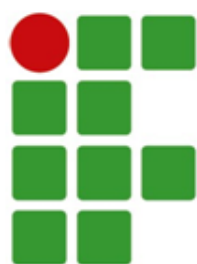


Instituto Federal de Educação, Ciência e Tecnologia de Goiás
Câmpus Inhumas
Bacharelado em Sistemas de Informação

**Proposta de uma arquitetura de sistema de
sistemas de informação (SoIS) para gestão de
dados acadêmicos**



INSTITUTO FEDERAL
Goiás

Câmpus
Inhumas

Autor: Carlos Eduardo da Rocha
Orientador: (Prof. Me. Victor Hugo Lázaro Lopes)

Inhumas, GO
2021



INSTITUTO FEDERAL
Goiás

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
SISTEMA INTEGRADO DE BIBLIOTECAS

TERMO DE AUTORIZAÇÃO PARA DISPONIBILIZAÇÃO NO REPOSITÓRIO DIGITAL DO IFG - ReDi IFG

Com base no disposto na Lei Federal nº 9.610/98, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia de Goiás, a disponibilizar gratuitamente o documento no Repositório Digital (ReDi IFG), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, em formato digital para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IFG.

Identificação da Produção Técnico-Científica

- | | |
|--|---|
| <input type="checkbox"/> Tese | <input type="checkbox"/> Artigo Científico |
| <input type="checkbox"/> Dissertação | <input type="checkbox"/> Capítulo de Livro |
| <input type="checkbox"/> Monografia – Especialização | <input type="checkbox"/> Livro |
| <input checked="" type="checkbox"/> TCC - Graduação | <input type="checkbox"/> Trabalho Apresentado em Evento |
| <input type="checkbox"/> Produto Técnico e Educacional - Tipo: _____ | |

Nome Completo do Autor: Carlos Eduarodo da Rocha

Matrícula: 20161030090072

Título do Trabalho: Proposta de uma arquitetura de sistema de sistemas de informação (SoIS) para gestão de dados acadêmicos

Autorização - Marque uma das opções

1. (x) Autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso aberto);
2. () Autorizo disponibilizar meu trabalho no Repositório Digital do IFG somente após a data ____/____/____ (Embargo);
3. () Não autorizo disponibilizar meu trabalho no Repositório Digital do IFG (acesso restrito).

Ao indicar a opção **2 ou 3**, marque a justificativa:

- () O documento está sujeito a registro de patente.
() O documento pode vir a ser publicado como livro, capítulo de livro ou artigo.
() Outra justificativa: _____

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O/A referido/a autor/a declara que:

- i. o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;
- ii. obteve autorização de quaisquer materiais incluídos no documento do qual não detém os direitos de autor/a, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia de Goiás os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;
- iii. cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia de Goiás.


Local

Data

Inhumas

16/06/2021.

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
SISTEMA INTEGRADO DE BIBLIOTECAS

A handwritten signature in dark ink, reading "Carlos Eduardo da Rocha", is written over a horizontal line.

Assinatura do Autor e/ou Detentor dos Direitos Autorais

Carlos Eduardo da Rocha

Proposta de uma arquitetura de sistema de sistemas de informação (SoIS) para gestão de dados acadêmicos

Monografia submetida ao curso de graduação em Bacharelado em Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Câmpus Inhumas, como requisito parcial para obtenção do Título de Bacharel em Sistemas de Informação.

Instituto Federal de Educação, Ciência e Tecnologia de Goiás
Câmpus Inhumas

Orientador: (Prof. Me. Victor Hugo Lázaro Lopes)

Inhumas, GO

2021

Dados Internacionais de Catalogação na Publicação

Rocha, Carlos Eduardo da

R672 Proposta de uma arquitetura de sistema de sistemas de informação (SolS) para gestão de dados acadêmicos [Manuscrito]. / Carlos Eduardo da Rocha – Inhumas: IFG, 2021.

85 f.

Bibliografia.

Orientador: Prof. Me. Victor Hugo Lázaro Lopes

Trabalho de conclusão de curso de graduação em Bacharelado em Sistemas de Informação – IFG/Câmpus Inhumas, 2021.

1. System of Systems 2. System of Information Systems 3. Interoperabilidade. 4. Lopes, Victor Hugo Lázaro (orientador). I. Título.

CDD 371.9

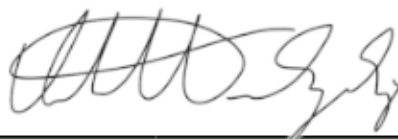
Ficha catalográfica elaborada pela bibliotecária
Larissa Stefane Rodrigues de Lima - CRB/1-3424
Instituto Federal de Educação, Ciência e Tecnologia de Goiás
Câmpus Inhumas – Biblioteca Atena

Carlos Eduardo da Rocha

Proposta de uma arquitetura de sistema de sistemas de informação (SoIS) para gestão de dados acadêmicos

Monografia submetida ao curso de graduação em Bacharelado em Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Câmpus Inhumas, como requisito parcial para obtenção do Título de Bacharel em Sistemas de Informação.

Trabalho —. Inhumas, GO, 08 de junho de 2021:



(Prof. Me. Victor Hugo Lázaro Lopes)
IFG-GO / Campus Inhumas
Orientador



Prof. Me. Ricardo Rodrigues Dias de Lima
IFG-GO / Campus Inhumas



Prof. Me. Ciro José Almeida Macedo
IFG-GO / Campus Cidade de Goiás

Inhumas, GO
2021

Este trabalho é dedicado aos amantes da computação.

Agradecimentos

A Deus por ter me dado saúde e força para superar as dificuldades. A esta instituição, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbro um horizonte superior. Ao meu orientador, pelo suporte necessário ao desenvolvimento deste trabalho, pelas suas correções e incentivos. Aos meus pais e minha irmã, pelo amor, incentivo e apoio incondicional. E a todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

Resumo

Neste trabalho é apresentado uma proposta de arquitetura baseada em sistema de sistemas de informação, SoIS, do inglês *System of Information Systems*, para gestão de dados acadêmicos. O objetivo é propor esta arquitetura e contribuir com o avanço nesta área, pois a mesma é recente, mostrando as facilidades e dificuldades, formas de possíveis implementações, para a aplicação ou construção do aplicativo. Desta forma é apresentado, desenhado e explicado, uma arquitetura baseada em SoIS, antes desta, foram apresentadas outras arquiteturas, que auxiliam nas definições e se comparam com SoIS. Como resultado foi apresentado, além da arquitetura, as telas do protótipo e um sistema que foi desenvolvido durante o andamento deste trabalho, para gestão de dados acadêmicos para uso da coordenação de curso da instituição, o SisCoord.

Palavras-chaves: *System of Systems*, *System of Information Systems*, interoperabilidade, integração sistêmica, dados acadêmicos.

Abstract

This work presents an architecture proposal based on system of information systems (SoIS) for academic data management. The objective is to propose this architecture and contribute to the advancement in this area, as it is recent, showing the facilities and difficulties, ways of possible implementations, for the application or construction of the application. In this way, an architecture based on SoIS is presented, designed and explained, before this, other architectures were presented, which help in the definitions and compare with SoIS. As a result, in addition to the architecture, the prototype screens and a system that was developed during the course of this work were presented, for the management of academic data for the use of the institution's course coordination, the SisCoord.

Key-words: System of Systems, System of Information Systems, interoperability, systemic integration, academic data.

Lista de ilustrações

Figura 1 – Elementos de uma arquitetura, baseado em MSDN (MICROSOFT, 2019)	30
Figura 2 – Arquitetura MVC (GORLA; FOSCHINI, 2014)	30
Figura 3 – Arquiteturas, cliente servidor e <i>peer to peer</i> (CECIN, 2005)	31
Figura 4 – Exemplo, para descrição de um SOA	32
Figura 5 – Arquitetura baseada em SoIS (SALEH; ABEL, 2016a)	34
Figura 6 – Arquitetura proposta	52
Figura 7 – Modelo conceitual	55
Figura 8 – Modelo Lógico	55
Figura 9 – Tela de <i>login</i> do orquestrador de sistema	57
Figura 10 – Tela de cadastro de um novo sistema constituinte	57
Figura 11 – Tela de pesquisa de recurso	58
Figura 12 – Tela de pesquisa de recurso por sistema	58
Figura 13 – Tela de boas-vindas, no dispositivo móvel, após o <i>login</i>	59
Figura 14 – Tela de boas-vindas, no <i>laptop</i> , após o <i>login</i>	60
Figura 15 – Tela de gestão de curso	60
Figura 16 – Tela de gestão de setores	61
Figura 17 – Tela de gestão de documentos	61

Lista de quadros

2.1	Estrutura de um documento WSDL padrão	35
2.2	Algoritmo para acesso a recursos da web	39
2.3	Algoritmo para realização de <i>login</i>	40
4.1	Exemplo de descritor para um serviço via API	53
4.2	Modelo físico	55
A.1	Conector para o sistema SisCoord	73
B.1	Exemplo de um descritor para um serviço web não padrão	77

Lista de tabelas

Tabela 1 – Navegadores com suporte oficial pelo <i>Selenium</i>	38
---	----

Lista de abreviaturas e siglas

API	<i>Application Programming Interface</i>
DOM	<i>Document Object Model</i>
JSON	Notação de Objetos JavaScript
MVC	Modelo Visão Controlador
P2P	<i>Peer to Peer</i>
REST	<i>Representational State Transfer</i>
SGBD	Sistema de Gerenciamento de Banco de Dados
SIG	Sistema de Informações Gerenciais
SOA	Arquitetura Orientada a Serviços
SOAP	<i>Simple Object Access Protocol</i>
SoIS	Sistema de Sistemas de Informação
SoS	Sistema de Sistemas
TI	Tecnologia da Informação
UML	Linguagem de Modelagem Unificada
W3C	<i>World Wide Web Consortium</i>
WSDL	Linguagem de Descrição de Serviços Web
XML	Linguagem Extensível de Marcação Genérica

Sumário

I	INTRODUÇÃO	23
1	INTRODUÇÃO	25
1.1	Motivação	25
1.2	Justificativa	25
1.3	Objetivos	26
1.3.1	Objetivo Geral	26
1.3.2	Objetivos Específicos	26
II	REFERENCIAL TEÓRICO	27
2	APRESENTAÇÃO	29
2.1	Arquitetura de software	29
2.1.1	Modelo Visão Controlador (MVC)	30
2.1.2	Cliente servidor e <i>Peer to Peer</i>	31
2.1.3	Arquitetura Orientada a Serviços	31
2.1.4	Microserviços	32
2.2	<i>System of Information Systems</i>	33
2.3	Linguagem de descrição WSDL	35
2.4	<i>Selenium</i>	38
2.4.1	Exemplos	39
2.5	Trabalhos relacionados	42
2.5.1	SoS, SolS e SOA	42
2.5.2	Sistema de informação para gestão de dados acadêmicos	43
III	METODOLOGIA	45
3	METODOLOGIA APLICADA	47
IV	ARQUITETURA PROPOSTA E RESULTADOS	49
4	APRESENTAÇÃO	51
4.1	A arquitetura proposta	51
4.2	Protótipo em telas iniciais	56
4.3	Sistema SisCoord	59
4.4	Discussão	62

V	CONCLUSÃO	63
5	CONSIDERAÇÕES FINAIS	65
	REFERÊNCIAS	67
	APÊNDICES	71
	APÊNDICE A – O CONECTOR	73
	APÊNDICE B – EXEMPLO DE DESCRITOR PARA <i>SELENIUM</i> <i>WEB DRIVER</i>	77
	ANEXOS	79
	Artigo Secitec 2019	81

Parte I

Introdução

1 Introdução

Nas organizações temos que uma quantidade enorme de recursos heterogêneos são produzidos por vários usuários, que estão trabalhando em diferentes projetos e usando vários sistemas de informação, desta forma fazer uso de uma arquitetura baseada Sistema de Sistemas de Informação (SoIS), pode auxiliar e facilitar na gestão de dados. Neste trabalho, especificamente, propomos esta arquitetura para uma instituição pública de ensino, entretanto pode ser aplicada em outras organizações. Outro motivo é comum nestes cenários a existência de múltiplos sistemas de informação distintos, que naturalmente não possuem interoperabilidade, gerando complexidade às tarefas de gerenciamento dos dados acadêmicos da instituição.

1.1 Motivação

Contribuir com o avanço do conhecimento científico na área de Sistema de Sistemas (SoS, do inglês *System of Systems*) e SoIS, sendo que até o momento podemos considerar como uma tema na área de Tecnologia da Informação (TI) ainda pouco explorado, esse também é um dos desafios do trabalho, outro desafio e motivação desta proposta é garantir a interoperabilidade entre sistemas já existentes, tirando a necessidade de se ter projetos específicos para desenvolver, integrar ou implantar novos sistemas.

1.2 Justificativa

O tempo, recurso importante, pode ser otimizado, pelos docentes e técnicos administrativos, quando se trata de reunir e gerenciar informações de relevância em diferentes sistemas do meio acadêmico nas instituições públicas. Se economiza tempo, também economiza dinheiro. Assim, a proposta deste trabalho de uma arquitetura de sistema de sistemas de informação para gestão de dados acadêmicos visa à resolução de problemas ou questões multissetoriais, de forma centralizada, almejando uma gestão de forma menos complexa. Por vezes também temos alguns dados que já são existentes em sistemas prontos, se estes dados forem demanda para novos sistemas, com esta proposta, tais implementações podem ser economizadas.

Muitas vezes outros sistemas podem ser desenvolvidos para resolução de problemas específicos, em que o tema esteja envolvendo vários setores. Um dos problemas é que os recursos financeiros são limitados. Outro problema também são as licitações, a administração pública deve prestar contas e observar uma série de princípios e procedimentos previstos em lei. Desta forma, a centralização em um SoIS pode evitar que alguns siste-

mas sejam projetados e desenvolvidos, pois integrar um sistema constituinte em SoIS é de rápida adaptação.

