UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

VINÍCIUS DA ROSA SILVA

Restrições de integridade dentro do contexto da Web Semântica

RECIFE
2023

VINÍCIUS DA ROSA SILVA

Restrições de integridade dentro do contexto da Web Semântica

Trabalho de Conclusão de Curso apresentado ao Programa de Bacharelado em Sistemas de Informação, como parte dos requisitos necessários à obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Frederico Luiz Gonçalves de Freitas.

Coorientador: Prof. Pedro Porfirio Muniz Farias.

RECIFE

2023

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Silva, Vinícius Da Rosa.
Restrições de integridade dentro do contexto da Web Semântica / Vinícius Da Rosa Silva. - Recife, 2023.
78 p : il., tab.

Orientador(a): Frederico Luiz Gonçalves De Freitas
Cooorientador(a): Pedro Porfirio Muniz Farias
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Sistemas de Informação - Bacharelado, 2023.
10.

1. Web. 2. Semântica. 3. Interfaces. 4. RESTful. 5. Restrições. I. De Freitas, Frederico Luiz Gonçalves. (Orientação). II. Farias, Pedro Porfirio Muniz. (Coorientação). IV. Título.

000 CDD (22.ed.)

Vinícius da Rosa Silva

“Restrições de integridade dentro do contexto da Web Semântica”

TCC apresentado ao Curso de Sistemas de Informação da Universidade Federal de Pernambuco, Centro de Informática, como requisito para a obtenção do título de Bacharel em Sistemas de Informação.

Recife, 26 de setembro de 2023

BANCA EXAMINADORA

_____

Prof. Dr. Frederico Luiz Gonçalves de Freitas.(Orientadora)

Centro de Informática / UFPE

_____

Prof. Dr. Pedro Porfirio Muniz Farias. (Examinador Interno)

Centro de Informática / UFPE

_____

Prof. Dr. Nelson Souto Rosa. (Examinador Interno)

Centro de Informática / UFPE

Dedico estre trabalho a todos os meus professores, que me ajudaram em minha trajetória acadêmica e profissional, em especial aos professores Frederico Luiz, Pedro Porfirio e Vinícius Garcia. Também à minha família, que sempre esteve ao meu lado e me apoiando nos momentos mais difíceis.

“Não esmoreçam ou se desesperem: Se tiverdes fé, estareis em vantagem”.

AGRADECIMENTOS

Agradeço a Deus por guiar meus passos, a meus pais pelo amor incondicional e aos meus professores por compartilharem seu conhecimento e inspiração. Sou profundamente grato por todas as bênçãos, apoio e orientação que recebi ao longo da minha jornada.

Fim da jornada,
Tese pronta para brilhar,
Novos sonhos surgem.

“Não há sentido em tentar ser preciso se sequer houver conceituação do que se está falando” - John von Neumann

RESUMO

A verificação da integridade de dados é uma condição importante para o funcionamento adequado de sistemas web que permitem a inserção de informação. Serviços Web semânticos são serviços web incrementados com anotações semânticas. Interfaces RESTful semânticas adotam a hipótese de um mundo fechado localmente para especificar serviços web que atendam tais restrições de integridade. Este trabalho de conclusão de curso objetiva fazer uma revisão de literatura em relação à especificação de restrições de integridade dentro do contexto da web-semântica, no que elas consistem e nos tópicos associados às mesmas.

Palavras-chave: Web; Semântica; Interfaces; RESTful; Restrições.

ABSTRACT

Data integrity verification is an important condition for the proper functioning of Web systems that allow data insertion. Semantic web services are web services enhanced with semantic annotations. Semantic RESTful interfaces adopt the assumption of a locally closed world to specify web services that meet such integrity constraints. This undergraduate thesis aims to conduct a literature review regarding the specification of integrity constraints within the context of the semantic web, what they entail, and related topics.

Keywords: Web; Semantics; Interfaces; RESTful; Constraints.

LISTA DE ILUSTRAÇÕES

Figura 1 –	Exemplo de código OWL	15
Figura 2 –	Interação de Servidores num mundo fechado localmente	15
Figura 3 –	Suporte dos Verbos HTTP por área de intersecção nas API's públicas	17
Figura 4 –	Comunicação cliente-servidor nos primórdios da Web	17
Figura 5 –	Distribuição das API's por número de recursos	18
Figura 6 –	Descrição de Recurso em OpenAPI	18
Figura 7 –	Estrutura da Web Semântica	
Figura 8 –	Busca em SPARQL	
Figura 9 –	Modelo Descritivo WS3	
Figura 10 –	Anotações semânticas inseridas na descrição de serviços	
Figura 11 –	Ontologias descrevendo serviços	
Figura 12 –	Legenda	
Figura 13 –	Interface Implementada em Java	
Figura 14 –	Anotações à nível de classe	
Figura 15 –	Distribuição das Abordagens de Descrição RESTful em Slimani	
Figura 16 –	Descrição em OWL de propriedades com restrições de integridade	
Figura 17 –	Descrição de API RESTful com anotações semânticas	
Figura 18 –	Exemplo de definição de obrigatoriedade de propriedade em OpenAPI	

LISTA DE TABELAS

Tabela 1 – Palavras Chaves por Artigo 16

Tabela 2 – Incidência dos autores por artigo 16

Tabela 3 – Serviços Web Semânticos 51

SUMÁRIO

1	**INTRODUÇÃO**	**14**
2	**REVISÃO SISTEMÁTICA DE LITERATURA**	**15**
2.1	TERMOS RELEVANTES	16
2.2	AUTORES	16
2.3	REFERÊNCIAS DE ARTIGOS	16
3	**RESTRIÇÕES DE INTEGRIDADE**	**15**
3.1	CONTEXTO DE MUNDO ABERTO	16
3.2	CONTEXTO DE MUNDO FECHADO	16
3.3	RESTRIÇÕES DE INTEGRIDADE	16
3.4	ESTUDO DE CASO: RESTRIÇÕES DE INTEGRIDADE EM BANCOS DE DADOS RELACIONAIS	16
3.5	RESTRIÇÕES DE INTEGRIDADE DE API's	16
4	**SERVIÇOS WEB, WEB API's E RESTful**	**18**
4.1	EVOLUÇÃO DA ARQUITETURA DE SERVIÇOS DA WEB	16
4.2	WEB API's	16
4.3	PRINCÍPIOS RESTful	16
4.4	OpenAPI	16
5	**SERVIÇOS WEB SEMÂNTICOS**	**18**
5.1	WEB-SEMÂNTICA	16
5.2	LINKED DATA	16
5.3	SERVIÇOS WEB SEMÂNTICOS	16
5.3.1	**Serviços web semânticos usando anotações nas descrições dos serviços**	**18**
5.3.2	**Ontologias descrevendo serviços**	**18**
5.3.3	**Anotações em ontologias especificando interfaces semânticas**	**18**
5.4	INTERFACES ABSTRATAS PARA SERVIÇOS WEB	16
5.5	SERIN – INTERFACES RESTful SEMÂNTICAS	16
5.6	WEB API's SEMÂNTICAS RESTful	16
6	**RESTRIÇÕES DE INTEGRIDADE NA WEB SEMÂNTICA**	**18**

6.1	RESTRIÇÕES DE INTEGRIDADE EM INTERFACES RESTful SEMÂNTICAS	16
6.2	RESTRIÇÕES DE INTEGRIDADE EM SERVIÇOS WEB SEMÂNTICOS	16
6.3	ESTUDO DE CASO: RESTRIÇÕES EM OpenAPI E RESTRIÇÕES SEMÂNTICAS DE INTEGRIDADE	16
7	**CONCLUSÃO**	**18**
	REFERÊNCIAS	**19**

1 INTRODUÇÃO

A verificação da integridade de dados nos processos de inserção em sistemas Web atuais é essencial para a garantia do funcionamento perfeito do sistema e para a proteção do mesmo e de seus usuários. A inserção de dados não verificados pode acarretar inúmeros problemas como a corrupção de dados pré-existentes no sistema, a não adequação do sistema à métricas externas e internas de compliance nas organizações e até mesmo na não-continuidade do serviço a depender da gravidade dos problemas gerados.

A falta de adequação de sistemas a critérios de integridade de dados já foi explorada por exemplo em situações de ataques à sistemas de transporte de dados conforme demonstrado por Lin (2018). Em pesquisas previamente realizadas (HAUG, 2011), foi constatado que a baixa integridade e qualidade de dados também podem afetar negócios e empresas em diversos níveis como em relação à satisfação do cliente, a diminuição da performance das atividades desempenhadas pelos funcionários e até mesmo a diminuição do nível de satisfação dos desenvolvedores envolvidos na manutenção dos projetos.

Todavia, para implementar nos sistemas as verificações adequadas de dados em todos os seus níveis, é necessário que primeiramente sejam definidas restrições de integridade que garantam que somente dados que cumpram todos os requisitos possam ser adotados no sistema. No processo de descrição de interfaces e aplicações Web, é interessante que haja conhecimento por exemplo do contexto em que a aplicação está imersa em relação à sua postura no que tange a informações presentes ou ausentes no sistema.

Em contextos de mundo aberto, assume-se que a informação existente no sistema pode ser incompleta, permitindo a adição de detalhes e especificações que não foram previamente definidos na ontologia que explica o mundo dinamicamente. Isso inclui a ideia de que a ausência de informação não implica automaticamente na sua inveracidade, facilitando a adaptação da ontologia para a incorporação de novas informações.

Como destacado por Moore (2015), OWA (do inglês, *Open-World Assumption*) ou Presunção de mundo aberto, está predicada na suposição de que o valor de

verdade de uma sentença ou hipótese é independente de ser reconhecida como verdadeira por um agente ou observador único. Em contraste com contextos de mundo fechado, as especificações dos sistemas são pré-definidas e conhecidas pelas partes envolvidas, e aquelas especificações e informações que não são previamente conhecidas são dadas como falsas a partir da presunção da falsidade por ausência de informação.

McIlraith (2001) explica que Serviços Web Semânticos são serviços Web que devem fornecer informações sobre seus serviços, propriedades, interfaces de execução, pré e pós-condições, em um formato legível por máquina. Serviços Web semânticos podem ser implementados utilizando diferentes modelos arquiteturais e princípios distintos, todavia, na situação específica de Serviços Web semânticos RESTful, os recursos gerenciados, suas propriedades, suas relações internas e seus atributos precisam estar anotados para viabilizar os processos que são efetivados sobre esses serviços. (SALVADORI, 2019)

Interfaces RESTful Semânticas são interfaces que, implementando descrições semânticas num contexto de mundo fechado ou aberto, são capazes de descrever propriedades de sistemas que implementam princípios e conceitos arquiteturais RESTful, como: a ausência de permanência de estado, a descrição clara de recursos e a padronização da interface com a utilização de anotações (PATCH, POST, GET, DELETE etc., por exemplo). Tais anotações são semânticas pois descrevem compreensiva e formalmente a partir dos construtos da linguagem em que são implementadas as especificações dessas interfaces que representam as características dos sistemas.

Um exemplo pode ser encontrado no trabalho de Rauf (2015) onde interfaces semânticas são utilizadas para representar processos dentro de um sistema de locação de quartos de hotel que implementa conceitos RESTful. Nesse contexto, restrições de integridade são essenciais à descrição completa do sistema e de suas limitações, sendo parte integral do processo descritivo das funcionalidades da aplicação.

Uma Revisão Sistemática de Literatura (RSL) foi conduzida para explorar informações relacionadas às restrições de integridade relativas aos serviços Web

semânticos, às Interfaces RESTful semânticas e tópicos correlatos. A busca abrangeu uma variedade de fontes, incluindo artigos acadêmicos, trabalhos de pesquisa, documentação técnica e literatura relevante disponível publicamente em repositórios de acesso aberto como Arxiv, de acesso privado como ScienceDirect e em mecanismos de indexação massiva como o Google Scholar.

Após a coleta de um conjunto substancial de fontes, a informação resultante foi analisada e avaliada quanto à sua relevância e contribuição para o entendimento abrangente do tema em questão. Os dados obtidos a partir dessa revisão sistemática da literatura serviram como a base para a construção e desenvolvimento de todo o texto, fornecendo percepções valiosas e conceituações adequadas para apoiar as afirmações e argumentos apresentados ao longo do trabalho. Isso assegurou a credibilidade, fundamentação teórica e a confiabilidade das informações compartilhadas neste estudo sobre Web semântica e restrições de integridade.

2 REVISÃO SISTEMÁTICA DE LITERATURA

A escolha de uma RSL como metodologia deste estudo foi motivada não pela ausência de dados sobre o tema em ferramentas públicas de indexação, mas sim para aprimorar a filtragem de informações e identificar trabalhos relacionados de maneira mais eficiente. Nesse contexto, utilizamos os artigos compilados por Cheron (2021) como uma base sólida para fundamentar nossa metodologia, dada a proximidade dos tópicos do trabalho desenvolvido pelo referido autor em relação ao presente trabalho.

Cheron (2021) havia realizado um trabalho que abordava temas relacionados às Interfaces RESTful semânticas e Serviços Web semânticos RESTful e disponibilizava um repositório de artigos associados ao tema em seu perfil no Github. Essa compilação de artigos, totalizando 26 no espaço amostral, serviu como ponto de partida para as investigações realizadas nesta pesquisa. Essa abordagem permitiu uma análise mais completa e estruturada dos estudos relacionados ao nosso tema de interesse a partir dos três pilares:

- **Palavras-Chave:** As palavras-chave de cada artigo são escolhidas pelos seus respectivos autores justamente para guiar a busca de novos autores e para possibilitar uma indexação dos artigos condizentes com os temas abordados e

acesso mais simples à informação. Consequentemente, a utilização dessas palavras-chave foi relevante na pesquisa realizada, já que ela propiciava uma filtragem mais precisa das temáticas de artigos relacionados.

- **Autores:** A análise de autores se mostrou relevante e frutífera visto que existe uma probabilidade de autores que atuaram e escreveram trabalhos relativos às interfaces semânticas RESTful e serviços Web semânticos voltarem a tratar do tema em trabalhos posteriores. Apesar da reincidência de artigos de um autor sobre o mesmo tema por mais uma vez ter sido rara, de forma geral, é possível afirmar que os resultados em que a reincidência ocorreu trouxeram informações relevantes ao trabalho. Dezessete artigos foram considerados de interesse e coletados, tendo seu material revisado e fortalecendo a pesquisa.
- **Referências de Artigos:** Analisar quais artigos mais recentes fazem referência aos artigos previamente selecionados pode oferecer uma fonte interessante em relação à busca de novidades sobre o tema, visto que estes artigos tendem, de forma geral, a tratar do mesmo assunto e consequentemente podem ser inclusos nos artigos selecionados. Obtivemos um total de 12 artigos de interesse a partir desse método.

É válido dizer que, além dos artigos presentes no repositório inicial de Cheron (2021), também foram levados em conta documentos fundamentadores em relação aos conceitos abordados no trabalho como: as temáticas da Web3, documentações de frameworks encontradas na Web e artigos associados aos que foram encontrados nas buscas realizadas.

Nas buscas em si, apenas artigos publicados entre os anos de 2018 a 2023 foram considerados em nossa seleção, garantindo a presença de dados recentes que justifiquem a necessidade do desenvolvimento deste estudo. Nas próximas seções serão explorados os resultados obtidos a partir das buscas realizadas utilizando cada uma das três abordagens previamente explicadas.

2.1 Termos Relevantes

Todos os artigos foram analisados, e aqueles que possuíam as palavras chaves destacadas foram listadas na tabela abaixo. Como os trabalhos possuíam formatações distintas, por vezes, não foi possível coletar palavras chaves. Tais

trabalhos foram apontados com a marcação “[Sem palavras-chave propostas]”. Os resultados foram os seguintes:

Tabela 1 – Palavras-chave por Artigo

Título	Palavras-chave
A Framework for Self-descriptive RESTful Services	Web services; REST; Service Description; Information Extraction; Semantic Web
A Framework for Semantic Description of RESTful Web APIs	REST; Web API; Semantic Web Services; Hypermedia Control; Linked Dat
A Framework for the Description, Discovery and Composition of RESTful Semantic Web Services	Semantic Web; Intelligent Agents; Web Service; Semantic Web Service; REST; RESTful Architecture; Service Discovery; Service Composition; Microformats; Poshformats; RDF
A Framework for the Structural Analysis of REST APIs	REST; Interface Description Language; Analysis
A Semantic Description Language for RESTful Data Services to Combat Semaphobia	Semantic Web; Web services; REST; Web 3.0; Internet; SOA; SEREDASj; Semaphobia
A Maturity Model for Semantic RESTful Web APIs	Maturity Model; REST; Web API; Linked Data; Semantic Web
A model-driven approach for REST compliant services	REST; Architectural Style Compliance; Modeldriven Software Development; Architectural Constraints; Representational State Transfer
First International Workshop on RESTful Design (WS-REST 2010)	EST; HTTP; Web Architecture; SOA; Web Services; Service Design
Description Languages for REST APIs – State of the Art, Comparison, and Transformation	[Sem palavras-chave propostas]
Formal Modeling of RESTful Systems Using Finite-State Machines	Representational State Transfer; World Wide Web; Software Architectural Styles; Formal Model; Finite-state Machines; Hypermedia
From requirements to source code: a Model-Driven Engineering approach for RESTful web services	RESTful Web Services; Model-Driven Engineering; Software requirements; Automated Software Engineering
MIRA: A Model-Driven Framework for Semantic Interfaces for Web Applications	HCI; Interface; Adaptation; Semantic Web; Data-driven design; Model-Driven Interface Design
Modeling RESTful applications	REST; Metamodel; Modeling; Web Services
Ontology Modeling for REST Open APIs and Web Service Mash-up Method	[Sem palavras-chave propostas]
REST APIs: A Large-Scale Analysis of	REST; APIs; REST principles; Mobile Internet

Compliance with Principles and Best Practices	
REST web service description for graph-based service discovery	[Sem palavras-chave propostas]
RESTDoc: Describe, Discover and Compose RESTful Semantic Web Services using Annotated Documentations	Semantic Web; Intelligent Agents; Web Service; Semantic Web Service; REST; RESTful Architecture; Service Discovery; Service Composition; Microformats; Poshformats; RDF
RESTful Conversation with RESTalk The Use Case of Doodle	RESTful web services; Multiparty Conversations; Domain Specific Language; Real API; Use Case; Doodle
Rapido: A Sketching Tool for Web API Designers	API; Design; Sketching; REST; Web API; Usability
Semantic Matchmaking of Web Services using Model Checking	Web Service; Matchmaking; Model Checking
Supporting the Creation of Semantic RESTful Service Descriptions	[Sem palavras-chave propostas]
Towards Behaviorally Enriched Semantic RESTful Interfaces Using OWL2	[Sem palavras-chave propostas]
Architectural Styles and the Design of Network-based Software Architectures	[Sem palavras-chave propostas]
Model Your Application Domain, Not Your JSON Structures	Web; Distributed Systems; Contracts; Web Services; Web APIs, HTTP; REST; Linked Data; JSON-LD; Hydra; Domain-Driven Design
Third Generation Web APIs	[Sem palavras-chave propostas]

Fonte: O autor (2023).

