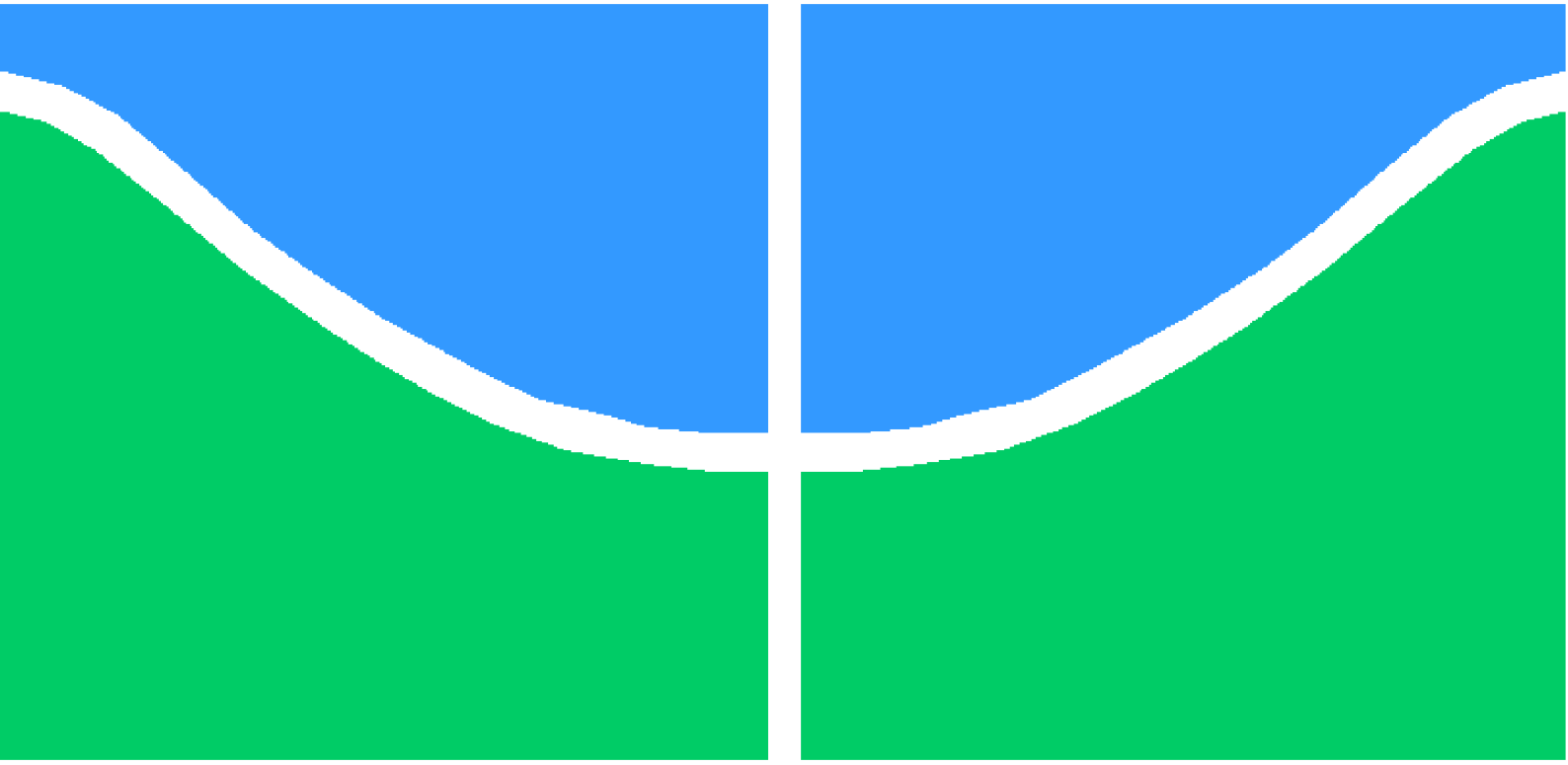

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de software

Proposta de uma arquitetura integrativa baseada em serviços para um ambiente virtual

Autora: Beatriz Ferreira Gonçalves

Orientador: Professor Doutor Sérgio Antônio Andrade de Freitas

Brasília, DF

2016

Beatriz Ferreira Gonçalves

Proposta de uma arquitetura integrativa baseada em serviços para um ambiente virtual

Monografia submetida ao curso de graduação em Engenharia de softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Professor Doutor Sérgio Antônio Andrade de Freitas

Brasília, DF

2016

Beatriz Ferreira Gonçalves
Proposta de uma arquitetura integrativa baseada em serviços para um ambiente virtual/ Beatriz Ferreira Gonçalves. – Brasília, DF, 2016-
46 p. : il. (algumas color.) ; 30 cm.

Orientador: Professor Doutor Sérgio Antônio Andrade de Freitas

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2016.

1. Palavra-chave01. 2. Palavra-chave02. I. Professor Doutor Sérgio Antônio Andrade de Freitas. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Proposta de uma arquitetura integrativa baseada em serviços para um ambiente virtual

CDU COLOCAR CDU

AQUI - DADOS DA FICHA CATALOGRAFICA

Beatriz Ferreira Gonçalves

Proposta de uma arquitetura integrativa baseada em serviços para um ambiente virtual

Monografia submetida ao curso de graduação em Engenharia de softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de software.

Trabalho aprovado. Brasília, DF, XX de Junho de 2016:

Professor Doutor Sérgio Antônio Andrade de Freitas
Orientador

Titulação e Nome do Professor Convidado 01
Convidado 1

Titulação e Nome do Professor Convidado 02
Convidado 2

Brasília, DF
2016

ESCREVER AQUI A DEDICATÓRIA

Agradecimentos

ESCREVER AQUI OS MEUS AGRADECIMENTOS

ESCREVER AQUI A MINHA EPÍGRAFE

Resumo

TODO: ESCREVER O RESUMO. DICAS SOBRE COMO ESCREVER O RESUMO ESTÃO LOGO ABAIXO

O resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. (. . .) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto. O texto pode conter no mínimo 150 e no máximo 500 palavras, é aconselhável que sejam utilizadas 200 palavras. E não se separa o texto do resumo em parágrafos.

Palavras-chaves: latex. abntex. editoração de texto.