1.3 Objetivos

1.3.1 Objetivo Geral

Propor uma arquitetura baseada em SoIS para ambientes acadêmicos para minimizar a complexidade e o tempo necessário de se capitalizar recursos de diferentes sistemas de informação.

1.3.2 Objetivos Específicos

- Comparar arquiteturas de sistemas de informação, com foco voltado para web;
- Propor os recursos para dar suporte à arquitetura proposta;
- Desenvolver e validar a arquitetura em um protótipo;
- Contribuir com o avanço do conhecimento científico na área de SoIS;
- Analisar e desenvolver um sistema para gestão de dados acadêmicos.

Parte II

Referencial Teórico

2 Apresentação

Neste capítulo são tratados grandes temas que convergem para a delimitação do escopo do trabalho. Dentre as temáticas estão um conjunto de arquiteturas, definições, que se relacionam com SoIS e contribui para o entendimento do mesmo, serão apresentados também trabalhos relacionados.

2.1 Arquitetura de software

Conceituando, representa um sistema em que existe um mapeamento de funcionalidade para componentes, hardware e software, e uma interação humana com esses, existem diversos conceitos ([D'ANUNCIAÇÃO, 2021](#)). Basicamente, envolve a descrição de elementos arquiteturais dos quais os sistemas serão construídos, interações entre esses elementos, paradigmas que guiam suas composições e restrições sobre estes padrões.

Especificando, a arquitetura de um software define em termos computacionais quais são seus elementos arquiteturais e como ocorre a interação entre eles. Um paradigma arquitetural como é caso de SoS ou SoIS, é ao padrão, de arquitetura de software, aplicado que está sendo referido.

Elementos arquiteturais: bancos de dados, servidores, clientes, um ou mais componentes, dentre outros, e a interação entre eles pode ocorrer através de chamadas de procedimentos, acesso a variáveis, uso de protocolos para acesso a clientes e servidores, bancos de dados, e outros eventos quaisquer.

Segundo ([SILVEIRA et al., 2012](#)) pag 141, arquitetura pode ser definida como uma possível interpretação da implementação do sistema. É quando olhamos para a aplicação para observar como diferentes partes do sistema afetam-se entre si. Essa visão mais global traz, porém, diversos desafios e indagações que demandam decisões difíceis que podem significar o sucesso ou o fracasso do projeto.

Desta forma, a arquitetura é também influenciada por fatores de implementação como arquitetura de computador, sistema operacional, Sistema de Gerenciamento de Banco de Dados (SGBD), protocolos de rede, linguagem de programação, ambiente de interface gráfica, bibliotecas de funções disponíveis, sistemas legados, necessidades de performance, portabilidade etc.

A Figura 1 apresenta as partes envolvidas em um projeto arquitetural de um software.

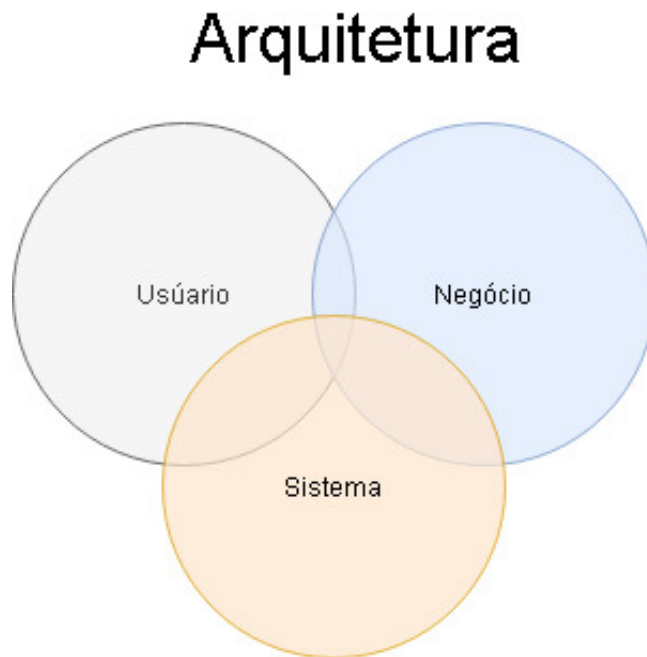


Figura 1 – Elementos de uma arquitetura, baseado em MSDN ([MICROSOFT, 2019](#))

Nesta ilustração temos o usuário, negócio e o sistema, cada um independente, mas também possuem relacionamentos entre si.

Nas subseções será demonstrado, soluções arquiteturais, exemplos de arquiteturas.

2.1.1 Modelo Visão Controlador (MVC)

MVC é um padrão de arquitetura de software, sua abordagem é composta de três tipos de objetos, modelo visão e controlador, antes deste padrão os projetos de interface com o usuário tendiam a agrupar esses objetos, desta forma o padrão separa estes objetos e aumenta a flexibilidade e a reutilização ([GAMMA, 2009](#)). Na Figura 2 os modelos de

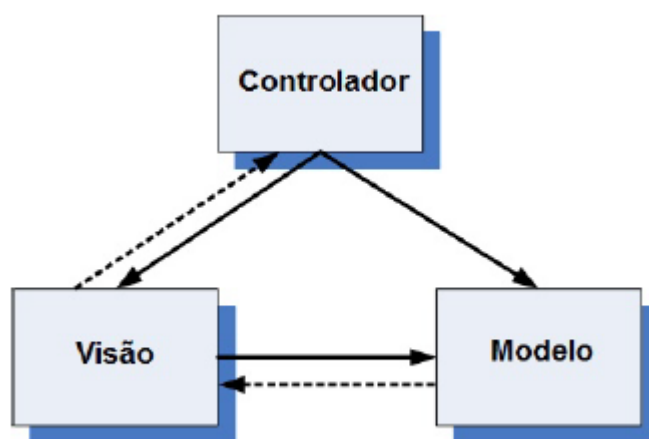


Figura 2 – Arquitetura MVC ([GORLA; FOSCHINI, 2014](#))

arquitetura que utilizam o padrão arquitetural MVC definem claramente a separação de responsabilidades e a comunicação entre os componentes de uma aplicação (TERUEL, 2012). A camada de visualização (*View*) é responsável pela interface com o usuário e controla as entradas e saídas gráficas e textuais. O modelo (*Model*) controla o comportamento e os dados do domínio da aplicação respondendo às solicitações e instruções para mudar seu estado, além de conter as regras de negócio. O controle (*Controller*) controla as solicitações do usuário repassando as mesmas para o modelo ou para a visualização, adequadamente. Esta divisão em camadas tem o objetivo de aumentar a flexibilidade e a reutilização do código (GAMA et al., 2000).

2.1.2 Cliente servidor e *Peer to Peer*

Na arquitetura cliente-servidor o processamento da informação é dividido em módulos ou processos distintos. Um processo é responsável pela manutenção da informação (servidores) e outros responsáveis pela obtenção dos dados (os clientes). No servidor ficam geralmente o banco de dados e outros sistemas mais “pesados”, pois os clientes são máquinas com hardware menos robusta que a do servidor. A arquitetura *peer to peer* (P2P), em português, par a par ou ponto a ponto, tem funcionalidade onde o nos da rede funciona tanto como cliente como servidor tirando a necessidade de um servidor central, esse tipo de arquitetura de rede é muito conhecida pelo compartilhamento de arquivos

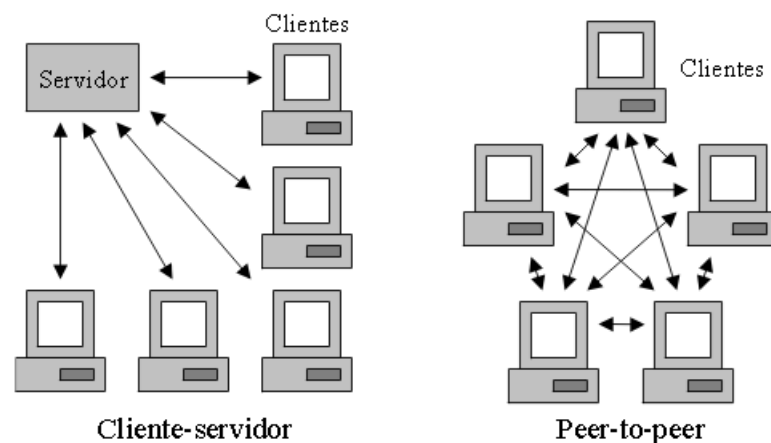


Figura 3 – Arquiteturas, cliente servidor e *peer to peer* (CECIN, 2005)

2.1.3 Arquitetura Orientada a Serviços

A arquitetura orientada a serviços (SOA) é o conjunto de procedimentos, técnicas ou padrões que são utilizados para construção do software, tais tem como principal a utilização de *web services*, que permite o baixo acoplamento da aplicação o que facilita sua manutenção. Os sistemas que fazem uso do *web services* podem ser construídos em diferentes linguagem o que eles precisam é conhecer como se faz uso do serviço, definindo

um serviço: este é um processo que pode ser utilizado por um software, este serviço pode ser protegido ou não, para o cliente que deseja utilizá-lo, para realizar a tarefa específica.

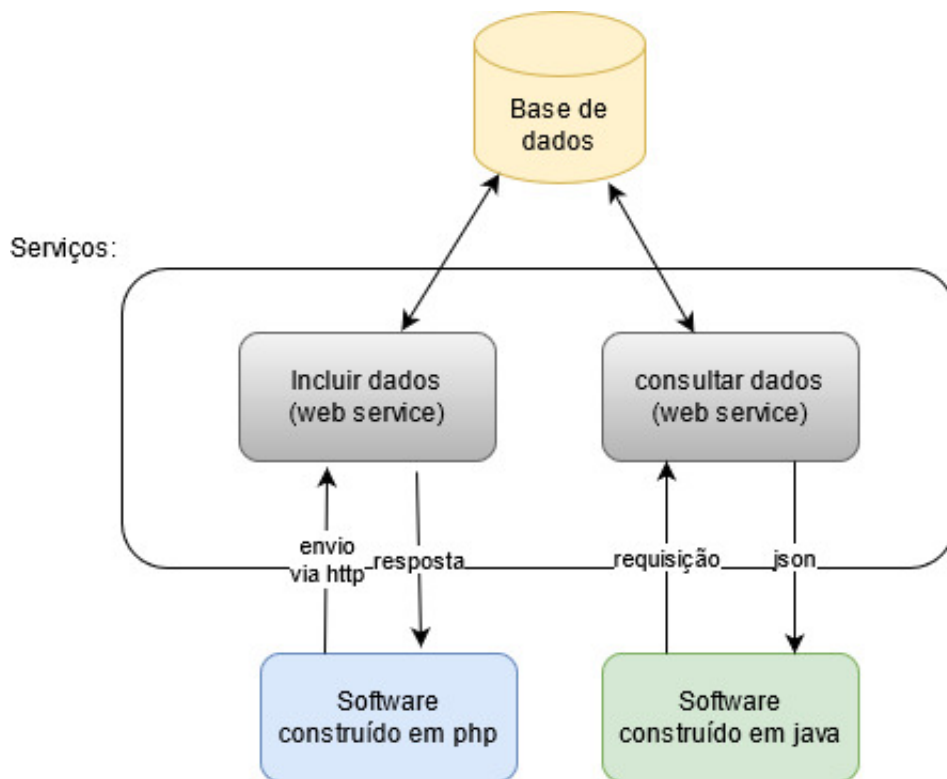


Figura 4 – Exemplo, para descrição de um SOA

Na Figura 4 temos dois softwares distintos consumindo serviços, *web services*, software construído em php alimenta o banco de dados, enviando os dados, há uma forma padrão deve ser observado para que este ou outro software envie dados, o *web service* (incluir dados) retorna uma mensagem se incluiu ou não. Já o software construído em java só consome os dados, onde mesmo faz a requisição, seguindo os padrões de uso do *web service*, e recebe a resposta no formato de Notação de Objetos JavaScript (JSON). Estes padrões de uso podem ser descritos por uma Linguagem de Descrição de Serviços Web (WSDL, do inglês *Web Services Description Language*).

2.1.4 Microserviços

Segundo (NEWMAN, 2015), microserviços são serviços com poucas responsabilidades, pequenos e autônomos que podem trabalhar de forma independente ou em conjunto com outros serviços. Ao contrário de sistemas monolíticos, cada microserviço é empacotado em um artefato separado, e portanto pode-se realizar a implantação de cada um independentemente. Este padrão de arquitetura possui diversas vantagens como: possibilitar escrever componentes de um único sistema utilizando linguagens diferentes se necessário, facilitar a escalabilidade do sistema, entre outras.

2.2 System of Information Systems

O conjunto de sistemas independentes e interoperantes combinados para atingir um objetivo comum é chamado de SoS, tratando-se de uma abordagem arquitetural de alto nível com foco na integração entre artefatos de software (BOARDMAN; SAUSER, 2006). Trata-se de um paradigma arquitetural em que tais artefatos são sistemas computacionais, chamados de sistemas constituintes, que carregam as seguintes categorias (SALEH; ABEL, 2017):

- Independência Operacional: cada sistema constituinte é independente;
- Independência Gerencial: eles funcionam e são administrados de forma independente;
- Desenvolvimento Evolutivo: capacidade de que os sistemas sigam sendo desenvolvidos dinamicamente, só juntando serviços existentes para se fazer um novo;
- Comportamento Emergente: situações emergenciais que geram a necessidade de uma arquitetura que permita montarmos novos sistemas mais rapidamente, desde que os serviços já estejam prontos;
- Arquitetura Dinâmica: que a arquitetura possa ir sendo alterada de acordo com a necessidade.