Com o propósito de equalizar termos de conteúdo idêntico ou que apresentavam apenas leves diferenças que configuravam variações sobre o mesmo tópico, foi realizado o tratamento dos resultados coletados. Para isso foi feito a contagem da incidência de cada termo somada à análise comparativa entre os valores apresentados. Os cinco termos de maior relevância encontrados foram:

- REST.
- Semantic Web.
- Web Services.
- Model-Driven Engineering.
- Service Discovery.

Alinhado aos interesses deste estudo, também foi adicionado o termo “Interface”, tendo em vista a relevância do referido termo previamente explanado na seção anterior. A partir desses termos, buscas foram realizadas com as possíveis combinações das palavras-chave, em todos os seus possíveis subconjuntos do conjunto das partes que possuíam quatro ou cinco elementos, como demonstrado a seguir:

(REST, Semantic Web, Web services, Model-Driven Engineering, Service discovery), (REST, Semantic Web, Web services, Model-Driven Engineering, Interfaces).

Para viabilizar a busca, apenas os dez primeiros artigos em ordem de relevância em cada busca foram considerados. Após uma seleção meticulosa, 13 artigos foram selecionados a partir desse processo que trazem informações relevantes aos objetivos deste estudo.

2.2 Autores

Antes da busca pelos artigos em particular, seria interessante analisar a incidência dos nomes no espaço amostral, de modo que se garanta que não haja repetição em particular de apenas alguns autores, oferecendo uma perspectiva mais ampla em termos de pesquisa. A seguinte tabela de incidência demonstra o número de incidência por autor e em quais trabalhos os nomes aparecem:

Tabela 2 – Incidência dos autores por artigo

Autor	Incidência	Artigos
Ivan Salvadori	2	A Framework for Semantic Description of RESTful Web APIs; A Maturity Model for Semantic RESTful Web APIs
Frank Siqueira	2	A Framework for Semantic Description of RESTful Web APIs; A Maturity Model for Semantic RESTful Web APIs
Davis John	2	A Framework for the Description, Discovery and Composition of RESTful Semantic Web Services; RESTDoc: Describe, Discover and Compose RESTful Semantic Web Services using Annotated Documentations
Dr. Rajasree	2	A Framework for the Description, Discovery and Composition of RESTful Semantic Web Services; RESTDoc: Describe, Discover and Compose

		RESTful Semantic Web Services using Annotated Documentations
Florian Haupt	2	A Framework for the Structural Analysis of REST APIs; A model-driven approach for REST compliant services
Frank Leymann	2	A Framework for the Structural Analysis of REST APIs; A model-driven approach for REST compliant services
Anton Schrerer	2	Description Languages for REST APIs - State of the Art, Comparison, and Transformation; A Framework for the Structural Analysis of REST APIs
Markus Lanthaler	3	A Semantic Description Language for RESTful Data Services to Combat Semaphobia; Model Your Application Domain, Not Your JSON Structures; Third Generation Web APIs Bridging the Gap between REST and Linked Data
Christian Gütl	2	A Semantic Description Language for RESTful Data Services to Combat Semaphobia; Model Your Application Domain, Not Your JSON Structures

Fonte: O autor (2023).

Há um total de 39 outros autores que não apareceram simultaneamente em mais de um trabalho, consequentemente, com um espaço amostral total de 48 autores iniciais, pode-se inferir que o espaço amostral é satisfatório para oferecer material para uma busca extensiva.

Durante a pesquisa, ferramentas como Google Scholar (Indexador de informações sobre autores e trabalhos acadêmicos) e a ferramenta Consensus (Ferramenta de IA generativa que realiza buscas de mesmo propósito) foram utilizadas para buscar o nome dos estudiosos envolvidos nos trabalhos previamente selecionados. Para cada um foi realizada uma busca e posteriormente outra filtragem por data (para garantir o período determinado no protocolo de busca deste estudo) foi realizada no perfil dos autores encontrados.

Trabalhos relacionados aos tópicos de Web-semântica, API’s, RESTful e restrições de integridade foram buscados. No total, dos 48 autores inicialmente identificados, apenas 11 desenvolveram trabalhos posteriores na área de interesse desta pesquisa. Os autores com mais proeminência nos resultados dessa pesquisa

foram Cesare Pautasso com quatro artigos relevantes e Ivan Salvadori, com três. Além disso, eles também figuram em outros artigos coletados, o que demonstra uma representação significativa de sua importância nos resultados da pesquisa. A pesquisa também evidência o fato de que, por razões diversas, não é necessariamente verdade que autores que previamente escreveram artigos sobre um tema serão reincidentes em trabalhos posteriores.

2.3 Referências de Artigos

No processo de busca por referências dos artigos, utilizamos uma abordagem semelhante à pesquisa por autores, recorrendo às ferramentas de indexação públicas de artigos como Google Scholar e repositórios como os da IEEE e Springer para identificar trabalhos relevantes. Foram consultadas todas as referências de artigos publicados entre 2018 a 2023, excluindo duplicatas e documentos em idiomas diferentes do português e inglês.

Após uma criteriosa filtragem, foram selecionados apenas 12 trabalhos para que auxiliassem na constituição da base deste estudo. Embora alguns desses artigos fundamentadores encontrados no repositório de Cheron (2021) contassem com um grande número de citações, muitas delas se concentravam em aspectos específicos do conteúdo ou apenas tangenciavam nosso tema de interesse. Foram priorizados e escolhidos então trabalhos relacionados aos tópicos de design de API’s (do inglês, *Application Programming Interfaces*), serviços Web semânticos, princípios RESTful e interfaces semânticas, garantindo assim uma base sólida e relevante para o embasamento deste trabalho.

3 RESTRIÇÕES DE INTEGRIDADE

Nesta seção, restrições de integridade serão abordadas, bem como todo o contexto que as cerca será delineado. A seção trata de uma conceitualização de contextos de mundo aberto e contextos de mundo fechado a princípio, essenciais para a compreensão do leitor do escopo das restrições de integridade em quaisquer contextos nos quais elas se aplicam. Posteriormente, uma definição formal de restrições de integridade será apresentada, assim como uma distinção entre elas, além de especificações gerais de sistema. Por último, abordaremos a perspectiva da importância das restrições de integridade para a construção adequada de API’s.

3.1 Contexto de Mundo Aberto

Para uma compreensão abrangente das restrições de integridade em interfaces semânticas de dados, especialmente em um contexto de mundo fechado, é essencial explorar inicialmente os conceitos relacionados aos contextos de mundo aberto e fechado. Essa compreensão geral dos diferentes contextos fornece o princípio do entendimento do porquê essas restrições se tornam cruciais na garantia da integridade do sistema, principalmente quando aplicadas em interfaces RESTful semânticas, serviços Web-semânticos e outras aplicações similares.

O contexto de mundo aberto, em sua concepção relativa à modelagem de sistemas, parte fundamentalmente do pressuposto de que a veracidade de uma informação é independente de sua observação por agentes terceiros, isto é, uma informação que não é conhecida previamente por um sistema não é necessariamente verdadeira ou falsa. Essa garantia é essencial, pois isso é a chave para a inserção dinâmica de informações num sistema e sua adequada atualização: Se as informações não foram explicitamente definidas num sistema ou não podem ser inferidas dele por meio de um processo dedutivo, elas são tratadas meramente como desconhecidas. (MOORE, 2015)

Pode-se dizer que no contexto de mundo aberto, a incompletude das informações presentes em um modelo é na verdade seu estado padrão. Consequentemente, o contexto de mundo aberto é ideal para a Web-semântica, visto que a abertura para novas informações e atualizações do conhecimento disponível no sistema são essenciais. Ademais, é no contexto de mundo aberto que surge a OWL (do inglês, *Web Ontology Language*). A Linguagem de Ontologia Web (OWL) é uma linguagem que define um padrão da Web-semântica para representar e compartilhar conhecimento em um formato estruturado.

OWL permite definir relações, propriedades e restrições entre classes e indivíduos, ou, como definido na wiki oficial da linguagem, “representar conhecimento rico e complexo sobre coisas, grupos de coisas e relações entre coisas.” (OWL WORKING GROUP, 2012). Ela é suficientemente versátil para dar suporte à diversos níveis de expressividade. A OWL tomou historicamente o protagonismo na construção de maior eficiência na interoperabilidade de dados na Web. A seguinte imagem

exemplifica a utilização da OWL para, neste exemplo, definir uma classe de nome “Person” dentro de uma ontologia:

Figura 1 – Exemplo de código OWL

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
    <!ENTITY ex "http://example.com/ontology#">
]>
<rdf:RDF xmlns="http://example.com/ontology#"
     xml:base="http://example.com/ontology"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

    <owl:Class rdf:about="#Person"/>

    <rdf:Description rdf:about="#John">
        <rdf:type rdf:resource="#Person"/>
        <ex:hasName>John Doe</ex:hasName>
        <ex:hasProperty>Some String Value</ex:hasProperty>
    </rdf:Description>

    <owl:ObjectProperty rdf:about="#ex:hasName"/>
    <owl:ObjectProperty rdf:about="#ex:hasProperty"/>

