação de graduação em andamento ou finalizada, 16 pessoas (aproximadamente 26,7%) possuem pós-graduação em andamento ou finalizada, 5 pessoas (aproximadamente 8,3%) são técnicos, 1 pessoa (aproximadamente 1,6%) possui doutorado e 3 pessoas (aproximadamente 5,1%) declararam ter outros tipos de formação. Com base nesses resultados verifica-se que mais de 80% dos respondentes possuem uma formação igual ou superior a uma graduação, o que pode ser observado no gráfico 1.

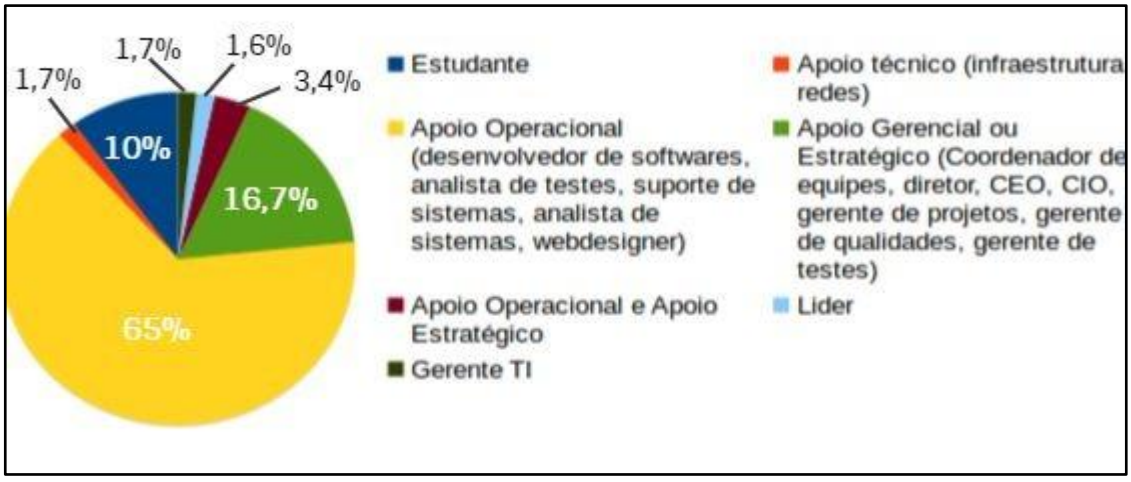
Gráfico 1 – Nível de Formação dos Respondentes



Fonte: Próprio autor

Em relação às funções desempenhadas profissionalmente, 39 pessoas (65%) disseram desempenhar funções ligadas ao apoio operacional, 10 pessoas (aproximadamente 16,7%) responderam que desempenham funções ligadas ao apoio gerencial ou estratégico, 6 pessoas (10%) são estudantes, 1 pessoa (aproximadamente 1,6%) desempenha funções ligadas ao apoio técnico e 4 pessoas (aproximadamente 6,7%) disseram ter outras funções. Com base nesses resultados verifica-se que a maior parte dos respondentes possui experiência na área de TI, sendo a maioria no desenvolvimento de *software*, como demonstrado no gráfico 2.

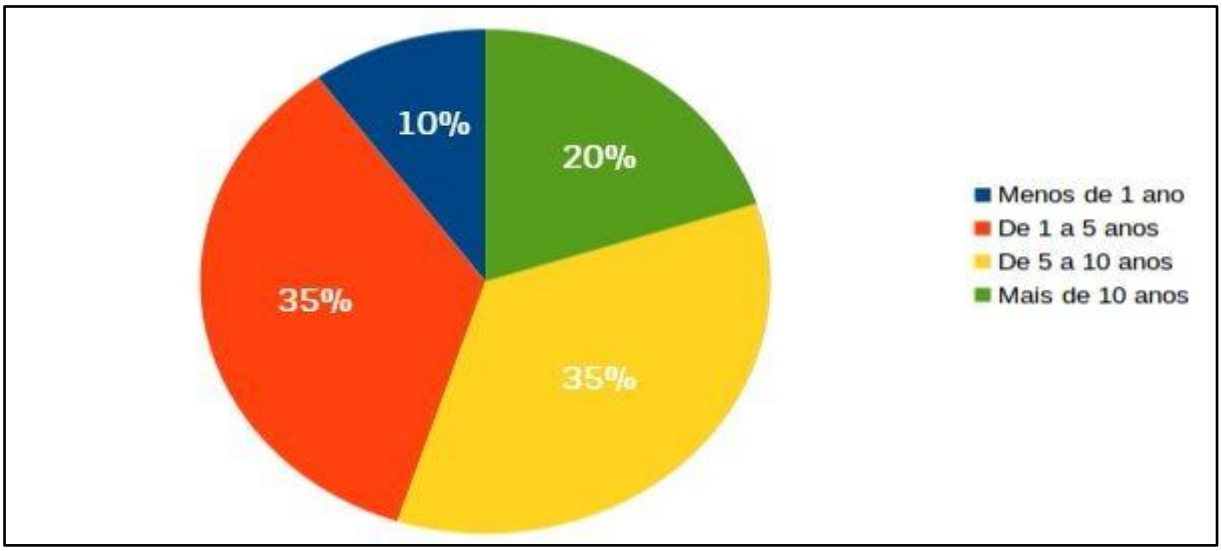
Gráfico 2 – Função desempenhada pelos respondentes



Fonte: Próprio autor

Com relação ao tempo de experiência na área de tecnologia da informação, 12 pessoas (20%) disseram que já trabalham na área de TI a mais de 10 anos, 21 pessoas (35%) disseram que já trabalham na área de TI de 5 a 10 anos, 21 pessoas (35%) disseram trabalhar na área de TI de 1 a 5 anos e 6 pessoas (10%) trabalham na área de TI a menos de 1 ano. Com base nesses resultados verifica-se que a maioria dos respondentes possuem 5 anos ou mais de experiência na área de TI, o qual pode ser observado no gráfico 3.

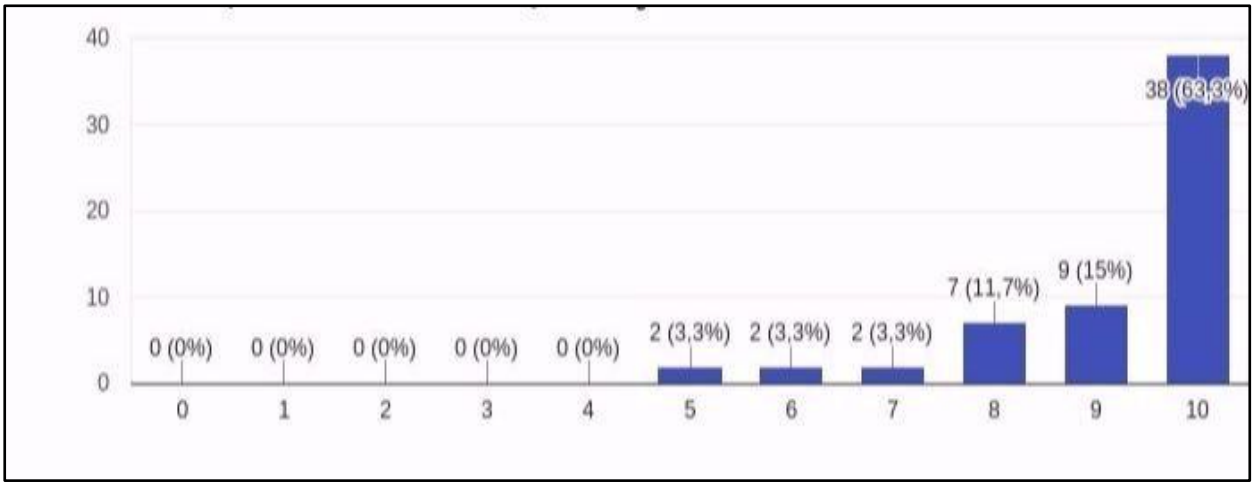
Gráfico 3 – Tempo de experiência na área de TI



Fonte: Próprio autor

Visto que a manutenibilidade é um dos objetos de estudo deste trabalho, quando os respondentes foram perguntados sobre a importância de se aplicar a manutenibilidade em projetos de *software*, considerando uma escala de 0 a 10 onde 0 pode ser considerado como pouco importante e 10 pode ser considerado como muito importante, 38 pessoas (aproximadamente 63,3%) responderam 10, considerando assim a aplicação da manutenibilidade em projetos de *software* de suma importância, 9 pessoas (15%) responderam 9, 7 pessoas (aproximadamente 11,7%) responderam 8, 2 pessoas (aproximadamente 3,3%) responderam 7, 2 pessoas (aproximadamente 3,3%) responderam 6 e 2 pessoas (aproximadamente 3,3%) responderam 5. Com base nesses resultados verifica-se que aproximadamente 90% dos respondentes consideram a aplicação da manutenibilidade em projetos de *software* como muito importante, o que pode ser observado no gráfico 4.

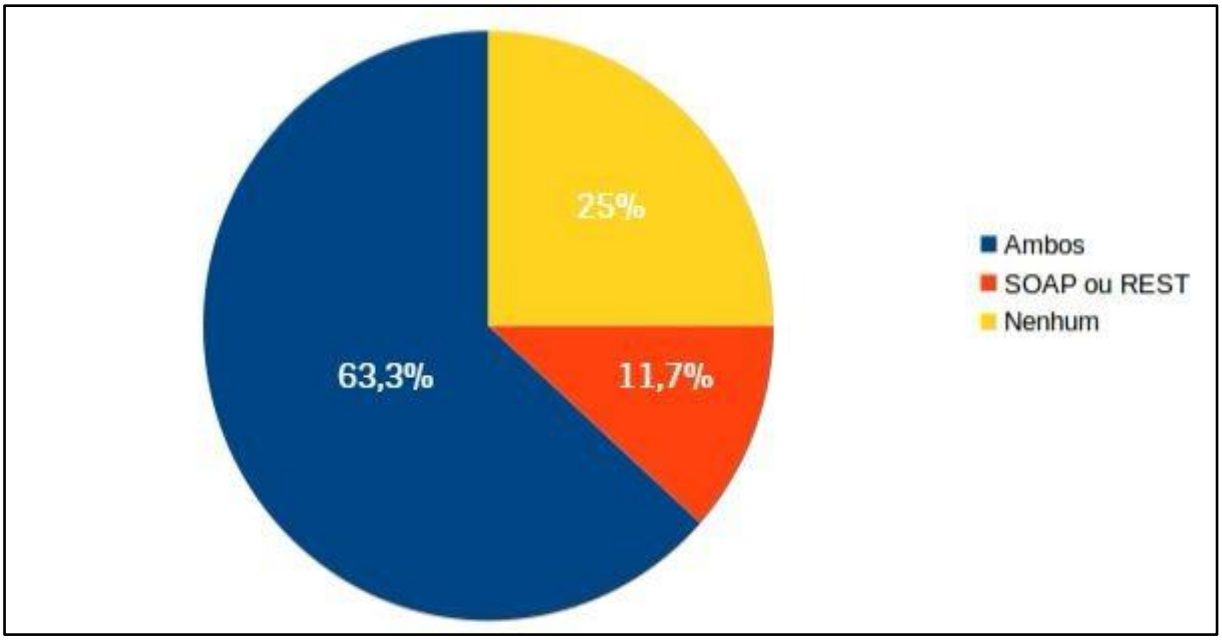
Gráfico 4 – Importância da Aplicação da Manutenibilidade



Fonte: Próprio autor

Com relação à experiência ou conhecimento nas arquiteturas SOAP e REST, 38 pessoas (aproximadamente 63,3%) disseram conhecer ou já ter trabalhado com ambas as arquiteturas, 7 pessoas (aproximadamente 11,7%) disseram conhecer ou já ter trabalhado somente com a arquitetura REST ou somente com a arquitetura SOAP e 15 pessoas (25%) disseram não conhecer nenhuma das arquiteturas. Com base nesses resultados verifica-se que a maioria dos respondentes conhecem ambas as arquiteturas, isso pode ser observado no gráfico 5.

Gráfico 5 – Experiência ou Conhecimento com SOAP e REST



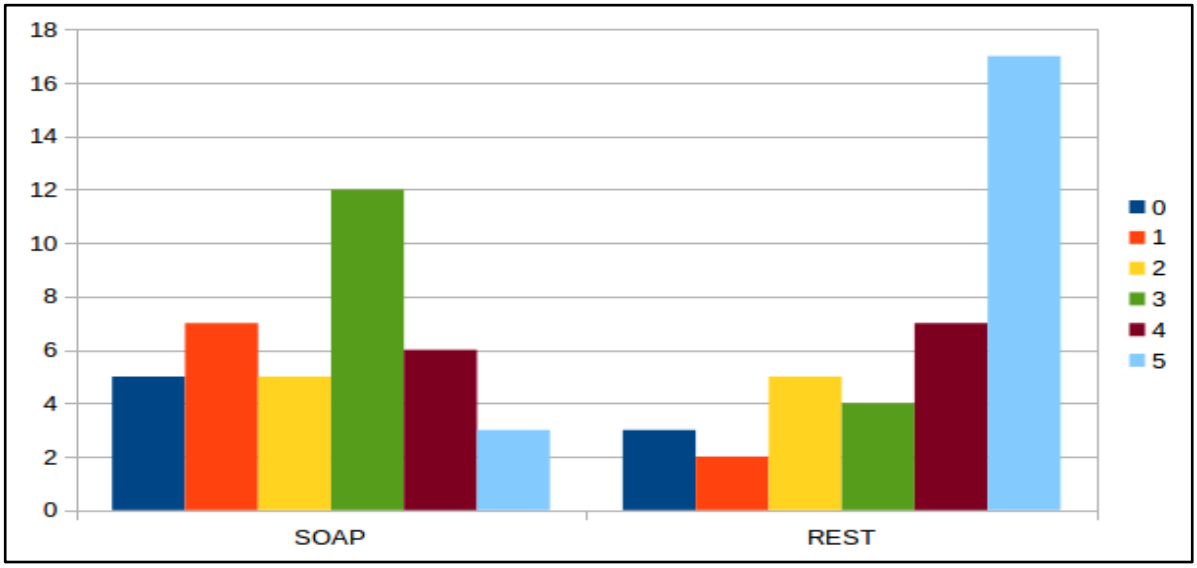
Fonte: Próprio autor

3.2 Conhecimento ou experiência com ambas as arquiteturas

Com relação aos respondentes que possuem conhecimento ou já têm alguma experiência com SOAP e REST, que totalizaram 38 pessoas, foram feitas algumas perguntas que serviam de comparação entre estas duas arquiteturas.