A partir desta abordagem, surge uma classe especial de SoS em que um ou mais destes constituintes são sistemas de informação, potencialmente independentes interoperando segundo um processo de negócio, chamado de SoIS, (FERNANDES; NETO; SANTOS, 2018). Um sistema de informação é um conjunto de componentes associados que coletam (ou recuperar), processar, armazenar e distribuir informações (TEIXEIRA et al., 2019).

O autor (SALEH; ABEL, 2016a) é uma referência, para a arquitetura que foi construída durante este trabalho. Como visto na Figura 5, o SoIS agregará informações e recursos de vários Sistemas de Informação (Sistema A, Sistema B etc.). Esses sistemas estão funcionando separadamente. Cada dos quais possui serviços / funções e bancos de dados próprios. Enquanto alguns sistemas estão fornecendo abertamente uma API (conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web) para solicitar seus serviços, outros sistemas são fechados e operar como caixas pretas para o mundo exterior e fornecer apenas funções invocadas no próprio sistema. As informações podem ser representados de diferentes maneiras em diferentes sistemas, portanto, o SoIS pode ter problemas para acessar informações, a menos que o os serviços desse sistema estão disponíveis por meio de uma API. Assim sendo, existe uma necessidade crescente de uma solução para essa interoperabilidade questão.

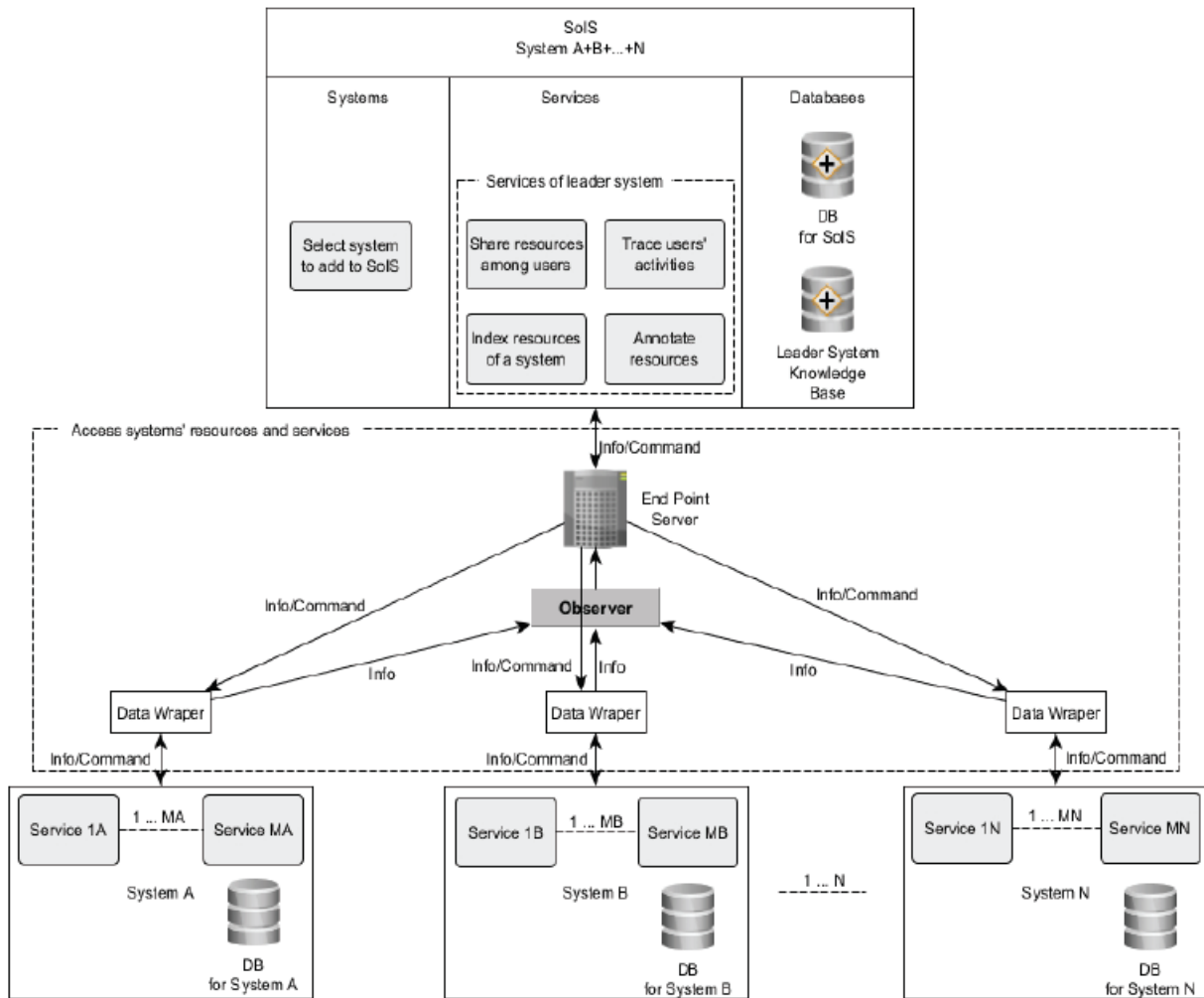


Figura 5 – Arquitetura baseada em SoIS (SALEH; ABEL, 2016a)

Uma solução para o problema de garantir a interoperabilidade dentro do SoIS é controlar o meio de comunicação entre os sistemas. Dois métodos incluem:

- Criando um modelo de software para cada sistema, onde o modelo de software coleta dados do sistema e gera as saídas;
- Criando uma linguagem comum para descrever dados, onde cada sistema pode representar seus dados de forma que outros sistemas podem interpretar.

Devido a restrições de sobrecarga em uma linguagem comum, frequentemente não é possível ter uma linguagem padrão para interpretação dos dados usado em todo o SoIS. Portanto, a criação de software individual modelos para os vários sistemas é o mais utilizado abordagem para garantir a interoperabilidade dentro de um SoIS. Isso pode explicar a necessidade de envolver dados como um software mediador entre o SoIS e seus sistemas constituintes. O SoIS é representado como um grupo de conectores de sistemas, serviços e bancos de dados. Os serviços residentes no SoIS podem ser a utilização dos

serviços existentes nos Sistemas de Informação incluindo o SoIS ou serviços emergentes criados a partir da agregação de diferentes sistemas de informação.

As ideias presentes nesta arquitetura do SoIS podem ser resumido na lista a seguir:

- O usuário recebe acesso a vários Sistemas de informações;
- O usuário pode escolher quais sistemas de informação ele / ela gostaria de se conectar;
- Conectar-se a vários sistemas de sua escolha, o usuário podem acessar recursos e serviços em seus respectivos meio;
- É então possível trabalhar com os recursos produzidos por diferentes Sistemas de Informação de dentro do SoIS;
- Os recursos produzidos por diferentes sistemas de informação são gerenciados no SoIS por meio dos serviços fornecidos pelo sistema líder;
- Utilizar os serviços do sistema líder como forma de gerenciar as informações com o SoIS permitirá que o usuário de indexação, compartilhamento e rastreamento de recursos dentro o SoIS.

2.3 Linguagem de descrição WSDL

O WSDL é escrito em uma Linguagem Extensível de Marcação (XML, do inglês *Extensible Markup Language*), que emprega marcações para definir detalhes sobre um serviço web (*web service*). Um documento WSDL define serviços como coleções de terminais de rede ou portas (normalmente chamados *endpoints*). Basicamente, em um documento WSDL a definição abstrata de *endpoints* e mensagens é separada de sua implementação de rede concreta ou ligações de formato de dados, isto é, não são acoplados às implementações. Isso permite a reutilização de definições abstratas, isto é, as mensagens, que são descrições abstratas dos dados sendo trocados, e tipos de portas, que são coleções abstratas das operações disponíveis no serviço web. Tais definições permitem construir descrições dos serviços de forma reutilizável. Um *endpoint* é definido pela associação de um endereço de rede a um serviço web em específico, sendo que serviços web mais elaborados podem possuir diversos *endpoints*. Como se trata de um documento XML, o qual pode ser observado no exemplo que apresenta a sua estrutura na Lista 2.1, o mesmo é composto de marcadores, ou *tags*.

Lista 2.1 – Estrutura de um documento WSDL padrão

```
1 <wsdl:definitions name="nmtoken"? targetNamespace="uri"?>
  <import namespace="uri" location="uri"/>*
```



```
44         <-- extensibility element --> *
45         </wsdl:fault>
46     </wsdl:operation>
47 </wsdl:binding>
48
49 <wsdl:service name="nmtoken"> *
50     <wsdl:documentation .... />?
51     <wsdl:port name="nmtoken" binding="qname"> *
52         <wsdl:documentation .... /> ?
53         <-- extensibility element -->
54     </wsdl:port>
55     <-- extensibility element -->
56 </wsdl:service>
57 <-- extensibility element --> *
58 </wsdl:definitions>
```

Conforme a estrutura de exemplo vista na Lista 2.1, um documento WSDL define serviços que serão consumidos por aplicações ou orquestradores, sendo que os serviços são descritos usando seis principais elementos (CHRISTENSEN et al., 2001):

- *types*, que fornece definições de tipo de dados usados para descrever as mensagens trocadas. Também *element* inclui definições de tipo de dados que são relevantes para as mensagens trocadas.
- *message*, que representa uma definição abstrata dos dados sendo transmitidos. Uma mensagem consiste em partes lógicas, cada uma das quais associada a uma definição em algum sistema de tipo.
- *portType*, que é um conjunto de operações abstratas. Cada operação se refere a uma mensagem de entrada e mensagens de saída. Uma operação é nomeada por meio do atributo *name*.
- *binding*, que especifica especificações concretas de protocolo e formato de dados para as operações e mensagens definidas por um portType específico.
- *port*, que especifica um endereço para uma ligação, definindo assim um único terminal de comunicação.
- *service*, que é usado para agregar um conjunto de portas relacionadas.

2.4 Selenium

O projeto de automação do navegador *Selenium* é muito empregado em testes de software (RAMYA; SINDHURA; SAGAR, 2017; SANTOS et al.,), mas possui características que permitem o seu uso para outros fins (MANJARI et al., 2020; LI et al.,), como a construção de artefatos de software automatizados (*bots*) para os mais variados fins. Ele fornece extensões para emular a interação do usuário com navegadores, um servidor de distribuição para dimensionar a alocação do navegador e a infraestrutura para implementações da especificação W3C WebDriver que permite escrever código intercambiável para todos os principais navegadores da web (SELENIUM, 2020). Estes navegadores estão detalhados na tabela 1, eles possuem os drivers necessários para a execução, já as linguagens suportadas são: java, python, c#, ruby, JavaScript e kotlin. Estas necessitam de suas respectivas bibliotecas.

Uma das suas principais capacidades é a possibilidade de manipulação completa de formulários e componentes de interfaces de sistemas em plataforma web, como o clique em botões, a procura por elementos na interface e a capacidade de digitar textos e manipulação de teclas alfanuméricas com utilização elementos DOM (do inglês *Document Object Model*) e os eventos de *keyboard*, através de métodos como o *sendKeys* (digita uma sequência de teclas no elemento DOM), *keyDown* (usado para simular a ação de pressionar uma tecla modificadora (*CONTROL*, *SHIFT*, *ALT*)), *keyUp* (usado para simular a liberação de uma tecla modificadora) e *clear* (Limpa o conteúdo de um elemento editável).

Tabela 1 – Navegadores com suporte oficial pelo *Selenium*

Navegador	Mantenedor	Versões Suportadas
Chrome	Chromium	Todas as versões
Firefox	Mozilla	54 ou mais nova
Edge	Microsoft	84 ou mais nova
Internet Explorer	Selenium	6 ou mais nova
Opera	Opera Chromium / Presto	10.5 ou mais nova
Safari	Apple	10 ou mais nova

O *Selenium WebDriver* conduz um navegador nativamente, como um usuário faria, localmente ou em uma máquina remota usando o servidor *Selenium*, marcando um salto em termos de automação do navegador. *WebDriver* é uma API compacta orientada a objetos e pode ser combinada com os recursos da linguagem, como, por exemplo, *loops* e condicionais. Possui uma técnica, *web scraping*, eficiente usada para extração de grandes quantidades de dados de vários sites. Com uso de padrões como *Page Objects*, torna a leitura e manutenção de código mais simples. É uma ferramenta *open source* bastante difundida na comunidade de testes do mundo, tendo até aqui atualizações constantes, correções e materiais de apoio (AVASARALA, 2014).

2.4.1 Exemplos

No Algoritmo 2.2 vemos um exemplo da implementação de um *driver web*, escrito em java, porém o mesmo poderia ser escrito nas linguagens que possuem suportes, com algoritmos desse tipo é possível o acesso à qualquer recurso em plataforma web, sendo que no algoritmo exemplificado, obtemos acesso ao buscador do google e realizamos a busca pela palavra "cheese". Assim, tal ferramenta se mostra bastante útil na construção de artefatos de software que permitam emular um usuário humano interagindo com a interface.

Lista 2.2 – Algoritmo para acesso a recursos da web

```
1 import org.openqa.selenium.By;
2 import org.openqa.selenium.Keys;
3 import org.openqa.selenium.WebDriver;
4 import org.openqa.selenium.WebElement;
5 import org.openqa.selenium.firefox.FirefoxDriver;
6 import org.openqa.selenium.support.ui.WebDriverWait;
7 import static
    org.openqa.selenium.support.ui.ExpectedConditions.presenceOfElementLocated;
8 import java.time.Duration;
9
10 public class HelloSelenium {
11
12     public static void main(String[] args) {
13         WebDriver driver = new FirefoxDriver();
14         WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
15         try {
16             driver.get("https://google.com/ncr");
17             driver.findElement(By.name("q")).sendKeys("cheese" + Keys.ENTER);
18             WebElement firstResult =
                wait.until(presenceOfElementLocated(By.cssSelector("h3")));
19             System.out.println(firstResult.getAttribute("textContent"));
20         } finally {
21             driver.quit();
22         }
23     }
24 }
```

No algoritmo 2.3, também escrito em java, neste exemplo objeto é de uma página, construída a sua classe com atributo básico do tipo *WebDriver*, temos um método padrão para realização de *login* em uma página da web, que necessite deste processo, assim o usuário e a senha são passados como parâmetro em seguida verifica se ele está logado e,

se desejar, acessar os dados da página inicial.