</rdf:RDF>
```

Fonte: O autor (2023).

O código XML representado na Figura 1 define uma ontologia simples usando a linguagem OWL, que é baseada na estrutura do RDF (do inglês, *Resource Description Framework*). Nesta ontologia, existe uma classe chamada “Person” que representa indivíduos do tipo pessoa. Em seguida, é criada uma instância chamada “John,” que é um exemplo de uma pessoa. John é caracterizado por duas propriedades: *“hasName”* (temNome), com o valor *“John Doe”* e *“hasProperty”* (temPropriedade), com o valor *“Some String Value”* (Algum Valor de String).

Essas propriedades descrevem características de John na ontologia. As definições de classes e propriedades estão contidas nos *name spaces* definidos no cabeçalho do documento. Cada ontologia representa um conjunto específico de informação que juntas, no contexto da Web, formam uma vasta rede de conhecimento não-exclusivo em relação à adição de novas informações à ontologia.

É importante ressaltar que, apesar do contexto de mundo aberto não implementar restrições que são intrínsecas a contextos de mundo fechado como a presunção de um identificador único (UNA), restrições de integridade também podem ser implementadas e desenvolvidas, como foi demonstrado no trabalho de Tao (2010), que sugeriu implementações restrições em OWL. Todavia, essas restrições não são baseadas nos mesmos princípios de um contexto de mundo fechado e as tentativas de implementação de restrições de integridade semânticas num contexto de mundo aberto ainda não são amplamente difundidas.

Os trabalhos de McGuiness (2010) também evidencia por exemplo a necessidade restrições de integridade no contexto de OWL, porém tais restrições só puderam ser instanciadas a partir da sugestão de extensões para OWL que a façam numa linguagem que implemente restrições de integridade.

Outro ponto importante é o de que apesar da linguagem OWL ser mais adequada e adaptada para lidar com situações de mundo aberto como o contexto da Web, existem evidências de que há vantagens em expandir suas especificações para situações de mundo fechado, como no contexto de interfaces RESTful semânticas. Motik (2007) propõe uma extensão de OWL que permite a utilização da linguagem para a definição de restrições de integridade no contexto de bancos de dados relacionais, mostrando a maior eficiência da utilização de tal abordagem em comparação à utilização de algoritmos tradicionais para verificação de integridade de dados.

3.2 Contexto de Mundo Fechado

O contexto de mundo fechado, no qual interfaces RESTful semânticas e serviços Web semânticos muitas vezes estão imersos, pode ser visto como o oposto do contexto de mundo aberto. Fundamentalmente, esse conceito parte da premissa de que, para todas as informações que são desconhecidas no sistema, a presunção é de que estas sejam falsas. A ideia pode ser sumarizada na concepção de “negação-como-falha”, isto é, pelo sistema falhar em inferir o valor verdade de uma informação em relação ao que ele já possui previamente, se assume que a informação é falsa, a informação não é aceita pois é incompreensível.

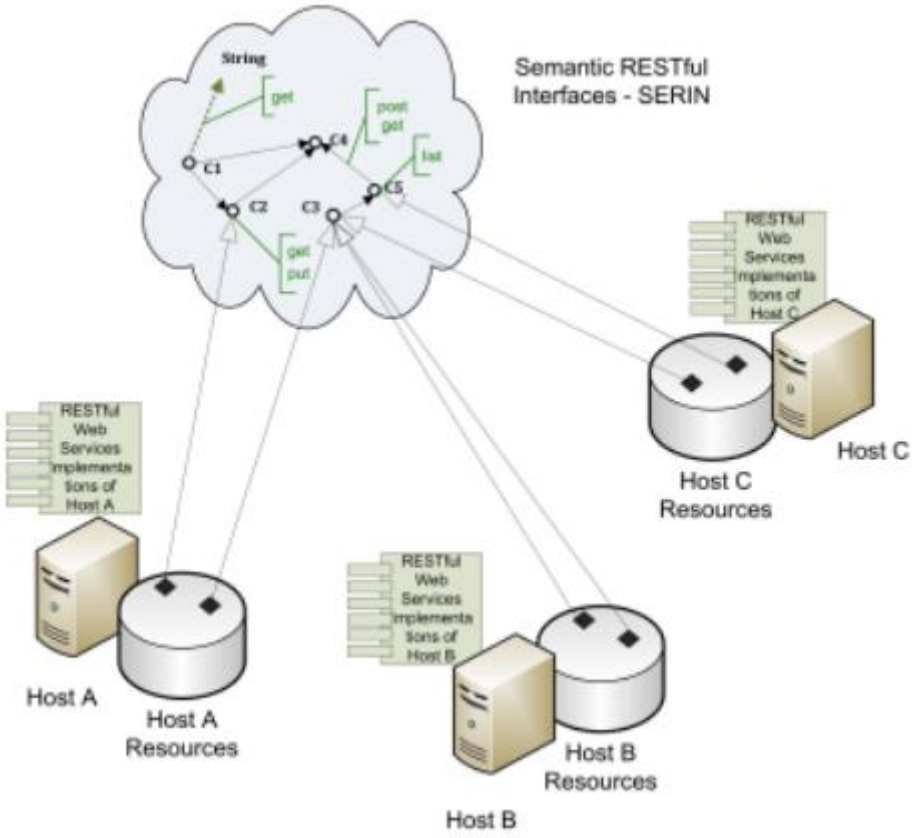
Um exemplo clássico para a compreensão do contexto de um mundo fechado é a ideia de um banco de dados relacional. Além de assumir para os itens registrados em si a ideia de UNA, bancos de dados relacionais também configuram em sua constituição princípios que garantem a manutenção dos axiomas do contexto de mundo fechado, como atomicidade, consistência, isolamento de dados e durabilidade (também conhecidos como princípios ACID). (MOORE, 2015)

Em um banco de dados relacional, se uma informação não se configura como presente, uma busca não retorna resultados e consequentemente, questionamentos atrelados a tais buscas terão uma resposta negativa, visto que a ausência de informações configura negação (“Existem bancos abertos 24h na cidade?”, por exemplo).

Um mundo fechado localmente é uma variação de um contexto de mundo fechado na qual vários elementos distintos possuem suas próprias condições enquanto ainda mantém a presunção de mundo aberto para contextos externos, ou seja, cada um dos elementos em seu próprio contexto possui a presunção de que as informações que não estão presentes são falsas, para buscas relacionadas. Todavia, na visão de um agente externo, que pode extrair informações de cada uma dessas fontes diferentes, o contexto é de mundo aberto, visto que o fato de uma informação não estar presente em uma das fontes não necessariamente resulta na falsidade desta informação. (LIRA, 2015)

A Figura 2 exemplifica a interação de servidores num contexto de mundo fechado localmente. Esse contexto é adequado para a conceituação de situações em que há o consumo de diferentes interfaces de programação de aplicativos (API), muitas das quais podem ser modeladas a partir da criação de Interfaces RESTful semânticas ou implementadas a partir de serviços Web-semânticos, temas que serão tratados nas próximas seções.

Figura 2 – Interação de Servidores num mundo fechado localmente



Fonte: LIRA et al. (2015)

3.3 Restrições de Integridade

No contexto de serviços existentes na Web, várias situações relativas à leitura, inserção e modificação de dados são possíveis. Ao menos de uma perspectiva de protocolos de acesso, serviços Web podem ser distintos entre aqueles que possuem protocolos simples de acesso (SOAP) e aqueles que são baseados numa arquitetura de estado de transferência representacional (REST). (TANDIONO, 2019)

Sistemas com arquitetura RESTful representam várias similaridades em relação a outros que operam num contexto de mundo fechado e compartilham de restrições de integridade semelhantes. Restrições de integridade de dados são limitações impostas aos processos de inserção de dados em um sistema para a manutenção de sua coerência interna, funcionamento adequado e disponibilidade.

As restrições de integridade são mecanismos essenciais em contextos de mundo fechado, visto que é por meio delas que as definições necessárias para a prevenção da inserção de informações incorretas ou incoerentes são impostas.

Restrições de integridade podem ser propostas para diversas características dos dados: em relação à sua unicidade (a exemplo em consonância com o princípio de UNA de mundos fechados), à sua adequação em relação às classes de dados definidas para seus atributos ou em relação à presença ou ausência de atributos (tal como chaves de partição e de organização em bancos de dados não-relacionais como atributos obrigatórios), dentre outras características.

As restrições de integridade são parte integrante do modelo de dados relacional proposto por Codd na década de 1970. Restrições de integridade relacional são um conjunto de regras que podem ser usadas para manter a integridade dos dados durante operações de inserção, exclusão e atualização em tabelas de um banco de dados relacional. Restrições de integridade normalmente são criadas e eliminadas usando SQL. Portanto, as restrições de integridade são utilizadas para evitar a persistência de dados inconsistentes no banco de dados. Como estão presentes na norma ANSI do SQl (ISO/IEC, 2023) as restrições são contempladas por todos os Sistemas Gerenciadores de Banco de Dados (SGBD's) que seguem esta norma.

De acordo com os manuais de desenvolvimento SGBD da Oracle (2021), no contexto de bancos de dados relacionais, usar os mecanismos de restrições de integridade do SGBD é uma opção preferível a checar a integridade de dados usando *stored procedures* ou embutindo regras no código da aplicação. Essa preferência se dá pelas seguintes razões:

- **Facilidade declarativa:** Como a definição de restrições de integridade perpassa a utilização de instruções SQL, nenhuma programação adicional é necessária ao definir ou alterar uma tabela. As instruções SQL são fáceis de escrever e eliminam erros de programação.
- **Regras Centralizadas:** As restrições de integridade são definidas para tabelas e armazenadas no dicionário de dados. Assim, os dados inseridos por todas as aplicações devem obedecer às mesmas restrições de integridade. Se as regras mudarem no nível da tabela, os aplicativos não precisarão mudar. Além disso, os aplicativos podem usar metadados no dicionário de dados para informar imediatamente os usuários sobre violações, mesmo antes de o banco de dados verificar a instrução SQL.

- **Flexibilidade no carregamento de dados:** Restrições de integridade podem ser desativadas de maneira temporária para evitar sobrecarga de desempenho ao carregar grandes quantidades de dados. Quando o carregamento de dados for concluído, elas podem ser facilmente reativadas.

Restrições de integridade podem ser definidas de diversas formas. Semanticamente, as mesmas podem ser definidas a partir da utilização de especificações que utilizam RDF (do inglês, *Resource Description Framework*) e implementam ontologias. Um exemplo dessa implementação foi demonstrado por Vitvar (2016) ao criar um conjunto de dados com a descrição de mais de onze mil API's, as quais demonstravam restrições relativas à formatação adequada dos dados. A criação de padrões de desenvolvimento também pode ser uma forma efetiva de especificar restrições de integridade.

3.4 Estudo de Caso: Restrições de Integridade em Bancos de Dados Relacionais

Bancos de dados são aplicações que armazenam conjuntos de dados de formas estruturadas. Existem inúmeros modelos diferentes de bancos de dados (relacionais, não relacionais, baseadas em grafo etc.). Contudo, para os propósitos deste estudo de caso, nosso foco será em bancos de dados relacionais. Bancos de dados relacionais podem ser descritos como um conjunto de informações dispostas num conjunto de tabelas, que são definidas a partir de relações entre as tabelas e seus objetos. (JATANA, 2012)

As definições dessas relações, inclusive as restrições, são dispostas no esquema do banco de dados. Existem diferentes bancos de dados relacionais disponíveis no mercado de tecnologia, como OracleDB, PostgreSQL, CassandraDB, dentre outros. Para propósitos de comparação, nosso estudo se limitará ao banco de dados MySQL, que é reconhecidamente um dos mais utilizados na indústria de Engenharia de Software.

Implementado em servidores e tendo o acesso à informação disponibilizado a partir da utilização de uma linguagem estruturada de consulta de informações (SQL), o banco de dados visto como uma aplicação torna-se um exemplo ideal para ilustrar restrições de integridade de forma prática. Algumas das restrições de integridade aplicáveis aos bancos de dados relacionais são:

- **Não-Nulidade:** Bancos de dados MySQL toleram, no contexto de certos atributos, a presença de valores nulos com a notação “NULL”. Todavia, especialmente em situações em que a modelagem de banco de dados exige que o valor de um atributo não seja nulo, seja porque o atributo é um identificador ou por ter relevância essencial para as regras de negócio, MySQL aceita a implementação da notação “NOT NULL” durante o processo da modelagem da tabela como uma forma de delinear essa restrição de integridade para todos os objetos inseridos no contexto desta relação.
- **Unicidade:** MySQL não permite a atribuição de mais de um valor para um atributo único de uma tabela. Por essa razão é completamente coerente com a restrição de integridade relacionada à unicidade dos atributos. Apesar de permitir atributos que representem listas ou conjuntos de subobjetos, estes são vistos apenas como um objeto único e podem ser definidos utilizando a notação “ARRAY”.
- **Identificação:** Todo objeto em MySQL deve possuir um identificador único que o distinga dos demais, incluindo outros objetos que possuem os valores dos atributos semelhantes. Durante o processo de modelagem do banco de dados, este valor é definido com a notação “PRIMARY KEY”. Também é possível especificar que outros atributos tenham valores únicos para si, tornando-os assim possíveis identificadores e diferenciadores dos objetos nas tabelas, com a notação “UNIQUE”.
- **URI Estrangeira:** Em SQL, para atributos cujo conteúdo referência são outras notações presentes no sistema, a utilização da notação “FOREIGN KEY” é obrigatória. Isso garante a consistência e integridade dos dados no sistema, sendo assim coerente com a restrição de integridade especificada.
- **Criação de Classes Internas:** No processo de modelagem de bases de dados em MySQL, se existe a necessidade da implementação de uma relação de subordinação entre duas classes, é possível utilizar a notação “FOREIGN KEY” junto da notação “CASCADE” para um atributo da tabela subordinada. Essa especificação determina que, se qualquer coisa acontecer com a tabela superior na hierarquia, como a remoção dos itens da mesma por exemplo, o mesmo ocorrerá na tabela subordinada aos seus itens associados. Todavia, essa restrição não é padrão na ferramenta, sendo necessária a especificação.

- **Aninhamento:** MySQL não implementa, por padrão, o processo de aninhamento no retorno de informações. Porém, quando consultas são efetuadas, é possível trazer informações relacionadas de diferentes tabelas associadas a busca principal utilizando a notação “JOIN”.
- **Verificação:** Por padrão, os bancos de dados MySQL possuem a notação “CHECK” para delimitação de condições relativas à limitação de valores possíveis dos atributos. Essa restrição de integridade pode ser utilizada para impedir que atributos possuam valores não adequados as especificações necessárias para o sistema.
- **Manipulação Referencial:** A partir da notação “REF”, administradores do sistema são capazes de definir restrições em relação à modificação e leitura de campos em objetos que fazem referência a outros objetos no sistema. Esta é mais uma restrição de integridade que garante a consistência dos dados no sistema de modo que alterações efetivadas nos atributos de um objeto possam ser restringidas caso esses atributos façam referência a outros objetos presentes no sistema.

Todas as informações relativas às notações de MySQL foram extraídas da documentação da ferramenta Oracle (2021). Apesar de bancos de dados serem excelentes estudos de caso para a representação da ideia de restrições de integridade, com o desenvolvimento da Web, essas restrições também aparecem em inúmeros outros contextos, como no de API’s.

3.5 Restrições de Integridade de API’s

As restrições de integridade de dados oferecem recursos importantes para as APIs, pois garantem a precisão, consistência e segurança dos dados trocados entre sistemas. Essas restrições impedem que dados inválidos ou maliciosos corrompam bancos de dados, garantem a qualidade dos dados e servem como salvaguarda contra alterações que não sejam condizentes com a estrutura vigente da modelagem de dados da aplicação.

Além disso, restrições de integridade também auxiliam no tratamento de erros, na validação de dados e na documentação, assegurando que as API’s permaneçam confiáveis, seguras e em conformidade com regulamentos, ao mesmo tempo em que acomodam as necessidades de dados em evolução e versões de software. No

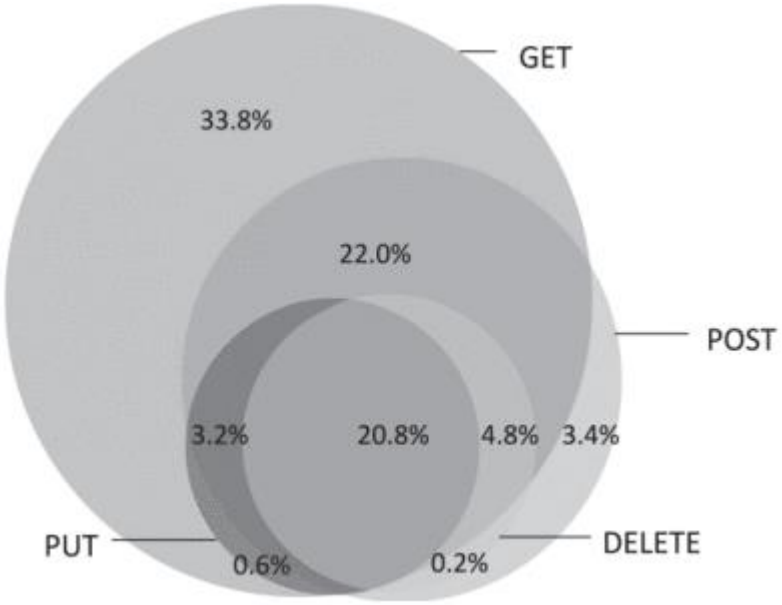
trabalho de Pautasso (2021) uma demonstração clara da importância da modelagem adequada de restrições de integridade de dados em API’s pode ser observada ao evitar certos padrões de desenvolvimento referentes à implementação de recursos.

O autor explica o caso de “domínios anêmicos”, que são contextos em que a API não reflete adequadamente as características e restrições de dados associadas à camada de persistência de dados acessada pela mesma. Tais designs frequentemente carecem de limites de transação e consistência de dados, resultando em desempenho e escalabilidade deficientes da API. Além disso, a falta de atenção às restrições de integridade pode gerar também inconsistência e dificuldade na manutenção futura da API.

A compreensão de restrições de integridades também auxilia no processo de modelagem de API’s em si. No trabalho de Schreier (2011), a definição de tipos de dados, atributos e da forma de identificação e acesso é levada em conta na concepção da construção da modelagem estrutural das API’s. Isso se reflete na qualidade dos dados na aplicação e na permissividade do sistema em relação ao processo de inserção de dados, reiterando mais uma vez a importância dessas delimitações para a construção de API’s robustas e adequadamente modeladas.

Ademais, restrições de integridade também podem ser utilizadas para reforçar a necessidade da utilização adequada de recursos em API’s, reforçando por exemplo a adequação de chamadas e de objetos enviados pelos clientes aos modelos apropriados para cada requisição. O trabalho de Neumann et al. (2021) demonstra que apesar de uma adequação de 88% no geral, existem sistemas que não utilizam os verbos HTTP (GET, PUT, POST etc.) de forma adequada, gerando inconformidades que podem posteriormente comprometer a integridade do sistema. Com restrições de integridade adequadamente implementadas no recebimento das requisições, esse desafio seria mitigado.