Abstract

TODO: ESCREVER AQUI O ABSTRACT DO TRABALHO This is the english abstract.

Key-words: latex. abntex. text editoration.

Lista de ilustrações

Figura 1 – Interoperabilidade em uma arquitetura baseada no modelo SOA. 31
Figura 2 – Proposta da arquitetura baseada no modelo SOA com o uso de um ESB. 33
Figura 3 – Fluxo básico do protocolo de comunicação. 34
Figura 4 – Exemplo de uma mensagem de requisição de serviço em formato JSON de uma aplicação usuário escrita em Python e Django. 35
Figura 5 – Exemplo de uma mensagem de resposta de requisição em formato JSON. 35

Lista de tabelas

Tabela 1 – Cronograma de atividades relacionadas ao TCC 2 39
Tabela 2 – Cronograma de atividades relacionadas ao TCC 1 39

Lista de abreviaturas e siglas

SOA	*Service-Oriented Architecture*
SOAP	*Simple Object Access Protocol*
REST	*Representational State Transfer*
ESB	*Enterprise Service Bus*

Lista de símbolos

Γ	Letra grega Gama

Sumário

1 INTRODUÇÃO 25
1.1 Objetivos 25
1.1.1 Objetivos Geral 25
1.1.2 Objetivos específicos 25
1.1.3 Questão de pesquisa 25
1.2 Motivação 25
1.3 Metodologia 25
1.3.1 Classificação da pesquisa 25
1.3.2 Referencial teórico 26
1.3.3 Proposta 26
1.3.4 Engenharia de software 26
1.4 Estrutura da Monografia 26

2 REFERENCIAL TEÓRICO 27
2.1 Section 1 - Titulo 27
2.2 Engenharia de Software 27
2.2.1 Conceito de ESW adotados 27

3 A PROPOSTA 29
3.1 Introdução 29
3.2 Proposta de arquitetura 30
3.2.1 Requisitos 30
3.2.2 A arquitetura 30
3.2.2.1 Ferramentas ESB 32
3.2.3 Protocolo de comunicação 33
3.2.3.1 Formato das Mensagens 35

4 DESENVOLVIMENTO DA PROPOSTA 37
4.1 Introdução 37
4.2 Metodologia de desenvolvimento 37
4.2.1 Fases de desenvolvimento 38
4.3 Cronograma de execução 39

5 CONSIDERAÇÕES FINAIS 41

REFERÊNCIAS 43

1 INTRODUÇÃO

ESCREVER O INICIO AQUI. INTRODUÇÃO DEVE SER ABRANGENTE.

1.1 Objetivos

São apresentados nessa seção os objetivos gerais, objetivos específicos e a questão de pesquisa.

1.1.1 Objetivos Geral

DESCREVER AQUI O OBJETIVO GERAL.

1.1.2 Objetivos específicos

Os objetivos deste trabalho são:

- DESCREVER OS OBJETIVOS ESPECÍFICOS.

1.1.3 Questão de pesquisa

EXPOR QUESTÃO DE PESQUISA.

1.2 Motivação

EXPOR MOTIVAÇÃO PARA A REALIZAÇÃO DO TRABALHO.

1.3 Metodologia

Esss seção apresenta a metodologia que será usada no desenvolvimento desse trabalho:

1.3.1 Classificação da pesquisa

CLASSIFICAR A PESQUISA.

1.3.2 Referencial teórico

DESCREVER COMO A PESQUISA FOI REALIZADA E ONDE O REFERENCIAL TEÓRICO ESTA CONTIDO NO DOCUMENTO.

1.3.3 Proposta

DESCREVER A PROPOSTA.

1.3.4 Engenharia de software

DESCREVER AS PRÁTICAS DE ENGENHARIA DE SOFTWARE ADOTADAS PARA O DESENVOLVIMENTO DO TRABALHO.

1.4 Estrutura da Monografia

DESCREVER COMO O DOCUMENTO ESTA ESTRUTURADO.

2 Referencial Teórico

"Este capítulo tem como objetivo apresentar o referencial teórico que embasa a pesquisa deste trabalho, incluindo conceitos abordados"

2.1 Section 1 - Titulo

2.2 Engenharia de Software

2.2.1 Conceito de ESW adotados

3 A PROPOSTA

Este capítulo apresenta a proposta do trabalho de conclusão de curso, exibindo detalhes da implementação a ser realizada acerca da arquitetura bem como o protocolo de comunicação dentro desta.

3.1 Introdução

Avanços tecnológicos e a criação de linguagens de programação, diferentes técnicas e paradigmas e outros conceitos relacionados ao desenvolvimento de software contribui para que a necessidade de interação entre estes elementos seja emergente, de modo a viabilizar a construção de sistemas cada vez mais robustos e inteligentes. Esta interação entre elementos de software não consistem de aplicações robustas que executam todas as suas atividades de forma independente de outras aplicações, pois cada vez mais diversos sistemas de software interagem com outros, podendo ser estes desenvolvidos tomando como base outros paradigmas ou escritos em outras linguagens de programação e utilizando-se diferentes técnicas.

A fim de suprir esta necessidade de interação entre sistemas diversos, foi criado um modelo arquitetural conhecido como Arquitetura Baseada em Serviços (ou *Service-Oriented Architecture* - SOA). Este modelo arquitetural utiliza o conceito de serviço como uma unidade que representa uma funcionalidade do sistema, além de trazer consigo os conceitos de interoperabilidade, flexibilidade, extensibilidade e baixo acoplamento entre os diversos sistemas ou serviços.

Para este trabalho de conclusão de curso, a proposta consiste em desenvolver uma arquitetura baseada no modelo SOA para um ambiente virtual, propiciando que diversos trabalhos já desenvolvidos e também aqueles em desenvolvimento não sejam mais "engavetados". Por meio do uso do modelo arquitetural baseado em serviços, será possível integrar tais aplicações, ou serviços, de modo que estas possam trocar dados e fazer uso do serviço disponibilizado por outras, indepentemente das tecnologias utilizadas para o desenvolvimento das mesmas.

Também faz parte da proposta, o estabelecimento de um protocolo de comunicação entre as aplicações, bem como o padrão de comunicação a ser utilizado, uma vez que as aplicações produzidas por outros podem se comunicar de modo a se tornarem mais robustas e completas enquanto ferramentas.