Primeiramente foi perguntado aos respondentes com que frequência eles utilizam-se destas arquiteturas. Em uma escala de 0 a 5 onde 0 pode ser considerado como pouco frequente e 5 pode ser considerado como muito frequente, com relação ao SOAP, 5 pessoas responderam 0; 7 pessoas responderam 1; 5 pessoas responderam 2; 12 pessoas responderam 3; 6 pessoas responderam 4 e 3 pessoas responderam 5. Já com relação ao REST, 3 pessoas responderam 0; 2 pessoas responderam 1; 5 pessoas responderam 2; 4 pessoas responderam 3; 7 pessoas responderam 4 e 17 pessoas responderam 5. Com base nesses resultados verifica-se que a maioria utiliza ambas as arquiteturas, o que pode ser observado no gráfico 6.

Gráfico 6 – Frequência de utilização das arquiteturas SOAP e REST

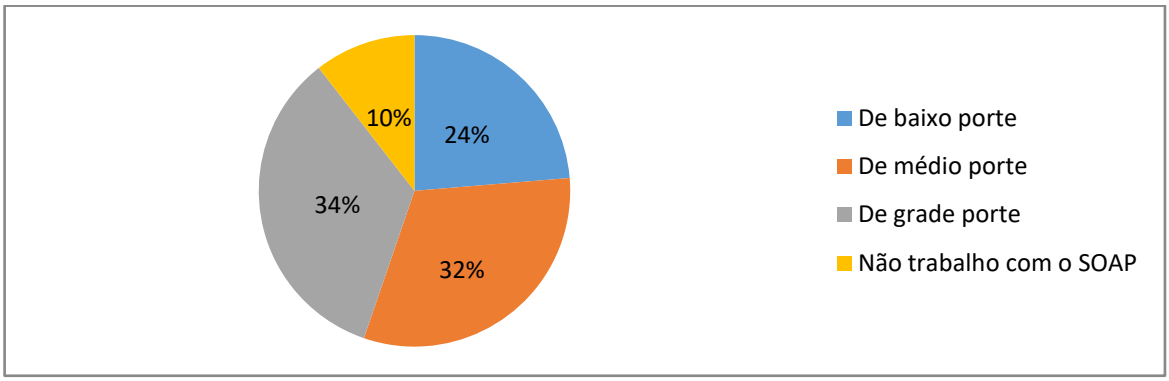


Fonte: Próprio autor

Quando perguntados sobre o tipo de projetos em que os respondentes utilizam o SOAP 13 pessoas (aproximadamente 34,2%) responderam que utilizam em projetos de grade porte, 12 pessoas (aproximadamente 31,6%) responderam que utilizam em projetos de médio porte, 9 pessoas (aproximadamente 23,7%)

responderam que utilizam em projetos de baixo porte e 4 pessoas (aproximadamente 10,5%) responderam que não utilizam o SOAP em seus projetos. Com base nesses resultados verifica-se que a maioria dos respondentes utilizam a arquitetura SOAP em projetos de médio ou grade porte, o que está de acordo com o gráfico 7.

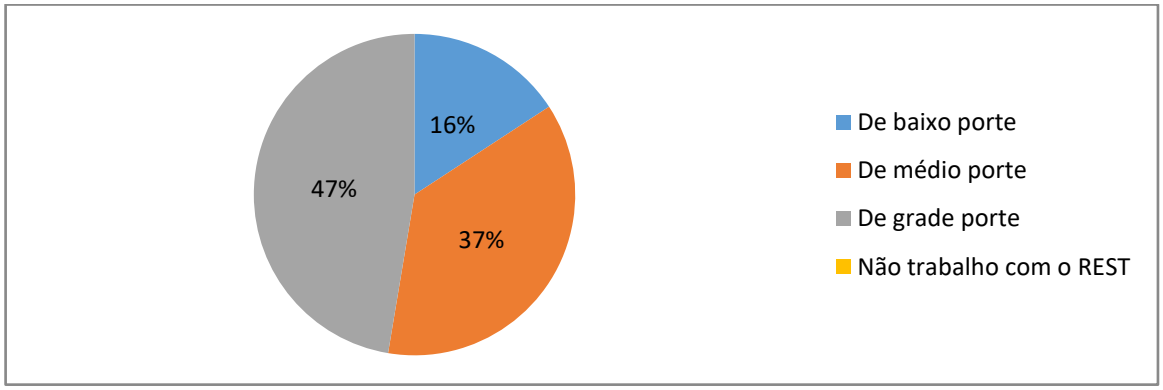
Gráfico 7 – Uso da arquitetura SOAP



Fonte: Próprio autor

Quando perguntados sobre o tipo de projetos em que os respondentes utilizam o REST 18 pessoas (aproximadamente 47,4%) responderam que utilizam em projetos de grande porte, 14 pessoas (aproximadamente 36,8%) responderam que utilizam em projetos de médio porte e 6 pessoas (aproximadamente 15,8%) responderam que utilizam em projetos de baixo porte. Com base nesses resultados verifica-se que todos os respondentes disseram utilizar o REST em algum de seus projetos, sendo estes em sua maioria de médio ou grade porte, o que pode ser observado no gráfico 8.

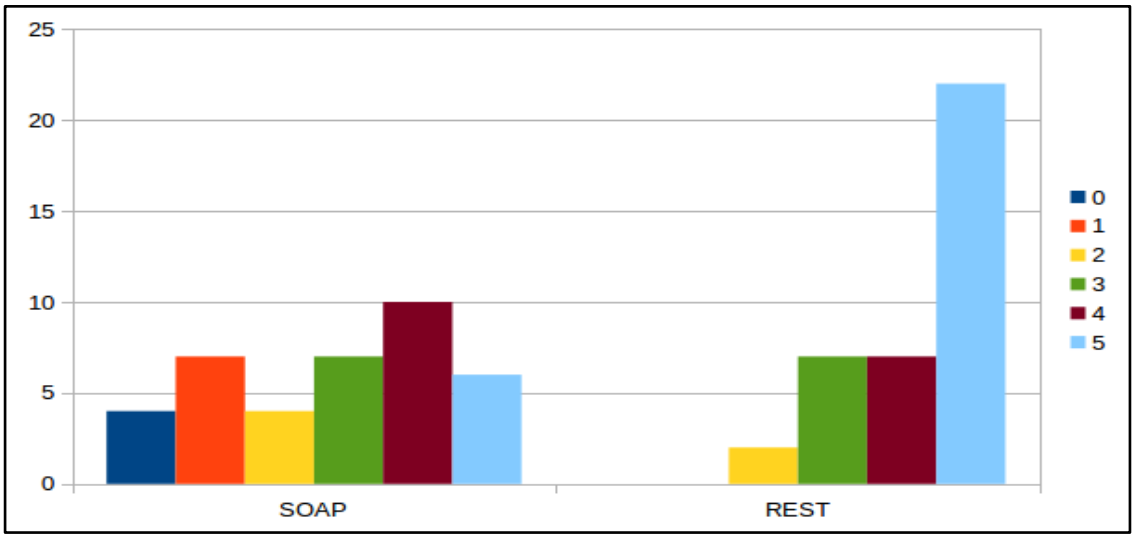
Gráfico 8 – Uso da arquitetura REST



Fonte: Próprio autor

Foi perguntado também aos respondentes qual o grau de aceitação destas arquiteturas. Em uma escala de 0 a 5 onde 0 pode ser considerado como baixa aceitação e 5 pode ser considerado como alta aceitação, com relação ao SOAP, 4 pessoas responderam 0; 7 pessoas responderam 1; 4 pessoas responderam 2; 7 pessoas responderam 3; 10 pessoas responderam 4 e 6 pessoas responderam 5. Já com relação ao REST, nenhuma pessoa respondeu 0; nenhuma pessoa respondeu 1; 2 pessoas responderam 2; 7 pessoas responderam 3; 7 pessoas responderam 4 e 22 pessoas responderam 5. Com base nesses resultados verifica-se que ambas as arquiteturas são bem aceitas sendo que o REST leva uma ligeira vantagem, o que pode ser observado no gráfico 9.

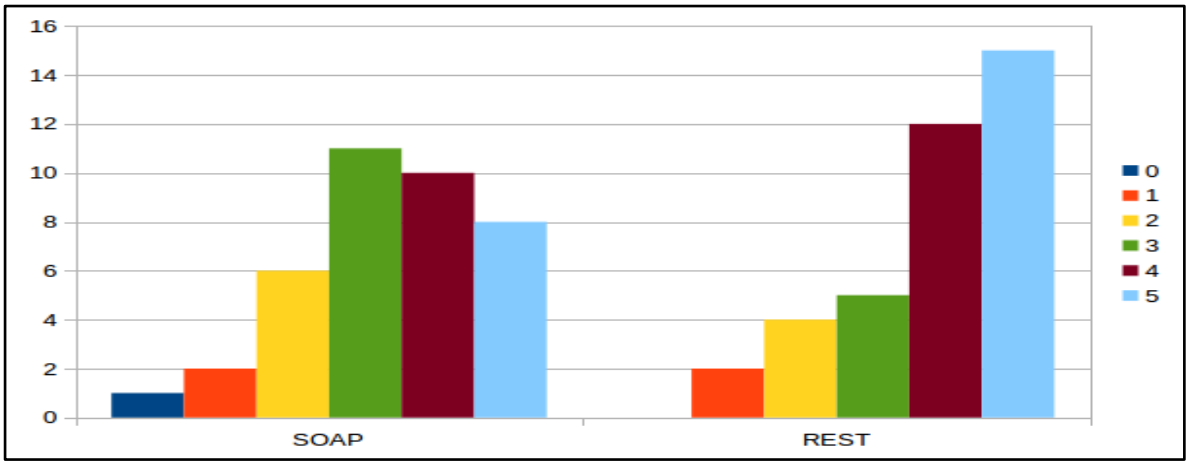
Gráfico 9 – Grau de aceitação das arquiteturas



Fonte: Próprio autor

Foi perguntado também aos respondentes qual o grau de aprendizado ao se utilizar estas arquiteturas. Em uma escala de 0 a 5 onde 0 pode ser considerado como aprendizado baixo e 5 pode ser considerado como aprendizado alto, com relação ao SOAP, 1 pessoa respondeu 0; 2 pessoas responderam 1; 6 pessoas responderam 2; 11 pessoas responderam 3; 10 pessoas responderam 4 e 8 pessoas responderam 5. Já com relação ao REST, nenhuma pessoa respondeu 0; 2 pessoas responderam 1; 4 pessoas responderam 2; 5 pessoas responderam 3; 12 pessoas responderam 4 e 15 pessoas responderam 5. Com base nesses resultados verifica-se que ambas as arquiteturas apresentam um bom nível de aprendizado ao utilizá-las, o que pode ser observado no gráfico 10.

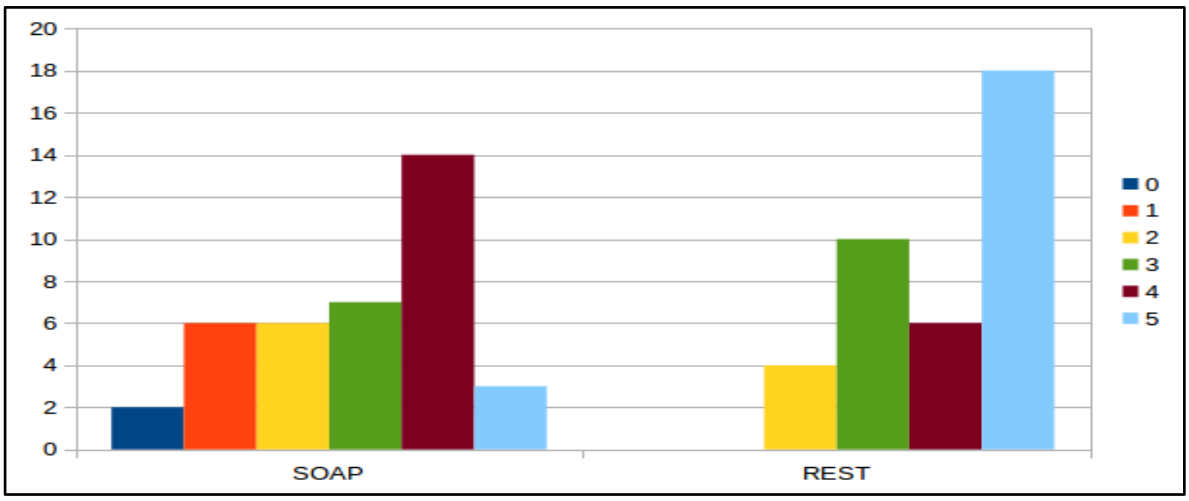
Gráfico 10 – Grau de aprendizado das arquiteturas



Fonte: Próprio autor

Foi perguntado também aos respondentes qual o grau de facilidade de manutenção apresentado pelas arquiteturas. Em uma escala de 0 a 5 onde 0 pode ser considerado como pouca facilidade e 5 pode ser considerado como alta facilidade, com relação ao SOAP, 2 pessoas responderam 0; 6 pessoas responderam 1; 6 pessoas responderam 2; 7 pessoas responderam 3; 14 pessoas responderam 4 e 3 pessoas responderam 5. Já com relação ao REST, nenhuma pessoa respondeu 0; nenhuma pessoa respondeu 1; 4 pessoas responderam 2; 10 pessoas responderam 3; 6 pessoas responderam 4 e 18 pessoas responderam 5. Com base nesses resultados verifica-se que ambas as arquiteturas apresentam facilidades para se aplicar manutenção, o que pode ser observado no gráfico 11.