Lista 2.3 – Algoritmo para realização de *login*

```
1 import org.openqa.selenium.By;
2 import org.openqa.selenium.WebDriver;
3
4 /**
5  * Page Object encapsula a página de login.
6  */
7 public class SignInPage {
8     protected WebDriver driver;
9
10    // <input name="user_name" type="text" value="">
11    private By usernameBy = By.name("user_name");
12    // <input name="password" type="password" value="">
13    private By passwordBy = By.name("password");
14    // <input name="sign_in" type="submit" value="SignIn">
15    private By signinBy = By.name("sign_in");
16
17    public SignInPage(WebDriver driver){
18        this.driver = driver;
19    }
20
21    /**
22     * Login como um usuário válido
23     *
24     * @param userName
25     * @param password
26     * @return HomePage object
27     */
28    public HomePage loginValidUser(String userName, String password) {
29        driver.findElement(usernameBy).sendKeys(userName);
30        driver.findElement(passwordBy).sendKeys(password);
31        driver.findElement(signinBy).click();
32        return new HomePage(driver);
33    }
34 }
35
36 /**
37  * Page Object encapsula a Home Page
38  */
39 public class HomePage {
```

```
40  protected WebDriver driver;
41
42  // <h1>Hello userName</h1>
43  private By messageBy = By.tagName("h1");
44
45  public HomePage(WebDriver driver){
46      this.driver = driver;
47      if (!driver.getTitle().equals("Home Page of logged in user")) {
48          throw new IllegalStateException("This is not Home Page of logged in user," +
49              " current page is: " + driver.getCurrentUrl());
50      }
51  }
52
53  /**
54   * Get message (h1 tag)
55   *
56   * @return String message text
57   */
58  public String getMessageText() {
59      return driver.findElement(messageBy).getText();
60  }
61
62  /* Mais métodos fornecendo o serviços representados pela Home Page
63   do usuário logado. Esses métodos por sua vez podem retornar mais Page Objects
        por exemplo clicar no botão Compor Email poderia retornar um objeto
        ComposeMail */
64  }
65
66  /**
67   * Tests login feature
68   */
69  public class TestLogin {
70
71      @Test
72      public void testLogin() {
73          SignInPage signInPage = new SignInPage(driver);
74          HomePage homePage = signInPage.loginValidUser("userName", "password");
75          assertThat(homePage.getMessageText(), is("Hello userName"));
76      }
77
78  }
```

2.5 Trabalhos relacionados

2.5.1 SoS, SoIS e SOA

"*Interoperability in Systems-of-Information Systems: A Systematic Mapping Study*" (FERNANDES; NETO; SANTOS, 2018), em português "Interoperabilidade em Sistemas de Sistemas de Informação: Um Estudo Sistemático de Mapeamento" A principal contribuição deste artigo é a externalização de fatores (por exemplo, questões técnicas, organizacionais e humanas) que afetam os sistemas de interoperabilidade de sistemas no contexto do SoIS através de um estudo sistemático de mapeamento. Os resultados contribuem para a caracterização de soluções de interoperabilidade para sistemas, que são de grande importância para identificar desafios e oportunidades.

"*Moving from digital ecosystem to system of information systems*" (SALEH; ABEL, 2016a), em português, Passando do ecossistema digital para o sistema de sistemas de informação. Neste artigo, estamos passando do Ecossistema Digital para o SoIS, definindo as semelhanças entre os dois conceitos e, em seguida, propondo um modelo de arquitetura do SoIS em correspondência ao modelo DE.

"*Resources management and decision support in a system of information systems*" (SALEH; ABEL, 2016b), em português Gestão de recursos e suporte à decisão em um sistema de sistemas de informação, Os autores nesse artigo planejam abordar esse problema, procurar recursos em diferentes sistemas de informação, introduzindo um modelo de arquitetura do SoIS. Este modelo fornecerá orientação para construir um primeiro protótipo.

"*Modeling and developing a system of information systems for managing heterogeneous resource*" (SALEH; ABEL, 2017), em português, Modelagem e desenvolvimento de um sistema de sistemas de informação para gerenciamento de recursos heterogêneos, este artigo enfoca a modelagem e o desenvolvimento de um SoIS para gerenciar recursos heterogêneos.

"*System of Information Systems and Organizational Memory*" (SALEH; ABEL, 2018a), em português Sistema de Sistemas de Informação e Memória Organizacional, Os autores Mostram neste artigo como as instalações de retenção (aquisição, retenção, recuperação e artefatos de cooperação) nas etapas da memória organizacional podem ser superadas pelo SoIS. Neste trabalho, foi desenvolvido um protótipo chamado MEMORA-eSoIS.

"*System of Information Systems to support learners (a case study at the University of Technology of Compiègne)*" (SALEH; ABEL, 2018b) (Saleh, 2018), em português, Sistema de Sistemas de Informação para apoiar os alunos (um estudo de caso na Universidade de Tecnologia de Compiègne). Este artigo apresenta a abordagem do SoIS para

apoiar os alunos com um estudo de caso na Universidade de Tecnologia de Compiègne.

“*A Conceptual Model for Systems-of-Information Systems*” (FERNANDES et al., 2019), em português, Um modelo conceitual para Sistemas de Sistemas de Informação. A principal contribuição deste artigo é o estabelecimento de um modelo conceitual para apoiar pesquisadores e profissionais no reconhecimento de um SoIS.

2.5.2 Sistema de informação para gestão de dados acadêmicos

“*FACTORS EXPLAINING THE DEGREE OF ACCEPTANCE OF AN ACADEMIC INFORMATION SYSTEM: A CASE STUDY WITH THE LECTURERS OF A PRIVATE UNIVERSITY*” (REIS; PITASSI; BOUZADA, 2013), em português, OS FATORES QUE EXPLICAM O GRAU DE ACEITAÇÃO DE UM SISTEMA DE INFORMAÇÃO ACADÊMICA: UM ESTUDO DE CASO COM DOCENTES DE UMA IES PRIVADA. Os autores neste trabalho tiveram como objetivo identificar os fatores que explicam o grau de aceitação do Sistema de Informação Acadêmica (SIA) utilizado nos processos de apoio à gestão docente em uma IES privada.

“A contribuição dos sistemas de informações na gestão universitária” (BERNARDES; ABREU, 2004), neste artigo os autores explicam sobre o sucesso ou fracasso da implantação de Sistemas de Informações Gerenciais –SIG, estão diretamente ligados ao estudo do contexto da organização na qual eles serão utilizados e na consequente criação de um ambiente propício, capaz de garantir o desenvolvimento, a implantação, a aceitação e o uso de um novo sistema. Tudo isso passa pelas pessoas da organização envolvidas no processo e na estratégia a ser utilizada para garantir o sucesso. Assim o desenvolvimento e a implantação de SIG, não são meras ações de instalação e treinamento de usuários. Nos últimos anos, porém, surgiu um interesse crescente em tratar sistemas de informações como ferramenta propícia como recurso organizacional significativo para ajuda na gestão de instituições.

“Integração entre o sistema de gestão acadêmica e o sistema de gestão da aprendizagem: identificando necessidades e prototipando requisitos favoráveis a prática docente” (CARVALHO et al., 2012). Este trabalho tem como objetivo identificar a forma adequada da integração entre os sistemas de gestão acadêmica e de gestão da aprendizagem favorável à prática docente.

Parte III

Metodologia

3 Metodologia Aplicada

Neste capítulo é apresentada a metodologia empregada para realização do trabalho.

O estudo bibliográfico foi a etapa inicial, teve como fim averiguar outras pesquisas na mesma linha de pesquisa deste tema, de forma a prover uma visão geral sobre as possíveis abordagens para a solução do problema aqui enfrentado. Neste sentido, foi empregada a metodologia de revisão sistemática da literatura, que emprega um protocolo bem definido, a se considerar as etapas: Planejamento, Condução e Relatório (KITCHENHAM; CHARTERS, 2007), (PETERSEN; VAKKALANKA; KUZNIARZ, 2015).

A partir dessas buscas, foram feitas leituras e foram escolhidos artigos científicos para serem analisados, segundo o artigo “The status quo of systems-of-information systems” (TEIXEIRA et al., 2019), para melhor mostrar o estado da arte, de forma que ao fim do processo, constatou-se que a arquitetura SoIS mostra-se como a solução mais viável.

Também a partir da *string* de busca, (*"software architecture"AND (survey OR "state % of % the % art"OR "literature review") AND ("SOA"OR "SOS"OR "SoIS")*), aplicada nas bases como IEEE Xplore, Google Acadêmico, foram feitas leituras dos metadados dos quais foram selecionados os artigos da seção 2.5.1.

Outra *string* de busca utilizada, (sistema de informação para gestão de dados acadêmicos) *AND (information system for academic data management)*, dos quais foram selecionados os artigos da seção 2.5.2.

Em paralelo ao estudo bibliográfico, foi feito a análise de sistema (modelagem), a qual consistiu no entendimento do problema e assim, também, na definição do escopo do trabalho, foi utilizada a ferramenta lucidchart (LUCIDCHART, 2019) para o desenho da arquitetura proposta. A Linguagem de Modelagem Unificada (UML) provê diversos tipos de diagramas para a especificação dos modelos. Segundo (BOOCH; RUMBAUGH; JACOBSON, 2006), a UML proporciona uma forma padrão para a preparação de planos de arquitetura de projetos de sistemas.

Foi utilizada a prototipação, processo que tem como objetivo facilitar o entendimento dos requisitos, apresentar conceitos e funcionalidades do software. Desta forma, pode-se propor uma solução adequada para o problema. De acordo com (JR, 2010), o protótipo da solução é um instrumento extremamente útil e importante para validação de requisitos. Esses protótipos servem para demonstrar o sistema, como será a navegação entre as interfaces, os relatórios previstos e assim por diante, reproduzindo o comportamento do futuro sistema a ser implementado. Foi utilizada a ferramenta Pencil para o

esboço, desenho, das telas do sistema líder.

Para o aprofundamento nas análises do tema a partir das pesquisas encontradas no levantamento bibliográfico, foi também desenvolvido um texto de estado da arte, o resumo expandido.

No desenvolvimento do sistema, descrito na seção 4.3, SisCoord 1.0, foram utilizadas tecnologias como java web, bootstrap, javascript, JSON e para o banco de dados, o MySQL. Sobre o java web este gera páginas web interativas e conteúdo dinâmico, o bootstrap é um *framework* web com código-fonte aberto utilizado em *front-end* faz uso principalmente de css e javascript para tornar *front* responsivo, o javascript foi utilizado para aprimorar ainda mais a interação com o usuário, JSON é como a maioria dos dados trafega entre cliente e servidor, MySQL é o SGBD utilizado para o banco de dados.

Parte IV

Arquitetura proposta e resultados

4 Apresentação

Neste capítulo são apresentados e discutidos os resultados do trabalho realizado. Como resultado são apresentados a arquitetura proposta neste trabalho e as telas do protótipo e também um sistema desenvolvido para analisar a questão do *web service*, assim será exposto este sistema, o SisCoord, desenvolvido para atividades no meio acadêmico, desta forma este foi utilizado para pequenos testes e entendimento da arquitetura proposta. Por fim, a discussão desses resultados para melhor entendimento da arquitetura SoIS.

4.1 A arquitetura proposta

Como ponto fundamental do presente trabalho, em que se propõe a definição de uma arquitetura flexível para permitir a interoperabilidade entre sistemas de informação distintos, conforme já descrito, a Figura 6 apresenta visualmente a arquitetura proposta, que é composta dos seguintes elementos fundamentais:

- O orquestrador;
- Os descritores;
- Os conectores;
- Sistemas constituintes;
- Base de dados local;
- Serviços;
- Estruturas de Dados.