Figura 3 – Suporte dos Verbos HTTP por área de intersecção nas API's públicas



Fonte: NEUMANN et al. (2021)

4 SERVIÇOS WEB, WEB API's E RESTful

4.1 Evolução da Arquitetura de Serviços da Web

Desde 1990, os sistemas e serviços na Web evoluíram e mudaram significativamente. De forma geral, pode-se afirmar que os três elementos principais que, nas primeiras proposições feitas por Berners-Lee e Cailliau (1996) figuravam como base motriz da intercomunicação de sistemas (URL, HTTP e HTML) se mantiveram fundamentais nos desenvolvimentos posteriores de tecnologias relacionados à web.

Além disso, inúmeros outros elementos e desenvolvimentos tecnológicos foram introduzidos ao longo do tempo e modificaram significativamente a forma como a Web é acessada pelos seus usuários. Berners-Lee e Cailliau (1996) propuseram que a internet evoluiria das seguintes maneiras: "A primeira envolve a melhoria da infraestrutura, para fornecer um serviço mais funcional, robusto, eficiente e disponível. A segunda é aprimorar a web como um meio de comunicação e interação entre pessoas. A terceira é permitir que a web, além de ser um espaço navegável por humanos, contenha dados ricos em um formato compreensível por máquinas,

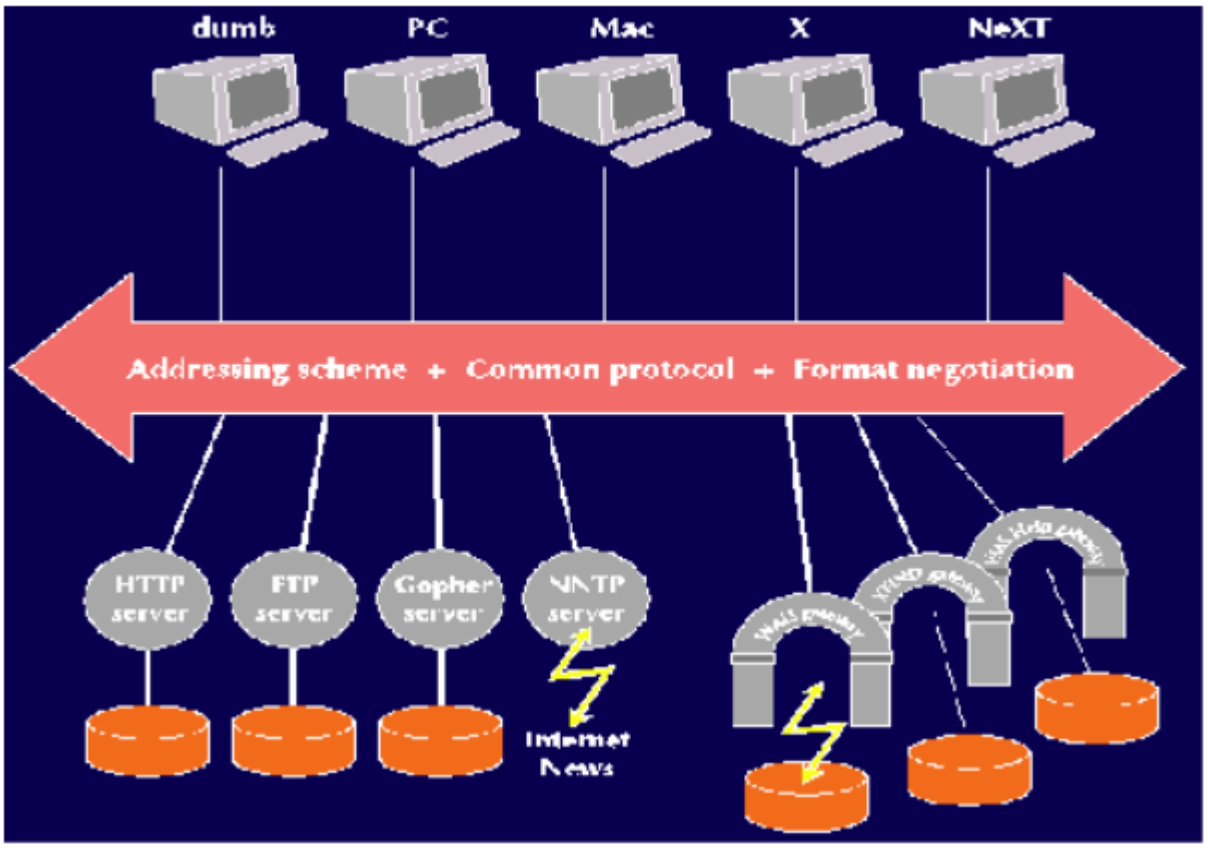
permitindo assim que as máquinas desempenhem um papel mais forte na análise da web e na resolução de problemas para nós.”

Apesar de suas previsões se provaram corretas, elas não possuem um enfoque particular na evolução dos serviços Web em si. Segundo o trabalho de Cheron (2021), os principais fatores que levaram às mudanças arquiteturais e evoluções de serviços na Web foram:

- **Evolução das ferramentas de navegação:** As possibilidades oferecidas pelas ferramentas de navegação em relação ao acesso às informações ofereceram motivações relevantes na evolução dos serviços oferecidos na Web. Desde as formas mais básicas de especificação de informações utilizando HTML e CSS, à evolução nas formas de especificação e avanços em relação a capacidade das ferramentas (como a capacidade de suportar arquivos do tipo *flash*) foram fatores essenciais na moldagem dos processos na WWW (do inglês, *World Wide Web*) mudaram.
- **Evolução no modo como a Web é utilizada:** Se no princípio a utilização de sistemas Web era extremamente limitada, sendo possível apenas expor e inserir informações num modelo de páginas estáticas, com o passar do tempo e com evoluções tecnológicas (tal como o aumento na velocidade da conexão disponibilizada aos usuários para acesso à internet), formas mais diversificadas e adequadas de se utilizar a Web foram desenvolvidas. Desde a utilização de protocolos mais eficientes até a adoção de modelos arquiteturais mais robustos para lidar com o aumento no número de requisições.

Além dos fatores previamente mencionados, para o escopo desse trabalho, é importante considerar a evolução do processo de consumo de informação dos servidores. Nos primórdios da Web, os sites eram, no geral, resumidos a páginas geradas dinamicamente a partir do cálculo de servidores ou estaticamente. O protocolo de comunicação entre a página local e os servidores e o formato de negociação das chamadas deveria ser definido durante esta comunicação.

Figura 4 – Comunicação cliente-servidor nos primórdios da Web



Fonte: BERNERS-LEE E CAILLIAU (1996)

Com os propósitos de manter a interoperabilidade e troca livre de informações na Web, preservando sua capacidade comunicativa, o protocolo SOAP para a troca de informações foi criado. De acordo com sua documentação oficial, o SOAP é “um protocolo leve para a troca de informações em um ambiente descentralizado e distribuído.

É um protocolo baseado em XML que consiste em três partes: um envelope que define uma estrutura para descrever o que está em uma mensagem e como processá-la, um conjunto de regras de codificação para expressar instâncias de tipos de dados definidos pela aplicação e uma convenção para representar chamadas e respostas de procedimentos remotos”. (W3C, 2000) Principalmente para o contexto de uma Web que era, majoritariamente, de páginas Web estáticas, o protocolo SOAP era uma solução satisfatória.

Posteriormente, com o desenvolvimento progressivo dos *Web Standards* (Padrões para o desenvolvimento Web), as buscas se tornaram mais padronizadas e

junto ao desenvolvimento da tecnologia AJAX (do inglês, *Asynchronous JavaScript And XML*), que possibilitou a criação de páginas web mais dinâmicas, os servidores HTTP se tornaram hegemônicos em relação a prover informações para aplicações web.

Além disso, a arquitetura SOAP foi progressivamente despriorizada em relação ao modelo RESTful, principalmente pelo fato dessa arquitetura ser baseada em um modelo de RPC (do inglês, *Remote Procedure Call*) e apresentar falhas funcionais e de expressividade em relação aos servidores que adotam princípios RESTful. (CHERON, 2021)

Atualmente, a maioria das aplicações de servidores de *backend* disponíveis na Web pública implementam parte dos princípios RESTful. Entretanto, é necessário esclarecer que, de acordo com as definições originais propostas por Fielding (2000), a adoção dos princípios RESTful demanda a utilização de HATEOAS (do inglês, *Hypermedia as the Engine of Application State*).

Dessa forma, o retorno de informações relativas ao estado da aplicação dos sistemas a partir de sua chamada. Como a maioria dos serviços Web atuais presentes não apresentam estas características, de forma geral, os mesmos são atualmente caracterizados como REST Web API’s, e não como API’s RESTful propriamente. De fato, é possível afirmar que atualmente o modelo de consumo de informações a partir de Serviços Web que implementam API’s REST se tornou a forma mais popular de se estruturar a comunicação entre serviços web. (NEUMANN, 2021)

Protocolos como FTP, apesar de ainda existirem e serem utilizados em alguns contextos, foram despriorizados quanto à utilização de API’s, pois elas apresentam maior facilidade de difusão e uma arquitetura mais uniforme para a construção de chamadas para servidores. Nas duas próximas seções, apresentaremos a definição dos conceitos de API, de Serviços Web e dos princípios RESTful.

4.2 Web API’s

As API’s são, por definição, interfaces usadas como pontos de entrada para entidades de software reutilizáveis. Elas não são entidades de software independentes; em vez disso, são empacotadas com as bibliotecas de software, estruturas ou serviços da Web que as oferecem. (LAMOTHE, 2021)

API’s são utilizadas para prover acesso à busca e modificação de informações para inúmeros tipos diferentes de aplicações. No contexto de aplicações Web, elas são essenciais, pois servem como uma fonte de contato entre o *frontend* das aplicações e seu *backend*, onde geralmente encontra-se a aplicação onde as API’s estão anotadas.

A seguir, usaremos Web API’s como nosso principal exemplo para discutir API’s em geral. Essa abordagem foi escolhida porque as Web API’s são o tipo mais comum de API’s encontradas na internet atualmente, servindo como a base para a troca de dados e comunicação entre vários serviços e aplicativos da web. Ao examinar os princípios e características das Web API’s, podemos obter *insights* valiosos sobre o conceito mais amplo de API’s, suas funcionalidades e seu papel fundamental em possibilitar interações eficientes dentro do ecossistema digital.

De forma geral, além de conceituações básicas sobre a constituição de Web API’s, é trabalhoso encontrar consensos em relação aos fatores gerais e distintivos presentes nelas. O trabalho de Wilde (2018) apresenta um panorama conceitual interessante sobre a diversidade que pode existir nas Web API’s: Além dos conceitos fundamentais da identificação de recursos por meio de URI’s (do inglês, *Uniform Resource Identifier*) e da interação com os recursos por meio da utilização de métodos HTTP, existe uma grande flexibilidade em como as API’s são estruturadas.

Em relação à forma como os dados estruturados podem ser retornados ao cliente que realizou a requisição no servidor, distintas opções possíveis podem ser identificadas (RDF, XML, JSON etc.). Uma situação semelhante é apresentada em relação às formas de retorno do cabeçalho de requisições, dos possíveis valores do status HTTP retornados, dentre outros.

Apesar da variedade de possibilidades, todos esses possíveis valores estão enquadrados como *Web Concepts*. Não obstante a natureza da constante evolução desta área, concordamos que não é possível deter o conhecimento absoluto de todos

os *Web Concepts* existentes. Existem bases de dados disponíveis para consulta sobre esses conceitos na internet, disponível em: https://webconcepts.info/.

A natureza flexível das API's traz diversos benefícios relacionados à sua capacidade de representatividade e de retorno dos mais diversos tipos de informações. Porém, isso pode dificultar a identificação de sua adequação e a distinção entre API's consumíveis ou impróprias para tipos distintos de serviços.

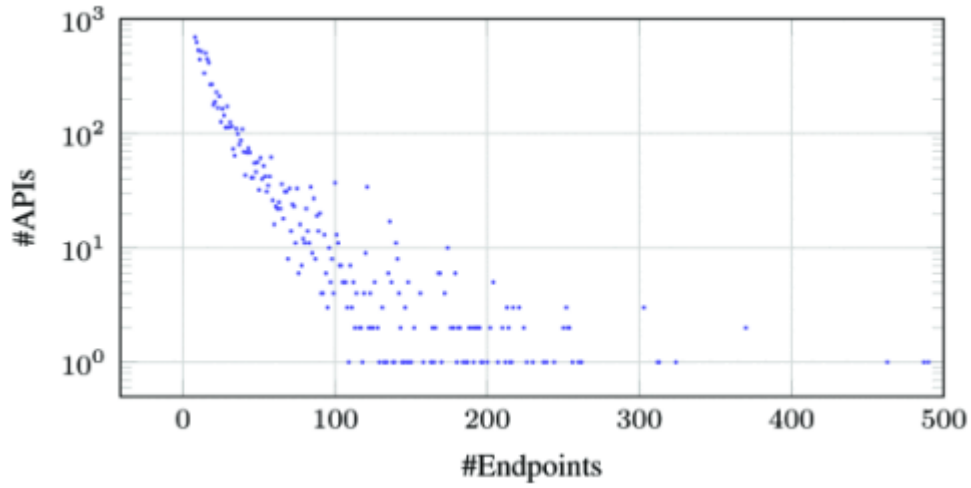
Soluções possíveis já foram apresentadas para tentar adequadamente lidar com a problemática, como a proposta por Pautasso (2020) na qual a construção de um framework para viabilizar a criação de rótulos para API's chamado AFL (do inglês, *American Fuzzy Lop*). De maneira geral, esse tipo de prática ainda não foi adotado em larga escala.

Como API's funcionam fundamentalmente oferecendo seus recursos por meio de *endpoints* únicos, uma prática existente na indústria é a de composição, onde recursos de diferentes API's são sintetizados em interfaces únicas para facilitar a implementação de desenvolvedores de serviços que precisem buscar para a construção de suas aplicações, evitando o retrabalho na implementação, em interfaces diferentes, da mesma funcionalidade.

Um estudo realizado por Serbout (2022) demonstrou um número relativamente alto de API's compatíveis em relação aos seus recursos quando combinam seus modelos de dados apenas no nível do tipo de dados. No entanto, esse número diminui quando o escopo é restringido apenas às API's que lidam com objetos de dados com nomes de propriedade idênticos.

A probabilidade média de encontrar duas API's compatíveis, com correspondência tanto nos nomes de propriedade quanto nos tipos de dados, atinge 21%. Além disso, o número de *endpoints* compatíveis dentro da mesma API é baixo. Parte disso pode estar associado ao fato de que a maioria das API's apresentam um número muito conciso de recursos, contribuindo para esta sucintez e diminuindo a redundância de URI's. A Figura 5, evidenciada no trabalho de Serbout (2022), demonstra a distribuição de API's em relação aos seus recursos.

Figura 5 – Distribuição das API’s por número de recursos



Fonte: SERBOUT (2022)

4.3 Princípios RESTful

REST (do inglês, *Representational State Transfer*) é um estilo arquitetônico para o design de aplicações em rede. A inspiração fundamental para os princípios RESTful pode ser encontrada na estrutura de HTTP. O princípio fundamental do REST é a adoção do paradigma sem estado, ou seja, a ideia de que nada é armazenado pelos servidores relativos às operações, exceto durante as operações realizadas no próprio servidor.

Cada evento é autocontido em si e não há nada salvo para outras operações futuras. Por essa razão, cada solicitação REST deve ser também autocontida, contendo todas as informações necessárias para que o servidor execute a ação solicitada, tornando cada ação solicitada um evento independente. (FU, 2010)

Além das características já previamente citadas, outras particularidades de sistemas que implementam os princípios arquiteturais RESTful são a utilização de recursos unicamente identificáveis (URI), a implementação de um sistema de criação, leitura, atualização e remoção de dados (CRUD) baseado na utilização dos métodos HTTP e a especificação necessária da representação de dados.

Fielding (2000) definiu RESTful como “uma abstração dos elementos arquitetônicos dentro de um sistema de hipermídia distribuída.” O REST ignora os detalhes da implementação de componentes e a sintaxe de protocolo para se

concentrar nos papéis dos componentes, nas restrições sobre a interação entre eles e na interpretação de elementos de dados significativos. Ele abrange as restrições fundamentais sobre componentes, conectores e dados que definem a base da arquitetura da Web e, assim, a essência de seu comportamento como uma aplicação baseada em rede.

Aplicações com características RESTful tem presença quase hegemônica no processo de desenvolvimento de sistemas Web modernos. A facilidade do acesso às informações por meio de requisições para API's REStful, também torna esse princípio arquitetural recomendável para a criação de interfaces públicas de acesso à dados.

Segundo o trabalho de Shang (2021), mais de 7.000 API's/Serviços Web RESTful registrados na *ProgrammableWeb* foram identificados e esse número está aumentando rapidamente. O mesmo trabalho também analisa as limitações das ferramentas atuais que executam o processo de indexação de API's RESTful para a busca e propõe a implementação de um sistema que possibilite a busca de API's RESTful baseado em conceitos de similaridade semântica entre elas.

Além do contexto Web, os princípios RESTful também são utilizados para o desenvolvimento de sistemas que tem como principal objetivo prover serviços para contextos de Internet das Coisas (IoT) e sistemas críticos, que precisam implementar restrições rígidas em relação à coerência lógica e matemática dos seus recursos, visto que quaisquer erros podem prejudicar a garantia de suas funcionalidades.

Rauf (2018) prova a adequação de princípios arquiteturais RESTful para estes propósitos por meio da implementação de um sistema que analisa modelos de design implementados na linguagem de programação lógica Event-B e utilizam os mesmos para a geração de trechos de código íntegros de uma perspectiva formal utilizando o framework Django, da linguagem de programação Python.

A necessidade de documentação de Web API's com características RESTful impulsionou o desenvolvimento de várias formas de anotação de recursos, incluindo anotações semânticas. No entanto, uma das contribuições mais significativas foi o surgimento do framework OpenAPI (anteriormente conhecido como Swagger).

4.4 OpenAPI

OpenAPI é uma linguagem de especificação padronizada e agnóstica que permite uma comunicação clara das definições de API, desacopladas das linguagens de programação, para facilitar a transferência de conhecimento entre fornecedores e consumidores, promovendo o sucesso na economia de AP'Is. OpenAPI é agnóstica pois não tem, fundamentalmente, conexão direta com os componentes do software, sendo assim uma linguagem descritiva que existe de forma distinta ao código em si. Também não é intrinsecamente relacionada a nenhuma linguagem ou formatação de documentos, podendo por exemplo ser utilizada para gerar descrições de software em modelos JSON e YAML.

A OpenAPI é usada para descrever serviços de API criando um documento estruturado que delineia detalhes essenciais da API. Este documento é dividido estruturalmente por meio de inúmeros objetos, que representam cada um definições de especificações relacionadas a certos elementos da API. O objeto de base, chamado *OpenAPI Object* é o objeto fundamental da descrição, sob o qual todas as outras definições são dependentes.

Este objeto possui atributos que representam a descrição dos servidores associados à aplicação, os recursos existentes, os mecanismos de segurança que podem ser implementados etc. De forma geral, a definição dos objetos subjacentes é feita de maneira similar a definição de uma estrutura de dados de árvore, onde os objetos de menor abrangência estão associados ao objeto OpenAPI e seus objetos subordinados estão associados entre si, gerando um certo aninhamento dentro da estrutura da definição, semelhante à que acontece em objetos descritos como JSON.

Essa abordagem estruturada melhora o desenvolvimento de APIs, fornecendo documentação clara, promovendo consistência e simplificando a colaboração entre os fornecedores e consumidores de APIs. (Miller, 2021) A descrição de API's utilizando OpenAPI é altamente difundida. O estudo de Serbout (2022) demonstra como o conjunto de dados atualmente disponível referente às descrições de serviços Web que utilizam OpenAPI é suficientemente robusto para suportar análises extensas sobre a estrutura de aplicações Web.

A partir de um conjunto de dados coletados, foram extraídas 20.587 descrições válidas de API's. Os critérios para as descrições selecionadas foram a presença de

URL’s (do inglês, *Uniform Resource Locator*) de servidor e recursos de base coerentes, descrições de esquemas de dados nos recursos e um mínimo de três recursos avaliáveis. Um exemplo da utilização possível de OpenAPI para a descrição de API’s é representada na Figura 6 abaixo, que foi retirada da especificação oficial de OpenAPI (MILLER, 2021):

Figura 6 – Descrição de Recurso em OpenAPI