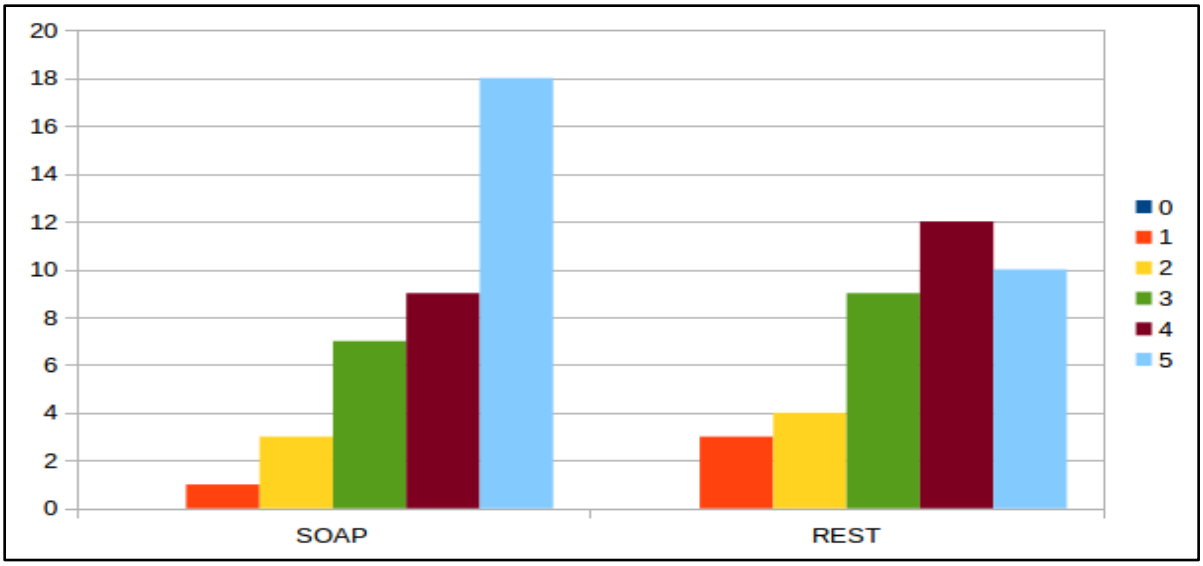
Gráfico 11 – Grau de facilidade de manutenção das arquiteturas



Fonte: Próprio autor

Foi perguntado também aos respondentes qual o grau de segurança apresentado pelas arquiteturas. Em uma escala de 0 a 5 onde 0 pode ser considerado como baixa segurança e 5 pode ser considerado como alta segurança, com relação ao SOAP, nenhuma pessoa respondeu 0; 1 pessoa respondeu 1; 3 pessoas responderam 2; 7 pessoas responderam 3; 9 pessoas responderam 4 e 18 pessoas responderam 5. Já com relação ao REST, nenhuma pessoa respondeu 0; 3 pessoas responderam 1; 4 pessoas responderam 2; 9 pessoas responderam 3; 12 pessoas responderam 4 e 10 pessoas responderam 5. Com base nesses resultados verifica-se que ambas as arquiteturas apresentam uma boa segurança com o SOAP apresentando alguma vantagem neste quesito, o que pode ser visto no gráfico 12.

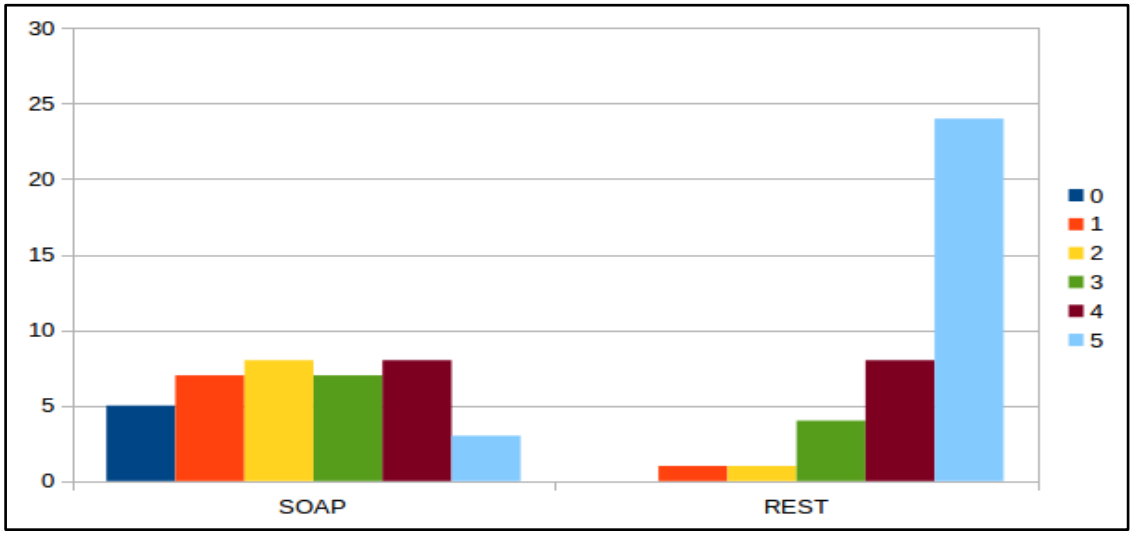
Gráfico 12 – Grau de segurança das arquiteturas



Fonte: Próprio autor

Foi perguntado também aos respondentes qual o grau de simplicidade apresentado pelas arquiteturas. Em uma escala de 0 a 5 onde 0 pode ser considerado como pouco simples e 5 pode ser considerado como muito simples, com relação ao SOAP, 5 pessoas responderam 0; 7 pessoas responderam 1; 8 pessoas responderam 2; 7 pessoas responderam 3; 8 pessoas responderam 4 e 3 pessoas responderam 5. Já com relação ao REST, nenhuma pessoa respondeu 0; 1 pessoa respondeu 1; 1 pessoa respondeu 2; 4 pessoas responderam 3; 8 pessoas responderam 4 e 24 pessoas responderam 5. Com base nesses resultados verifica-se que a arquitetura REST apresenta grande vantagem em relação à arquitetura SOAP neste quesito, o que pode ser observado no gráfico 13.

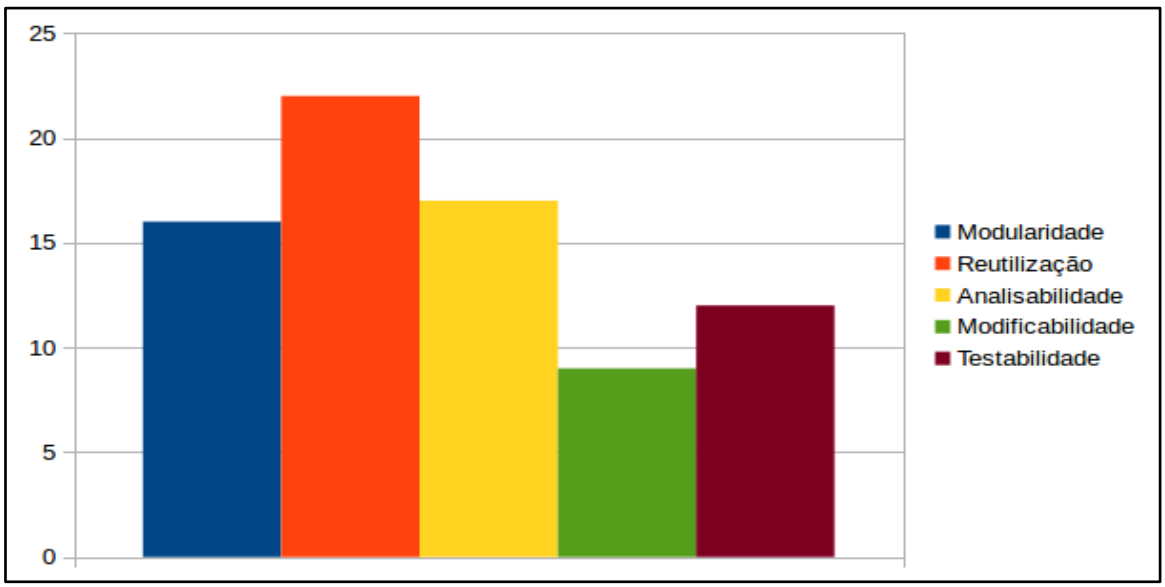
Gráfico 13 – Grau de simplicidade das arquiteturas



Fonte: Próprio autor

Quando perguntados sobre as principais características da arquitetura SOAP, 22 pessoas responderam que é a reutilização, 17 pessoas responderam que é a analisabilidade, 16 pessoas responderam que é a modularidade, 12 pessoas responderam que é a testabilidade e 9 pessoas responderam que é a modificabilidade. Com base nesses resultados verifica-se que a reutilização é a principal característica do SOAP para os respondentes, o que pode ser observado no gráfico 14.

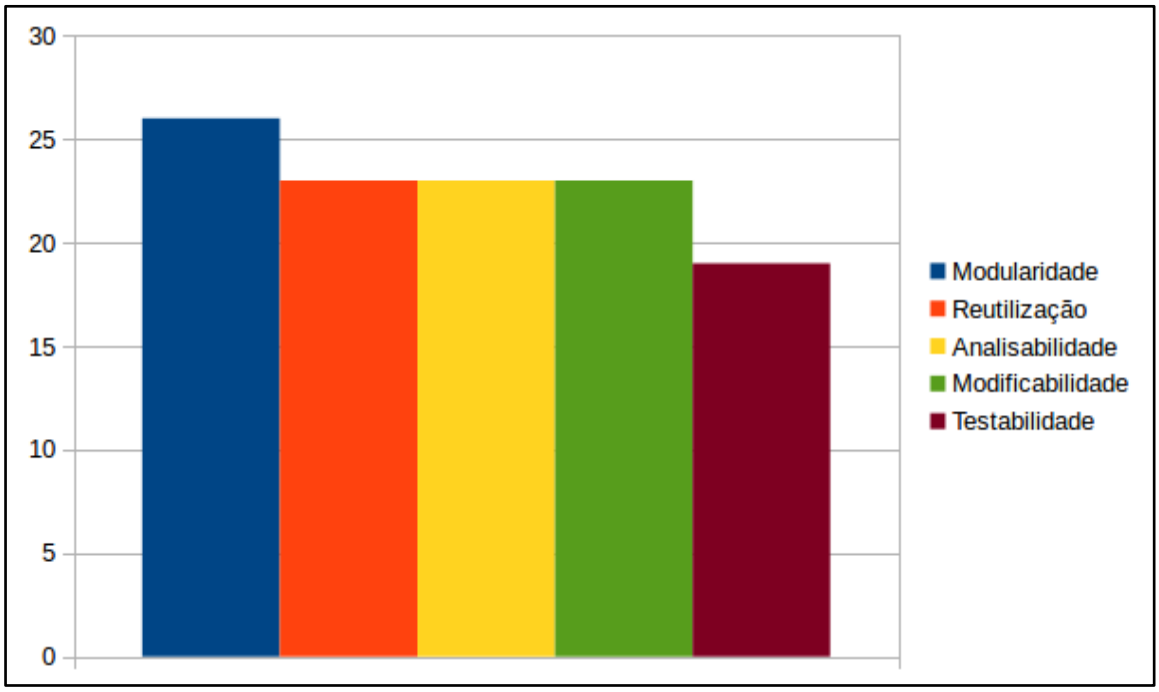
Gráfico 14 – Características da arquitetura SOAP



Fonte: Próprio autor

Quando perguntados sobre as principais características da arquitetura REST, 26 pessoas responderam que é a modularidade, 23 pessoas responderam que é a analisabilidade, 23 pessoas responderam que é a reutilização, 23 pessoas responderam que é a modificabilidade e 19 pessoas responderam que é a testabilidade. Com base nesses resultados verifica-se que a modularidade é a principal característica do REST para os respondentes, o que pode ser observado no gráfico 15.

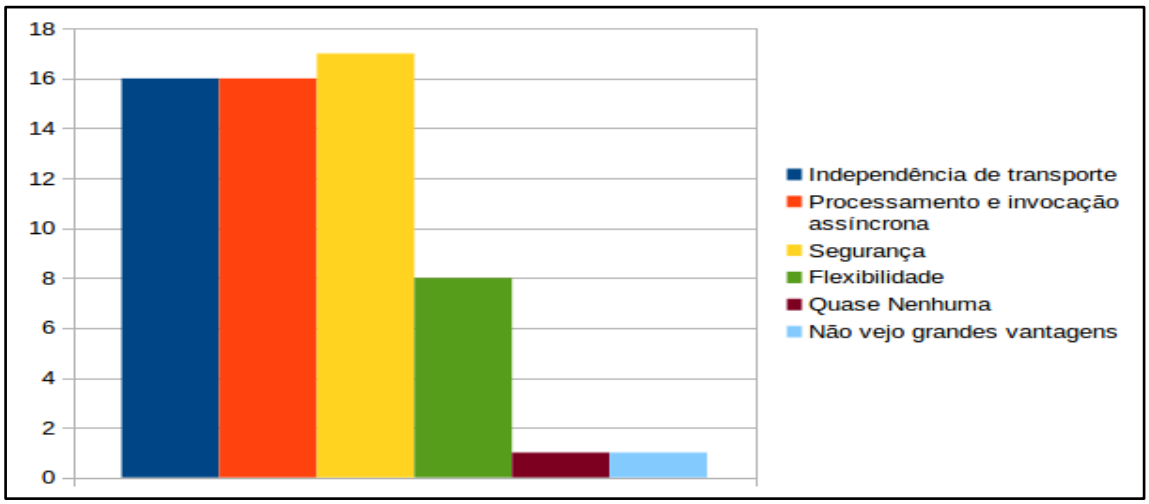
Gráfico 15 – Características da arquitetura REST



Fonte: Próprio autor

Quando perguntados sobre as principais vantagens da arquitetura SOAP, 17 pessoas responderam que é a segurança, 16 pessoas responderam que é a independência de transporte, 16 pessoas responderam que é o processamento e invocação assíncrona, 8 pessoas responderam que é a flexibilidade e 2 pessoas responderam não verem vantagens na utilização da arquitetura. Com base nesses resultados verifica-se que a segurança é a principal vantagem do SOAP para os respondentes, o que pode ser observado no gráfico 16.