Desta forma, será viável a disponibilização à sociedade, interna e externa à Universidade, de uma plataforma virtual que conterá os trabalhos realizados dentro da mesma,

sejam oriundos de projetos de conclusão de curso, disciplinas ou projetos de extensão e pesquisa.

3.2 Proposta de arquitetura

3.2.1 Requisitos

A partir da necessidade identificada de disponibilizar aplicações que foram desenvolvidas, bem como aquelas que estão em desenvolvimento e que serão desenvolvidas, através de uma plataforma virtual único, algumas das principais características arquiteturais deste ambiente que influenciaram na escolha do modelo arquitetural para a construção da plataforma são:

- A comunicação entre as aplicações deve permitir a troca de dados independentemente das tecnologias utilizadas para seu desenvolvimento.
- O acoplamento entre aplicações deve ser o mínimo possível.
- Extensibilidade, permitindo que novas aplicações/componentes sejam inseridas à plataforma.
- Escalabilidade, fornecendo suporte para que diversas aplicações (ou componentes) sejam aderidas à plataforma.
- Flexibilidade, possibilitando a extensão da plataform sem que a arquitetura original seja modificada drasticamente.

A partir das características acima, foi proposto o uso do modelo arquitetural SOA - *Service-Oriented Architecture* -, pois desta forma a plataforma virtual terá conhecimento sobre as aplicações por meio das interfaces disponibilizadas, mas não precisará ter conhecimento sobre como ou quais tecnologias foram utilizadas para o desenvolvimento das aplicações. As aplicações neste contexto também podem ser denominadas serviços ou funcionalidades da plataforma virtual.

3.2.2 A arquitetura

A proposta de arquitetura a ser implementada faz uso da abordagem de implementação de SOA chamada "*Hub-and-spoke*", onde a interface de comunicação entre os serviços é única e pode ser realizada com o uso de um barrameneto de serviços ou um Enterprise Service Bus (ESB).

O barramento de serviços é um recurso a ser utilizado na implementação da arquitetura baseada no modelo SOA para facilitar a troca entre mensagens entre as aplicações

- ou serviços. Este barramento é uma ferramenta que implementa funcionalidades que roteiam as mensagens entre os usuários e provedores de um determinado serviço, transformam as mensagens e os dados para o formato aceito pelas aplicações e aceita protocolos múltiplos de comunicação através de adaptadores. Na arquitetura proposta, o protocolo de comunicação será padronizado e unificado e apenas as outras características do barramento de serviços serão utilizadas.

Sendo interoperabilidade um dos requisitos relevantes para a escolha do modelo arquitetural, o barramento de serviços é visto como um recurso que pode ser utilizado para ajudar a promover a interoperabilidade na arquitetura definida e na validação de políticas e critérios de segurança a serem definidas em trabalhos posteriores.

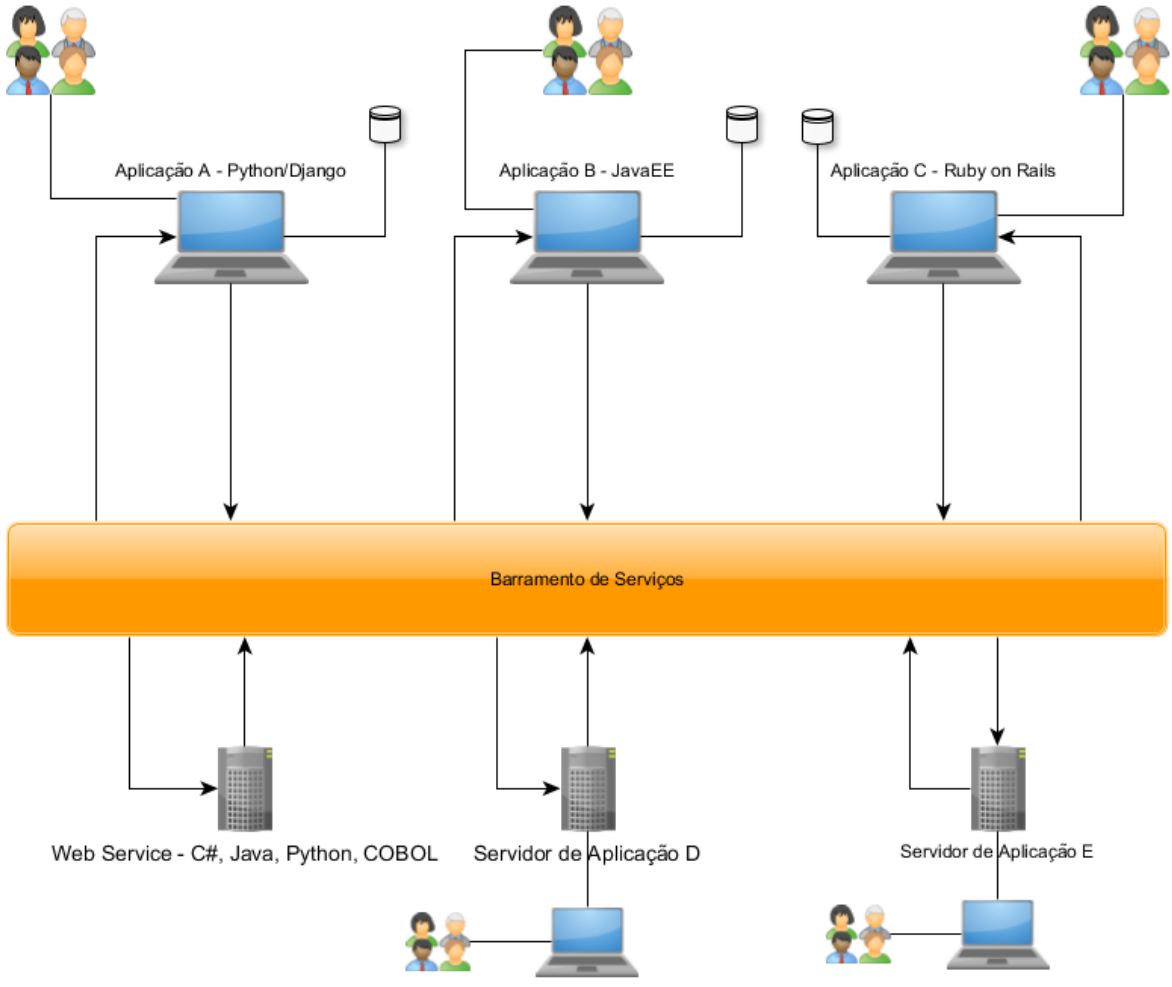

Figura 1: Interoperabilidade em uma arquitetura baseada no modelo SOA.