Como o objetivo do emprego desta arquitetura é gerar a possibilidade de criação de um novo sistema de informação, ou uma nova funcionalidade a um sistema já existente, observa-se a necessidade de uma entidade que orquestre tais serviços existentes, assim como visto em arquiteturas SOA (DOSPINESCU et al., 2017). O orquestrador é o sistema líder, que permitirá o consumo dos serviços disponíveis através dos sistemas constituintes. Através dele encontraremos as opções para acessar todos os outros sistemas, isto é, consumir os serviços por eles disponibilizados. Em outras palavras, o orquestrador terá o papel de uma API para o sistema que consumirá os serviços dispersos nos sistemas constituintes, o que gera a facilidade de concentrar todo o conhecimento necessário para

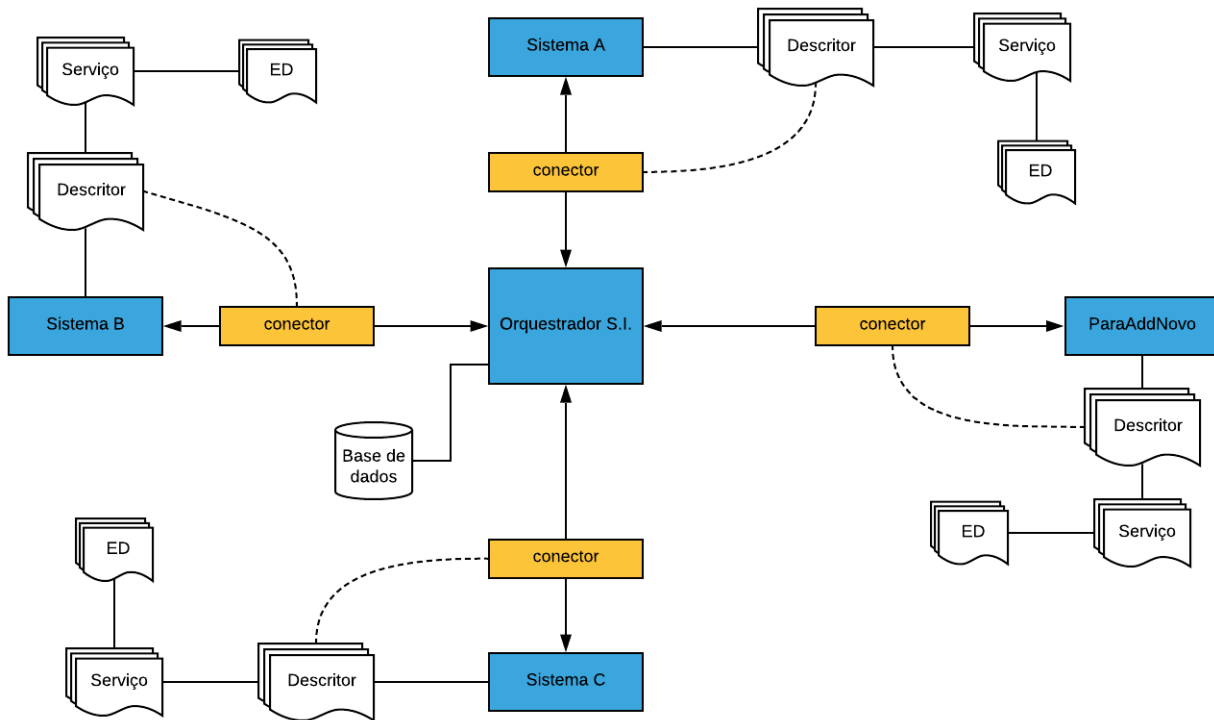


Figura 6 – Arquitetura proposta

a integração dos sistemas em um único ponto de acesso. Assim, o desenvolvimento e/ou adaptação de novos sistemas de informação tendem a retirar dos projetistas e desenvolvedores a necessidade do conhecimento de todos os sistemas envolvidos. Adicionalmente, o orquestrador também pode ser o próprio sistema fim, isto é, possuir as próprias interfaces de usuário que permitam aos usuários o acesso/consumo dos serviços por ele orquestrados, como apresentado nos protótipos do sistema SoIS de gestão de dados acadêmicos (Seção 4.2).

Como o SoIS será baseado em serviços disponíveis em sistemas de informação independentes, é por meio de conectores especializados que a comunicação entre o orquestrador e os sistemas constituintes deverá ocorrer. Os conectores são artefatos de software que vão utilizar os descritores para consumir os serviços oferecidos. E ele pode ser desenvolvido das mais diferentes formas, usando diferentes linguagens de programação, o que gera a flexibilidade já destacada. Aqui surge a primeira diferenciação entre a arquitetura aqui proposta e arquiteturas baseadas APIs tradicionais, que obrigatoriamente precisam consumir, apenas, serviços web formalmente definidos em APIs do estilo SOAP ou REST, por exemplo. Com a presente arquitetura proposta, um conector poderá se conectar com tais APIs, mas também estará apto a consumir sistemas de informação através das próprias interfaces de usuário, utilizando mecanismos implementados em ferramentas no estilo do *Selenium Web Driver* (SELENIUM, 2020), por exemplo.

Tais conectores serão instruídos por artefatos que carregam informações sobre

como os serviços web devem ser acessados. Tais artefatos nesta arquitetura são definidos como descritores, sendo arquivos definidos por meio da linguagem de descrição de serviços web WSDL (Seção 2.3). Embora WSDL seja empregado em soluções baseadas em serviços web tradicionais, um ganho tecnológico da presente proposta é a sua adoção para descrever diferentes formas de consumo de sistemas de informação em plataforma web que não são instanciados ou descritos como tal. A seguir temos a Lista 4.1, que demonstra um descritor de um serviço web para o cadastro de um setor através do consumo de um serviço web disponibilizado pelo sistema constituinte SisCoord, em que é possível observar a definição da própria estrutura de dados empregada (dois campos do tipo *string*: nomeSetor e siglaSetor), bem como as possíveis mensagens de troca e o próprio endereço do serviço (linha 40). Importante observar que *tags* especializadas podem ser incluídas em um descritor, de forma a permitir a correta instrução da implementação dos recursos necessários no orquestrador para consumo de serviços não padronizados, como no caso de acesso via *Selenium Web Driver*, como exemplificado no apêndice B.1.

Lista 4.1 – Exemplo de descritor para um serviço via API

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!--
3 IFG
4 2019-08-29
5 Serviço para cadastro de setor
6 Online WSDL 1.1 SOAP generator 0.2
7 Julien Blitte
8 -->
9 <definitions name="Serviço para cadastro de setor"
10     targetNamespace=".wsdl"
11     xmlns:tns=".wsdl"
12     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
13     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
14     xmlns:xsd1=".xsd"
15     xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
16     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
17     xmlns="http://schemas.xmlsoap.org/wsdl/">
18     <!-- definition of datatypes -->
19     <types>
20     <schema targetNamespace=".xsd" xmlns="http://www.w3.org/2000/10/XMLSchema">
21         <element name="nomeSetor">
22             <complexType><all><element name="value" type="string"
23                 /></all></complexType>
24         </element>
25         <element name="siglaSetor">
```



```

25     <complexType><all><element name="value" type="string"
26         /></all></complexType>
27     </element>
28 </schema>
29 </types>
30 <!-- response messages -->
31 <!-- request messages -->
32 <!-- server services -->
33 <portType name='Cadastro de setor'></portType>
34 <!-- server encoding -->
35 <binding name='Cadastro de setor_webservices' type='tns:Cadastro de setor'>
36 <soap:binding style='rpc' transport='http://schemas.xmlsoap.org/soap/http' />
37 </binding>
38 <!-- access to service provider -->
39 <service name='setor'>
40 <port name='_0' binding='Cadastro de setor_webservices'>
41 <soap:address location='http://localhost:8080/SisCoord/cms/setor' />
42 </port>
43 </service>
44 </definitions>

```

Sistemas de informação independentes observados, constituintes (SisCoord, API RestFULL, Sara), na instituição para pequenos testes e validação, do protótipo, da arquitetura:

O SisCoord é sistema que foi construído durante o andamento deste trabalho, temos uma seção 4.3 para explicar o mesmo.

API RestFULL é um sistema de informação para controle de eventos institucionais este aceita métodos de *post*, para cadastro de novos eventos, e de *get*, que busca um JSON com todos já cadastrados, para o uso do orquestrador.

Sara (Sistema para alocação de recursos acadêmicos) é um sistema da fábrica de software do ifg, desenvolvido inicialmente em php, sua funcionalidade é para fazer reservas de recursos como projetores, reservar salas e laboratórios, dentre outros.

A base de dados é para guardar as senhas para realizar *login* nos sistemas constituintes e do próprio orquestrador, no qual o banco de dados está conectado, também para persistir os descritores. Aqui temos, apresentados para a arquitetura proposta, do banco de dados os modelos conceitual, lógico e físico.

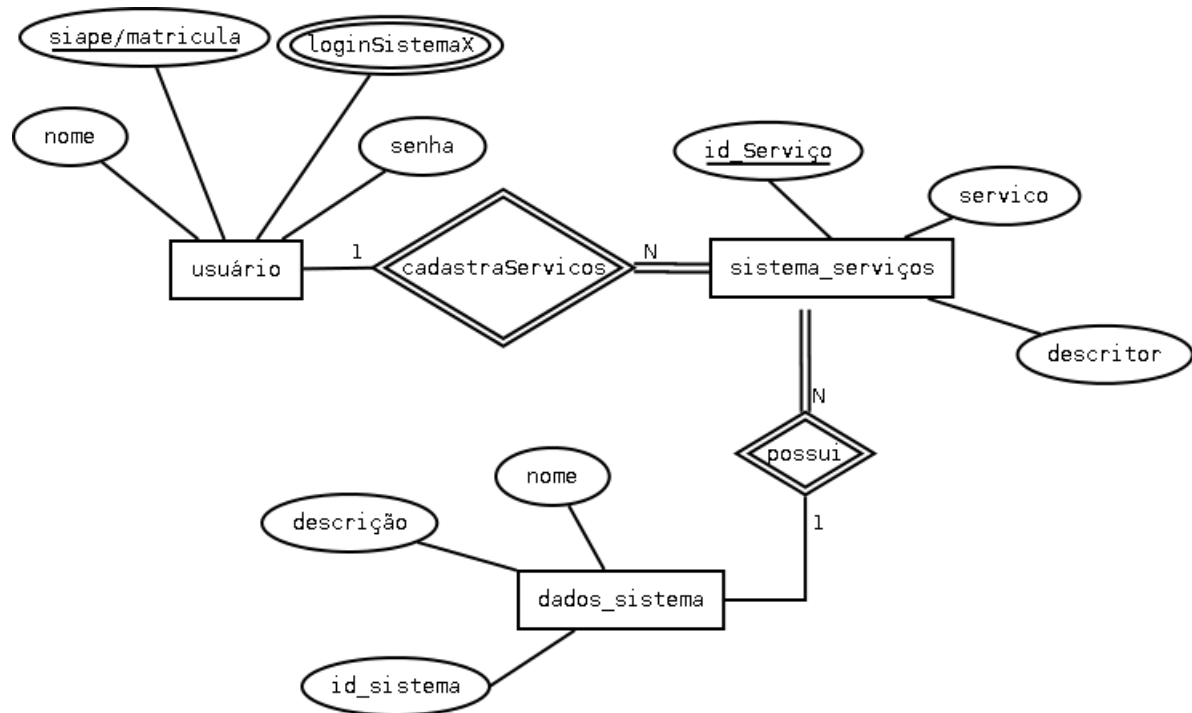


Figura 7 – Modelo conceitual

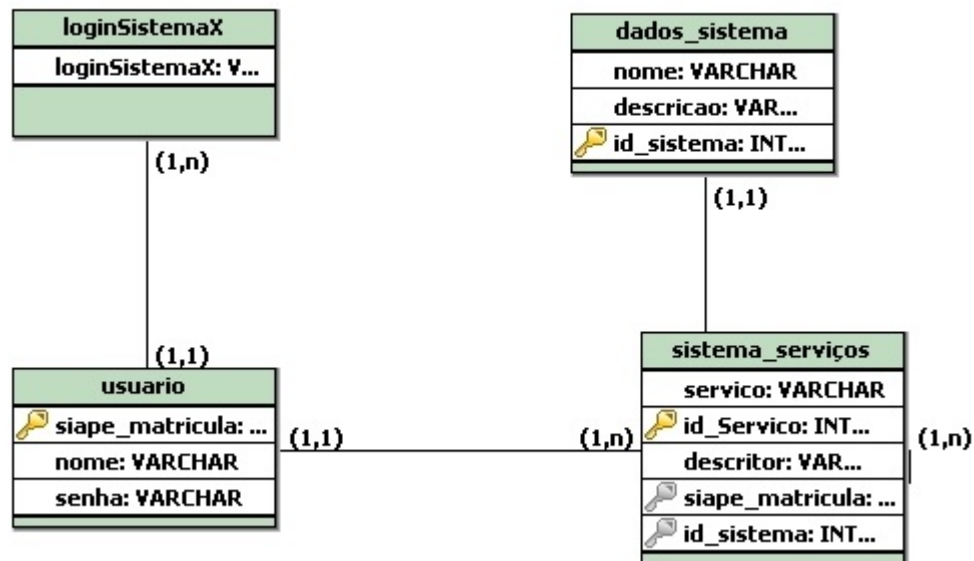


Figura 8 – Modelo Lógico

Lista 4.2 – Modelo físico

```

1 -- Geração de Modelo físico
2 -- Sql ANSI 2003 - brModelo.
3
4 CREATE TABLE sistema_serviços (
5  serviço VARCHAR,

```