```
{
  "/pets": {
    "get": {
      "description": "Returns all pets from the system that the user has access to",
      "responses": {
        "200": {
          "description": "A list of pets.",
          "content": {
            "application/json": {
              "schema": {
                "type": "array",
                "items": {
                  "$ref": "#/components/schemas/pet"
                }
              }
            }
          }
        }
      }
    }
  }
}
```

Fonte: MILLER (2021)

Esta especificação do OpenAPI descreve um *endpoint* GET em “*/pets*” que recupera uma lista de animais de estimação no formato JSON. A resposta será uma matriz de objetos de animais de estimação, com a estrutura de cada animal de estimação descrita pelo esquema “*pet*” definido em outro lugar nos componentes da API. O *endpoint* destina-se a fornecer aos usuários acesso aos animais de estimação armazenados no sistema.

5 SERVIÇOS WEB SEMÂNTICOS

5.1 Web-Semântica

A informação disponível na Internet está definida e exposta de diversas formas. Enquanto para leitores humanos, mesmo documentos não-estruturados podem conter significado claro; para computadores, a legibilidade e “compreensão” de informações não se dão de forma óbvia. Na primeira Conferência Mundial da *World Wide Web,* em 1994, Berners-Lee destaca as limitações da Web para máquinas, observando a falta de informações legíveis por máquinas e a natureza plana e dependente de contexto de seus links.

Para possibilitar maior interoperabilidade e intercomunicação entre máquinas, a ideia de uma Web interpretável veio à tona. A diferença entre uma Web interpretável e uma Web processável está na capacidade dos algoritmos de navegar e coletar dados de forma inteligente, seguindo recursos significativos e ricos em semântica. (CHERON, 2021)

Enquanto o HTML torna a web processável, descritores semânticos em dados e recursos são cruciais para que os algoritmos diferenciem e interpretem os recursos com base em seu significado, possibilitando a recuperação inteligente de informações de uma forma automática em uma web acessível aos computadores. Segundo Chaves (2011), o propósito da semântica é “ao fazer uso das ontologias, anotar o conteúdo da Web, através de metadados interpretáveis por sistemas capazes de processar esse conteúdo a um nível semântico”.

Inicialmente, é necessário fazer uma distinção entre sintaxe e semântica no contexto da Web. A sintaxe refere-se às regras e convenções que governam a estrutura e o formato de uma linguagem ou representação de dados. Concentrando-se em como os dados são organizados, incluindo elementos, atributos e suas relações hierárquicas. A semântica, por sua vez, ocupa-se com a noção de sentido e interpretação dos dados, possibilitando sua compreensão por parte de outras máquinas.

Para a camada sintática, a web-semântica implementa a linguagem XML. Segundo a documentação oficial da linguagem, “documentos XML são compostos por unidades de armazenamento chamadas entidades, que contêm dados analisados ou não. Dados analisados são compostos por caracteres, alguns dos quais formam dados de caracteres e outros formam marcações. As marcações codificam uma

descrição do *layout* de armazenamento e estrutura lógica do documento. O XML fornece um mecanismo para impor restrições no *layout* de armazenamento e na estrutura lógica". (W3C, 2008)

Com relação à perspectiva semântica, para os propósitos descritivos da Web, o *framework* RDF (do inglês, *Resource Description Framework*) foi criado para representar informações na Web. (HARTIG, 2023) A sintaxe abstrata possui grafos que são instanciados a partir da utilização de triplas (principal unidade estrutural do RDF), que possibilitam a descrição de recursos na Web.

A estrutura de RDF, por funcionar utilizando uma estrutura de sujeito-objeto-predicado em suas descrições, possibilita até certo nível a descrição de sentido semântico de recursos na Web com a utilização de RDF-S (do inglês, *RDF Schema*). O RDF-S é um vocabulário dentro do RDF usado para definir e descrever a estrutura de ontologias e grafos de conhecimento, contendo classes básicas para a definição de ontologias simples.

Este vocabulário não é suficiente para garantir todos os níveis de expressividade necessários para a Web. Bechhofer (2004) define a linguagem OWL como uma extensão do RDF, sendo toda ontologia OWL construída sobre triplas RDF. A OWL permite a descrição mais completa de ontologias com definições mais abrangentes relativas às classes, propriedades e relações entre objetos e classes.

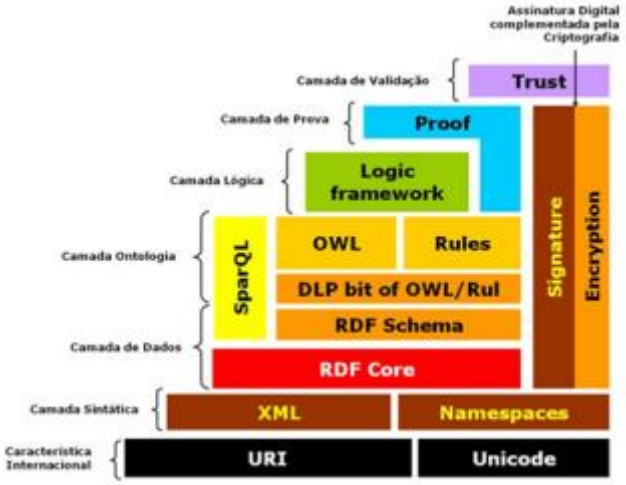
Ontologias em OWL são implementadas para os mais diversos propósitos. *Dublin Core* e o *Simple Knowledge Organization System* (SKOS) são ontologias populares no domínio do OWL. O *Dublin Core* é amplamente reconhecido por descrever metadados relacionados a recursos digitais, como documentos, páginas da web e conteúdo multimídia. Enquanto isso, o SKOS se destaca na representação de sistemas de organização do conhecimento, incluindo taxonomias, tesauros e vocabulários controlados.

No entanto, o mundo do OWL abriga um espectro diversificado de ontologias que atendem a domínios mais especializados. Um exemplo é o JudO, uma moderna biblioteca de ontologia OWL2 focada explicitamente no conhecimento jurídico. Ao aproveitar os metadados presentes em documentos judiciais, o JudO captura as interpretações feitas por juízes durante o raciocínio jurídico para a adjudicação de

casos. Essa ontologia, aproveitando recursos do OWL2, possibilita uma conexão com documentos de origem e busca conferir atribuição automática de significado para máquinas e análises do campo jurídico. (CECI, 2016)

Os frameworks da web semântica são capazes de descrever serviços da Web, fornecendo uma maneira estruturada e padronizada de representar e transmitir a semântica de dados e interações. Esses frameworks permitem a definição de ontologias, vocabulários e modelos de dados que tornam o significado de recursos e operações de serviço da web explícitos. A Figura 7 demonstra a estrutura hierárquica da Web-Semântica:

Figura 7 – Estrutura da Web-Semântica



Fonte: disponível em https://mundodalogica.files.wordpress.com/2013/05/as.png

Dentro do contexto da Web-Semântica, um tema relevante é o de *Linked Data*. *Linked Data* ou Dados associados e Web-Semântica são conceitos relacionados, mas distintos. *Linked Data* concentra-se principalmente nos princípios e melhores práticas para interconectar dados na Web de maneira padronizada e estruturada. Em contraste, a Web-Semântica, é um conceito mais amplo que abrange a ideia de adicionar significado e contexto aos dados da Web.Enquanto o Linked Data é uma abordagem específica para alcançar os objetivos da Web-Semântica, esta abrange um espectro mais amplo de tecnologias e princípios para aprimorar a riqueza semântica e a acessibilidade dos dados da Web.

5.2 Linked Data

O conceito de dados associados está relacionado à ideia de interconexão no âmbito das tecnologias da Web-Semântica. Isso enfatiza o uso de URI's para estabelecer conexões significativas entre conjuntos de dados e recursos diversos, criando uma rede de conhecimento onde a informação não está isolada, mas interconectada.

Os dados interligados permitem a navegação e a descoberta contínuas de informações relacionadas, fomentando uma rede de recursos interligados que aprimoram a integração de dados e o compartilhamento de conhecimento. Essa abordagem facilita a criação de uma Web de informações estruturadas e semanticamente ricas, capacitando aplicativos e usuários a explorar, consultar e derivar *insights* dos dados de maneira mais coerente e significativa, promovendo uma maior interoperabilidade e utilidade dos dados.

Para Berner-Lee (2009), criadores do conceito de *Linked Data*, os princípios fundamentais sob os quais essa estruturação da Web viria à tona seriam a utilização de URI's para a identificação de recursos, a padronização da documentação presente na URI de cada recurso e a interconexão entre os recursos de dados e informações associadas sendo fidedignamente demonstradas.

Apesar disso, Berners-Lee (2009) também afirma que a Web de dados interligados atual é majoritariamente uma Web apenas de leitura, e que embora essa Web permita a integração, navegação e consultas estruturadas em grandes conjuntos de dados, ainda falta um conceito geral para uma Web de dados de leitura e escrita.

O conceito de *Linked Data* também traz, no contexto da Web-Semântica, a possibilidade de realizar buscas associadas às informações disponíveis no sistema. No contexto de bases de dados implementadas em RDF, é possível a utilização da linguagem de busca SPARQL (do inglês, *Protocol and RDF Query Language*) para encontrar características e informações sobre os dados descritos.

Conforme explica Segundo (2014), as consultas que utilizam SPARQL possuem os seguintes elementos:

- **Prefixos:** Declarações de prefixos, para abreviar URI's, simplificando as consultas para que os textos das buscas não fiquem muito extensos.

- **Conjunto de dados:** Informação relativa à quais conjuntos de dados estão sendo buscados nas buscas de SPARQL.
- **Cláusula de resultado:** Define como a informação deve ser retornada da consulta SPARQL.
- **Padrão de triplas:** Utilizando o conceito de triplas de RDF, define os critérios para a busca ser realizada em relação às condições dos objetos.
- **Modificadores:** Condições especiais e termos que, se colocados nas buscas, podem modificar o resultado esperado de uma busca.

Uma busca simples de SPARQL pode ser exemplificada na Figura 8, onde a busca tem por objetivo retornar todos os objetos da classe "*person*" no banco de dados que cumprem a condição "*hasName*", isto é, que possuem um nome:

Figura 8 – Busca em SPARQL