A figura acima visa demonstrar a interoperabilidade em uma arquitetura baseada no modelo SOA: é possível inferir que as diversas aplicações fazem a requisição dos serviços disponíveis por meio do uso do barramento de serviços, que também pode ser interpretado como um barramento de aplicações. As aplicações podem ser desenvolvidas utilizando-se tecnologias e paradigmas distintos e a troca de dados entre elas se darão de

forma bidirecional via mensagens de requisição e resposta entre as aplicações usuário, que requisitam operações dos serviços, e os serviços, responsáveis por processar as requisições e fornecer a resposta correspondente.

A figura também exibe um fato interessante na arquitetura proposta: não é necessário que um serviço seja apenas um serviço hospedado em um servidor. As aplicações poderão operar tanto em modo *standalone*, sendo executadas de forma independente dos outros serviços ou aplicações, quanto como um serviço para a plataforma virtual ou para outras aplicações que tenham conhecimento da existência e do protocolo de uso deste serviço.

O ESB é uma ferramenta que fornece as funcionalidades do barramento de serviços. Seu uso irá garantir que, sempre que este estiver ativo, as requisições que serão realizadas sempre terão uma resposta, mesmo sendo algo que indique a inatividade do serviço requerido ou a não autorização para acesso à este. Ao adicionar um novo serviço à arquitetura utilizando o ESB, deverão ser especificados tanto os procedimentos quando uma nova requisição for feita quanto aqueles passos que conduzem ao envio de uma resposta de acesso ao serviço especificado, sendo esta resposta advinda do próprio serviço ou uma resposta padrão, também especificada, em caso de falha ou sucesso ao executar a requisição.

Com base nos requisitos essenciais levantados e no estudo realizado sobre o modelo arquitetural SOA, o modelo proposto pode ser visto na imagem abaixo:

A comunicação entre aplicações e serviços deverão seguir o padrão de protocolo também definido durante o desenvolvimento deste trabalho de conclusão de curso, para que seja mantida uma regra de execução na troca de informações. O protocolo também facilitará a adição de um novo serviço à arquitetura no que diz respeito aos procedimentos de transformação dos dados e adaptação entre tecnologias e protocolos de transporte e comunicação adotados.

3.2.2.1 Ferramentas ESB

Existem diversas ferramentas deste tipo disponíveis e em uso por grandes organizações, tais como JBoss ESB, Mule ESB, Zato e WSO2 ESB. Para o conhecimento sobre a viabilidade de execução do trabalho aqui proposto, algumas destas ferramentas já foram levantadas, e, sendo o ESB um elemento importante para a implementação aqui proposta, uma análise prévia destas ferramentas já foi realizada. O levantamento feito mostrou que as ferramentas que podem ser utilizadas para a implementação da arquitetura são o JBoss ESB, WSO2 ESB e ErlangMS. Os critérios utilizados foram:

- Ser uma ferramenta de código aberto e/ou *free*;

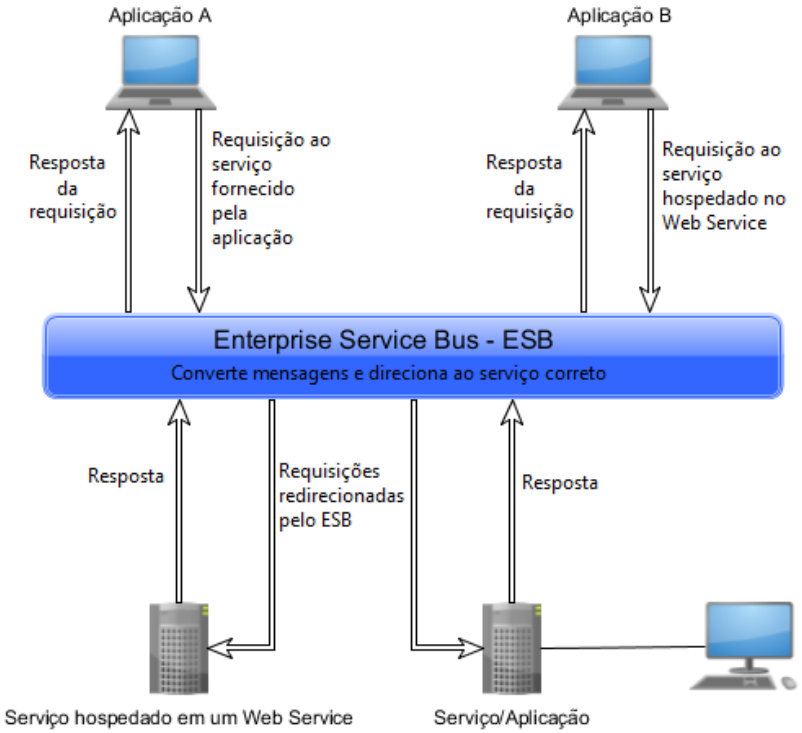

Figura 2: Proposta da arquitetura baseada no modelo SOA com o uso de um ESB.

- Possuir documentação e tutoriais disponíveis;
- Facilidade para implantação;
- Facilidade para uso;
- Possibilida de de uso de conectores (customizados e existentes);
- Suporte ao formato de mensagem escolhido para a implementação do protocolo.

3.2.3 Protocolo de comunicação

O protocolo de comunicação proposto para que ocorra a troca de dados entre usuários e provedores de serviço é um protocolo geral a ser utilizado por estes e adaptados sempre que necessário, porém não fugindo dos padrões estabelecidos. O protocolo proposto utiliza o modelo REST e, portanto, deve ser aderente à arquitetura deste. Este protocolo foi escolhido por ser de fácil uso, podendo ser implementado em diversas linguagens de programação (principalmente aquelas que são destinadas ao desenvolvimento de plataformas para a web) e em diversos sistemas operacionais. Além disso, o modelo REST utiliza o HTTP como protocolo de transporte, contribuindo para que a comunicação entre diversas aplicações seja realizada de maneira mais estável.