```
6 id_Servico INTEGER PRIMARY KEY,
7 descritor VARCHAR,
8 siape_matricula VARCHAR,
9 id_sistema INTEGER
10 )
11 CREATE TABLE usuario (
12 siape_matricula VARCHAR PRIMARY KEY,
13 nome VARCHAR,
14 senha VARCHAR
15 )
16 CREATE TABLE dados_sistema (
17 nome VARCHAR,
18 descricao VARCHAR,
19 id_sistema INTEGER PRIMARY KEY
20 )
21 CREATE TABLE loginSistemaX (
22 loginSistemaX VARCHAR
23 )
24 ALTER TABLE sistema_servicos ADD FOREIGN KEY(siape_matricula) REFERENCES usuario
    (siape_matricula)
25 ALTER TABLE sistema_servicos ADD FOREIGN KEY(id_sistema) REFERENCES dados_sistema
    (id_sistema)
```

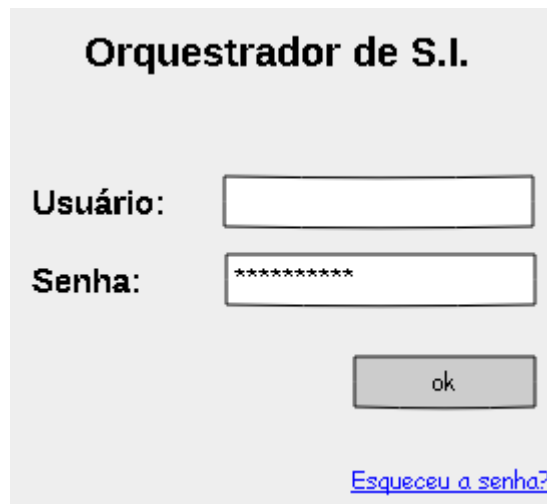
Serviços este são os que os descritores oferecem.

ED vai ser usado para salvar os dados no orquestrador. Uma tabela vai ser dinamicamente criada para cada ED, assim que ele for salvo.

Detalhando a figura 6, no centro temos o orquestrador, este possui o componente base de dados ligado por uma linha simples, que se comunica com os constituintes através de conectores, as setas observadas apontam os conectores para os sistemas, isto significa que a comunicação entre eles depende dos conectores, a linha pontilhada entre conectores e descritores, indica que o conector é construído através de um conhecimento do descritor que por sua vez está ligado através de uma linha simples com componentes que o compõem.

4.2 Protótipo em telas iniciais

O protótipo está demonstrado até aqui em telas iniciais, nesta parte do trabalho será exibido e apresentado as telas com suas respectivas explicações, e futura perspectiva de implementação. A seguir o registro para utilizar o software, sistema líder, o *login*:



Orquestrador de S.I.

Usuário:

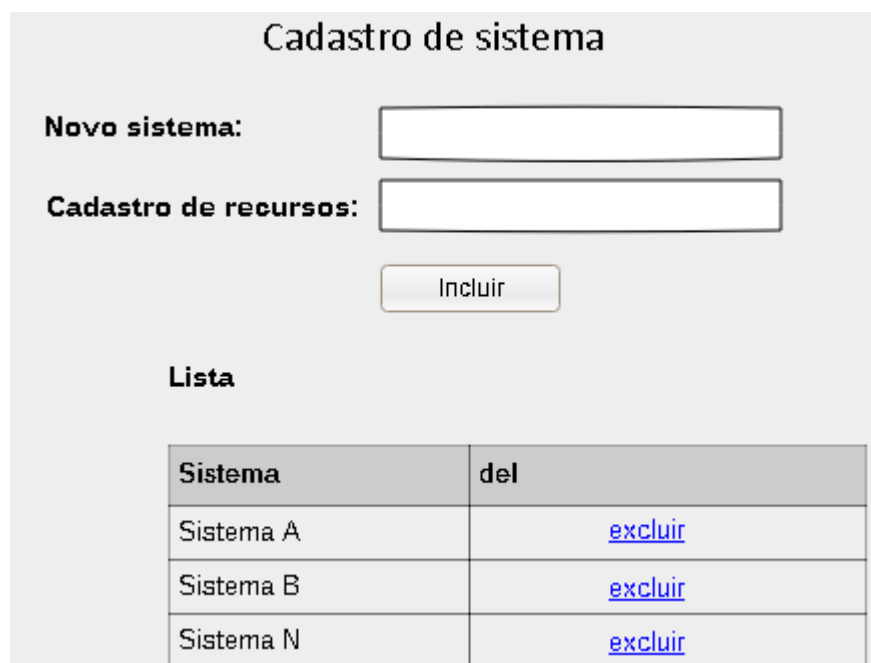
Senha:

[Esqueceu a senha?](#)

Figura 9 – Tela de *login* do orquestrador de sistema

Nesta tela 9 temos a ideia de como será o registro para utilizar o orquestrador do sistema, já que através do mesmo teremos acesso aos sistemas constituintes, por isso é importante esta tela inicial, após realizado este procedimento de autenticação pode ser exibido uma mensagem de bem-vindo com nome do usuário, no primeiro acesso o usuário pode mudar sua senha, pois o mesmo será cadastrado por um gestor.

Cadastro de um sistema constituinte:



Cadastro de sistema

Novo sistema:

Cadastro de recursos:

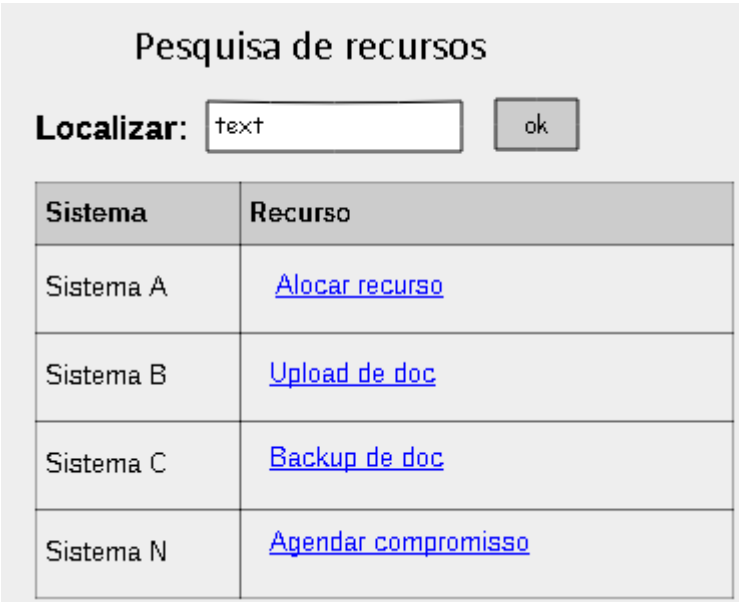
Lista

Sistema	del
Sistema A	excluir
Sistema B	excluir
Sistema N	excluir

Figura 10 – Tela de cadastro de um novo sistema constituinte

Nesta tela 10 temos um CRUD (*Create, Read, Update and Delete*), as quatro operações básicas (criação, consulta, atualização e destruição de dados) utilizadas em bases de dados relacionais, de sistemas constituintes. O cadastro de recursos é feito por meio de um descritor, escrito por meio de um WSDL.

Pesquisa no sistema para encontrar um recurso:

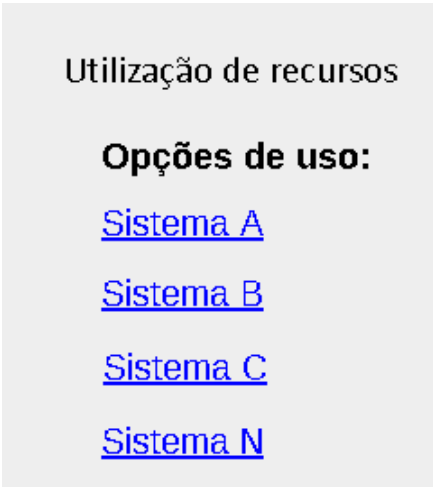


Sistema	Recurso
Sistema A	Alocar recurso
Sistema B	Upload de doc
Sistema C	Backup de doc
Sistema N	Agendar compromisso

Figura 11 – Tela de pesquisa de recurso

Nesta tela 11 vemos como poderá ser utilizado os recursos no orquestrador, através de um campo, *input*, de busca. Ao clicar no recurso desejado o mesmo será utilizado no próprio do orquestrador, através de sua base de dados e conectores.

Pesquisa de recursos, por sistema, no orquestrador para encontrar um recurso:



Utilização de recursos

Opções de uso:

[Sistema A](#)

[Sistema B](#)

[Sistema C](#)

[Sistema N](#)

Figura 12 – Tela de pesquisa de recurso por sistema

A ideia é mesma que a Figura 11, porém nessa tela a busca de recursos é feita por sistema caso o usuário escolha buscar o recurso desta forma.

4.3 Sistema SisCoord

O SisCoord é um dos sistemas constituintes que foi desenvolvido para ser utilizado pela coordenação, este é um dos resultados do trabalho, também para realizar pequenos testes, a fim de validar o protótipo da arquitetura, como temos o código fonte podemos criar descritores, como este do quadro 4.1 e requisições foram utilizadas pelo sistema líder (orquestrador) do protótipo, por exemplo, para o cadastro de um novo setor, a seguir será detalhado o funcionamento do SisCoord.

O sistema possui *design* responsivo, demonstração exibida em capturas de telas.

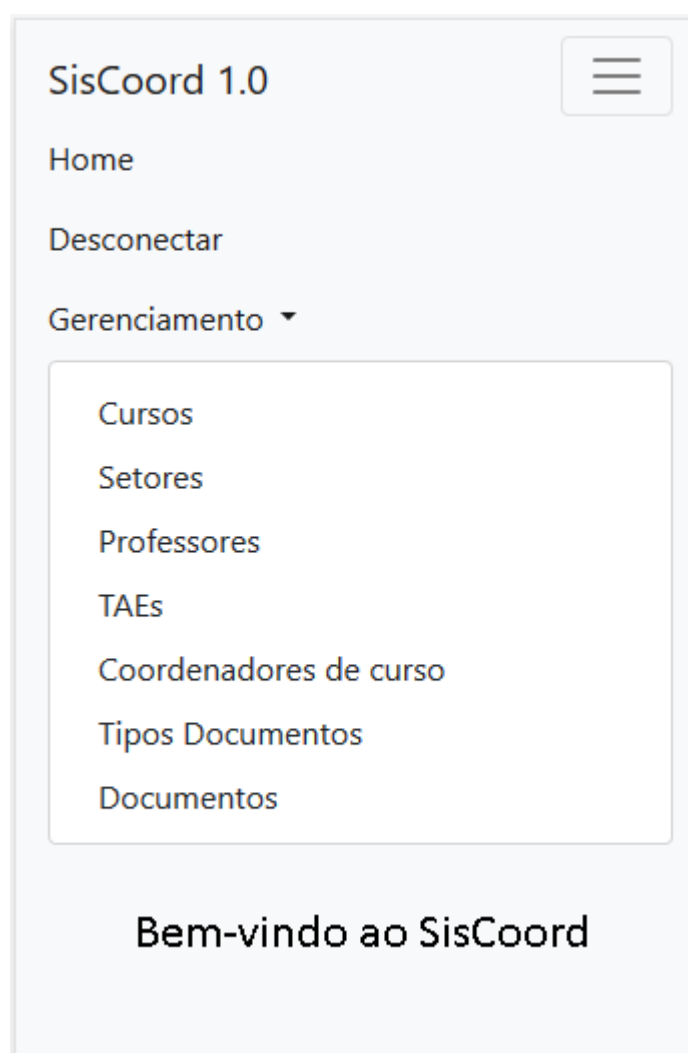


Figura 13 – Tela de boas-vindas, no dispositivo móvel, após o *login*

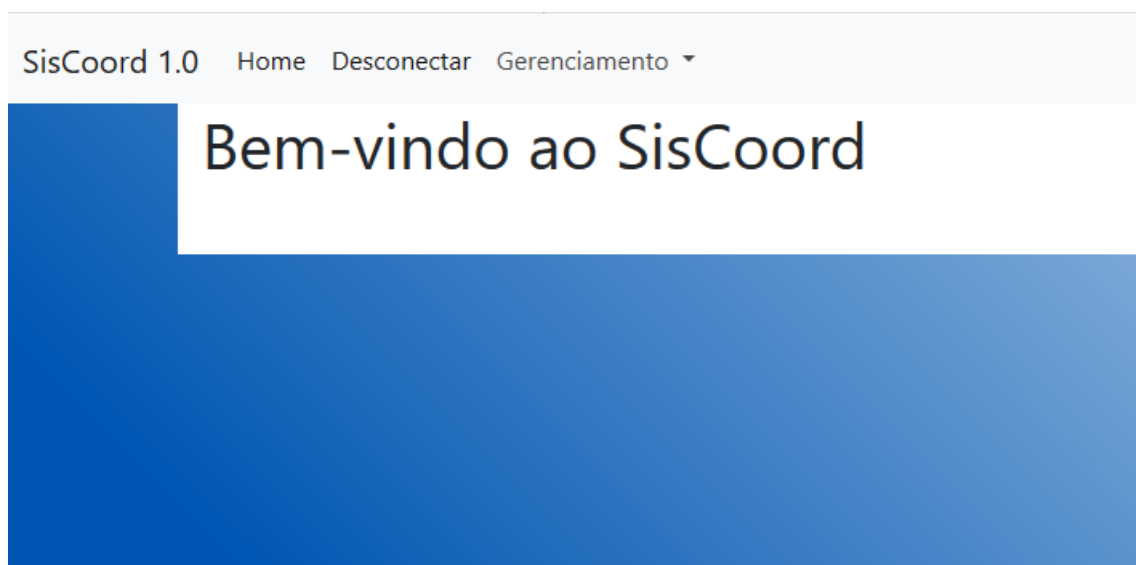


Figura 14 – Tela de boas-vindas, no *laptop*, após o *login*

Desta forma podemos utilizar o SisCoord tanto no dispositivo móvel quanto no *desktop* ou *laptop*, a seguir veremos o que pode ser gerenciado no SisCoord.

A screenshot of the 'SisCoord - Módulo de gestão de cursos' page. The header shows 'SisCoord 1.0' and a hamburger menu icon. The main title is 'SisCoord - Módulo de gestão de cursos'. Below it are three input fields: 'Nome do Curso' with the placeholder 'Nome', 'Sigla do Curso (suap)' with the placeholder 'sigla', and a dropdown menu for 'Selecione o coordenador do curso' with 'Adão' selected. A blue 'Incluir' button is positioned below the dropdown. Underneath is the section 'Lista de Cursos Cadastrados' which contains a table. The table has a dark header with columns '#', 'Nome', 'Sigla', and 'Del'. The first row of data shows the number '1', the name 'Sistemas de Informação', the sigla 'CBSI', and a button with a trash icon and the text 'Excluir'.

#	Nome	Sigla	Del
1	Sistemas de Informação	CBSI	Excluir

Figura 15 – Tela de gestão de curso

Nesta parte do sistema é possível manter os cursos. Com um clique sobre o nome do curso ou sigla é possível alterar o mesmo.

SisCoord 1.0

SisCoord - Módulo de gestão de Setores

Nome do Setor

Sigla do Setor (suap)

Incluir

Lista de Setores Cadastrados

#	Nome do Setor	Sigla do Setor	Del
1	GEPEX	GEPEX	Excluir
2	CORAE	CORAE	Excluir

Figura 16 – Tela de gestão de setores

Assim as outras partes, gestão de professores, taes, coordenadores de curso e tipos de documentos, do sistema segue este mesmo padrão demonstrado até aqui, uma parte diferente do SisCoord é o *upload* de arquivos, que será explicado a seguir.

SisCoord 1.0

SisCoord - Módulo de gestão de documentos

Selezione o tipo de documento

Data

Escolher arquivo Browse... No file selected.

Novo

Lista

#	Doc	Data	Del
4	Apresentacao.pdf	2019-05-12	Excluir

Figura 17 – Tela de gestão de documentos

Nesta parte do sistema, o arquivo é enviado ao servidor, é guardado a data de envio, assim quando for necessário utilizar o mesmo, o usuário deve clicar sobre o *link* do arquivo, desta forma este ficará disponível para leitura ou para fazer o *download*.

4.4 Discussão

A qualquer momento podemos inserir um sistema novo, nesta arquitetura, desde que o mesmo possua os descritores, para atender, por exemplo, uma situação emergencial, na Figura 6 esta situação está representada como ParaAddNovo, esta é uma das vantagens da arquitetura, por não conhecer o código fonte, pode ser uma desvantagem até construir um descritor, conseqüentemente um conector.

O orquestrador pode monitorar diferentes recursos de diferentes sistemas ao mesmo tempo em caso de uma situação emergencial, este é um requisito importante para que usuário tenha maiores informações para tomar determinadas decisões.

Parte V

Conclusão

5 Considerações finais

Nesta proposta foi desenhada e explicada a arquitetura, baseada em SoIS, por isso foi feito um estudo mais aprofundado sobre arquiteturas, com exemplos e explicações dos mesmos, foi feito a telas do protótipo e demonstrado um sistema web construído durante este trabalho para gestão de dados acadêmicos, o SisCoord.

O SisCoord é um sistema proposto como um dos constituintes, feito com objetivo de validar arquitetura, nele foram realizados pequenos testes e construídos descritores para o sistema. SisCoord é também a automação do processo de emissão e controle dos documentos, garantindo a emissão e gestão de cada documento com segurança e agilidade.

Produzir o presente trabalho foi de suma importância para ampliar os conhecimentos do autor sobre o tema tão recente na realidade profissional na sua área de estudo, desta forma também este ajuda a contribuir com o avanço nesta área.

Em trabalhos futuros esta pode ser melhorada, ser feita de maneira diferente, e ou pode ocorrer a implementação da mesma, construindo a sua aplicação ou aplicativo, destacando melhor as dificuldades e facilidades encontradas, como a questão da interoperabilidade entre os sistemas.

Referências

- AVASARALA, S. *Selenium WebDriver practical guide*. [S.l.]: Packt Publishing Ltd, 2014. Citado na página 38.
- BERNARDES, J. F.; ABREU, A. F. d. A contribuição dos sistemas de informações na gestão universitária. INPEAU, 2004. Citado na página 43.
- BOARDMAN, J.; SAUSER, B. System of systems-the meaning of of. In: IEEE. *2006 IEEE/SMC International Conference on System of Systems Engineering*. [S.l.], 2006. p. 6–pp. Citado na página 33.
- BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. *UML: guia do usuário*. [S.l.]: Elsevier Brasil, 2006. Citado na página 47.
- CARVALHO, R. S. et al. Integração entre o sistema de gestão acadêmica e o sistema de gestão da aprendizagem: identificando necessidades e prototipando requisitos favoráveis a prática docente. *Revista Brasileira de Computação Aplicada*, v. 4, n. 1, p. 81–91, 2012. Citado na página 43.
- CECIN, F. R. Freemmg: uma arquitetura cliente-servidor e par-a-par de suporte a jogos maciçamente distribuídos. 2005. Citado 2 vezes nas páginas 13 e 31.
- CHRISTENSEN, E. et al. *Web services description language (WSDL) 1.1*. [S.l.]: Citeseer, 2001. Citado na página 37.
- D'ANUNCIAÇÃO, G. T. UFPE, 2021. Disponível em: <https://www.cin.ufpe.br/~gta/rup-vc/core.base_rup/guidances/concepts/system_architecture_5F3B1E17.html>. Citado na página 29.
- DOSPINESCU, O. et al. Rest soa orchestration and bpm platforms. *Informatica Economica*, v. 21, n. 1, 2017. Citado na página 51.
- FERNANDES, J. et al. A conceptual model for systems-of-information systems. In: IEEE. *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*. [S.l.], 2019. p. 364–371. Citado na página 43.
- FERNANDES, J. C.; NETO, V. V. G.; SANTOS, R. P. d. Interoperability in systems-of-information systems: A systematic mapping study. In: *Proceedings of the 17th Brazilian Symposium on Software Quality*. [S.l.: s.n.], 2018. p. 131–140. Citado 2 vezes nas páginas 33 e 42.
- GAMA, E. et al. *Padrões de projeto: Reutilizáveis de software orientado a objeto*. [S.l.]: Bookman,, 2000. Citado na página 31.
- GAMMA, E. *Padrões de projetos: soluções reutilizáveis*. [S.l.]: Bookman editora, 2009. Citado na página 30.
- GORLA, J. P. F.; FOSCHINI, I. J. Arquitetura para desenvolvimento web baseado em jsf 2.0 utilizando padrões de projeto. *Revista TIS*, v. 2, n. 3, 2014. Citado 2 vezes nas páginas 13 e 30.

- JR, H. E. Engenharia de software na prática, 1ª edição. *São Paulo: Editora Novatec*, 2010. Citado na página 47.
- KITCHENHAM, B.; CHARTERS, S. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. 2007. Citado na página 47.
- LI, X. et al. Good bot, bad bot: Characterizing automated browsing activity. Citado na página 38.
- LUCIDCHART. *LUCIDCHART*. 2019. Disponível em: <<https://www.lucidchart.com>>. Acesso em: 17, setembro 2019. Citado na página 47.
- MANJARI, K. U. et al. Extractive text summarization from web pages using selenium and tf-idf algorithm. In: IEEE. *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)*(48184). [S.l.], 2020. p. 648–652. Citado na página 38.
- MICROSOFT. *Elementos de uma arquitetura*. 2019. Disponível em: <msdn.microsoft.com>. Acesso em: 04 abril 2019. Citado 2 vezes nas páginas 13 e 30.
- NEWMAN, S. Building microservices: designing fine-grained systems. O'Reilly Media, Inc, 2015. Citado na página 32.
- PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, Elsevier, v. 64, p. 1–18, 2015. Citado na página 47.
- RAMYA, P.; SINDHURA, V.; SAGAR, P. V. Testing using selenium web driver. In: IEEE. *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. [S.l.], 2017. p. 1–7. Citado na página 38.
- REIS, P. N. C.; PITASSI, C.; BOUZADA, M. A. Os fatores que explicam o grau de aceitação de um sistema de informação acadêmica: um estudo de caso com docentes de uma ies privada. *Revista Eletrônica de Sistemas de Informação*, v. 12, n. 3, 2013. Citado na página 43.
- SALEH, M.; ABEL, M.-H. Moving from digital ecosystem to system of information systems. In: IEEE. *2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. [S.l.], 2016. p. 91–96. Citado 4 vezes nas páginas 13, 33, 34 e 42.
- SALEH, M.; ABEL, M.-H. Resources management and decision support in a system of information systems. In: IEEE. *2016 International Conference on Industrial Informatics and Computer Systems (CIICS)*. [S.l.], 2016. p. 1–5. Citado na página 42.
- SALEH, M.; ABEL, M.-H. Modeling and developing a system of information systems for managing heterogeneous resources. In: IEEE. *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. [S.l.], 2017. p. 2672–2677. Citado 2 vezes nas páginas 33 e 42.
- SALEH, M.; ABEL, M.-H. System of information systems and organizational memory. In: IEEE. *2018 13th Annual Conference on System of Systems Engineering (SoSE)*. [S.l.], 2018. p. 477–484. Citado na página 42.

- SALEH, M.; ABEL, M.-H. System of information systems to support learners (a case study at the university of technology of compiègne). *Behaviour & Information Technology*, Taylor & Francis, v. 37, n. 10-11, p. 1097–1110, 2018. Citado na página 42.
- SANTOS, J. P. R. dos et al. Selenium as a free tool to test for java web application. Citado na página 38.
- SELENIUM. *WebDriver*. 2020. Disponível em: <<https://www.selenium.dev/documentation/en/webdriver/>>. Citado 2 vezes nas páginas 38 e 52.
- SILVEIRA, P. et al. *Introdução à arquitetura e design de software: uma visão sobre a plataforma Java*. [S.l.]: Rio de Janeiro: Elsevier, 2012. ISBN 9788535250299. Citado na página 29.
- TEIXEIRA, P. G. et al. The status quo of systems-of-information systems. In: IEEE. *2019 IEEE/ACM 7th International Workshop on Software Engineering for Systems-of-Systems (SESoS) and 13th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (WDES)*. [S.l.], 2019. p. 34–41. Citado 2 vezes nas páginas 33 e 47.
- TERUEL, E. Arquitetura de sistemas para web com java utilizando design patterns e frameworks. ed. *Ciência Moderna*, 2012. Citado na página 31.

Apêndices

APÊNDICE A – O conector

Neste apêndice está o conector, construído em java, para demonstrar sua aplicação e ser utilizado pelo o orquestrador.

Lista A.1 – Conector para o sistema SisCoord