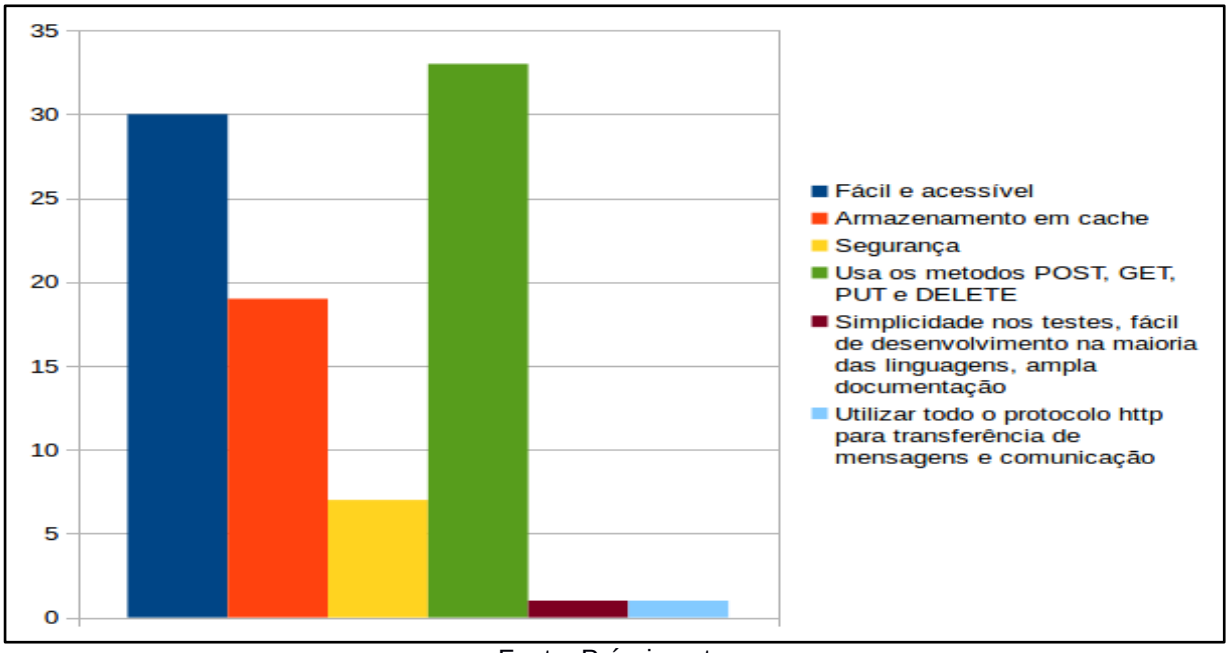
Gráfico 16 – Principais vantagens da arquitetura SOAP



Fonte: Próprio autor

Quando perguntados sobre as principais vantagens da arquitetura REST, 33 pessoas responderam que é o fato da arquitetura permitir o uso de métodos como *post*, *get*, *put* e *delete*, 30 pessoas responderam que é a facilidade e acessibilidade, 19 pessoas responderam que é o armazenamento de *cache* e 2 pessoas responderam outras vantagens. Com base nesses resultados verifica-se que a capacidade de usar métodos como *post*, *get*, *put* e *delete* é a principal vantagem do REST para os respondentes, o que pode ser observado no gráfico 17.

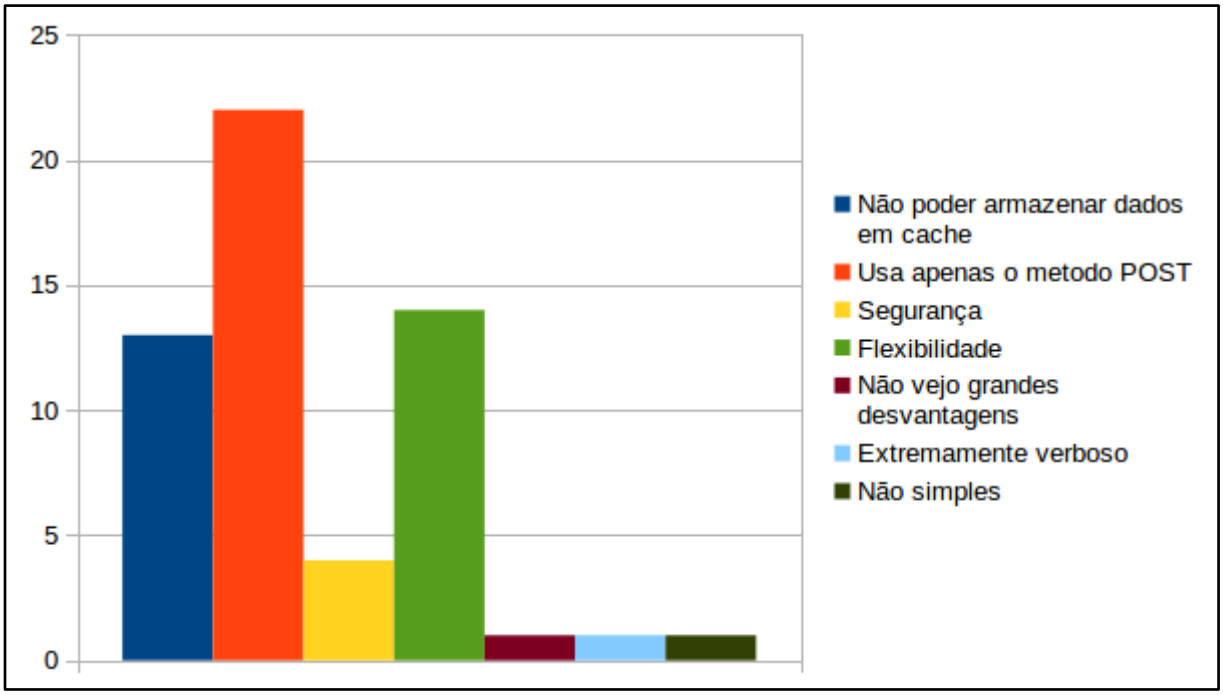
Gráfico 17 – Principais vantagens da arquitetura REST



Fonte: Próprio autor

Quando perguntados sobre as principais desvantagens da arquitetura SOAP, 22 pessoas responderam que é o fato da arquitetura permitir o uso apenas do método *post*, 14 pessoas responderam que é a flexibilidade, 13 pessoas responderam que é o fato de não poder armazenar dados em *cache*, 4 pessoas responderam que é a segurança, 1 pessoa respondeu não ver desvantagem na utilização da arquitetura e 2 pessoas responderam outras desvantagens. Com base nesses resultados verifica-se que o fato da arquitetura SOAP permitir apenas o uso do método *post* é considerado a principal desvantagem desta arquitetura para os respondentes, o que pode ser observado no gráfico 18.

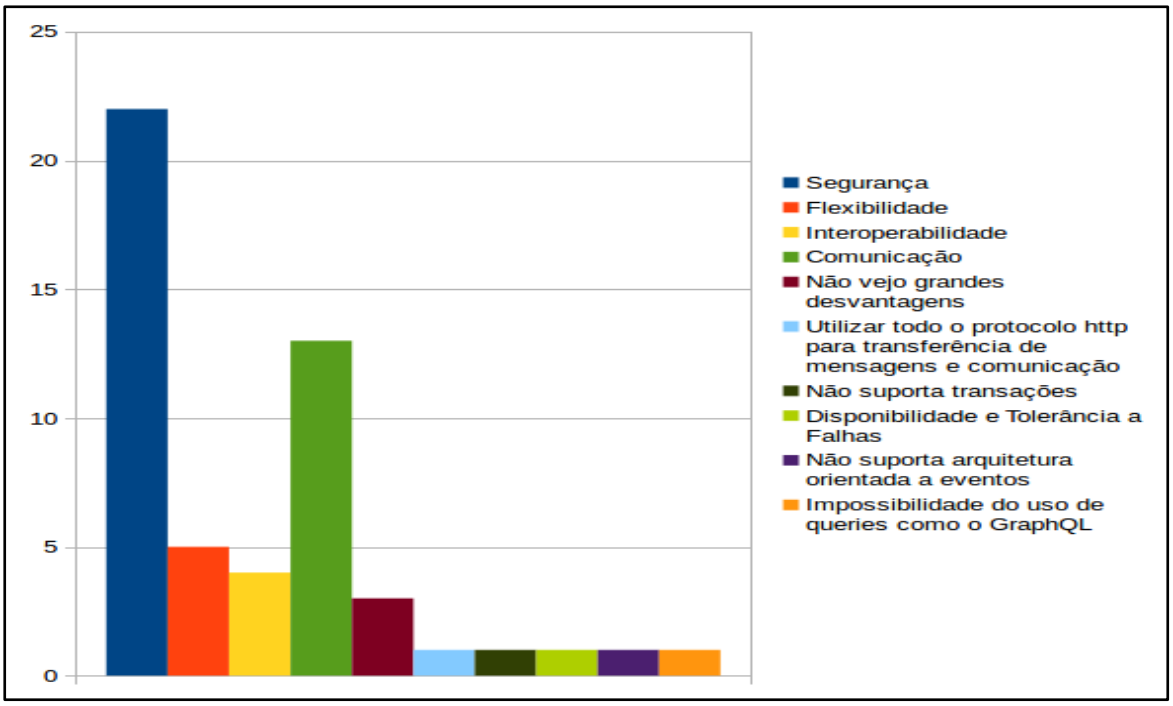
Gráfico 18 – Principais desvantagens da arquitetura SOAP



Fonte: Próprio autor

Quando perguntados sobre as principais desvantagens da arquitetura REST, 22 pessoas responderam que é a segurança, 13 pessoas responderam que é a comunicação, 5 pessoas responderam que é a flexibilidade, 4 pessoas responderam que é a interoperabilidade, 3 pessoas responderam não verem desvantagens na utilização da arquitetura e 5 pessoas responderam outras desvantagens. Com base nesses resultados verifica-se que a segurança é a principal desvantagem do REST para os respondentes, o que pode ser observado no gráfico 19.

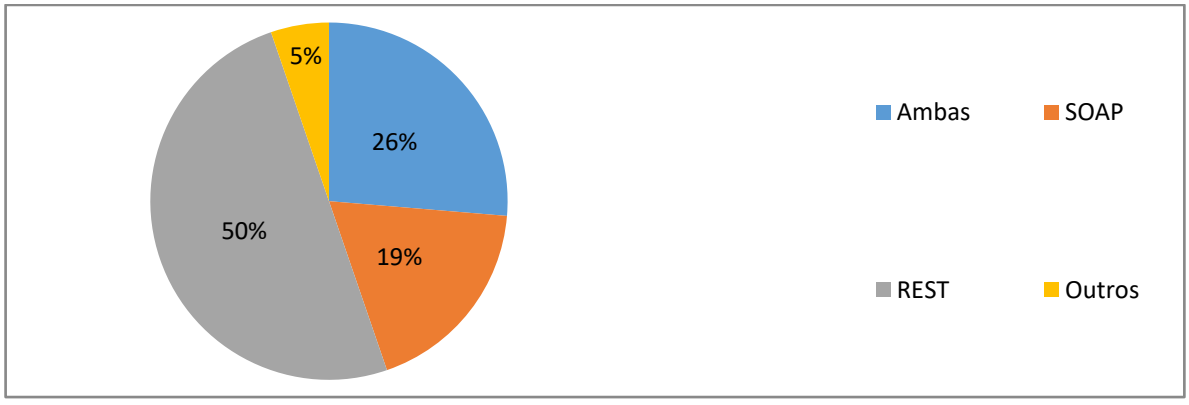
Gráfico 19 – Principais desvantagens da arquitetura REST



Fonte: Próprio autor

Quando perguntados sobre qual das arquiteturas SOAP ou REST é melhor para ser aplicada em projetos de software, 19 pessoas (50%) disseram que é o REST, 7 pessoas (aproximadamente 18,4%) disseram que é o SOAP, 10 pessoas (aproximadamente 26,3%) disseram que ambas são boas para ser aplicada em projetos de *software* e 2 pessoas (aproximadamente 5,3%) disseram outras coisas. Com base nesses resultados verifica-se que o REST é considerado pelos respondentes como melhor para ser aplicado em projetos de *software*, o que pode ser observado no gráfico 20.

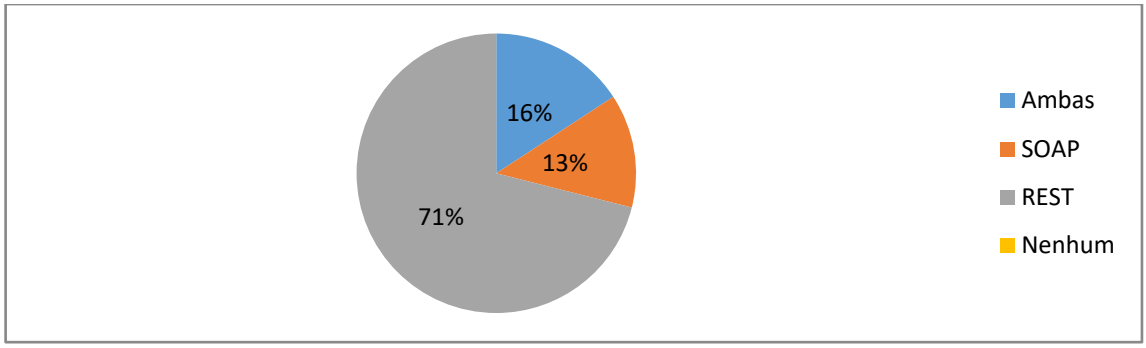
Gráfico 20 – Melhor arquitetura para ser aplicada em projetos



Fonte: Próprio autor

Quando perguntados sobre qual das arquiteturas SOAP ou REST é mais de fácil para ser aplicada em projetos de *software*, 27 pessoas (aproximadamente 71,1%) disseram que é o REST, 5 pessoas (aproximadamente 13,2%) disseram que é o SOAP e 6 pessoas (aproximadamente 15,7%) disseram que ambas são fáceis de ser aplicada em projetos de *software*. Com base nesses resultados verifica-se que o REST é considerado pelos respondentes como mais fácil de ser aplicado em projetos de *software*, o que pode ser visto no gráfico 21.

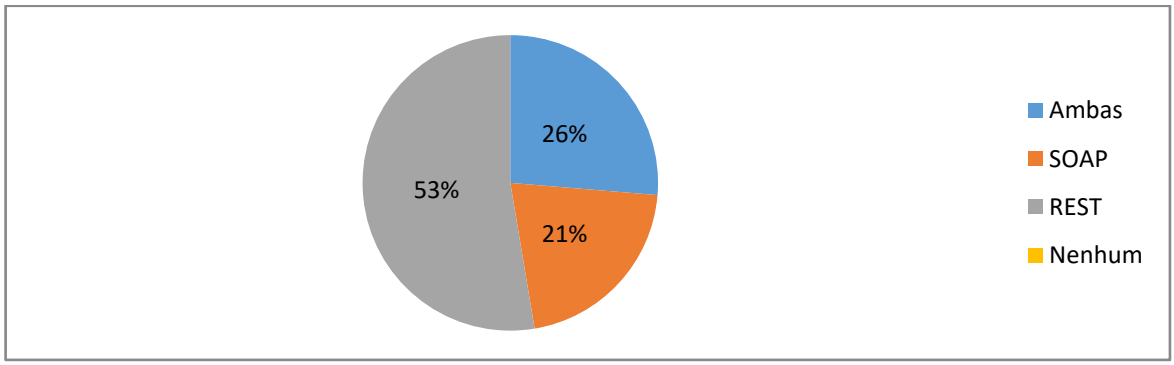
Gráfico 21 – Qual arquitetura é mais fácil para ser aplicada em projetos?



Fonte: Próprio autor

Quando perguntados sobre qual das arquiteturas SOAP ou REST apresenta um alto nível de manutenibilidade, 20 pessoas (aproximadamente 52,6%) disseram que é o REST, 8 pessoas (aproximadamente 21,1%) disseram que é o SOAP e 10 pessoas (aproximadamente 26,3%) disseram que ambas apresentam um alto nível de manutenibilidade. Com base nesses resultados verifica-se que o REST é considerado pelos respondentes como uma arquitetura que melhor apresenta um alto nível de manutenibilidade, o que pode ser visto no gráfico 22.