A arquitetura do protocolo REST aceita alguns formatos tais como JSON, CSV e texto simples, como mencionado nos capítulos anteriores. O formato definido para uso neste protocolo é o JSON, por ser um formato que permite a composição da mensagem através de chaves e valores. Assim, quando de posse da mensagem, os valores podem extraídos de acordo com a chave. Estas informações deverão ser publicadas pelo serviço na especificação de suas operações e valores de retorno quando forem acessados.

A fim de permitir o acesso ao serviço, as aplicações devem disponibilizar uma API REST para que os recursos sejam manipulados através das operações. Alguns serviços podem necessitar de autenticação e autorização para acesso ao mesmo, porém outros estarão disponpiveis para uso sem a necessidade de que tais recuros de segurança sejam implementados. Os requisitos de segurança na troca de dados deverão ser planejados em trabalhos futuros.

Desta forma, o fluxo básico do protocolo de comunicação entre os provedores e usuários dos serviços pode ser visto na imagem abaixo:

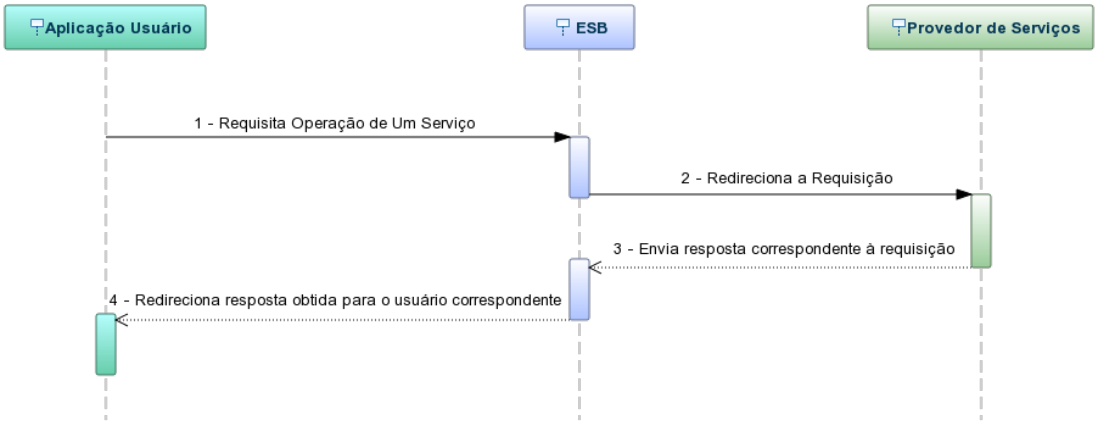

Figura 3: Fluxo básico do protocolo de comunicação.

A imagem exibe o fluxo geral do protocolo de comunicação estabelecido entre o usuário e o provedor de serviços, sendo esta comunicação intermediada pelo ESB: a aplicação que faz uso do serviço requisita que uma operação disponibilizada pelo serviço seja executada, esta requisição, por sua vez, é tratada pelo ESB e redirecionada à aplicação provedora do serviço; a resposta da requisição feita também será intermediada pelo ESB e redirecionada à aplicação que realizou a requisição.

O ESB é o elemento da comunicação que detêm o conhecimento sobre os servidos providos na plataforma e é o ator responsável por realizar a entrega das mensagens de requisição e de resposta dos serviços. Caso necessário, é também uma responsabilidade do ESB, realizar a transformação/adaptação dos formatos das mensagens e dos protocolos usados.

3.2.3.1 Formato das Mensagens

Como anteriormente mencionado, o formato escolhido para a troca de mensagens entre as aplicações será o JSON. A escolha foi realizada pelo fato de este formato de mensagem ser leve, de fácil entendimento e implementação, além de permitir que não seja difícil a recuperação dos dados em qualquer linguagem de programação, bem como pelo ESB.

O formato JSON é baseado em um esquema de chave-valor, onde a chave identifica um atributo, um dado, e o valor é o dado em si, o valor quantitativo ou qualitativo do atributo indicado pela chave. Este formato de mensagem adotado, pode ser tratado na aplicação como uma mensagem JSON ou como uma cadeia de caracteres, a depender da linguagem de programação adotada na construção da plataforma virtual e dos serviços disponibilizados.

Assim, para realizar uma requisição, a aplicação que executa o papel de usuário de um serviço deverá indicar o serviço e a operação desejada, o formato da mensagem (JSON por padrão) e os valores necessários para que o serviço seja executado corretamente, indicados pela API disponibilizada pelo mesmo. Da mesma forma, a resposta também deverá ser gerada em formato JSON, porém a aplicação que prover serviços retorna apenas a resposta da mensagem no padrão chave-valor. Abaixo podem ser visualizados um exemplo de requisição e outro de reposta, ambos em formato JSON.

```
url = "http://localhost:8280/services/facebookConnector"
headers = {'Action': 'urn:getUserDetails',
           'Content-type':'application/json'}
payload = {'apiUrl':'https://graph.facebook.com','apiVersion':'v2.5',
           'accessToken':access_token,
           'fields':'id,name,email,age_range,birthday'}
```

Figura 4: Exemplo de uma mensagem de requisição de serviço em formato JSON de uma aplicação usuário escrita em Python e Django.

```
{'id': 'user_id', 'name':'user_name',  'email':'user@email.com',
 'age_range':'20-25', 'birthday':'user_birthday'}
```

Figura 5: Exemplo de uma mensagem de resposta de requisição em formato JSON.

A primeira imagem exibe os valores necessários para realizar uma requisição ao serviço fornecido pela rede social Facebook através de sua API. Para a chamada do serviço, são necessários a especificação do endereço do serviço, indicado pela *url*; *headers* guarda os valores da operação a ser executada pelo serviço (’*Action*’) e o formato da mensagem (’*Content-type*’); os valores necessários para a execução da operação requisitada estão contidos no *payload* também em formato *’chave’:’valor’*.

As imagens acima mostram apenas exemplos do uso do formato de mensagem JSON para realizar uma requisição e de mensagem obtida como resposta advinda do serviço. Pode-se ver que os valores são correspondentes à uma chave conhecida por ambas as aplicações, permitindo que as aplicações (provedora e usuária de serviços) possam comunicar-se entre si de forma padronizada e conhecida por ambas as partes.