```

1  import java.io.BufferedReader;
2  import java.io.IOException;
3  import java.io.InputStreamReader;
4  import java.io.UnsupportedEncodingException;
5  import java.util.ArrayList;
6  import java.util.List;
7  import org.apache.http.Consts;
8  import org.apache.http.HttpResponse;
9  import org.apache.http.NameValuePair;
10 import org.apache.http.client.entity.UrlEncodedFormEntity;
11 import org.apache.http.client.methods.HttpGet;
12 import org.apache.http.client.methods.HttpPost;
13 import org.apache.http.impl.client.DefaultHttpClient;
14 import org.apache.http.message.BasicNameValuePair;
15 import org.apache.http.util.EntityUtils;
16
17 /**
18  *
19  * @author CARLOS
20  */
21
22 /*
23 bibliotecas (jar) utilizadas:
24 org.apache.commons.codec-1.6
25 httpcore-4.2.2
26 httpclient-4.2.3
27 commons-logging-1.1.1-api
28 */
29 public class Conector {
30
31     private final DefaultHttpClient client = new DefaultHttpClient();
32
33     public boolean login(final String url, final String user, final String
        password) throws

```

```
34         UnsupportedEncodingException, IOException {
35
36         /* Método POST */
37         final HttpPost post = new HttpPost(url);
38         boolean result = false;
39
40         /* Configura os parâmetros do POST */
41         final List<NameValuePair> nameValuePairs = new ArrayList<NameValuePair>();
42         nameValuePairs.add(new BasicNameValuePair("usuario", user));
43         nameValuePairs.add(new BasicNameValuePair("password", password));
44
45         /*
46          * Codifica os parametros.
47          */
48         post.setEntity(new UrlEncodedFormEntity(nameValuePairs, Consts.UTF_8));
49
50         /* Define navegador */
51         post.addHeader("User-Agent", "Mozilla/5.0 (Windows NT 5.1; rv:18.0)
52             Gecko/20100101 Firefox/18.0");
53
54         /* Efetua o POST */
55         HttpResponse response = client.execute(post);
56
57         /* Resposta HTTP
58         System.out.println("Login form get: " + response.getStatusLine());
59
60         /*
61          * Consome o conteúdo retornado pelo servidor
62          */
63         EntityUtils.consume(response.getEntity());
64
65         /*
66          * Testa se o login funcionou.
67          */
68         final HttpGet get = new
69             HttpGet("http://localhost:8080/siscoord/cms/home.jsp");
70         response = client.execute(get);
71
72         /*
73          * Verifica se a String: "Bem-vindo" está presente
74          */
75         if (checkSuccess(response)) {
```

```
74         System.out.println("Conexao Estabelecida!");
75         result = true;
76     } else {
77         System.out.println("Login não-efetuado!");
78     }
79
80     return result;
81 }
82
83 private boolean checkSuccess(final HttpServletResponse response) throws IOException {
84     final BufferedReader rd = new BufferedReader(new InputStreamReader(
85         response.getEntity().getContent()));
86     String line;
87     boolean found = false;
88     // System.out.println(rd.readLine());
89     /* Deixa correr todo o laço, mesmo achando a String, para consumir o
90        content */
91     while ((line = rd.readLine()) != null) {
92         //System.out.println(rd.readLine());
93         if (line.contains("Bem-vindo")) {
94             found = true;
95         }
96     }
97     return found;
98 }
99
100 public void openPage(final String url) throws IOException {
101     final HttpGet get = new HttpGet(url);
102     final HttpServletResponse response = client.execute(get);
103
104 }
105
106 public void close() {
107     client.getConnectionManager().shutdown();
108 }
109
110 public static void main(String[] args) {
111
112     Conector conector = new Conector();
113
114     String usuario = "bruno", senha = "bruno", nomeSetor =
```