Gráfico 22 – Alto nível de manutenibilidade



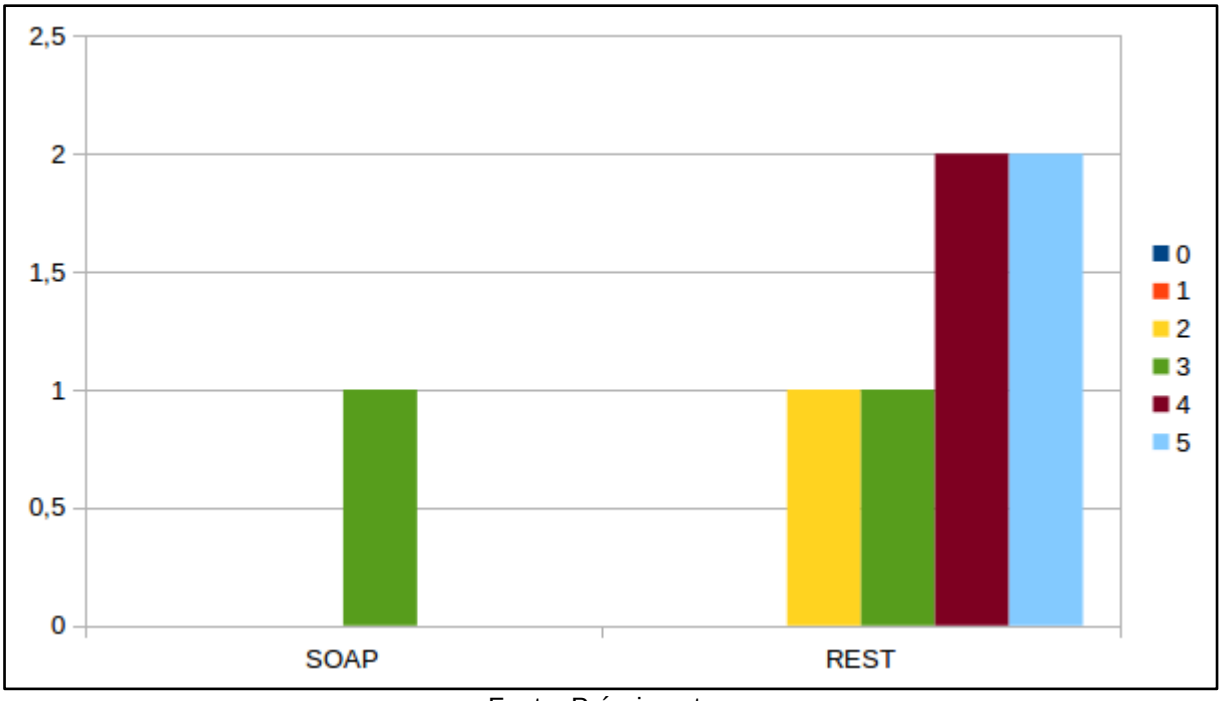
Fonte: Próprio autor

3.3 Conhecimento ou experiência com uma das arquiteturas

Com relação aos respondentes que possuem conhecimento ou já têm alguma experiência somente com o SOAP ou somente com o REST, que totalizaram 7 pessoas, foram feitas algumas perguntas que serviam de comparação entre estas duas arquiteturas.

Primeiramente foi perguntado aos respondentes com que frequência eles utilizam-se destas arquiteturas. Em uma escala de 0 a 5 onde 0 pode ser considerado como pouco frequente e 5 pode ser considerado como muito frequente, com relação ao SOAP, nenhuma pessoa respondeu 0; nenhuma pessoa respondeu 1; nenhuma pessoa respondeu 2; 1 pessoa respondeu 3; nenhuma pessoa respondeu 4 e nenhuma pessoa respondeu 5. Já com relação ao REST, nenhuma pessoa respondeu 0; nenhuma pessoa respondeu 1; 1 pessoa respondeu 2; 1 pessoa respondeu 3; 2 pessoas responderam 4 e 2 pessoas responderam 5. Com base nesses resultados verifica-se que a maioria utiliza ambas as arquiteturas, o que pode ser observado no gráfico 23.

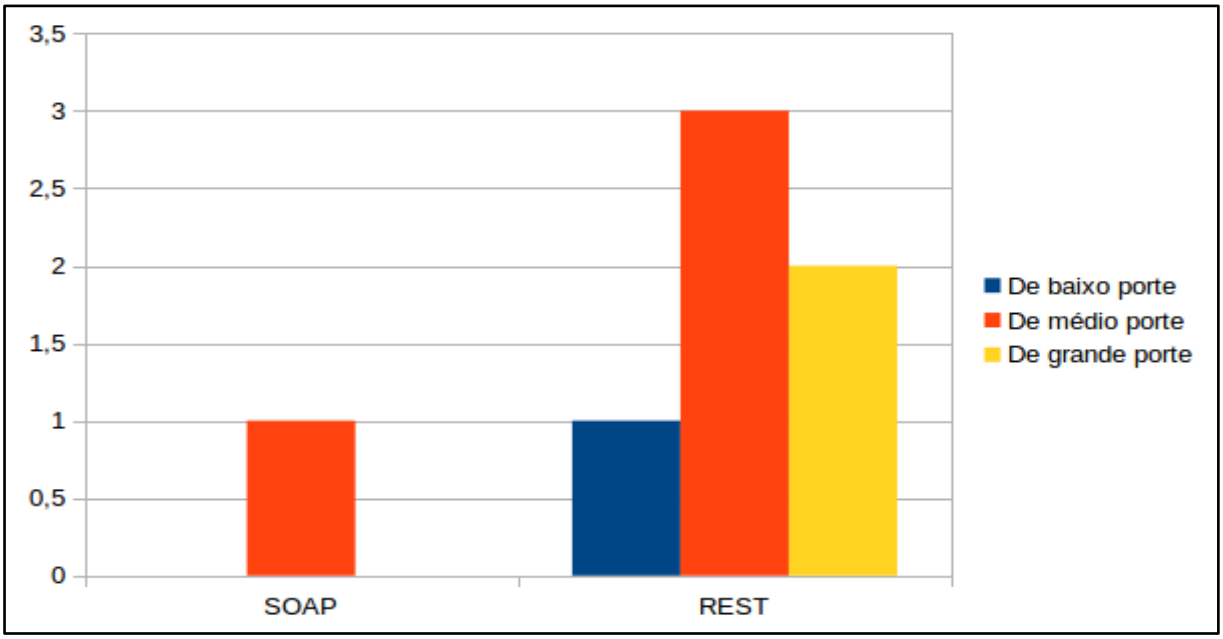
Gráfico 23 – Frequência de utilização das arquiteturas SOAP ou REST



Fonte: Próprio autor

Quando perguntados sobre o tipo de projetos em que os respondentes utilizam o SOAP ou o REST, com relação ao SOAP, 1 pessoa respondeu utilizar a arquitetura em projetos de médio porte. Já com relação ao REST, 2 pessoas responderam utilizá-la em projetos de grande porte, 3 pessoas responderam utilizá-la em projetos de médio porte e 1 uma pessoa respondeu utilizá-la em projetos de baixo porte. Com base nesses resultados verifica-se que a maioria dos respondentes utiliza a arquitetura SOAP ou REST em projetos de médio porte, o que está de acordo com o gráfico 24.

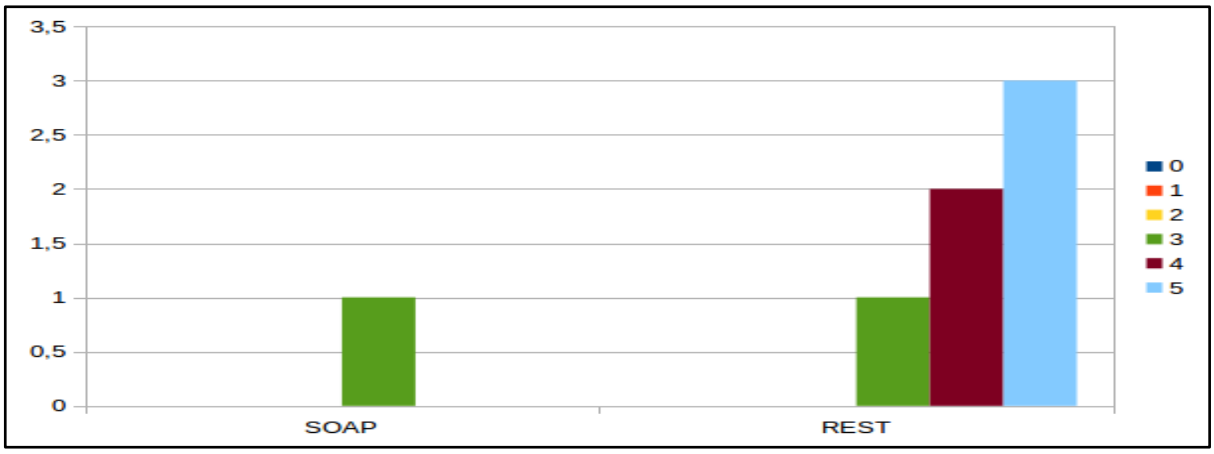
Gráfico 24 – Uso da arquitetura SOAP ou REST



Fonte: Próprio autor

Foi perguntado também aos respondentes qual o grau de aceitação destas arquiteturas. Em uma escala de 0 a 5 onde 0 pode ser considerado como baixa aceitação e 5 pode ser considerado como alta aceitação, com relação ao SOAP, nenhuma pessoa respondeu 0; nenhuma pessoa respondeu 1; nenhuma pessoa respondeu 2; 1 pessoa respondeu 3; nenhuma pessoa respondeu 4 e nenhuma pessoa respondeu 5. Já com relação ao REST, nenhuma pessoa respondeu 0; nenhuma pessoa respondeu 1; nenhuma pessoa respondeu 2; 1 pessoa respondeu 3; 2 pessoas responderam 4 e 3 pessoas responderam 5. Com base nesses resultados verifica-se que ambas as arquiteturas são bem aceitas sendo que o REST leva uma ligeira vantagem, o que pode ser visto no gráfico 25.

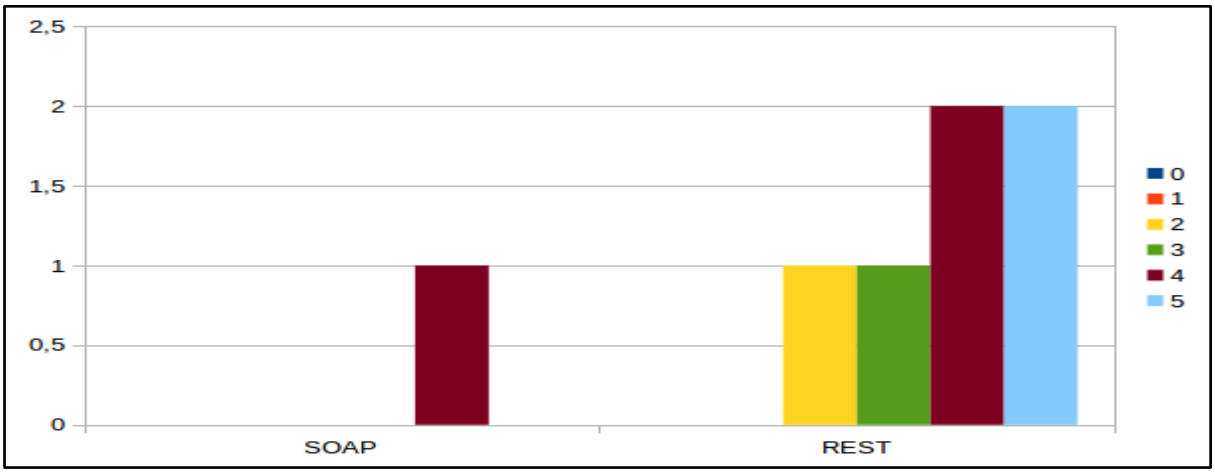
Gráfico 25 – Grau de aceitação das arquiteturas



Fonte: Próprio autor

Foi perguntado também aos respondentes qual o grau de aprendizado ao se utilizar estas arquiteturas. Em uma escala de 0 a 5 onde 0 pode ser considerado como aprendizado baixo e 5 pode ser considerado como aprendizado alto, com relação ao SOAP, nenhuma pessoa respondeu 0; nenhuma pessoa respondeu 1; nenhuma pessoa respondeu 2; nenhuma pessoa respondeu 3; 1 pessoa respondeu 4 e nenhuma pessoa respondeu 5. Já com relação ao REST, nenhuma pessoa respondeu 0; nenhuma pessoa respondeu 1; 1 pessoa respondeu 2; 1 pessoa respondeu 3; 2 pessoas responderam 4 e 2 pessoas responderam 5. Com base nesses resultados verifica-se que ambas as arquiteturas apresentam um bom nível de aprendizado ao utilizá-las, o que pode ser observado no gráfico 26.