4 Desenvolvimento da Proposta

Este capítulo apresenta o uso de metodologias e conceitos relacionados à Engenharia de Software que serão aplicados no desenvolvimento da proposta descrita no capítulo anterior.

4.1 Introdução

Um projeto de Engenharia de Software deve ser realizado utilizando-se de metodologias e técnicas relacionadas à esta área de conhecimento, sendo sempre adaptadas de acordo com o projeto a ser desenvolvido e podendo ser modificado quando necessário, visando o sucesso na conclusão do projeto.

O projeto de Engenharia de Software a ser desenvolvido como parte do trabalho de conclusão de curso consiste em uma arquitetura para uma plataforma virtual onde as funcionalidades serão tratadas como serviços no contexto arquitetural. Os serviços ou funcionalidades desta plataforma virtual consiste em trabalhos já realizados, em processo de desenvolvimento e também de trabalhos futuros que, quando incorporados a esta arquitetura, poderão ser disponibilizados para uso pela comunidade tanto acadêmica quando externa, deixando que ser "projetos de gaveta".

A proposta feita, que foi descrita no capítulo anterior, será desenvolvida utilizando-se de alguns conceitos de metodologias de desenvolvimento e de gerenciamento de projetos de software, adaptadas conforme as necessidades deste projeto.

4.2 Metodologia de desenvolvimento

Dentre as diversas metodologias de desenvolvimento conhecidas na Engenharia de Software, foi decidido pela elaboração de uma metodologia que utiliza as fases do RUP (iniciação, elaboração, construção e transição) e algumas das atividades também deste método de desenvolvimento de software. Além disso, o método adotado terá um ciclo de vida incremental, onde partes da arquitetura serão desenvolvidas e incrementadas a cada ciclo.

Por se tratar de um projeto mais técnico do que funcional, a adoção de uma outra metodologia de desenvolvimento, como a metodologia ágil, tornou-se complexa para a identificação de unidades funcionais unitárias, como histórias de usuário (um conceito da metodologia ágil utilizado para descrever as funcionalidades de um sistema de software), o estabelecimento de ciclos bem definidos e a distribuição das unidades funcionais neste

ciclo. Isto se deve ao fato de que o estabelecimento da arquitetura deve ser realizado em conjunto com o uso do protocolo de comunicação de acordo com o padrão estabelecido e isto deve ser realizado de maneira incremental, pois estas partes são dependentes uma da outra.

4.2.1 Fases de desenvolvimento

Adotando o RUP como metodologia de desenvolvimento, as fases a serem executadas são:

- *Iniciação*: fase de concepção do projeto, onde as necessidades são identificadas e a solução é proposta.
- *Elaboração*: fase em que é realizado o planejamento da solução proposta. Também são identificados os recursos necessários, tanto de pessoal quanto de infraestrutura física.
- *Construção*: fase em que os requisitos identificados são de fato desenvolvidos/construídos. Também nesta fase são realizados testes a fim de garantir a menor taxa de erros no sistema.
- *Transição*: fase em que o software é homologado e implantado no ambiente do cliente, dono do software.

Os ciclos de desenvolvimento adotados serão de forma incremental de maneira a adaptar parte de um sofwtare desenvolvido em trabalhos anteriores para que este seja incorporado a plataforma virtual como um serviço e a comunicação entre a plataforma virtual e este serviço esteja de acordo com o padrão de comunicação adotado. O objetivo principal desta iteração é garantir que exista um modelo prático a ser seguido quando novas aplicações forem desenvolvidas e incorporadas à plataforma virtual.

—DEFINIR QUANTIDADE DE CICLOS A SEREM RODADOS— —COLOCAR UMA FIGURA EXPLICANDO O CICLO DE DESENVOLVIMENTO— —O CICLO ADOTADO DEVERÁ CONTER UMA GRANDE FASE DE INICIAÇÃO E ELABORAÇÃO NO PRIMEIRO CICLO. O SEGUNDO, TERCEIRO E QUARTO CICLOS DEVERÃO SER DE 7 A 10 DIAS. NESTES CICLOS, A FASE DE INICIAÇÃO DEVE SER BEM PEQUENA, SÓ INDICANDO DE NOVAS NECESSIDADES PODEM SER IDENTIFICADAS; A FASE DE ELABORAÇÃO DEVE SER UM POUCO MAIOR QUE A INICIAÇÃO, INDICANDO O PLANEJAMENTO DO NOVO CICLO E A IDENTIFICAÇÃO DE NECESSIDADE DE NOVOS RECURSOS DE HARDWARE; A FASE DE CONSTRUÇÃO DEVE INDICAR IMPLEMENTAÇÃO DA PLATAFORMA, ADAPTAÇÃO DO SOFTWARE E TESTES. A IMPLANTAÇÃO DEVERÁ SER FEITA AO

FINAL DOS CICLOS, E TAMBÉM DEVE INDICAR A EXISTÊNCIA DE TESTES—
—

A imagem acima mostra o ciclo de desenvolvimento a ser adotado com base nas fases do modelo RUP. A imagem mostra as fases de iniciação, elaboração, construção e implantação a serem executadas durante o trabalho de conclusão de curso. A parte incremental é realizada durante a construção da arquitetura, onde pequenos ciclos são executados e o foco maior está na fase de construção.

A fase de construção consiste em ciclos de desenvolvimento onde os requisitos e necessidades serão revistos e implementados usando-se de tecnologias mais atuais e adequadas ao contexto. Este ciclo é exibido desta maneira na figura a fim de demonstrar que, embora as fases de iniciaão, elaboração e implantação tenham seus períodos definidos e maior parte dos trabalhos de identificação das necessidades, planejamento e homologação da arquitetura proposta sejam realizados em suas respectivas fases, estas tarefas também devem ser realizadas durante a construção, principalmente se o serviço que for incorporado à arquitetura tiver que ser adaptado para tal finalidade. Para que as aplicações sejam adaptadas, as fases indicadas devem ser executadas de maneira cíclica e incremental, de modo que as necessidades de adaptação sejam identificadas e depois implementadas. A implantação ao final de cada ciclo será um teste para verificar e validar a adequação realizada.