```
        "SistemasDeInformação", siglaSetor = "SI";
115
116    try {
117        // Tenta efetuar login
118        boolean ok = conector.login("http://localhost:8080/siscoord/cms/login",
119            usuario, senha);
120        if (ok) {
121            // Acessa página restrita
122            conector.openPage(
123                "http://localhost:8080/siscoord/cms/setor?nomeSetor=" +
124                nomeSetor + "&siglaSetor=" + siglaSetor);
125            // System.out.println("fazendo o get!");
126        }
127        conector.close();
128    } catch (UnsupportedEncodingException ex) {
129        ex.printStackTrace();
130    } catch (IOException ex) {
131        ex.printStackTrace();
132    }
133 }
```

APÊNDICE B – Exemplo de descritor para *Selenium Web Driver*

Lista B.1 – Exemplo de um descritor para um serviço web não padrão

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <!--
3  IFG
4  2021-05-25
5  retorna um JSON com os eventos da empresa cadastrados
6
7  Online WSDL 1.1 SOAP generator 0.2
8  Julien Blitte
9  Com adaptações
10 -->
11 <definitions name="retorna um json com os eventos da empresa cadastrados"
12   targetNamespace="us.world_corp.my_soap_forum.wsdl"
13   xmlns:tns="us.world_corp.my_soap_forum.wsdl"
14   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
15   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
16   xmlns:xsd1="us.world_corp.my_soap_forum.xsd"
17   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
18   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
19   xmlns="http://schemas.xmlsoap.org/wsdl/">
20 <!-- definition of datatypes -->
21 <types>
22 <schema targetNamespace="us.world_corp.my_soap_forum.xsd"
23   xmlns="http://www.w3.org/2000/10/XMLSchema">
24 <element name="dados">
25 <complexType><all><element name="value" type="json" /></all></complexType>
26 </element>
27 <element name="void">
28 <complexType><sequence></sequence></complexType>
29 </element>
30 </schema>
31 </types>
32 <!-- response messages -->
33 <message name='returns_dados'>
34 <part name='dados' type='xsd:dados' />

```



```
34 </message>
35 <!-- request messages -->
36 <!-- server s services -->
37 <portType name='Consumo dos dados de eventos institucionais'>
38   <operation name='get'>
39     <output message='tns:returns_dados' />
40   </operation>
41 </portType>
42 <!-- server encoding -->
43 <binding name='Consumo dos dados de eventos institucionais_webservices'
44   type='tns:Consumo dos dados de eventos institucionais'>
45   <soap:binding style='rpc' transport='http://schemas.xmlsoap.org/soap/http' />
46   <operation name='get'>
47     <soap:operation soapAction='urn:xmethods-delayed-quotes#get' />
48     <output><soap:body use='encoded' namespace='urn:xmethods-delayed-quotes'
49       encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' /></output>
50   </operation>
51 </binding>
52 <!-- access to service provider -->
53 <service name='API RestFULL'>
54   <port name='API RestFULL_0' binding='Consumo dos dados de eventos
55     institucionais_webservices'>
56     <soap:address location='http://localhost:5000/events' />
57   </port>
58 </service>
59 </definitions>
```

Anexos

Proposta de uma arquitetura de sistema de sistemas de informação para gestão de dados acadêmicos

Carlos Eduardo da Rocha, carloscaed2011@gmail.com ¹

Victor Hugo Lázaro Lopes, victor.lopes@ifg.edu.br ¹

¹NumbERS/NETI - IFG Inhumas

Introdução

O tempo necessário para se coletar, organizar e processar as informações distribuídas em diferentes Sistemas de Informação (SIs) nas organizações, e especificamente em instituições de ensino públicas, quando se faz necessário, seja qual for o objetivo, como para a produção de relatórios, documentos gerenciais, realização de análises, organização da vida acadêmica, dentre outros, pode ser demasiadamente grande. Além disso, é complexo encontrar pontos em comum que proporcionem uma análise sistêmica do caso do aluno e ainda é muito comum a estas instituições a inexistência de recursos financeiros, em tempo hábil, para o desenvolvimento de novos sistemas de informação, que resolvam questões específicas, a todo momento. Por esse motivo, é comum nestes cenários a existência de múltiplos SIs distintos, que naturalmente não possuem interoperabilidade, gerando complexidade às tarefas de gerenciamento dos dados acadêmicos da instituição. Neste sentido, o presente trabalho apresenta uma arquitetura baseada em Sistema-de-Sistemas de Informação (SdSI ou SoIS, do inglês System-of-Information Systems), que é um tipo especializado de SoS (Systems-of-Systems), para gestão integrada dos dados acadêmicos, a fim de minimizar este tempo, a complexidade e o custo. Esta proposta tem como desafio gerar a garantia da interoperabilidade esperada entre sistemas de informação já existentes, tirando a necessidade de se ter projetos específicos para desenvolver, integrar e implantar novos sistemas. Por se tratar de uma arquitetura de alto nível, apesar da presente proposta ser direcionada a uma instituição de ensino pública, observa-se aderência às demandas de outros tipos de organizações, tanto públicas quanto privadas.

Referencial Teórico

O conjunto de sistemas independentes e interoperantes combinados para atingir um objetivo comum é chamado de Sistema-de-Sistemas (SdS ou SoS, do inglês System-of-Systems), tratando-se de uma abordagem arquitetural de alto nível com foco na integração entre artefatos de *software* (BOARDMAN e SAUSER, 2006). Trata-se de um paradigma arquitetural em que tais artefatos são sistemas computacionais, chamados de sistemas constituintes, que carregam as seguintes categorias (SALEH e ABEL, 2017):

- Independência Operacional: cada sistema constituinte é independente.
- Independência Gerencial: eles funcionam e são administrados de forma independente.
- Desenvolvimento Evolutivo: capacidade de que os sistemas sigam sendo desenvolvidos dinamicamente, só juntando serviços existentes para se fazer um novo.
- Comportamento Emergente: situações emergenciais que geram a necessidade de uma arquitetura que permita montarmos novos sistemas mais rapidamente, desde que os serviços já estejam prontos.
- Arquitetura Dinâmica: que a arquitetura possa ir sendo alterada de acordo com a necessidade.

A partir desta abordagem, surge uma classe especial de SoS em que um ou mais destes constituintes são SIs, potencialmente independentes interoperando segundo um processo de negócio, chamado Sistema-de-Sistemas de Informação (SdSI ou SoIS, do inglês System-of-Information Systems) (FERNANDES et al., 2018). Um SI é um conjunto de componentes associados que coletam (ou recuperar), processar, armazenar e distribuir informações (TEIXEIRA et al., 2019).

Material e Método

O estudo bibliográfico foi a etapa inicial, teve como fim averiguar outras pesquisas na

mesma linha de pesquisa deste tema, de forma a prover uma visão geral sobre as possíveis abordagens para a solução do problema aqui enfrentado. Neste sentido, foi empregada a metodologia de revisão sistemática da literatura, que emprega um protocolo bem definido, a se considerar as etapas: Planejamento, Condução e Relatório (Kitchenham e Charles, 2007), (Petersen et al., 2015).

A partir dessas buscas, foram feitas leituras e foram escolhidos artigos científicos para serem analisados, segundo o artigo “The status quo of systems-of-information systems” (TEIXEIRA et al., 2019), para melhor mostrar o estado da arte, de forma que ao fim do processo, constatou-se que a arquitetura SoIS mostra-se como a solução mais viável.

Em paralelo ao estudo bibliográfico, sendo um deles: O artigo “Moving from digital ecosystem to system of information systems” (SALEH, 2016), em português, Passando do ecossistema digital para o sistema de sistemas de informação, foi feito a análise de sistema (modelagem), a qual consistiu no entendimento do problema e assim, também, na definição do escopo do trabalho, foi utilizada a ferramenta lucidchart para o desenho da arquitetura proposta. A UML (Unified Modeling Language - Linguagem de Modelagem Unificada) provê diversos tipos de diagramas para a especificação dos modelos. Segundo Booch (2006), a UML proporciona uma forma padrão para a preparação de planos de arquitetura de projetos de sistemas.

Foi utilizada a técnica de prototipação, processo que tem como objetivo facilitar o entendimento dos requisitos, apresentar conceitos e funcionalidades do software. Desta forma, pode-se propor uma solução adequada para o problema.

Resultados e Discussão

No geral foram criadas as telas do protótipo do orquestrador, sendo um sistema de informação que agrupa e administra as informações adquiridas pelos sistemas constituintes, a modelagem da arquitetura inicial da proposta, que é ilustrada na Figura 1.

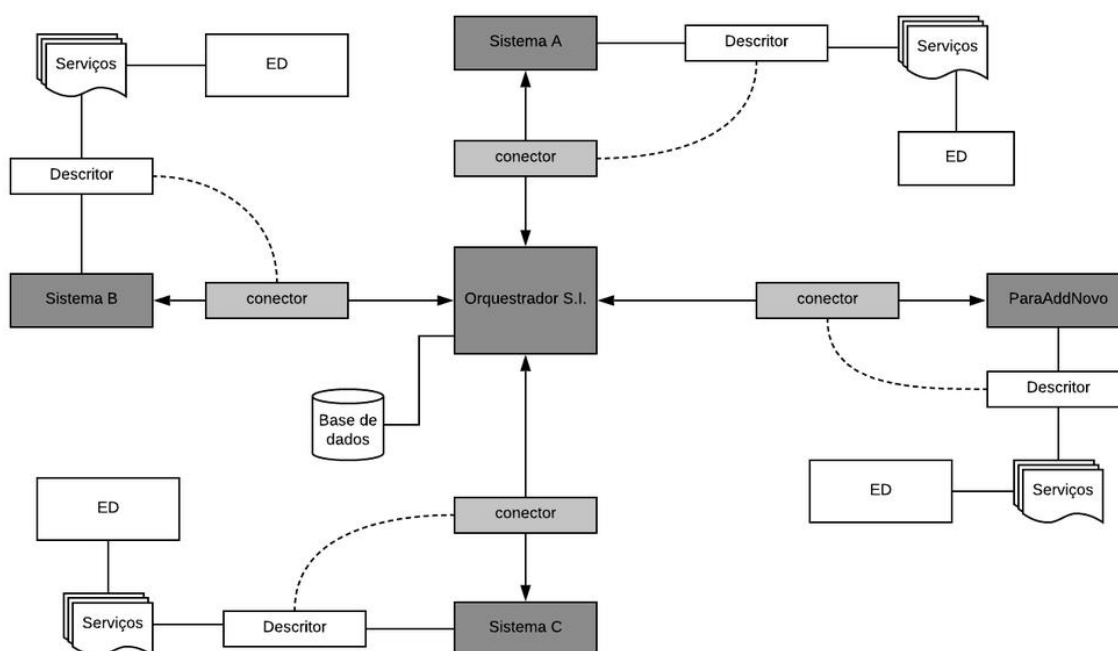


Figura 1. Arquitetura proposta.

A arquitetura proposta, ilustrada na Figura 1 é baseada em diversos componentes: O orquestrador; Sistemas Constituintes; Os Descritores; Os Conectores; Base de dados; Serviços; ED (Estrutura de Dados), que podem ser assim descritos:

O orquestrador é o sistema líder, nele encontraremos as opções para acessar todos os outros sistemas, assim para ter acesso ao mesmo precisa de um alto nível de segurança, nele se concentra toda a acessibilidade aos sistemas constituintes, coletando e administrando os dados.

Os sistemas constituintes, nesta arquitetura especificamente, apresentada na Figura 1, eles são genéricos, sendo os sistemas de informação que serão consumidos pelo orquestrador.

Os descritores descrevem, por meio de um wsdl, Web Services Description Language (Linguagem de Descrição de Serviços Web), como será feito o uso pelo orquestrador de determinado recurso de um sistema constituinte.

Os conectores são artefatos de software que vai usar um descritor para consumir os serviços oferecido. E ele pode ser desenvolvido das mais diferentes formas, usando diferentes linguagens de programação.

A base de dados é para guardar as senhas para realizar login nos sistemas constituintes e do próprio orquestrador, no qual o banco de dados está conectado, também para persistir os descritores.

Serviços este são os que os descritores oferecem.

ED vai ser usado para salvar os dados no orquestrador. Uma tabela vai ser dinamicamente criada para cada ED, assim que ele for salvo.

Discussão

Observa-se que a arquitetura proposta permite o desenvolvimento de novas aplicações no orquestrador baseando-se em serviços que possam ser fornecidos por sistemas constituintes já existentes, gerando a flexibilidade exigida nas organizações atualmente. Neste tipo de arquitetura a complexidade que envolve a interoperabilidade se concentra na construção dos descritores e na definição dos conectores. Em um serviço oferecido por um sistema constituinte em forma de *Webservice*, por exemplo, o descritor deve prover os passos necessários para o consumo do serviço, e o conector pode ser um simples artefato de *software* que seja capaz de implementar os passos necessários, independentemente da linguagem de programação estabelecida.

Referências Bibliográficas

BOARDMAN, John; SAUSER, Brian. System of Systems-the meaning of of. In: **2006 IEEE/SMC International Conference on System of Systems Engineering**. IEEE, 2006. p. 6 pp.

SALEH, Majd; ABEL, Marie-Hélène. Modeling and developing a system of information systems for managing heterogeneous resources. In: **2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)**. IEEE, 2017. p. 2672-2677.

FERNANDES, Juliana Costa; NETO, Valdemar V. Graciano; SANTOS, Rodrigo Pereira dos. Interoperability in Systems-of-Information Systems: A Systematic Mapping Study. In: **Proceedings of the 17th Brazilian Symposium on Software Quality**. ACM, 2018. p. 131-140.

TEIXEIRA, Paulo Gabriel et al. The status quo of systems-of-information systems. In: **Proceedings of the 7th International Workshop on Software Engineering for Systems-of-Systems and 13th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems**. IEEE Press, 2019. p. 34-41.

B. Kitchenham and S. Charters, “Guidelines for performing systematic literature reviews in software engineering,” **Keele University and Durham University Joint Report, Tech. Rep.** EBSE 2007-001, 2007.

K. Petersen, S. Vakkalanka, and L. Kuzniarz, “Guidelines for conducting systematic mapping studies in software engineering: An update,” **Information and Software Technology**, vol. 64, pp. 1–18, 2015.

SALEH, Majd; ABEL, Marie-Hélène. Moving from digital ecosystem to system of information systems. In: **2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD)**. IEEE, 2016. p. 91-96.

LUCIDCHART. lucidchart, 2019. Disponível em: <<https://www.lucidchart.com>>. Acesso em: 17, setembro 2019.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. UML: guia do usuário. **Elsevier Brasil**, 2006.