```
PREFIX ex: <http://example.com/ontology#>

SELECT ?personName
WHERE {
  ?person ex:hasName ?personName .
}
```

Fonte: O autor (2023)

Outro propósito da implementação de *Linked Data* é a diminuição da redundância de dados nas informações encontradas em todos os bancos de dados conectados às redes de conhecimento. Com a geração automática de URI's para a identificação de recursos de dados, o reuso de dados entre aplicações também é maximizado, diminuindo a necessidade de recursos de memória alocados. (MARJIT, 2013).

Além disso, por permitir a identificação massiva de recursos de dados, o *Linked Data* vem sendo utilizado para o registro de bibliotecas de conhecimento. O estudo de Araraki (2017) demonstra como diversas bibliotecas têm adotado o *Linked Data* para aprimorar o acesso e a análise de informações bibliográficas. Alguns dos principais esforços incluem a *Dublin Core Metadata Initiative* (DCMI), a *Library of Congress* com o BIBFRAME e a *Online Computer Library Center* (OCLC) com o *WorldCat*.

O conceito de *Linked Data* é essencial também para a criação de microsserviços semânticos orientados à dados. O trabalho de Salvadori (2019) discute a integração de microsserviços com tecnologias da Web-Semântica para criar uma rede de recursos interligados. Enquanto soluções existentes se concentram na publicação de dados interligados, o trabalho apresentado por Salvadori (2019) introduz o *sdd-μs*, um serviço em nuvem especializado que converte dados não semânticos em dados interligados, maximiza a reutilização de dados e infere novos conhecimentos a partir de recursos disponíveis.

Apesar da potencialidade das concepções de *Linked Data*, Lanthaler (2011) argumenta que em vez de fornecer dados interligados por meio de serviços Web RESTful, os esforços atuais implantam *endpoints* SPARQL ou simplesmente fazem *upload* de dados estáticos (*Dump*), a fim de proporcionar o acesso aos dados. O autor afirma que isso raramente reflete a natureza dos dados fornecidos.

Lanthaler (2011) acrescenta ainda outro problema das abordagens atuais com relação à Web de dados interligados. Tais abordagens geralmente fornecem interfaces apenas de leitura para os dados subjacentes. Portanto, ainda há uma expressiva carência de serviços de acesso a dados em nível de entidade provedores de dados interligados, e os que existem possuem limitações como a falta de suporte à escrita.

5.3 Serviços Web-Semânticos

Os Serviços Web-Semânticos foram propostos por McIlraith (2001) e podem ser definidos como serviços Web cujas propriedades, capacidades, interfaces e efeitos são codificados de forma não ambígua, possíveis de serem interpretados por máquinas. Em outras palavras, a semântica referente aos dados é descrita por meio da utilização de ontologias.

Outra forma de definir serviços web-semânticos é a exposta por Cheron (2021), onde estes são serviços Web que tem sua descrição definida por meio da utilização de ontologias que podem ser compreendidas por máquinas, na qual a descrição dos serviços segue padrões definidos previamente e amplamente conhecidos e implementados em diversas máquinas e que assim, possibilitam uma

interoperabilidade e compreensão compartilhada da maneira como os serviços operam.

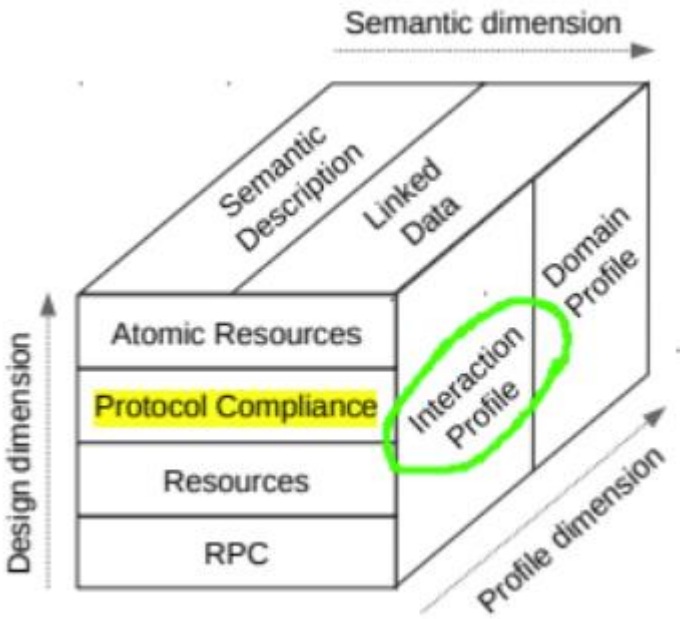
A visão dos serviços web-semânticos proposta por Mcilraith a Martin (2003) é combinar serviços Web com a web-semântica e através disso ser capaz de uma interação dinâmica e automática entre os sistemas de informações. Prazeres et al. (2009) acrescenta explicando que Serviços Web-Semânticos fornecem a oportunidade para aumentar a automação das tarefas de descoberta, seleção, invocação e composição de serviços Web.

Desde os princípios da Web, o modelo de descrição de aplicações utilizando HTML havia sido priorizado. Ademais, para fazer sua incrementação semântica, tecnologias como hREST e RDFa foram criadas, visto que essas tecnologias implementam microformatos para possibilitar descrições interpretáveis por máquinas das descrições de serviços já pré-existentes utilizando HTML. Evoluções posteriores com *frameworks* como MicroWSMO e SA-REST focam na interoperabilidade e modificação de formatos de definições pré-existentes.

Além das descrições previamente citadas que são mais focadas na perspectiva da exposição de contratos sintáticos entre aplicações, é possível utilizar *frameworks* como OWL-S, OWL2, RESTDesc e WSMO para fazer a definição de descrições semânticas das funcionalidades dos serviços. (CHERON, 2021)

Serviços Web-Semânticos viabilizam também a criação de modelos de qualidade e maturidade relacionados às aplicações que adotam essa abordagem descritiva. O estudo de Salvadori (2015) demonstra como a utilização de um modelo de maturidade chamado WS3, que analisa interfaces RESTful a partir de uma perspectiva semântica, auxilia na compreensão de métricas de qualidade e maturidade nas aplicações. A Figura 9 representa o modelo visualmente, na qual três dimensões estão representadas.

Figura 9 – Modelo Descritivo WS3



Fonte: SALVADORI (2015)

A dimensão de perfil é referente ao enquadramento da documentação oferecida pela interface, diferenciando aquelas que são mais focadas na descrição dos processos de interação com o sistema das documentações focadas na descrição de todo o domínio da aplicação. A dimensão de design explora a estratégia de definição adotada pela documentação da interface. Finalmente, a dimensão semântica avalia se a interface suporta, factualmente, descrições semânticas ou se é limitada nesse sentido.

Devido à resistência de desenvolvedores em relação à utilização de descrições semânticas para a incrementação de seus serviços web, por sua ausência de familiaridade com as tecnologias de descrições semânticas da Web, várias metodologias foram propostas para o desenvolvimento de metodologias automáticas de incrementação semântica de serviços web pré-existentes.

Salvadori (2019) propõe a arquitetura *OntoGenesis* para mapeamento de possíveis definições semânticas para serviços de interação com dados e um microsserviço capaz de devolver dados semanticamente organizados a partir de fontes não-semanticamente descritas. Apesar de ser uma abordagem complexa, a metodologia de Salvadori (2019) não se encontra como única representante da tentativa de implementação de formas automatizadas de descrição semântica de serviços Web.

Como demonstra Chavez (2019) em seu estudo, onde as técnicas apresentadas utilizam desde o mapeamento de características (utilizando algoritmos baseados em grafos) à implementação de serviços de mineração de dados. Os serviços web-semânticos pretendem utilizar a semântica previamente estabelecida para os conceitos de uma ontologia associando esta semântica aos componentes dos serviços Web.

Existem três abordagens principais utilizadas para a descrição de serviços utilizando web-semântica: Anotar as descrições dos serviços fazendo-as apontar para ontologias, anotar as ontologias de domínio indicando quais serviços Web existem referentes aquela ontologia ou criar uma especificação nova, independente das duas primeiras. Esta nova especificação é, normalmente, uma meta-ontologia que descreve os serviços. A distribuição das abordagens e as respectivas tecnologias que elas utilizam podem ser encontradas abaixo:

Tabela 3 – Serviços Web Semânticos

Serviços Web	Anotações Semânticas na Descrição de Serviços	Ontologias sobre Serviços	Anotações em Ontologias especificando Interface Semânticas
Baseados em SOAP/WSDL	SAWSDL	OWL-S, SWSF WSMO	Active OWL
Baseados em REST	SA-REST, SBWS(SA-WADL)	EXPRESS	SERIN

Fonte: O autor (2023).

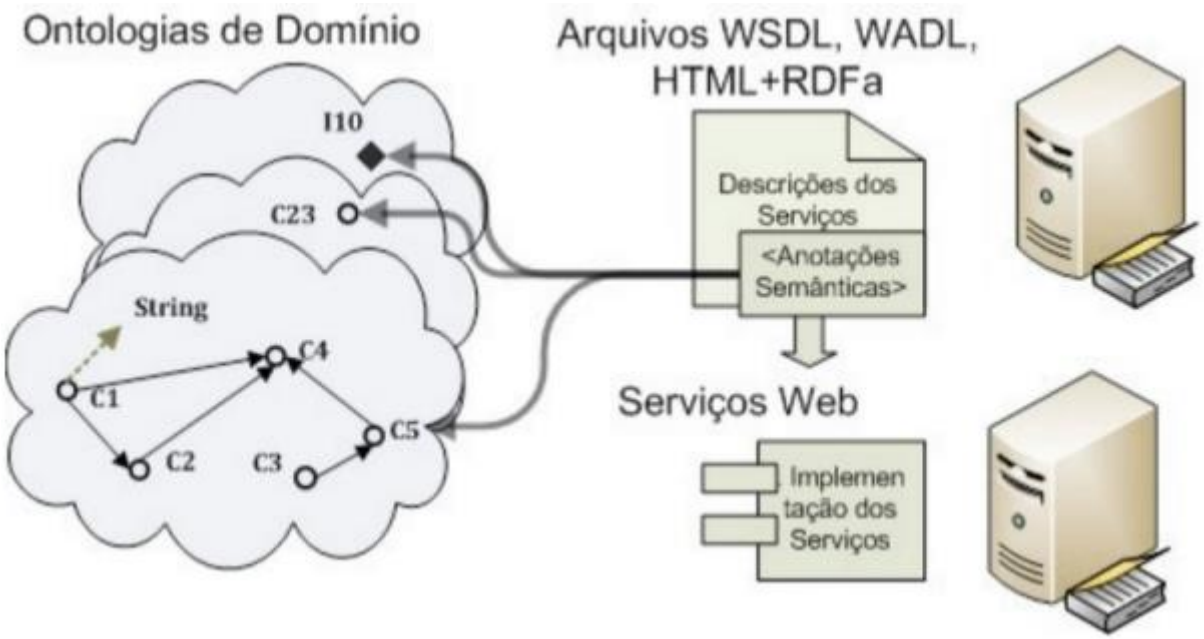
Os serviços web-semânticos podem ser baseados em duas categorias de serviços Web: serviços Web baseados em SOAP/WSDL e serviços Web baseados em RESTFul. Para cada uma dessas categorias, que serão discutidas nas subseções a seguir, detalharemos as três estratégias apresentadas na Tabela 3, além das tecnologias e ferramentas utilizadas para a implementação das estratégias em si.

5.3.1 Serviços web semânticos usando anotações nas descrições dos serviços

Uma primeira estratégia para associar semântica a serviços Web é incluir anotações na descrição dos serviços. Essas anotações associam as entradas e saídas dos serviços a componentes de ontologias de domínio. A Figura 10 mostra o

relacionamento entre conceitos de uma ontologia com anotações semânticas que são inseridas nos documentos de especificação dos serviços web.

Figura 10 – Anotações semânticas inseridas na descrição de serviços



Fonte: MUNIZ (2014)

Entre os serviços Web semânticos SOAP/WSDL, o SAWSDL (do inglês, *Semantic Annotation for WSDL*), baseado no WSDL-S (do inglês, *Web Service Semantics*), Akkiraju (2005) segue a estratégia de anotar semanticamente a descrição dos serviços. Para isso, Chinnici (2007) define três novos atributos de extensibilidade aos elementos WSDL 2.0 de forma a permitir as anotações semânticas em componentes WSDL:

- **modelReference:** É um atributo de extensão para especificar a associação entre um componente WSDL e um conceito em algum modelo semântico. O atributo *modelReference* é utilizado para anotar definições de tipos complexos, XML *Schema*, definições de tipos simples, declaração de atributos, assim como interfaces WSDL, operações e falhas.
- **liftingSchemaMapping:** Este atributo define a forma como o sistema recebe como entrada dados XML (que seguem um esquema XML fornecido) e produz dados semânticos (que seguem um modelo semântico, no nosso caso uma ontologia RDF) como saída.

- **loweringSchemaMappping:** Faz o processo contrário do atributo *liftingSchemaMapping*, isto é, define a forma como o sistema recebe como entrada dados semânticos a partir de descrições RDF e as transforma em XML. (W3C, 2007)

O SA-REST, SBWS (do inglês, *Semantic Bridge for Web Services*) e SAWSDL são exemplos de estratégias que anotam semanticamente as descrições dos serviços web semânticos. SA-REST adiciona metadados às páginas XHTML a partir de modelos diferentes, como uma ontologia, taxonomia ou *tag cloud*. (GOMADAM, 2005), (BATTLE, 2007) e (W3C, 2007)

Estes metadados se propõem a melhorar a pesquisa, facilitar a mediação de dados e proporcionar uma integração mais fácil entre serviços. A proposta SA-REST, conforme demonstrou Gomadam (2005), foi criada para adição de semântica aos serviços RESTful, por empréstimo da ideia do SAWSDL. A principal distinção entre SAWSDL e SA-REST é que na ferramenta SAWSDL as anotações semânticas são acrescentadas ao código WSDL (que não são normalmente adotados para serviços Web RESTful).

As anotações do SA-REST serão adicionadas na descrição HTML de páginas Web que descrevem os serviços através da utilização de microformatos (um subconjunto de poshformatos) como RDFa e "*Gleaning Resource Descriptions from Dialects of Language*s". (HERMAN, 2015) e (CONNOLLY, 2007)

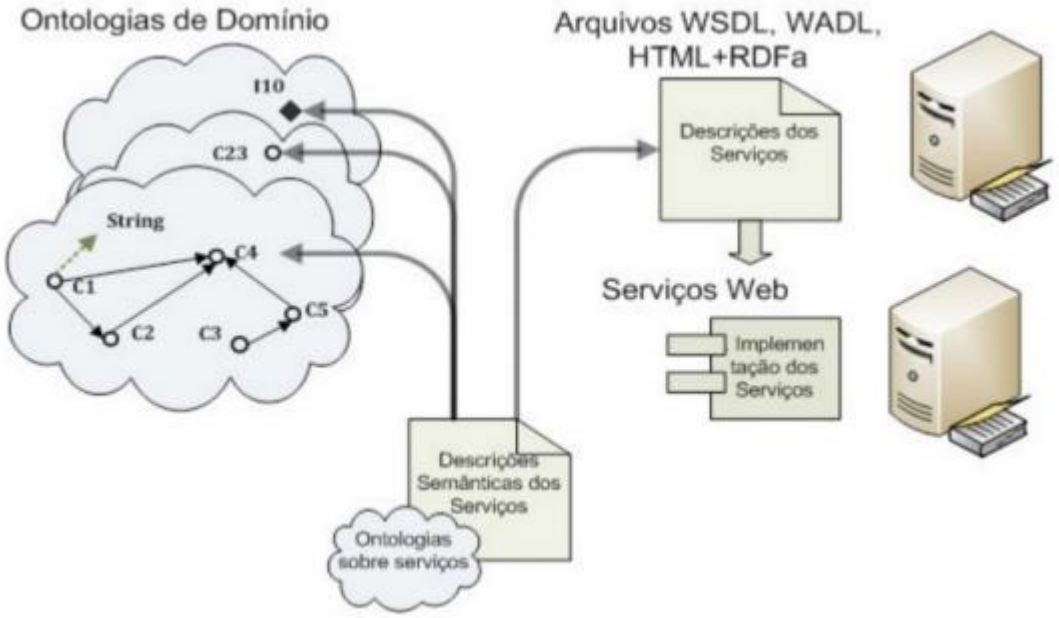
O SBWS é uma ferramenta Java utilizada para integrar serviços web e criar *endpoints* SPARQL comunicando estes serviços. O objetivo desta proposta é prover uma ferramenta que analise comandos SPARQL e decida qual a combinação de serviços web que atendem à solicitação. Para tanto, esta especificação adiciona uma descrição semântica aos serviços Web RESTful, a partir de anotações realizadas em documentos WADL que mapeiam as entradas e saídas do serviço a ontologias. (PRUD'HOMMEAUX, 2008)

5.3.2 Ontologias descrevendo serviços

Outra abordagem utilizada é adicionar uma especificação nova que descreva o serviço. Desta forma, não se modificam nem a descrição do serviço Web nem a

ontologia de domínio. Em vez disso, especifica-se uma nova descrição de serviços Web utilizando uma ontologia sobre os serviços, como mostrado na Figura 11:

Figura 11 – Ontologias descrevendo serviços



Fonte: MUNIZ (2014)

Ankolekar (2004) define a OWL-S, uma ontologia OWL de serviços para descrever as propriedades e recursos dos serviços Web. A ontologia define o conceito de *service* (serviço) e três sub-ontologias OWL-S, conhecidas como *service profile*, *service model* e *service grounding* (NETO, 2007).

A classe s*ervice* é uma base genérica para serviços Web, representando conceitos amplos e abstratos que podem ser aplicados a diversas áreas, mas não incluem detalhes específicos de domínios. Por essa razão, a classe *service* pode ser considerada uma forma de ontologia superior (*Upper Ontology*).

Ela serve como estrutura para a criação de ontologias de domínio mais específicas, como as médicas, financeiras ou de engenharia. As sub-ontologias podem ser explicadas da seguinte forma:

- **Service Profile:** Essa sub-ontologia descreve as funções de um serviço de uma maneira que permite a um agente de busca de serviços determinar se ele atende aos critérios de busca. Inclui informações sobre as ações do serviço, suas limitações, qualidade e requisitos para os clientes.

- **Service model:** Informa aos clientes como usar o serviço, explicando a semântica das requisições, as condições para resultados específicos e, quando necessário, os passos a serem seguidos para alcançar esses resultados. Em resumo, descreve como chamar o serviço e o que esperar quando ele é executado.
- **Service grounding:** Especifica como um agente pode acessar o serviço, detalhando o protocolo de comunicação, formatos de mensagens e outros detalhes específicos, como números de portas. Também aborda a serialização, ou seja, como as entradas e saídas são formatadas.

O *Framework* de Serviços Web-Semânticos, SWSF, é fundamentado no padrão OWL-S. Ele consiste em uma ontologia chamada *Semantic Web Services Ontology* (SWSO) e uma linguagem mais robusta que o OWL, chamada *Semantic Web Service Language* (SWSL). A SWSL é composta por duas sublinguagens: SWSLFOL, baseada na lógica de primeira ordem, usada para descrever conceitos de serviços web, e SWSL-Rules, baseada em lógica de programação por regras, para apoiar o uso de ontologias em inferências e ambientes de execução. (BATTLE et al., 2005)

Brujin (2005) explica que a *Web Services Modeling Ontology* (WSMO) “tem como objetivo descrever todos os aspectos relevantes relacionados a serviços que são acessíveis através de uma interface de serviço Web. Seu objetivo principal é permitir a automação (total ou parcial) de tarefas (e.g. descoberta, seleção, composição, mediação, execução e monitoramento)”. O autor define quatro elementos principais para a descrição de semântica em serviços Web:

- **Ontologias:** que fornecem a terminologia utilizada por outros elementos WSMO.
- **Descrição de serviços Web:** que descreve os aspectos funcionais e comportamentais de um serviço Web.
- **Metas:** que representam as intenções dos usuários.
- **Mediadores:** que tem por objetivo tratarem automaticamente os problemas de interoperabilidade entre os elementos WSMO.

O EXPRESS (*RESTful Expressing Semantic Services*) descreve serviços web RESTful usando ontologias de domínio para criar interfaces de acesso a recursos em hosts. Os provedores usam ontologias OWL para definir entidades e relações. Um

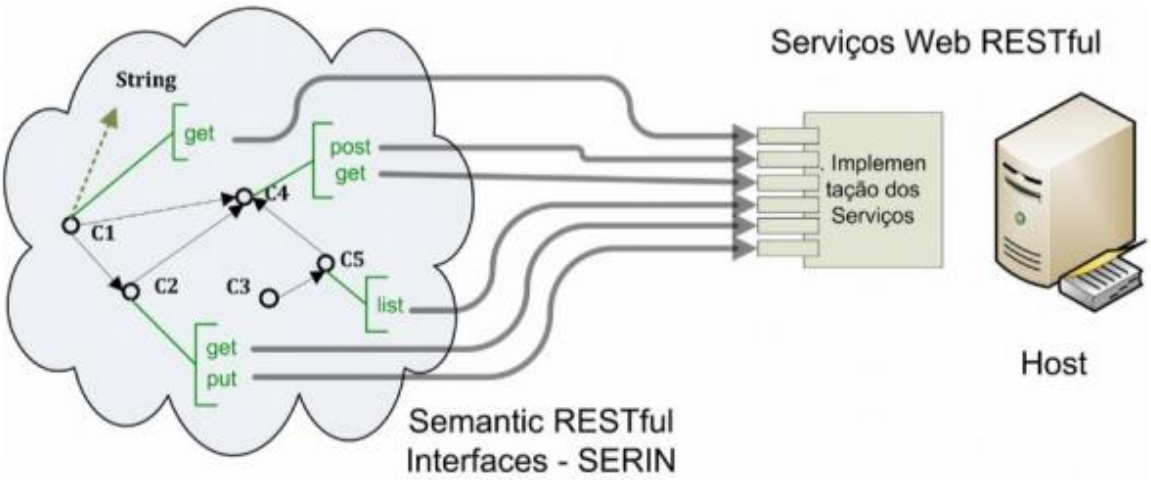
motor EXPRESS gera interfaces e URI's com base nisso. O provedor indica operações HTTP disponíveis. No entanto, o mecanismo de implantação EXPRESS ainda não foi implementado e requer trabalho futuro. (ALOWISHEQ, 2009)

5.3.3 Anotações em Ontologias especificando Interface Semânticas

O *Active OWL* adota uma estratégia de associar serviços Web às ontologias, utilizando anotações para indicar a existência de tipos de serviços Web em classes e propriedades. Essas anotações são definidas em OWL por meio *de Annotation Properties*, mas não se aplicam a classes anônimas ou criadas a partir de expressões, devido à falta de referência explícita.

Além disso, as anotações não são herdadas, o que significa que classes-filhas precisam especificar seus próprios serviços web. Essa abordagem inicial tem limitações devido à complexidade de inferência por máquinas e à ausência de conceito de herança em serviços web. No entanto, uma futura expansão do *Active OWL* poderia abordar essas limitações, possibilitando o tratamento de falhas na execução de serviços web.

Figura 12 – Ontologia anotadas - Interfaces Semânticas



Fonte: MUNIZ (2014)

5.4 Interfaces Abstratas para Serviços WEB

O termo interface é frequentemente utilizado com pelo menos dois significados diferentes na academia. O primeiro significado é associado ao padrão de *design* "*Facade*" e o segundo, que será enfatizado no presente estudo, refere-se a uma

interface abstrata, um contrato a ser seguido por classes concretas que implementarão a *interface*. Muitas vezes a *facade* (fachada) é também uma interface abstrata o que reforça o duplo sentido do termo.

O padrão de *design* estrutural *facade* propõe a utilização de uma classe que serve como fachada a subsistemas complexos. Isto permite que detalhes internos sejam modificados enquanto a fachada permanece inalterada preservando também inalterados os sistemas que acessam a fachada. De forma distinta, as interfaces classificadas como *facade* desempenham um importante papel na interoperabilidade de programas em vários níveis de abstração: a nível de API, de objetos distribuídos, de serviços Web, dentre outros.

A nível de API's, interfaces são utilizadas para interligação de camadas de software. Fabricantes de um sistema operacional, por exemplo, podem especificar uma API a ser seguida pelos elaboradores de *drivers* de dispositivos, ou para a integração com funcionalidades internas de administração de ferramentas padrão. Frequentemente API's são padronizadas por um consórcio de fabricantes de *software* que compartilham o entendimento de seu significado. Neste caso a semântica é informada de maneira também informal através de documentações textuais referentes a cada API.

A nível de objetos distribuídos são especificadas interfaces, por exemplo, utilizando IDL (do inglês, *Interface Definition Language*) como Corba, ODL (do inglês, *Object Definition Language*) como COM, RMI (do inglês, *Remote Method Interface*) como Java e pode-se definir um repositório de interfaces.

A nível serviços Web, interfaces são um constituinte interno da especificação do WSDL (somente a partir da versão 2.0). Entretanto as especificações WSDL além das interfaces incluem também o "*binding*" (ligação), indicando quem concretamente implementa os serviços Web. Portanto, a nível Web, interfaces abstratas normalmente não são utilizadas de forma independente de sua implementação. Os serviços Web RESTFul também se referem a recursos concretos, mas não a interfaces abstratas.

As linguagens orientadas a objetos, como Java, popularizaram a segunda acepção do termo interface ao especificar as interfaces abstratas. Nas linguagens orientadas a objetos interfaces especificam tanto um conjunto de constantes como um

conjunto de assinaturas de métodos a serem concretamente construídos por quem vier a implementar a interface. A semântica das operações preconizadas na interface abstrata é comunicada de maneira informal aos interessados.

Segundo os tutoriais oficiais disponibilizados para o Java pela Oracle (2022), a implementação de interfaces é adequada para situações em que os mesmos comportamentos do sistema devem ser efetuados por diversas classes não relacionadas. Um exemplo de definição de Interface implementada na Figura 13.

Figura 13 – Interface Implementada em Java

```
public interface GroupedInterface extends Interface1, Interface2, Interface3 {

    // constant declarations

    // base of natural logarithms
    double E = 2.718282;