A execução destes ciclos menores se faz necessário visto que, como existem aplicações já desenvolvidas, um dos objetivos a serem alcançados ao final do TCC 2 é a adaptação e implantação de uma destas aplicações e torná-la um serviço dentro da plataforma virtual.

4.3 Cronograma de execução

Para o desenvolvimento da proposta e completude da mesma com sucesso foi criado um cronograma para a fase de prova de conceito e implementação da proposta aqui feita. O cronograma contém a descrição das atividades a serem realizadas, bem como os prazos relacionados a cada uma das atividades e pode ser visualizado na tabela abaixo.

Tabela 1: Cronograma de atividades relacionadas ao TCC 2

Atividade	Jul	Ago	Set	Out	Nov	Dez
Análise de ferramentas a serem utilizadas	X					
Implantação da ferramenta escolhida		X				
Adaptação de uma aplicação já desenvolvida			X	X		
Implementação do uso da ferramenta com a aplicação adaptada			X	X		
Escrita do TCC 2					X	X

— TEM QUE INDICAR NO CRONOGRAMA OU EXPLICAR EM FORMA DE TEXTO QUE OS CICLOS VÃO SER EXECUTADOS FURANTE AGOSTO E SETEMBRO (OU É SETEMBRO E OUTUBRO) E QUE A CARACTERIZAÇÃO DAS FERRAMENTAS esb FAZEM PARTE DA ELABORAÇÃO/INICIAÇÃO (VERIFICAR ISSO, CHECAR AS FASES DO RUP)—

As atividades realizadas durante o desenvolvimento do TCC 1 foram registradas e podem ser visualizadas na tabele seguinte.

Tabela 2: Cronograma de atividades relacionadas ao TCC 1

Atividade	Jan	Fev	Mar	Abr	Mai	Jun
Identificação da necessidade	X					
Leituras sobre modelos arquiteturais	X					
Pesquisa sobre o modelo arquitetural mais adequado	X	X				
Pesquisa sobre trabalhos relacionados	X	X				
Levantamento superficial de ferramentas ESB		X	X			
Organização das referências			X			
Escrita formal da proposta				X		
Estabelecimento de metodologia				X		
Escrita do TCC 1					X	X

5 Considerações finais

Referências

Adaptive Ltd et al. *Service oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS)*. OMG - Object Management Group, 2009. Disponível em: <http://www.uio.no/studier/emner/matnat/ifi/INF5120/v10/undervisningsmateriale/>. Nenhuma citação no texto.

BASS, L.; CLEMENTS, P. C.; KAZMAN, R. *Software Architecture in Practice*. 2. ed. [S.l.]: Addison-Wesley Professional, 2003. ISBN 0-321-15495-9. Nenhuma citação no texto.

BELQASMI, F. et al. SOAP-Based vs. RESTful Web Services: A Case Study for Multimedia Conferencing. *IEEE Internet Computing*, v. 16, n. 4, p. 54–63, jul. 2012. ISSN 1089-7801. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6197173>. Nenhuma citação no texto.

BIANCO, P.; KOTERMANSKI, R.; MERSON, P. *Evaluating a Service-Oriented Architecture*. [S.l.], 2007. Disponível em: <http://www.sei.cmu.edu/library/abstracts/reports/07tr015.cfm>. Nenhuma citação no texto.

BIANCO, P. et al. *Architecting Service-Oriented Systems*. Pittsburgh, PA, 2011. Disponível em: <http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9829>. Nenhuma citação no texto.

CHEN, H. M.; KAZMAN, R.; PERRY, O. From Software Architecture Analysis to Service Engineering: An Empirical Study of Methodology Development for Enterprise SOA Implementation. *IEEE Transactions on Services Computing*, v. 3, n. 2, p. 145–160, jun. 2010. ISSN 1939-1374. Disponível em: <http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=5467023>. Nenhuma citação no texto.

DAI, W. et al. Bridging Service-Oriented Architecture and IEC 61499 for Flexibility and Interoperability. *IEEE Transactions on Industrial Informatics*, v. 11, n. 3, p. 771–781, jun. 2015. ISSN 1551-3203, 1941-0050. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7086296>. Nenhuma citação no texto.

DIKMANS, L.; LUTTIKHUIZEN, R. v. *SOA Made Simple*. BIRMINGHAM - MUMBAI: Packt Publishing, 2012. ISBN 978-1-84968-416-3. Nenhuma citação no texto.

ERL, T. *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004. ISBN 0-13-142898-5. Nenhuma citação no texto.

ERL, T. *Service-Oriented Architecture: Concepts, Technology, and Design*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005. ISBN 0-13-185858-0. Nenhuma citação no texto.

ERL, T. Computação orientada a serviços e SOA. In: GAMA, F. C. N. d.; BARBOSA, R. d. C. (Ed.). *SOA: Princípios de Design de Serviços*. São Paulo: Pearson Prentice Hall, 2009. p. 306. ISBN 978-85-7605-189-3. Nenhuma citação no texto.

ERL, T. Orientação a serviços. In: GAMA, F. C. N. d.; BARBOSA, R. d. C. (Ed.). *SOA: Princípios de Design de Serviços*. São Paulo: Pearson Prentice Hall, 2009. p. 306. ISBN 978-85-7605-189-3. Nenhuma citação no texto.

ERL, T. *SOA: Princípios de Design de Serviços*. 1. ed. São Paulo: Pearson Prentice Hall, 2009. ISBN 978-85-7605-189-3. Nenhuma citação no texto.

ESFAHANI, F. S. et al. Adaptable Decentralized Service Oriented Architecture. *Journal of Systems and Software*, v. 84, n. 10, p. 1591–1617, out. 2011. ISSN 01641212. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0164121211000744>. Nenhuma citação no texto.

EVDEMON, J. *Princípios do design de serviço: padrões e antipadrões de serviço*. 2005. Disponível em: <https://msdn.microsoft.com/pt-br/library/ms954638.aspx>. Nenhuma citação no texto.

FERREIRA, A. Apresentação Prezi, *SOA vs ESB*. 2013. Disponível em: <https://prezi.com/na8_yc87is6s/soa-vs-esb/>. Nenhuma citação no texto.

JANSEN, A.; BOSCH, J. Software Architecture as a Set of Architectural Design Decisions. In: . IEEE, 2005. p. 109–120. ISBN 978-0-7695-2548-8. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1620096>. Nenhuma citação no texto.