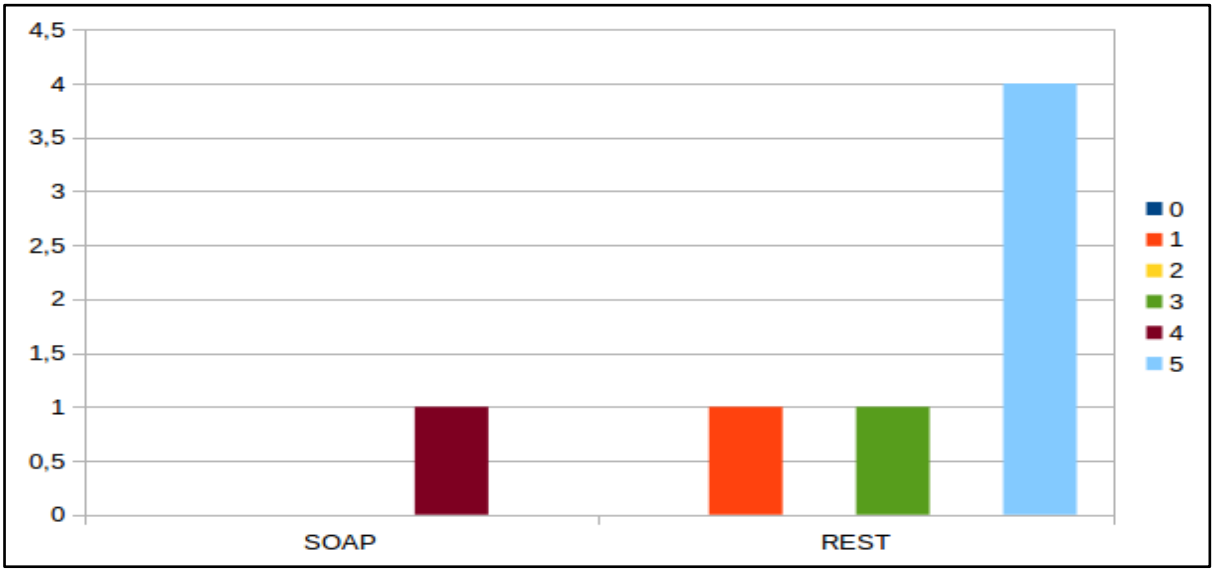
Gráfico 26 – Grau de aprendizado das arquiteturas



Fonte: Próprio autor

Foi perguntado também aos respondentes qual o grau de facilidade de manutenção apresentado pelas arquiteturas. Em uma escala de 0 a 5 onde 0 pode ser considerado como pouca facilidade e 5 pode ser considerado como alta facilidade, com relação ao SOAP, nenhuma pessoa respondeu 0; nenhuma pessoa respondeu 1; nenhuma pessoa respondeu 2; nenhuma pessoa respondeu 3; 1 pessoa respondeu 4 e nenhuma pessoa respondeu 5. Já com relação ao REST, nenhuma pessoa respondeu 0; 1 pessoa respondeu 1; nenhuma pessoa respondeu 2; 1 pessoa respondeu 3; nenhuma pessoa respondeu 4 e 4 pessoas responderam 5. Com base nesses resultados verifica-se que ambas as arquiteturas apresentam facilidades para se aplicar manutenção, o que pode ser observado no gráfico 27.

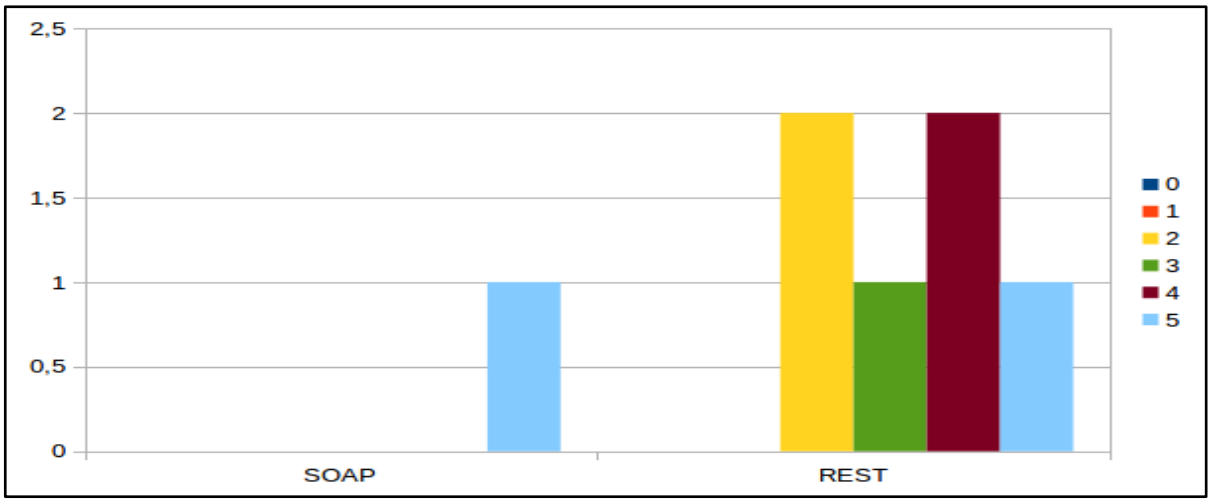
Gráfico 27 – Grau de facilidade de manutenção das arquiteturas



Fonte: Próprio autor

Foi perguntado também aos respondentes qual o grau de segurança apresentado pelas arquiteturas. Em uma escala de 0 a 5 onde 0 pode ser considerado como baixa segurança e 5 pode ser considerado como alta segurança, com relação ao SOAP, nenhuma pessoa respondeu 0, 1, 2, 3 ou 4 e 1 pessoa respondeu 5. Já com relação ao REST, nenhuma pessoa respondeu 0; nenhuma pessoa respondeu 1; 2 pessoas responderam 2; 1 pessoa respondeu 3; 2 pessoas responderam 4 e 1 pessoa respondeu 5. Com base nesses resultados verifica-se que ambas as arquiteturas apresentam uma boa segurança, o que pode ser visto no gráfico 28.

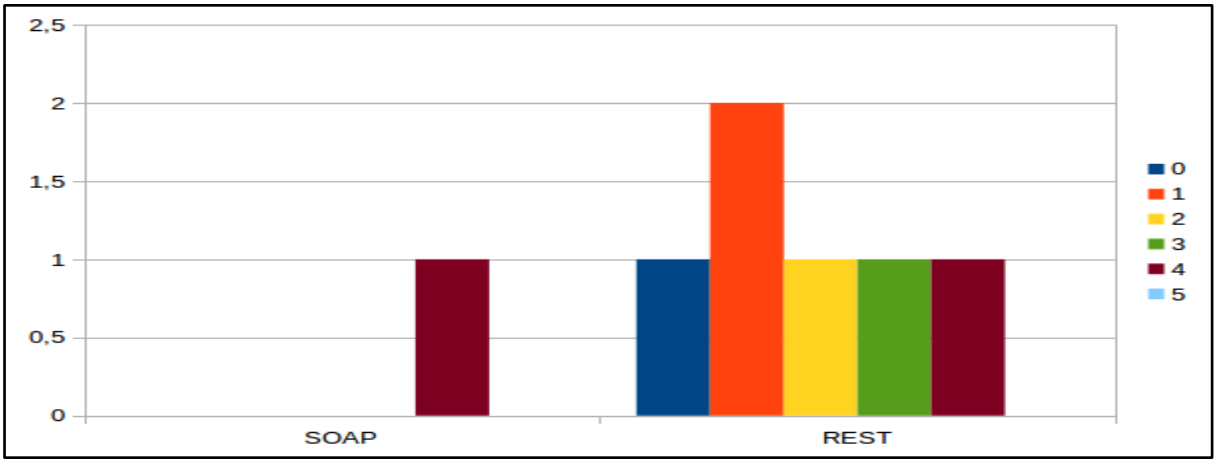
Gráfico 28 – Grau de segurança das arquiteturas



Fonte: Próprio autor

Foi perguntado também aos respondentes qual o grau de simplicidade apresentado pelas arquiteturas. Em uma escala de 0 a 5 onde 0 pode ser considerado com pouco simples e 5 pode ser considerado como muito simples, com relação ao SOAP, nenhuma pessoa respondeu 0; nenhuma pessoa respondeu 1; nenhuma pessoa respondeu 2; nenhuma pessoa respondeu 3; 1 pessoa respondeu 4 e nenhuma pessoa respondeu 5. Já com relação ao REST, 1 pessoa respondeu 0; 2 pessoas responderam 1; 1 pessoa respondeu 2; 1 pessoa respondeu 3; 1 pessoa respondeu 4 e nenhuma pessoa respondeu 5. Com base nesses resultados verifica-se que a arquitetura REST apresenta vantagem em relação à arquitetura SOAP neste quesito, o que pode ser observado no gráfico 29.

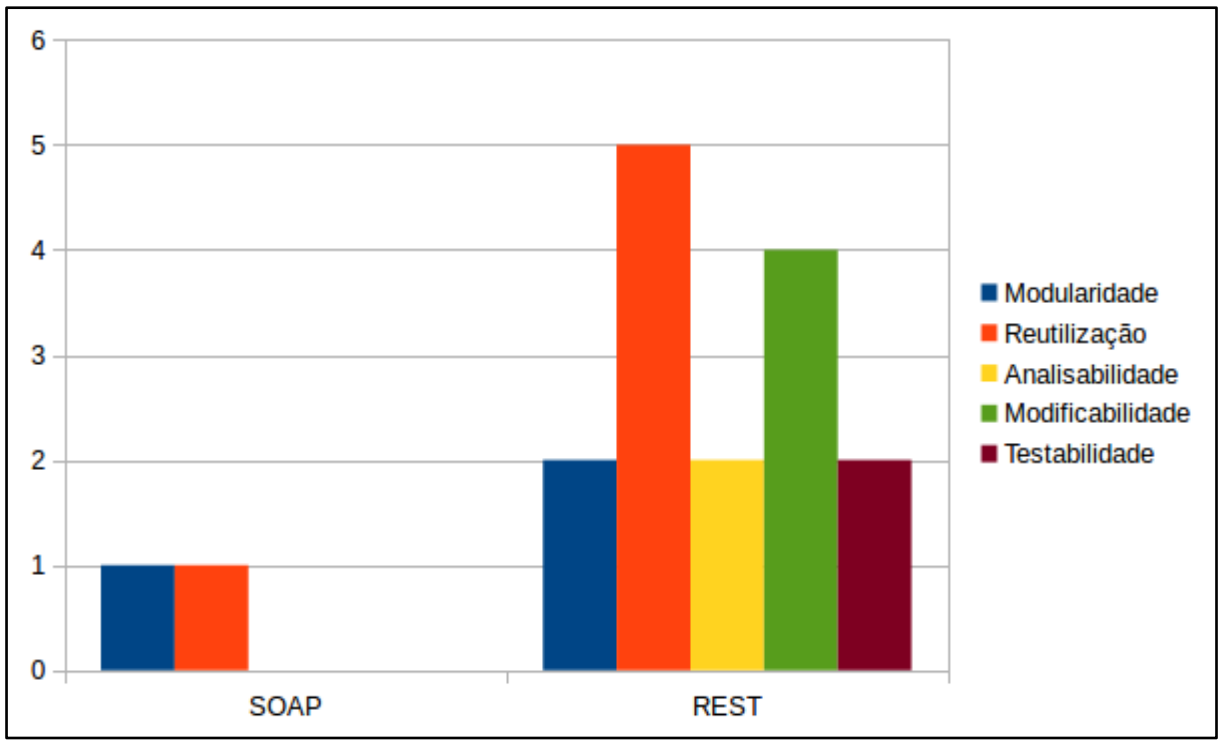
Gráfico 29 – Grau de simplicidade das arquiteturas



Fonte: Próprio autor

Quando perguntados sobre as principais características da arquitetura SOAP e REST, com relação ao SOAP, 1 pessoa respondeu que é a manutenibilidade e 1 pessoas respondeu que é a reutilização. Já com relação ao REST, 2 pessoas responderam que é a modularidade, 5 pessoas responderam que é a reutilização, 2 pessoas responderam que é a analisabilidade, 4 pessoas responderam que é a modificabilidade e 2 pessoas responderam que é a testabilidade. Com base nesses resultados verifica-se que a reutilização é a principal característica do SOAP e do REST para os respondentes, o que pode ser observado no gráfico 30.

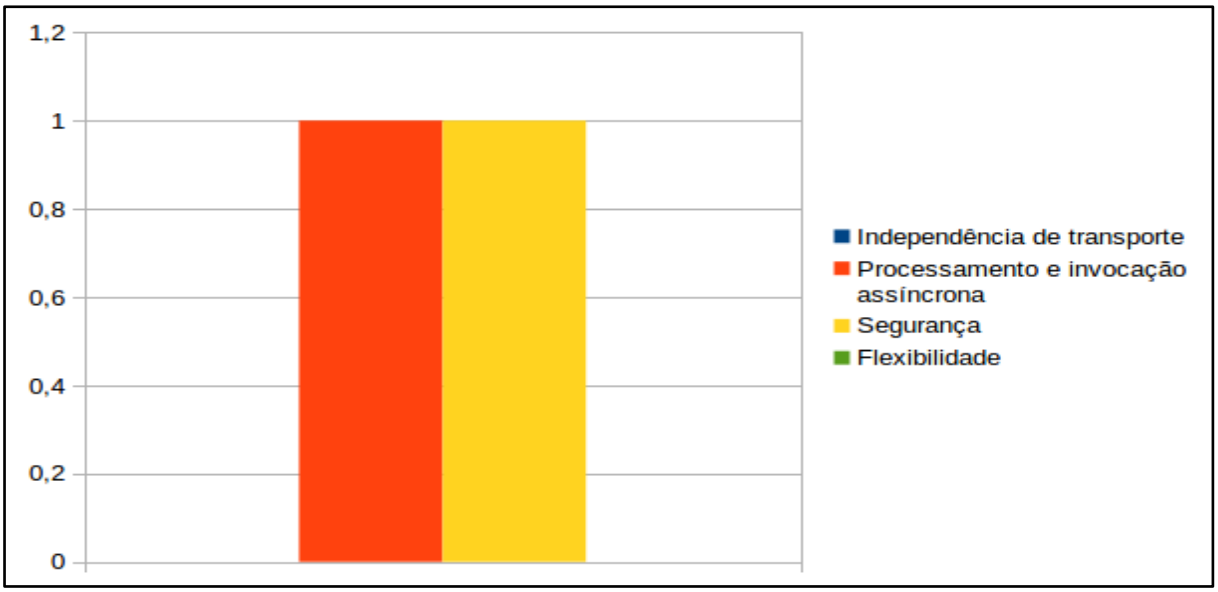
Gráfico 30 – Características das arquiteturas SOAP e REST



Fonte: Próprio autor

Quando perguntados sobre as principais vantagens da arquitetura SOAP, 1 pessoa respondeu que é a segurança e 1 pessoa respondeu que é o processamento e invocação assíncrona. Com base nesses resultados verifica-se que a segurança é a principal vantagem do SOAP para os respondentes, o que pode ser observado no gráfico 31.