    // method signatures
    void doSomething (int i, double x);
    int doSomethingElse(String s);
}
```

Fonte: disponível em https://docs.oracle.com/javase/tutorial/java/IandI/interfaceDef.html

Este código Java declara uma interface chamada “*GroupedInterface*” que herda de três outras interfaces (“*Interface1*”, “*Interface2*” e “*Interface3*”). Em seguida, define uma constante “E” com o valor da constante matemática “E”. Além disso, declara dois métodos abstratos: “*doSomething*”, com parâmetros inteiros e de ponto flutuante e “*doSomethingElse*”, com um parâmetro de *string*, os quais qualquer classe que implemente essa interface deve fornecer implementações concretas.

O conceito de interface normalmente presume que os envolvidos adotem uma mesma semântica na sua utilização. Uma ontologia se presta exatamente a descrever conceitos em relação aos quais os envolvidos concordam com sua semântica. Diante disso, parece conveniente a utilização de ontologias para descrever interfaces semânticas.

Muito embora a expressão interface semântica se revista de uma certa redundância, a palavra semântica empregada neste contexto, visa enfatizar a utilização das tecnologias relacionadas com a Web semântica na definição dessa

interface. Interface semântica abstrata é uma especificação que descreve um conjunto de operações abstratas nas quais os parâmetros, identificador da operação e o valor de retorno de cada operação têm sua semântica determinada através de uma ontologia.

5.5 SERIN - Interfaces RESTful Semânticas

O padrão REST se fundamenta na orientação a recursos e na utilização dos métodos HTTP como únicas operações disponíveis. Interfaces RESTful semânticas (SERIN) são basicamente ontologias anotadas nas quais classes e propriedades caracterizam semanticamente e univocamente os recursos disponíveis.

Através das anotações indica-se quais serviços Web RESTful deverão ser implementados para cada recurso. De forma mais específica, uma Interface RESTful Semântica é um tipo de interface semântica em que os recursos são representados como conjuntos de triplas (seguindo o modelo descritivo RDF), geralmente armazenados em um repositório de memória compartilhada.

Esses recursos são acessíveis a processos que participam de um mesmo contexto computacional. Cada recurso tem uma URI associada, que pode ser usada para acessar e modificar as informações e os dados associados aos recursos. A característica chave é que esses recursos e suas triplas podem ser manipulados usando solicitações HTTP padrão enviadas ao URI do recurso, seguindo os princípios REST.

Uma contribuição relevante da proposta SERIN é o de descrever interfaces abstratas para Serviços Web. Um segundo aspecto importante é o da semântica indicada através de ontologias OWL. Um terceiro aspecto, considerado útil e sempre presente nas linguagens orientadas a objetos, mas não nos serviços web, é o de herança. Este terceiro aspecto de adaptar o conceito de herança e consequentemente o de sobreposição para os serviços Web foi deixado como trabalho futuro na especificação SERIN.

SERIN é fundamentada na especificação anterior chamada Active OWL, que se aplica a serviços Web SOAP/WSDL. A estratégia do SERIN para vincular serviços web às ontologias envolve a adição de cinco tipos de anotações em classes e

propriedades de uma ontologia OWL. Essas anotações representam os tipos de serviços Web: “GET”, “POST”, “DELETE”, “PUT” e “LIST”. Em essência, uma Interface RESTful Semântica é uma ontologia OWL com essas anotações adicionadas. Esses cinco tipos de serviços Web são mapeados nos quatro métodos HTTP, com “GET” e “LIST” associados ao método GET. (NETO, 2007)

Este método requer uma chave como parâmetro para recuperar uma instância específica, enquanto o “LIST” não tem parâmetros e retorna uma lista de todas as instâncias da classe. É importante observar que o SERIN não define a semântica dos serviços Web em si, mas se concentra exclusivamente nos cinco tipos mencionados. Essa abordagem é adequada para serviços Web RESTful que também se baseiam em métodos HTTP.

Além disso, ao se restringir aos métodos HTTP, o SERIN é consideravelmente mais simples em comparação com outras propostas, pois após a definição da semântica dos métodos, resta apenas estabelecer a semântica dos argumentos e dos resultados. A limitação das Interfaces RESTful Semânticas à adequação aos princípios RESTful traz benefícios no que tange à simplicidade descritiva necessária em comparação aos serviços Web semânticos tradicionais. (CHAVES, 2011)

Por serviços Web RESTful se utilizarem dos métodos HTTP para organizar a descrição e distribuição de recursos, a anotação dos mesmos pode ser baseada nos métodos HTTP amplamente conhecidos (GET, PUT, POST etc.). O estudo de Chaves (2011) apresenta a Figura 14 para a anotação de propriedades em interfaces RESTful semânticas a nível de classe:

Figura 14 – Anotações a nível de classe

get	URL	GET <host>/serin/<Ont>/<Classe>/<rdf:id>
	Semântica	Recebe como parâmetro o identificador <rdf:id> do indivíduo e retorna o respectivo indivíduo
list	URL	GET <host>/serin/<Ont>/<Classe
	Semântica	Lista os indivíduos da Classe
post	URL	POST <host>/serin/<Ont>/<Classe>/<indivíduo>
	Semântica	Recebe como parâmetro o XML correspondente ao indivíduo e acrescenta-o ao conjunto de indivíduos da Classe
put	URL	PUT <host>/serin/<Ont>/<Classe>/<rdf:id>/<indivíduo>
	Semântica	Recebe como parâmetro o OWL correspondente ao indivíduo identificado por <rdf:id> e atualiza as informações do indivíduo da Classe
delete	URL	DELETE <host>/serin/<Ont>/<Classe>/<rdf:id>
	Semântica	Recebe como parâmetro o identificador do indivíduo e remove o indivíduo do conjunto de indivíduos da Classe

Fonte: CHAVES (2011)

5.6 Web API's Semânticas RESTful

Uma outra forma de disponibilização de anotações semânticas sobre sistemas da Web que implementam princípios RESTful são as Web API's Semânticas RESTful. Estas são aplicações que permitem que aplicativos acessem e manipulem dados na Web Semântica. Essas API's fornecem uma maneira padronizada de interagir com fontes de dados vinculados, como gráficos RDF e ontologias.

Elas permitem que desenvolvedores consultem e obtenham dados estruturados, realizem raciocínio semântico e vinculem conjuntos de dados diferentes para uma recuperação de informações mais significativa e contextualizada. Essa abordagem combina princípios REST com representação semântica de dados e atributos, permitindo a recuperação e modificação em dados de recursos de maneira distribuída e compatível com a Web. (GARROTE, 2010)

Uma pergunta razoável a ser levantada é o questionamento em relação às vantagens da utilização de Web API's Semânticas RESTful em comparação com a utilização de Web API's RESTful comuns combinada com uma descrição não-semântica de seus recursos, como por exemplo, a utilização da especificação OpenAPI para a definição de API's RESTful.

Embora a especificação OpenAPI sirva como um padrão amplamente aceito para descrever interfaces de serviços da Web, ela se concentra principalmente na interoperabilidade sintática, focando nos aspectos técnicos das capacidades dos serviços. No entanto, para uma descrição abrangente de serviços, especialmente em ambientes complexos e heterogêneos, as Web API's Semânticas RESTful demonstram maior capacidade de expressividade.

As Web API's Semânticas RESTful combinam descrições sintáticas com riqueza semântica para proporcionar uma compreensão mais ampla e significativa dos serviços da Web. Ao anotar descrições de serviços da Web com dados semânticos, as Web API's Semânticas RESTful permitem uma descoberta, composição e mediação de serviços aprimoradas. Essas API's eliminam ambiguidades e fornecem um contexto semântico claro para recursos, melhorando, em última análise, a qualidade das descrições de serviços e facilitando a interoperabilidade contínua com base em semânticas interpretáveis por máquinas. (MUHAMAD, 2020)

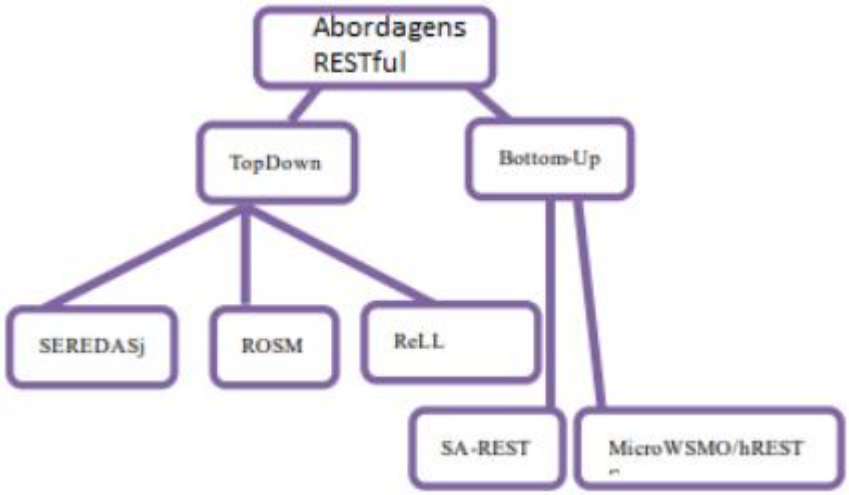
A Figura 15 explicita que, de acordo com Slimani (2013), existem dois tipos de abordagens comumente usada para a descrição de interfaces Web API's semânticas RESTful: As abordagens "*top-down*" (do nível mais generalista para as especificidades) e as abordagens "*bottom-up*" (do nível mais específico ao nível mais generalista).

O *framework* SEREDASj por exemplo é citado como uma ferramenta de abordagem "*top-down*" porque começa com uma descrição semântica de alto nível dos serviços, tem como objetivo transformar dados em recursos semânticos e trazer inicialmente uma perspectiva de descrição mais genérica. Essas características estão alinhadas com os princípios de design e descrição *top-down* no contexto de serviços da web semânticos e, consequentemente, com a descrição *top-down* de Web API's semânticas RESTful.

Em contraste com essa abordagem, o SA-REST é considerado uma abordagem "*bottom-up*" para a descrição de interfaces RESTful semânticas porque se concentra na anotação não intrusiva de documentos HTML/XHTML existentes, define propriedades específicas para elementos individuais e começa a partir do nível mais

específico para, potencialmente, criar uma descrição semântica mais geral à medida que mais documentos são anotados.

Figura 15 – Distribuição das Abordagens de Descrição RESTful em Slimani



Fonte: SLIMANI (2013)

6 RESTRIÇÕES DE INTEGRIDADE NA WEB SEMÂNTICA

6.1 Restrições de Integridade em Interfaces RESTful Semânticas

Restrições de integridade de dados podem ser achadas também no contexto de mundo aberto, todavia, as que são apresentadas em contextos de mundo fechado possuem particularidades distintivas alinhadas às limitações deste contexto. As restrições de integridade semântica desempenham um papel fundamental na manutenção da qualidade e consistência dos dados que podem ser obtidos e inseridos através de interfaces RESTful semânticas, mesmo quando operam dentro de uma suposição de mundo localmente fechado (LCWA).

A LCWA é uma variação da suposição de mundo fechado (CWA) que considera um contexto ou domínio específico como fechado, enquanto reconhece a possibilidade de um mundo aberto além desse contexto. Dentro da LCWA, as restrições de integridade semântica desempenham várias funções vitais. Elas garantem que os dados dentro do contexto definido sejam completos e coerentes, aderindo às regras e relações localizadas explicitamente definidas para esse domínio.

Segundo o elucidado por Denecker (2010), as primeiras conceituações relacionadas à noção de LCWA eram focadas justamente na descrição deste contexto

para a compreensão do processo de inserção e leitura de informações em bases de dados. Doherty (2000), por sua vez, explica que a presunção de mundo aberto e a presunção de mundo fechado representam dois extremos. Por isso, a utilização da presunção de um mundo fechado localmente ajuda, em situações como a que em Interfaces RESTful Semânticas estão por vezes imersas, descrições a serem suficientemente expressivas.

Essas restrições validam os dados em relação a critérios específicos do contexto, impondo tipos de dados, unicidade e integridade referencial. Elas também suportam a inferência contextual, permitindo deduções e inferências lógicas dentro do mundo localizado. Durante a avaliação de consultas, as restrições de integridade semântica ajudam a determinar a validade e relevância dos resultados dentro do contexto especificado e no processo de inserção de dados, a garantir a confiabilidade dos dados inseridos para a prevenção de erros e inconsistências no sistema.

Fundamentalmente, essas restrições adaptam a gestão de dados e o raciocínio ao escopo da LCWA, mantendo a confiabilidade dos dados, ao mesmo tempo em que reconhecem que as suposições de completude e consistência podem ser diferentes fora do contexto fornecido. Lira (2015) apresenta seis restrições de integridade distintivamente particulares ao contexto de mundo fechado localmente:

- **Não-Nulidade:** Valores que são registrados não podem possuir atributos nulos. A vantagem observada dessa restrição é de que os dados presentes no sistema estarão sempre consistentes com as expectativas postas pelo seu criador. Os dados sempre serão registrados e retornados de forma completa e cumprindo todos os critérios pré-definidos. Tal restrição também demonstra grande valor para a garantia do cumprimento de regras de negócio, visto que atributos que são especificados no sistema sempre cumprem um propósito funcional e por isso existem nos objetos.
- **Unicidade de Atributos:** Atributos que pertencem a um objeto dentro de um sistema devem ter um valor único para este objeto, isto é, um objeto não pode ter dois valores associados com um mesmo atributo. Este valor pode ser um conjunto de subvalores, porém a informação armazenada não pode possuir um conflito onde duas informações afirmam apresentar o verdadeiro valor de um atributo.

Essa restrição impossibilita uma possível não-determinação em relação ao valor correto de um atributo, garantindo assim a consistência dos dados do sistema.

- **Identificação:** Todo objeto inserido no banco de dados deve ter um identificador único relacionado a si. Em toda tupla de objetos onde eles possuem os mesmos valores para todos os atributos regulares, deve haver um identificador que os diferencie. Essa estratégia não somente auxilia na identificação de objetos presentes no sistema, mas também na distinção de objetos que possuem os mesmos valores para seus atributos em contextos diferentes, garantindo que toda mudança que seja feita em um valor seja associada ao objeto que possui um ID específico, impossibilitando uma atribuição ambígua de valor a uma instância de objeto de forma não-desejada.
- **URI Estrangeira em Atributos:** Uma URI é utilizada como uma ferramenta para identificar um recurso numa interface de uma aplicação. Um recurso é um intermédio pelo quais informações presentes no sistema podem ser modificadas e obtidas. Se uma URI estiver referenciada em um atributo anotado dentro de um objeto, essa URI deve ser associada diretamente a alguma informação presente na base de dados ou na interface relativa a ela. Essa verificação impossibilita que se façam conexões com informações que não estão presentes apropriadamente no sistema e impossibilitam referências sem lastro no contexto fechado localmente no qual o sistema está presente.
- **Criação de Classes Internas:** Uma classe interna só pode existir num sistema mediante a existência de uma classe forte que fundamente sua existência, ou seja, classes de objetos que derivam de uma classe forte previamente estabelecida não podem permanecer existindo caso haja remoção da classe forte. Dessa forma, em qualquer situação em que a existência de uma classe de objetos é diretamente relacionada a outra (a exemplo, a existência de uma classe “Voto” deve estar relacionada a existência de uma classe “Eleitor”), a referida restrição de integridade impossibilita que no processo de modificação do sistema haja objetos desconexos que não podem sequer ser acessados ou integrados ao resto da aplicação.
- **Aninhamento:** Quando uma chamada para a leitura das informações for realizada, todos os objetos subordinados ao objeto selecionado devem também ser retornados. Essa restrição de integridade é implementada para garantir que os dados completos do objeto sejam apresentados na busca dando um contexto

completo de sua situação e para impedir que buscas sucessivas para a obtenção da informação associada precisem ser realizadas.

É válido ressaltar o fato de que algumas dessas restrições listadas estão também associadas à perspectiva de uma descrição de uma Interface RESTful Semântica. De maneira que a forma de inserção, modificação e obtenção de informações deste contexto estão associadas a um conjunto particular de métodos (GET, POST, PUT, DELETE etc.).

Na descrição de restrições de integridade relacionadas a outros contextos, como o de bancos de dados não-relacionais, outra restrição de integridade que poderia ser considerada seria a obrigatoriedade da redundância da modificação de informações em diferentes instâncias de um mesmo objeto, visto que a unicidade de objetos não é uma necessidade absoluta num contexto não-relacional.

Entretanto, tal abordagem requer uma pesquisa mais aprofundada, que foge ao escopo do presente estudo. A descrição das restrições de integridade num contexto de mundo local fechado apresenta o desafio adicional da ausência de suporte extenso em linguagens de descrição semântica fundamentadas para apoiar a suas definições, no contexto de Interfaces RESTful Semânticas.

Enquanto abordagens para a extensão da linguagem OWL foram propostas para viabilizar a implementação da descrição de restrições de integridade em ontologias, a adoção dessas práticas ainda não aconteceu em larga escala e a ausência de engajamento por parte de desenvolvedores na utilização dessas tecnologias como forma de descrição de restrições de integridade ainda é problemática. (MOTIK, 2007) e (TAO, 2010)

Um exemplo da utilização de OWL para a descrição de restrições de integridade dentro do contexto de Interfaces RESTful Semânticas pode ser encontrado na seguinte implementação de código OWL proveniente do estudo de Lira (2015):

Figura 16 – Descrição em OWL de propriedades com restrições de integridade

```
<owl:DatatypeProperty
        rdf:about="#registryNumber">
<serin:id/>
<rdfs:domain rdf:resource="#Company"/>
<rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="#name">
<serin:notNull/>
<rdfs:domain rdf:resource="#Company"/>
<rdfs:range rdf:resource="&xsd;integer"/>
</owl:DatatypeProperty>

<owl:ObjectProperty
          rdf:about="#hasCompany">
<serin:foreignURI/>
<rdfs:domain
        rdf:resource="#SellProposal"/>
<rdfs:range rdf:resource="#Company"/>
</owl:ObjectProperty>
```

Fonte: LIRA (2023)

Na definição acima, o número de registro da empresa atua como um identificador único, garantindo a distinção de cada empresa das outras. Esse modelo reforça a noção de unicidade dos recursos identificadores, impedindo que duas empresas compartilhem o mesmo.

Além disso, a propriedade “nome”, ligada a classe “Empresa”, é marcada com uma anotação “*notNull*”, exigindo que todas as instâncias da “Empresa” tenham um valor para o “nome”, embora a singularidade não seja necessária. A propriedade “*hasCompany*” indica uma conexão entre “Empresa” e “Proposta de Venda”, com uma anotação de Interface RESTful Semântica, especificamente “*foreignURI*”, enfatizando a necessidade de um relacionamento explicitamente definido no banco de dados.

6.2 Restrições de Integridade em Serviços Web Semânticos

Restrições de integridade em serviços Web semânticos também são anotáveis. Apesar das diferenças entre as conceituações de Interfaces RESTful Semânticas e serviços Web semânticos, como ambos podem existir e serem definidos em um contexto de mundo localmente fechado, as definições de restrições de integridade para ambos apresentam semelhanças.

Ademais, como serviços Web semânticos são aplicações da Web que possuem recursos e semântica de dados anotados, as tecnologias que são utilizadas para a implementação das ontologias também são utilizadas para a definição de restrições de integridade dentro do contexto de serviços Web semânticos.

De acordo com sua documentação, a ferramenta SA-REST trata-se de “um *poshformat* para adicionar metadados adicionais a (mas não limitados a) descrições de APIs REST em HTML ou XHTML. Metadados de vários modelos, como uma ontologia, taxonomia ou uma nuvem de *tags*, podem ser incorporados nos documentos. Esses metadados incorporados permitem várias melhorias, como aprimorar a pesquisa, facilitar a mediação de dados e a integração mais fácil de serviços”. (GOMADAM, 2005)

Como *poshformats* são implementados utilizando semânticas de HTML, eles viabilizam incrementações semânticas dos serviços aos quais eles são incorporados e consequentemente possibilitam também a descrição de restrições de integridade. A seguir analisaremos o código encontrado no artigo de Gomadam (2005) que descreve uma anotação utilizando SA-REST em uma descrição de uma API RESTful em HTML.

Figura 17 – Descrição de API RESTful com anotações semânticas

```
<div class="section domain-rel" lang="en" title="sarest:Service" >
   <span class="domain-rel" title="sarest:Operation" >
   <div class="titlepage">
      <div>
         <div>
            <h3 id="JSON-RPCEndpoint">JSON-RPC Endpoint</h3>
         </div>
      </div>
   </div>
   <p>The JSON-RPC endpoint implements the <a class="ulink" href="http://json-rpc.org/wiki/specification" target="_top">JSON-RPC spec</a> on
   top of the Web service. Requests are serialized JavaScript following a specific data format.
   Each serialized JavaScript object contains the following properties:
   </p>