JúNIOR, D.; VIEIRA, M. *Uma arquitetura para aprendizagem colaborativa utilizando a integração web e TV digital integrativa*. Tese (text) — Universidade Federal de Alagoas, set. 2012. Disponível em: <http://www.repositorio.ufal.br/handle/riufal/849>. Nenhuma citação no texto.

JOSUTTIS, N. *Soa in Practice: The Art of Distributed System Design*. [S.l.]: O'Reilly Media, Inc., 2007. ISBN 0-596-52955-4. Nenhuma citação no texto.

KAZMAN, R. The essential components of software architecture design and analysis. In: . IEEE, 2005. p. 1 pp. ISBN 978-0-7695-2465-8. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1607129>. Nenhuma citação no texto.

LEWIS, G. *Getting Started with Service-Oriented Architecture (SOA) Terminology*. Software Engineering Institute Carnegie Mellon University, 2010. Disponível em: <http://www.sei.cmu.edu/library/assets/whitepapers/SOA_Terminology.pdf>. Nenhuma citação no texto.

LINTHICUM, D. et al. *Service Oriented Architecture (SOA) in the Real World*. [S.l.]: Microsoft Corporation, 2007. Nenhuma citação no texto.

LóPEZ-SANZ, M. et al. Modelling of Service-Oriented Architectures with UML. *Electronic Notes in Theoretical Computer Science*, v. 194, n. 4, p. 23–37, abr. 2008. ISSN 15710661. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S1571066108002028>. Nenhuma citação no texto.

MALLOY, B. et al. Improving the predictable assembly of service-oriented architectures. *IEEE Software*, v. 23, n. 2, p. 12–15, mar. 2006. ISSN 0740-7459. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1605172>. Nenhuma citação no texto.

MUELLER, J. *Understanding SOAP and REST Basics And Differences*. 2013. Disponível em: <http://blog.smartbear.com/apis/understanding-soap-and-rest-basics/>. Nenhuma citação no texto.

NICKULL, D. et al. *Service Oriented Architecture (SOA) and Specialized Messaging Patterns*. San Jose, CA, USA, 2007. Nenhuma citação no texto.

O que é SOA e por que usá-la? 2010. Disponível em: <http://www.celtainformatica.com.br/noticias/o-que-e-soa-e-por-que-usa-la>. Nenhuma citação no texto.

OLIVEIRA, M.; NAVARRO, R. Interoperabilidade em SOA: Desafios e Padrões. *SOA na prática*. Disponível em: <http://www.univale.com.br/unisite/mundo-j/artigos/37Interoperabilidade.pdf>. Nenhuma citação no texto.

ONOE, A. Y. *Proposta de governança SOA utilizando capacidades dinâmicas: uma aplicação em centro de comunicação digital universitário*. Tese (text) — Universidade de São Paulo, nov. 2010. Disponível em: <http://www.teses.usp.br/teses/disponiveis/3/3141/tde-19012011-095120/>. Nenhuma citação no texto.

PARK, S.; CHOI, J.; YOO, H. Integrated Model of Service-Oriented Architecture and Web-Oriented Architecture for Financial Software. *Journal of Information Science and Engineering*, v. 28, n. 5, p. 925–939, 2012. Disponível em: <http://www.iis.sinica.edu.tw/page/jise/2012/201209_07.pdf>. Nenhuma citação no texto.

PEREDO, R.; PEREDO, I. Software Architecture and Design with Semantic Web Technologies in Virtual Learning Environments. *Issues in Information Systems - A Journal of IACIS*, v. 15, n. 2, p. 190–196, 2014. Disponível em: <http://iacis.org/iis/2014/111_iis_2014_190-196.pdf>. Nenhuma citação no texto.

PEREIRA, M. Z. *PSOA: um framework de práticas e padrões SOA para projetos DDS*. Tese (masterThesis) — Pontifícia Universidade Católica do Rio Grande do Sul, Rio Grande do Sul, 2011. Disponível em: <http://hdl.handle.net/10923/1658>. Nenhuma citação no texto.

ROSCHELLE, J.; KAPUT, J. Educational Software Architecture And Systemic Impact: The Promise Of Component Software. *Journal of Educational Computing Research*, v. 14, n. 3, p. 217–228, 1996. Disponível em: <https://www.sri.com/sites/default/files/publications/imports/EdSoftwareArch.pdf>. Nenhuma citação no texto.

ROZLOG, M. *REST e SOAP: Usar um dos dois ou ambos?* 2013. Disponível em: <http://www.infoq.com/br/articles/rest-soap-when-to-use-each>. Nenhuma citação no texto.

SANGWAN, R. et al. Integrating a software architecture-centric method into object-oriented analysis and design. *Journal of Systems and Software*, v. 81, n. 5, p. 727–746, maio 2008. ISSN 01641212. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0164121207001872>. Nenhuma citação no texto.

SILVA, E. *Introdução aos Conceitos do WSO2 ESB*. 2014. Disponível em: <http://pt.slideshare.net/edgarsilva/introduo-aos-conceitos-do-wso2-esb>. Nenhuma citação no texto.

SIRIWARDENA, P. *Enterprise Integration with WSO2 ESB*. 1. ed. BIRMINGHAM - MUMBAI: Packt Publishing, 2013. ISBN 978-1-78328-019-3. Nenhuma citação no texto.

Software Engineering Institute - Carnegie Mellon University. *Software Architecture*. Disponível em: <http://www.sei.cmu.edu/architecture/>. Nenhuma citação no texto.

STEENDEREN, M. v.; DYK, P. v. Standard Object Access Protocol (SOAP). *InterWord Communications - South African Journal of Information Management*, v. 2, n. 2/3, set. 2000. ISSN 1560-683X. Disponível em: <http://www.sajim.co.za/index.php/SAJIM/article/viewFile/106/103>. Nenhuma citação no texto.

VANTAGENS e Desvantagens de SOA. 2013. Disponível em: <http://www.devmedia.com.br/vantagens-e-desvantagens-de-soa/27437>. Nenhuma citação no texto.

WSO2 Enterprise Service Bus. Disponível em: <http://wso2.com/products/enterprise-service-bus/>. Nenhuma citação no texto.