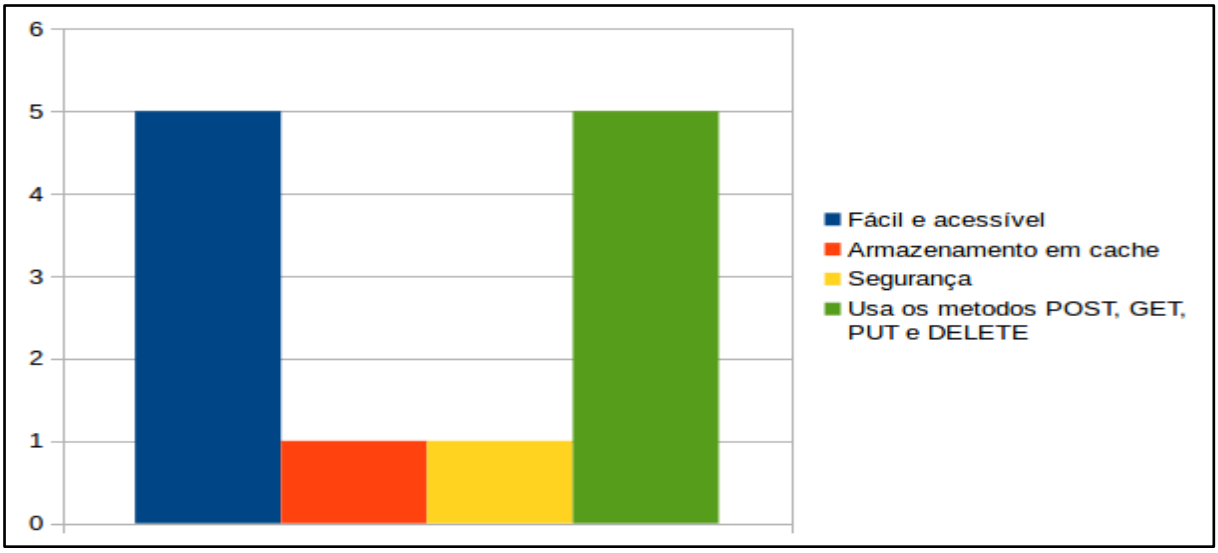
Gráfico 31 – Principais vantagens da arquitetura SOAP



Fonte: Próprio autor

Quando perguntados sobre as principais vantagens da arquitetura REST, 5 pessoas responderam que é o fato da arquitetura permitir o uso de métodos como *post*, *get*, *put* e *delete*, 5 pessoas responderam que é a facilidade e acessibilidade, 1 pessoa respondeu que é o armazenamento em *cache* e 1 pessoa respondeu que é a segurança. Com base nesses resultados verifica-se que a capacidade de usar métodos como *post*, *get*, *put* e *delete* é a principal vantagem do REST para os respondentes, o que pode ser observado no gráfico 32.

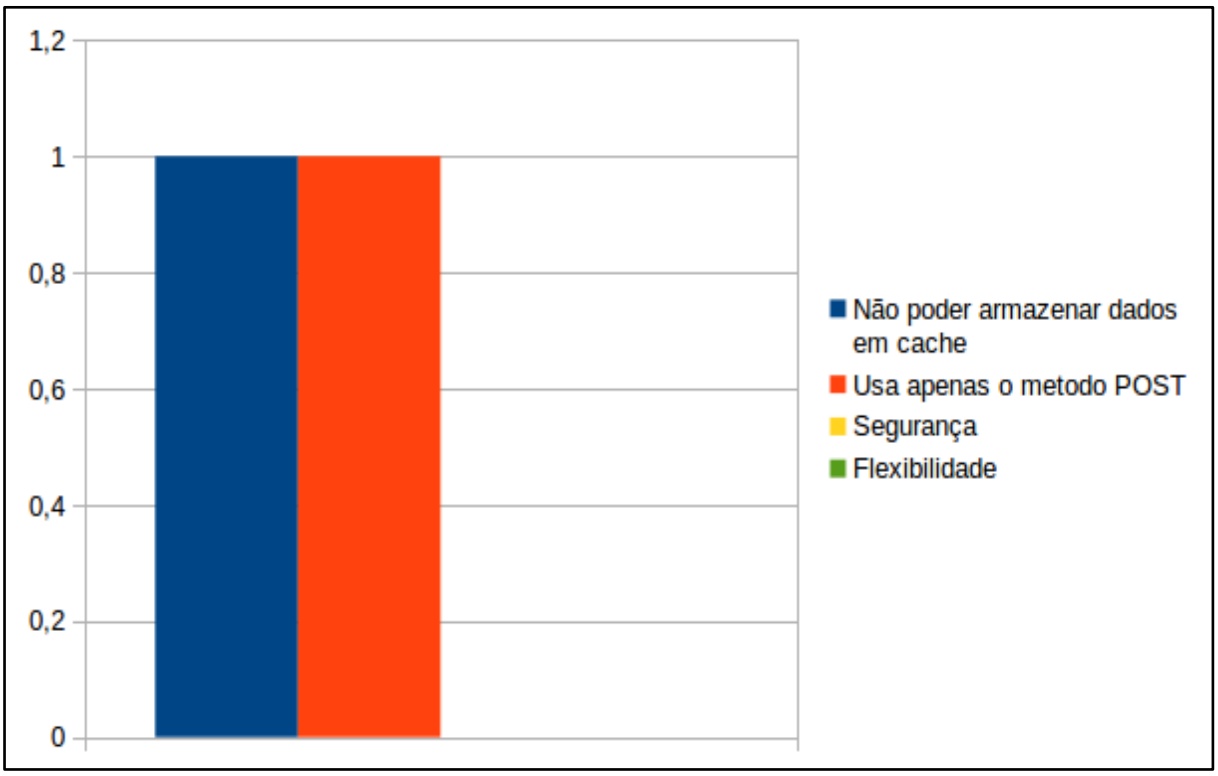
Gráfico 32 – Principais vantagens da arquitetura REST



Fonte: Próprio autor

Quando perguntados sobre as principais desvantagens da arquitetura SOAP, 1 pessoa respondeu que é o fato da arquitetura permitir o uso apenas do método *post* e 1 pessoa respondeu que é o fato de não poder armazenar dados em *cachê*. Com base nesses resultados verifica-se que o fato da arquitetura SOAP permitir apenas o uso do método *post* é considerado a principal desvantagem desta arquitetura para os respondentes, o que pode ser observado no gráfico 33.

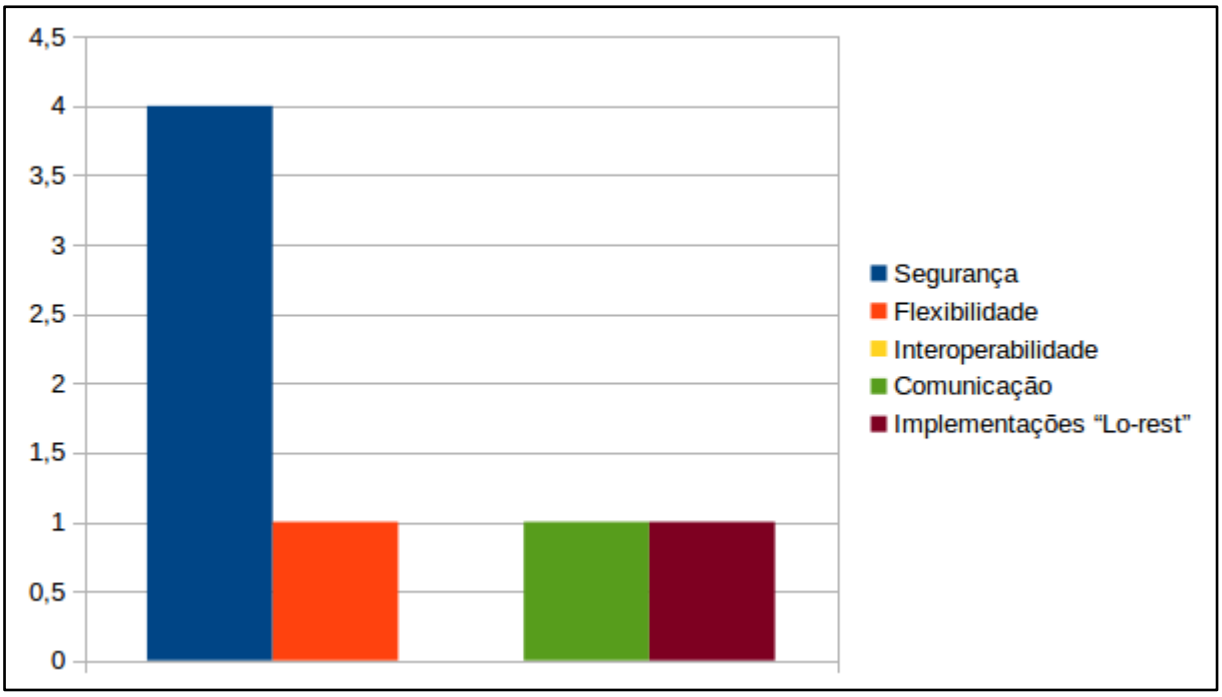
Gráfico 33 – Principais desvantagens da arquitetura SOAP



Fonte: Próprio autor

Quando perguntados sobre as principais desvantagens da arquitetura REST, 4 pessoas responderam que é a segurança, 1 pessoa respondeu que é a comunicação, 1 pessoa respondeu que é a flexibilidade, nenhuma pessoa respondeu que é a interoperabilidade e 1 pessoa respondeu outra desvantagem. Com base nesses resultados verifica-se que a segurança é a principal desvantagem do REST para os respondentes, o que pode ser observado no gráfico 34.

Gráfico 34 – Principais desvantagens da arquitetura REST



Fonte: Próprio autor

3.4 Discurssão dos resultados

A partir da análise do questionário foi possível observar que a grande maioria dos respondentes são pessoas com uma formação acadêmica igual ou superior a uma graduação, sendo que esta maioria possui 5 anos ou mais de experiência na área de TI, tendo o desenvolvimento de *software* como área predominante dos respondentes.

Segundo Pressman, “a manutenção de *software* é um fator de extrema importância, pois, é considerada como a fase onde a organização desprende maior esforço, consumindo em torno de 70% da atenção do pessoal de TI, e, este percentual continua aumentando gradativamente” (PRESSMAN, 1995 p. 876). Por meio das respostas do questionário foi possível observar que aproximadamente 90% dos respondentes consideram a aplicação da manutenibilidade em projetos de *software* como muito importante.

Um fator de grande importância que pode ser observado é que a maioria dos respondentes possui conhecimento tanto da arquitetura SOAP quanto da arquitetura

REST, o que contribui muito para a comparação das arquiteturas.

Com base nas opiniões dos respondentes pode-se verificar que ambas as arquiteturas comparadas são bastante utilizadas nos seus respectivos projetos sendo estes de médio a grande porte. Quando perguntados aos respondentes sobre o grau de aceitação das arquiteturas pode-se observar que a arquitetura REST possui uma vantagem sobre a arquitetura SOAP, o que também ocorre com relação à facilidade de manutenção. Já com relação à segurança a maioria dos respondentes considerou a arquitetura SOAP como mais eficaz neste quesito. E com relação à simplicidade a maioria dos respondentes considerou a arquitetura REST como a mais simples.

De acordo com o portal ISO 25000 (2017), a reutilização pode ser definida como um atributo de *software* em que um bem pode ser usado em mais de um sistema, ou na construção de outros sistemas. Para os respondentes do questionário esta foi considerada a característica da ISO 25010 que melhor caracteriza a arquitetura SOAP. Já com relação ao REST a modularidade que de acordo com o portal ISO 25000 (2017) pode ser considerada como um atributo de *software* em que o *software* é composto de componentes discretos, de modo que alterações em um componente tenham um impacto mínimo em outros componentes, foi considerada a característica da ISO 25010 que melhor se refere a esta arquitetura.

Com base em todos os dados fornecidos pelos respondentes do questionário pode-se verificar que a maioria destes considera a arquitetura REST como a melhor para ser aplicada em projetos de *software*, como a mais fácil de ser aplicada e também como a arquitetura que apresenta um melhor nível de manutenibilidade. Sendo que a arquitetura SOAP não foi considerada a melhor nestes quesitos, mas também se apresentou como uma boa arquitetura para se aplicada em projetos de software e um bom nível de manutenibilidade.

O quadro 1 apresenta uma comparação entre ambas as arquiteturas analisadas no decorrer deste trabalho, levando em consideração as respostas fornecidas pelos respondentes do questionário à perguntas sobre frequência de utilização, uso, aceitação, aprendizado, facilidade de manutenção, segurança, simplicidade e manutenibilidade.

Quadro 1 – Comparação das Características

	SOAP	**REST**
Frequência	Bem utilizada	Muito utilizada
Uso	Tem seu uso principalmente em projetos de grande e médio porte	Tem seu uso principalmente em projetos de grande e médio porte
Aceitação	Bom nível de aceitação	Excelente nível de aceitação
Aprendizado	Bom nível de aprendizado	Excelente nível de aprendizado
Facilidade	Bom nível de facilidade de manutenção	Excelente nível de facilidade de manutenção
Segurança	Excelente nível de segurança	Bom nível de segurança
Simplicidade	Bom nível de simplicidade	Excelente nível de simplicidade
Manutenibilidade	Bom nível de manutenibilidade	Excelente nível de manutenibilidade

Fonte: Próprio autor

CONCLUSÃO

O presente trabalho teve por objetivo comparar as arquiteturas de serviços web SOAP e REST, no intuito de avaliar qual arquitetura atende melhor à ISO 25010 no que se refere à manutenibilidade, para isto a princípio foram estudados todos os conceitos que forneceram o conhecimento necessário para o desenvolvimento deste trabalho.

Também foi realizada uma pesquisa via internet para profissionais da área de tecnologia da informação sobre a facilidade de manutenção em *software* que utilizam a arquitetura SOAP e REST, através de um questionário.

Com base nas respostas obtidas por meio do questionário foi possível constatar que os profissionais de Tecnologia da Informação têm ciência da necessidade de se aplicar a manutenibilidade em projetos de softwares, em função da complexidade dos sistemas produzidos e das exigências dos clientes que querem softwares de qualidade, confiáveis e eficientes.

Foi possível observar também que ambas as arquiteturas SOAP e REST, que foram o objeto de estudo deste trabalho são frequentemente utilizadas pelos profissionais da área. Sendo que cada uma das arquiteturas possui suas vantagens que podem ser aproveitadas de acordo com a necessidade. Para os respondentes a segurança é principal vantagem da arquitetura SOAP, já com relação à arquitetura REST os respondentes consideraram a flexibilidade como principal vantagem.

Com relação a ISO 25010 no quesito da manutenibilidade foi possível constatar que a arquitetura REST foi considerada a melhor para ser aplicada em projetos de *software*, a mais fácil de ser aplicada e também como a arquitetura que atende melhor o quesito da manutenibilidade, sendo que a modularidade, que é uma subcaracterística da manutenibilidade, foi considerada a principal característica desta arquitetura. Já a arquitetura SOAP foi considerada boa para ser aplicada em projetos de software e também com um bom nível de manutenibilidade, sendo que a reutilização, que é uma subcaracterística da manutenibilidade, foi considerada a principal característica desta arquitetura.

TRABALHOS FUTUROS

Como direção para possíveis trabalhos futuros tem-se:

- A aplicação de um questionário mais aprofundado sobre manutenibilidade de *software* envolvendo mais profissionais da área de TI que tenha conhecimento de *web service* SOAP e REST a fim de entender melhor a complexidade de se aplicar a manutenibilidade em *web services*.
- A avaliação de ferramentas para o desenvolvimento de *Web Services* que suportem serviços SOAP e REST. Como sugestão para essa avaliação tem-se *IBM Web Service Toolkit* (WSTK), *Java Web Service Developer Pack* (JWSDP) e *Web Application and Service Plataform* (WASP).
- O detalhamento e aprofundamento de metodologias para implementar segurança em serviços SOAP e REST, como o *WS-Security*.

REFERÊNCIAS

ABRAHAM Silberschatz, Henry F. KORTH, S. SUDARSHAN. *Sistema de Banco de Dados.* 5.ª Edição. Campus, 2006.

BLAKE, M. B. et al. Binding Now or Binding Later: The Performance of UDDI Registries. System Sciences 40th Annual Hawaii International Conference on, Janeiro 2007.

BOX, D. O'Reilly XML.Com. A Brief History of SOAP, 04 Abril 2001. Disponivel em: <http://www.xml.com/pub/a/ws/2001/04/04/soap.html>. Acesso em: 23 Out. 2017.

DEITEL, P. *Java Como Programar*. Tradução. São Paulo: Pearson Prentice Hall, 2010.

FIELDING, Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures.* Tese de Doutorado, University of california, Irvine, 2000. Disponível em: <http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm>. Acesso em: 16 de junho de 2017.

GONSALVES, A. Beginning Java EE 6 Plataform with GlassFish 3: From Novice to Professional. New York: Apress, 2009.

ISO 25000 software product quality. Disponível em: < http://iso25000.com/index.php/en/iso-25000-standards/iso-25010 >. Acesso em: 16 de junho de 2017.

MEDYK, Sérgio. Web Services em Gerência de Redes. Material de Apoio. Paraná, 2006. Disponível em: <http://www.ppgia.pucpr.br/~jamhour/Download/pub/Mestrado%202006/Web%20Service.pdf>. Acesso em: 23 Out. 2017.

ORT, E. The Source - Sum microsystems. Service Oriented Architecture and Web Services: Concepts, Tecnologies, and tools, 2005.

PRESSMAN, Roger S. Engenharia de Software. 3. ed. São Paulo: Makron Books, 1995.

PRESSMAN, Roger S. *Engenharia de Software: uma abordagem profissional*. 7.ed. Porto Alegre: AMGH Editora Ltda, 2011.

RIBEIRO, Ticianne; BRAGA, Antônio Rafael; SOUSA, Verônica Lima Pimentel. *Um ambiente virtual de ensino e aprendizagem baseado em web semântica e web services*. Disponível em: <http://artigocientifico.uol.com.br/uploads/artc_1160063078_11.pdf>. Acesso em: 16 de junho de 2017.

RICHARDSON, L.; RUBY, S. RESTful Web Services. [S.l.]: O'Reilly, 2007.

SHOMOYITA, J.; RALPH, D. Using a Cloud-Hosted Proxy to support Mobile Consumers of RESTful Services. Procedia Computer Science, v. Volume 5, p. 625-632, 2011.

SILVA, Marcos Cesar da. *Uma arquitetura de software para mediação flexível de Web services*. Rio de Janeiro, 2008.

SOMMERVILLE, Ian. *Engenharia de Software*. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

TIDWELL, D.; SNELL, J.; KULCHENKO, P. Programing Web Services with SOAP. 1. ed. [S.l.]: O'Reilly, 2001. 216 p.

W3C. *Web Services Architecture*. Disponível em: <http://www.w3.org/TR/ws-arch/>. Acesso em: 16 de junho de 2017.

ANEXO A – QUESTIONÁRIO

Este questionário tem o intuito de comparar as arquiteturas de serviços web SOAP e REST para analisar quais destas atendem melhor o quesito da manutenibilidade tendo como base a ISO 25010, além de analisar fatores que podem influenciar na manutenibilidade.

Nenhuma das informações fornecidas será divulgada, só serão usadas como base para um trabalho científico. Desde já agradeço por participar!

* Obrigatório

Perfil dos Respondentes

Se desejar receber o resultado do trabalho quando o mesmo estiver disponível, deixe seu e-mail.

_____

1 - Qual a cidade que você mora? *

_____

2 - Qual seu nível de formação acadêmica? *

- Técnico
- Graduando ou Graduado
- Pós-Graduando ou Pós-Graduado
- Outros: _____

3 - Há quanto tempo você exerce qualquer atividade relacionada à área da computação? *

- Menos de 1 ano
- De 1 a 5 anos
- De 5 a 10 anos

- Mais de 10 anos

4 - Qual sua atual função relacionada à instituição ou empresa que você se encontra? *

- Estudante
- Apoio técnico (infraestrutura, redes)
- Apoio Operacional (desenvolvedor de softwares, analista de testes, suporte de sistemas, analista de sistemas, webdesigner)
- Apoio Gerencial ou Estratégico (Coordenador de equipes, diretor, CEO, CIO, gerente de projetos, gerente de qualidades, gerente de testes)
- Outros: _____

5 - Em sua opinião, qual a importância da aplicação da manutenibilide em projetos de software? *

	0	1	2	3	4	5	6	7	8	9	10	
Pouco importante	○	○	○	○	○	○	○	○	○	○	○	Muito importante

6 - Possui conhecimento da arquitetura SOAP ou REST? *

- Ambos
- Somente SOAP
- Somente REST
- Nenhum

SOAP E REST

7 - No desenvolvimento de projetos de software, qual a frequência de utilização das arquiteturas SOAP e REST em seus projetos? *

	0	1	2	3	4	5
SOAP	○	○	○	○	○	○
REST	○	○	○	○	○	○

8 - Geralmente você, sua empresa ou equipe costumam fazer o uso da arquitetura SOAP em que tipo de projetos? *

- De baixo porte
- De médio porte
- De grande porte
- Não trabalho com SOAP

9 - Qual o grau de aceitação das arquiteturas SOAP e REST? *

	0	1	2	3	4	5
SOAP	○	○	○	○	○	○
REST	○	○	○	○	○	○

10 - Em sua opinião, qual o grau de aprendizado ao se utilizar as arquiteturas SOAP e REST? *

	0	1	2	3	4	5
SOAP	○	○	○	○	○	○
REST	○	○	○	○	○	○

11 - Qual o grau de facilidade de manutenção apresentado pelas arquiteturas SOAP e REST? *

	0	1	2	3	4	5
SOAP	○	○	○	○	○	○
REST	○	○	○	○	○	○

12 - Qual o grau de segurança apresentado pelas arquiteturas SOAP e REST? *

	0	1	2	3	4	5
SOAP	○	○	○	○	○	○
REST	○	○	○	○	○	○

13 - Qual o grau de simplicidade apresentado pelas arquiteturas SOAP e REST? *

	0	1	2	3	4	5
SOAP	○	○	○	○	○	○
REST	○	○	○	○	○	○

14 - De acordo com a ISO 25010, a manutenibilidade pode ser dividida em subcaracterísticas que estão listadas abaixo. Informe em qual característica a arquitetura SOAP apresenta melhor resultado. *

Pode ser informada mais de uma opção.

- Modularidade – Atributo pelo qual o software é composto de componentes discretos, de modo que alterações em um componente tenham um impacto mínimo em outros componentes.
- Reutilização – Atributo de software em que um bem pode ser usado em mais de um sistema, ou na construção de outros sistemas.
- Analisabilidade – Atributo de software que identifica a facilidade em se diagnosticar problemas e identificar as causas das falhas.
- Modificabilidade – Atributo de software que caracteriza a facilidade com que o software pode ser modificado.
- Testabilidade – Atributo de software que representa a capacidade de se testar o software modificado.

15 - Quais as principais vantagens do SOAP? *

Pode ser informada mais de uma opção.

- Independência de transporte
- Processamento e invocação assíncrona
- Segurança
- Flexibilidade
- Outros: _____

16 - Quais as principais desvantagens do SOAP? *

Pode ser informada mais de uma opção.

- Não poder armazenar dados em cachê
- Usa apenas o método POST

- Segurança
- Flexibilidade
- Outros: _____

17 - Geralmente você, sua empresa ou equipe costumam fazer o uso da arquitetura REST em que tipo de projetos? *

- De baixo porte
- De médio porte
- De grande porte
- Não trabalho com REST

18 - De acordo com a ISO 25010, a manutenibilidade pode ser dividida em subcaracterísticas que estão listadas abaixo. Informe em qual característica a arquitetura REST apresenta melhor resultado. *

Pode ser informada mais de uma opção.

- Modularidade – Atributo pelo qual o software é composto de componentes discretos, de modo que alterações em um componente tenham um impacto mínimo em outros componentes.
- Reutilização – Atributo de software em que um bem pode ser usado em mais de um sistema, ou na construção de outros sistemas.
- Analisabilidade – Atributo de software que identifica a facilidade em se diagnosticar problemas e identificar as causas das falhas.
- Modificabilidade – Atributo de software que caracteriza a facilidade com que o software pode ser modificado.
- Testabilidade – Atributo de software que representa a capacidade de se testar o software modificado.

19 - Quais as principais vantagens do REST? *

Pode ser informada mais de uma opção.

- Fácil e acessível
- Armazenamento em cachê

- Segurança
- Usar os métodos POST, GET, PUT e DELETE
- Outros: _____

20 - Quais as principais desvantagens do REST? *

Pode ser informada mais de uma opção.

- Segurança
- Flexibilidade
- Interoperabilidade
- Comunicação
- Outros: _____

21 - Dentre as arquiteturas SOAP e REST, qual delas você considera melhor para ser aplicada em projetos de software? *

- Ambas
- SOAP
- REST
- Outros: _____

22 - Dentre as arquiteturas SOAP e REST, qual delas você considera mais fácil de ser aplicada em projetos de software? *

- Ambas
- SOAP
- REST
- Nenhum

23 - De acordo com a ISO 25010, a manutenibilidade corresponde ao grau de eficácia e eficiência com que um produto ou sistema pode ser modificado para melhorá-lo, corrigi-lo ou adaptá-lo. Em sua opinião qual arquitetura apresenta um alto nível de manutenibilidade? *

- Ambos
- SOAP
- REST
- Nenhum

SOAP

7 - No desenvolvimento de projetos de software, qual a frequência de utilização da arquitetura SOAP em seus projetos? *

	0	1	2	3	4	5	
Baixa	○	○	○	○	○	○	Alta

8 - Geralmente você, sua empresa ou equipe costumam fazer o uso da arquitetura SOAP em que tipo de projetos?

- De baixo porte
- De médio porte
- De grande porte
- Não trabalho com SOAP

9 - Qual o grau de aceitação da arquitetura SOAP? *

	0	1	2	3	4	5	
Pouco	○	○	○	○	○	○	Muito

10 - Em sua opinião, qual o grau de aprendizado ao se utilizar a arquitetura SOAP? *

	0	1	2	3	4	5	
Pouco	○	○	○	○	○	○	Muito

11 - Qual o grau de facilidade de manutenção apresentado pela arquitetura SOAP? *

12 - Qual o grau de segurança apresentado pela arquitetura SOAP? *

13 - Qual o grau de simplicidade apresentado pela arquitetura SOAP? *

14 - De acordo com a ISO 25010, a manutenibilidade pode ser dividida em subcaracterísticas que estão listadas abaixo. Informe em qual característica a arquitetura SOAP apresenta melhor resultado. *

Pode ser informada mais de uma opção.

- Modularidade – Atributo pelo qual o software é composto de componentes discretos, de modo que alterações em um componente tenham um impacto mínimo em outros componentes.
- Reutilização – Atributo de software em que um bem pode ser usado em mais de um sistema, ou na construção de outros sistemas.
- Analisabilidade – Atributo de software que identifica a facilidade em se diagnosticar problemas e identificar as causas das falhas.
- Modificabilidade – Atributo de software que caracteriza a facilidade com que o software pode ser modificado.
- Testabilidade – Atributo de software que representa a capacidade de se testar o software modificado.

15 - Quais as principais vantagens do SOAP? *

Pode ser informada mais de uma opção.

- Independência de transporte
- Processamento e invocação assíncrona
- Segurança
- Flexibilidade
- Outros: _____

16 - Quais as principais desvantagens do SOAP? *

Pode ser informada mais de uma opção.

- Não poder armazenar dados em cachê
- Usa apenas o método POST
- Segurança
- Flexibilidade
- Outros: _____

REST

7 - No desenvolvimento de projetos de software, qual a frequência de utilização da arquitetura REST em seus projetos? *

	0	1	2	3	4	5	
Baixa	○	○	○	○	○	○	Alta

8 - Geralmente você, sua empresa ou equipe costumam fazer o uso da arquitetura REST em que tipo de projetos?

- De baixo porte
- De médio porte
- De grande porte
- Não trabalho com REST

9 - Qual o grau de aceitação da arquitetura REST? *

	0	1	2	3	4	5	
Pouco	○	○	○	○	○	○	Muito

10 - Em sua opinião, qual o grau de aprendizado ao se utilizar a arquitetura REST? *

	0	1	2	3	4	5	
Pouco	○	○	○	○	○	○	Muito

11 - Qual o grau de facilidade de manutenção apresentado pela arquitetura REST? *

	0	1	2	3	4	5	
Pouco	○	○	○	○	○	○	Muito

12 - Qual o grau de segurança apresentado pela arquitetura REST? *

	0	1	2	3	4	5	
Pouco	○	○	○	○	○	○	Muito

13 - Qual o grau de simplicidade apresentado pela arquitetura REST? *

	0	1	2	3	4	5	
Pouco	○	○	○	○	○	○	Muito

14 - De acordo com a ISO 25010, a manutenibilidade pode ser dividida em subcaracterísticas que estão listadas abaixo. Informe em qual característica a arquitetura REST apresenta melhor resultado. *

Pode ser informada mais de uma opção.

- Modularidade – Atributo pelo qual o software é composto de componentes discretos, de modo que alterações em um componente tenham um impacto mínimo em outros componentes.

- Reutilização – Atributo de software em que um bem pode ser usado em mais de um sistema, ou na construção de outros sistemas.
- Analisabilidade – Atributo de software que identifica a facilidade em se diagnosticar problemas e identificar as causas das falhas.
- Modificabilidade – Atributo de software que caracteriza a facilidade com que o software pode ser modificado.
- Testabilidade – Atributo de software que representa a capacidade de se testar o software modificado.

15 - Quais as principais vantagens do REST? *

Pode ser informada mais de uma opção.

- Fácil e acessível
- Armazenamento em cachê
- Segurança
- Usa os metodos POST, GET, PUT e DELETE
- Outros: _____

16 - Quais as principais desvantagens do REST? *

Pode ser informada mais de uma opção.

- Segurança
- Flexibilidade
- Interoperabilidade
- Comunicação
- Outros: _____