   <div class="itemizedlist domain-rel" title="sarest:InputMessage">
      <ul>
         <li class="bullist sem-class" title="sarest:Parameter">
         <code class="code">method</code>:
            name of the API method being called.
         </li>
         ...
      </ul>
   </div>
</span>
</div>
```

Fonte: GOMADAM (2010)

Caso o código apresentado na Figura 17 fosse modificado para anotar a representação de um endereço único associado a uma propriedade, efetivamente implementando a restrição de integridade de identificação única de atributos, o código poderia ser incrementado com a criação de um componente marcado com a notação “<div class=“section domain-rel” lang=“en” title=“sarest:URITemplate”>“, que de acordo com a documentação proposta, possibilita a anotação de URI’s, cumprindo papel análogo a anotação ”<serin : id>“ no contexto da descrição de serviços web semânticos.

6.3 Estudo de Caso: Restrições em OpenAPI e Restrições semânticas de integridade

O esquema do OpenAPI oferece suporte para definir algumas restrições de integridade de dados dentro de APIs. Por meio de recursos como palavras-chave para especificação de atributos como tamanho de campos e valores permitidos e opções de definições de formato, OpenAPI permite um controle preciso sobre os valores e estruturas de dados permitidos, reduzindo o risco de corrupção ou uso inadequado de dados.

Ao passo que, ao marcar campos como obrigatórios e definir estruturas de dados complexas, os desenvolvedores podem estabelecer dependências e relacionamentos entre elementos de dados, mantendo a integridade dos dados em todas as interações da API. O modelo da Figura 18 exemplifica a definição de um campo dentro de uma propriedade de corpo de retorno de uma requisição como obrigatório:

Figura 18 – Exemplo de definição de obrigatoriedade de propriedade em OpenAPI

Fonte: OpenAPI Specification v3.1.0 (2021)

O OpenAPI embora robusto na descrição de APIs, não possui a profundidade e abrangência encontradas em descrições semânticas de sistemas. Descrições semânticas que utilizam tecnologias como RDF e OWL, fornecem uma representação mais detalhada e abrangente de dados e relacionamentos dentro das APIs. Elas oferecem um contexto mais rico para entender os significados e comportamentos dos recursos, especialmente em sistemas complexos e interconectados.

Enquanto o OpenAPI se destaca na documentação básica e na interoperabilidade, as descrições semânticas oferecem uma abordagem mais precisa e flexível, essencial em cenários que requerem capacidades avançadas de semântica e raciocínio. Por essa razão, a capacidade de especificar restrições de integridade em OpenAPI também é limitada, demonstrando a vantagem na utilização de descrições semânticas, munidas da capacidade de descrições complexas e inferência, para a definição de restrições de integridade de dados em API's na Web.

7 CONCLUSÃO

Os resultados deste estudo permitiram elucidar uma visão ampla e contextualizada do conceito de restrições de integridade dentro do contexto da Web-Semântica. Ao analisar o escopo de tópicos que fundamentam o tema principal desta revisão de literatura, exemplos práticos de implementação e análises de estudos de caso, possibilitamos uma avaliação concisa do Estado da Arte e uma compreensão integral do contexto no qual se situam a interseção da Web-Semântica e de restrições de integridade.

Ao oferecer um panorama extenso dos conceitos de Restrições de Integridade no contexto de Interfaces RESTful Semânticas e Serviços Web-Semânticos, trabalhos futuros podem dar mais enfoque a aspectos que não tiveram uma avaliação minuciosa durante o processo de análise deste estudo ou a tópicos que não possuem abordagens notoriamente exploradas na literatura.

O número de aplicações que implementam princípios RESTful, por exemplo, só aumentam. É possível também que novas formas de descrever Interfaces RESTful Semânticas e Serviços Web-Semânticos surjam com o decorrer do tempo. Além disso, com o avanço das tecnologias de processamento de linguagem natural e aprendizado de máquina, modelos cada vez mais efetivos podem ser utilizados no processo de geração automática de descrições semânticas a partir de fontes de dados não semânticas, o que pode contribuir com a criação de novas abordagens de pesquisa.

Web API's e Serviços Web normalmente são analisados apenas como funções sem hierarquia, adotando a visão plana da programação estruturada. Com as interfaces semânticas, vinculadas a classes de ontologias, percebemos que os serviços Web passaram a ser agrupados como métodos agregados em torno de

objetos. Um passo futuro é o de investigar a especificação da herança e a consequente sobreposição de serviços Web, uma vez que tanto a herança como a sobreposição já se revelaram abstrações mais que úteis e importantes, sendo abstrações essenciais no contexto dos métodos na programação orientada a objetos.

Em relação as restrições de integridade dos bancos de dados impostas pelo comando CHECK, percebeu-se a ausência na especificação SERIN bem como na inclusão de instâncias, da especificação dos mecanismos, como o autoincremente, que na inserção de dados, permitem prescindir a especificação da chave primária. Trabalhos futuros podem contribuir com o estudo destes tópicos.

Em suma, esta monografia proporcionou uma base sólida para compreender o cenário das restrições de integridade no contexto da Web-Semântica, demonstrando a relevância e as oportunidades de pesquisa nesse campo. Embora tenha-se abordado uma considerável variedade de tópicos, há espaço para futuros estudos se concentrarem em áreas menos exploradas ou na adaptação das abordagens existentes às crescentes aplicações RESTful.

REFERÊNCIAS

LIRA, Hermano Albuquerque et al. **An approach to support data integrity for web services using semantic restful interfaces.** In: Proceedings of the 24th International Conference on World Wide Web. 2015. p. 1485-1490.

NEUMANN, Andy; LARANJEIRO, Nuno; BERNARDINO, Jorge. **An analysis of public REST web service APIs.** IEEE Transactions on Services Computing, v. 14, n. 4, p. 957-970, 2018.

Akkiraju, R., Farrell, J., Miller, J. A., Nagarajan, M., Sheth, A. P., & Verma, K. (2005). **Web service semantics-wsdl-s.**

ALOWISHEQ, A. **EXPRESS: EXPressing REstful Semantic Services.** In: ANAIS, 2009, Milan. IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology. Milan, 2009. p. 453 – 456.

ANKOLEKAR, Anupriya et al. **OWL-S'Relationship to Selected Other Technologies.** W3C Member Submission, 2004.

ARARAKI, F. A. **Linked data em bibliotecas: iniciativas e tendência.** In: ANAIS, Marília. Encontro Nacional de Pesquisa e Pós-graduação em Ciência da Informação. Marília. p. 1 – 17.

BATTLE, R. **Bridging the semantic Web and Web 2.0 with Representational State Transfer (REST).** Journal of Web Semantics, v. 6, n. 1, p. 61 – 69, Novembro 2007.

BATTLE, Steve et al. **Semantic web services ontology (swso).** Member submission, W3C, 2005.

BECHHOFER, Sean et al. **OWL web ontology language reference.** W3C recommendation, v. 10, n. 2, p. 1-53, 2004.

BIZER, Christian; HEATH, Tom; BERNERS-LEE, Tim. **Linked data.** International Journal on Semantic Web and Information Systems, v. 5, p. 33, 2009.

BERNERS-LEE, Tim. **The World Wide Web-past, present and future.** 1996.

DE BRUIJN, Jos et al. **Web service modeling ontology (wsmo).** Interface, v. 5, n. 1, p. 50, 2005.

CAMPOS NETO, J. C. Active OWL - **Uma arquitetura para integrar serviços Web e ontologias.** 2007. Dissertação (Mestrado em Ciência da Computação) — Universidade de Fortaleza (UNIFOR).

CECI, M. **An OWL ontology library representing judicial interpretations.** Semantic Web, v. 7, n. 3, p. 229 – 253, Março 2016

CHAVES, L. M. **DESCOBERTA DE SERVIÇOS WEB SEMÂNTICOS UTILIZANDO SERIN – SEMANTIC RESTFUL INTERFACES.** 2011. 90 p. Dissertação (Ciências da Computação) — UNIFOR.

CHÁVEZ, K**. A Systematic Literature Review on Composition of Microservices through the Use of Semantic Annotations: Solutions and Techniques.** In: ANAIS, 2019, Quito. International Conference on Information Systems and Computer Science (INCISCOS).

CHERON, A. **Design, maintenance and break-free evolution of REST APIs.** 2021. 169 p. Dissertação (Mathématiques et Sciences et Technologies de l'Information et de la Communication) — Universidade de Rennés.

CHINNICI, R. **Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language.** 2007.

CONNOLLY, D. **Gleaning Resource Descriptions from Dialects of Languages (GRDDL).** 2007.

DENECKER, M. **Towards a Logical Reconstruction of a Theory for Locally Closed Databases.** ACM Transactions on Database Systems, v. 35, n. 3, p. 1 – 45, Julho 2010.

DOHERTY, P. **Efficient Reasoning Using the Local Closed-World Assumption.** In: International Conference on Artificial Intelligence: Methodology, systems, and applications. Berlin: [s.n.], 2000. p. 49 – 58.

DUBLIN CORE. **DCMI Metadata Terms.** 2020.

FIELDING, R. **Architectural Styles and the Design of Network-based Software Architectures.** 2000. Dissertação (DOCTOR OF PHILOSOPHY in Information and Computer Science) — Universidade da Califórnia.

FU, C. **RESTful Web Services for Bridging Presence Service across Technologies and Domains: An Early Feasibility Prototype.** IEEE Communications Magazine, v. 48, n. 48, p.92 – 100, Dezembro 2010.

GARROTE, A. **A Formal Definition of RESTful Semantic Web Services.** In: ANAIS, 1., 2010, Raleigh. International Workshop on RESTful Design. Raleigh, 2010. p. 41 – 48.

GOMADAM , K.; RANABAHU , A.; SHETH , A. **SA-REST: Semantic Annotation of Web Resources.** Disponível em: <https://www.w3.org/submissions/SA-REST/>.

HARTIG, O. et al. RDF 1.2 **Concepts and Abstract Syntax.** Disponível em: <https://www.w3.org/TR/rdf12-concepts/>.

HAUG, A. **The costs of poor data quality.** Journal of Industrial Engineering and Management, Odense, v. 4, n. 2, p. 168 – 193, Agosto 2011. ISSN 2013-0953.

HERMAN, I. et al. **RDFa 1.1 Primer - Third Edition.** Disponível em: <https://www.w3.org/TR/rdfa-primer/>.

MICHELS, J.; ZEMKE, F.; BARTELS, J. **ISO/IEC 9075-11:2023 Information technology — Database languages SQL — Part 11: Information and definition schemas (SQL/Schemata).**

JATANA, N. **A Survey and Comparison of Relational and Non-Relational Database.** International Journal of Engineering Research & Technology (IJERT, v. 1, n. 6, p. 1 – 5, Agosto 2012. ISSN 2278-0181.

LAMOTHE, M. **A Systematic Review of API Evolution Literature.** ACM Computing Surveys, ACM, v. 54, n. 171, p. 1 – 36, Outubro 2021.

LIN, J. **Data Integrity Attacks Against Dynamic Route Guidance in Transportation-Based Cyber-Physical Systems: Modeling, Analysis, and Defense.** IEEE Transactions on Vehicular Technology, v. 67, n. 9, p. 8738 – 8753, Junho 2018.

LIRA, H. A. **An Approach to Support Data Integrity for Web Services Using Semantic RESTful Interfaces.** In: ANAIS, 2015, Florença. 24th World Wide Web Conference - WS Rest. Florença, 2015. p. 1485 – 1490.

MA, S. **Test-Oriented RESTful Service Discovery with Semantic Interface Compatibility.** EEE Transactions on Services Computing, v. 14, n. 5, p. 1571 – 1584, Outubro 2021.

MARJIT, U. **Publishing legacy data as linked data: a state-of-the-art survey.** Library Hi Tech, v. 31, n. 3, p. 520 – 535, Setembro 2013.

TAO, J. et al. **Extending OWL with Integrity Constraints.** CEUR Workshop Proceedings, [S.l.], v. 573, 2010.

MCILRAITH, S. A. **Semantic Web Services.** IEEE Intelligent Systems,, v. 16, n. 2, p. 46 – 53, Abril 2001.

Microformats Wiki. Disponível em: <https://microformats.org/wiki/Main_Page>. Acesso em: 17/09/2023.

MILLER, D. **OpenAPI Specification v3.1.0.** 2021. Disponível em: <https://spec:openapis:org/oas/latest:html#openapi-specification>. Acesso em: 09/07/2023.

MOORE, P. **On Context and the Open World Assumption.** In: ANAIS, 2015, Gwangju, Coréia do Sul. 2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops. Gwangju, Coréia do Sul, 2015. p. 387 – 392.

MOTIK, B. **Bridging the Gap Between OWL and Relational Databases.** In: PROCESSOS, 2007, Manchester. Décima Sexta Conferência Internacional sobre a World Wide Web. Manchester, 2007. p. 807 – 816.

MUHAMAD, W. **Designing Semantic Web Service Based on OAS 3.0 through Relational Database.** In: ANAIS, 2020, Bandung. International Conference on Information Technology Systems and Innovation. Bandung, 2020. p. 306 – 311.

NEUMANN, A. **An Analysis of Public REST Web Service APIs.** IEEE Transactions on Services Computing, IEE, v. 14, n. 4, p. 957 – 970, Agosto 2021.

ORACLE. **Data Integrity.** 2021. Disponível em: <https://docs:oracle:com/en/database/oracle/oracle-database/19/cncpt/data-integrity:html#GUID-6A89FF39-AD42-4399-BD1BE51ECEE50B4E>. Acesso em: 14/09/2023.

ORACLE. **Interfaces and Inheritance.** 2022. Disponível em: <https://docs:oracle:com/javase/tutorial/java/IandI/interfaceDef:html>. Acesso em: 17/09/2023.

ORACLE. **MySQL 8.0 Reference Manual.** Disponível em: <https://dev:mysql:com/doc/refman/8:0/en/>. Acesso em: 11/09/2023.

OWL WORKING GROUP. **Web Ontology Language (OWL).** 2012. Disponível em: <https://www:w3:org/2001/sw/wiki/OWL>. Acesso em: 10/09/2023.

PAUTASSO, C. **Advances in Service-Oriented and Cloud Computing: May Contain Nuts: The Case for API Labels.** Wittenberg: Springer, 2020. Disponível em: <http://dret:net/netdret/docs/wilde-wesoacs2018-api-labels:pdf>.

PAUTASSO, C. **Patterns on Designing API Endpoint Operations.** In: ANAIS, 2021, Nova Iorque. Conference on Pattern Languages of Programs. Nova Iorque, 2021. v. 21, p.1 – 29.

PRUD'HOMMEAUX, E. SPARQL **Query Language for RDF.** 2008. Disponível em: <https://www:w3:org/TR/rdf-sparql-query/#docConventions>. Acesso em: 08/09/2023.

RAUF, I. **Formal Verification of Stateful Services with REST APIs Using EventB.** In: ANAIS, 2018, São Francisco. IEEE International Conference on Web Services (ICWS). São Francisco, 2018. p. 131 – 138.

RAUF, I. **Towards Behaviorally Enriched Semantic RESTful Interfaces Using OWL2.** In: RESUMO, 2015, Rotterdam. International Conference on Web Engineering. Rotterdam, 2015. p. 407 – 410.

SALVADORI, I. **A Maturity Model for Semantic RESTful Web APIs.** In: ANAIS, 2015, Nova Iorque. IEEE International Conference on Web Services (ICWS). Nova Iorque, 2015. p. 88 – 93.

SALVADORI, I. **OntoGenesis: an architecture for automatic semantic enhancement of data services.** International Journal of Web Information Systems, v. 15, n. 1, p. 2 – 27, Dezembro 2018.

SALVADORI, I. L. **DATA LINKING AS A SERVICE: A MICROSERVICES INFRASTRUCTURE FOR PUBLISHING LINKED DATA.** 2019a. Dissertação (Programa de PósGraduação em Ciência da Computação) — Universidade Federal de Santa Catarina.

SALVADORI, I. L. **Semantic Data-Driven Microservices.** In: ANAIS, 2019, Milwaukee. 2019 IEEE 43rd Annual Computer Software and Applications Conference. Milwaukee, 2019b. p. 402 – 410.

SCHREIER, S. Modeling RESTful applications. In: ANAIS, 2011, Nova Iorque. Second International Workshop on RESTful Design. Nova Iorque, 2011. p. 15 – 21.

SEGUNDO, J. E. S. **WEB SEMÂNTICA: INTRODUÇÃO A RECUPERAÇÃO DE DADOS USANDO SPARQL.** In: XV Encontro Nacional de Pesquisas em Ciência da Informação (ENANCIB). Belo Horizonte: [s.n.]. p. 3863 – 3882.

SERBOUT, S., PAUTASSO, C. e ZDUN, U. **How Composable is the Web? An Empirical Study on OpenAPI Data model Compatibility.** In: ANAIS, 2022, Barcelona. 2022 IEEE International Conference on Web Services (ICWS). Barcelona, 2022. p. 415 – 424.

SLIMANI, T. **Semantic Description of Web Services.** International Journal of Computer Science Issues, v. 10, n. 3, p. 1 – 10, Janeiro 2013.

TANDIONO, E. K. **Transformation of REST Service Description for Structured Service Registry.** In: ANAIS, 2019, Pontianak. International Conference on Data and Software Engineering (ICoDSE). Pontianak, 2019. p. 1 – 6.

TAO, J. **Integrity Constraints in OWL.** In: ANAIS, 2010, Atlanta. AAAI Conference on Artificial Intelligence. Atlanta, 2010a. p. 1443 – 1448.

VITVAR, T. **Linked Web APIs Dataset: Web APIs meet Linked Data.** 2016. Semantic Web Journal.

W3C. **Extensible Markup Language (XML) 1.0 (Fifth Edition).** 2008a. Disponível em: <https://www:w3:org/TR/2008/REC-xml-20081126/#sec-intro>. Acesso em: 15/09/2023.

W3C. **SKOS Simple Knowledge Organization System RDF Schema.** 2008b. Disponível em: <https://www:w3:org/TR/2008/WD-skos-reference-20080829/skos:html>. Acesso em: 15/09/2023.

W3C. **Semantic Annotations for WSDL and XML Schema — Usage Guide.** 2007. Disponível em: <https://www:w3:org/TR/sawsdl-guide/#lifting>. Acesso em: 15/09/2023.

W3C. **Simple Object Access Protocol (SOAP) 1.1.** 2000. Disponível em: <https://www:w3:org/TR/2000/NOTE-SOAP-20000508/#_Toc478383486>. Acesso em: 14/09/2023.

WHAT is OpenAPI? Disponível em: https://www:openapis:org/what-is-openapi. Acesso em: 09/07/2023.

WILDE, E. **Surfing the API Web: Web Concepts.** In: ANAIS, 2018, Lyon. World Wide Web Conference. Lyon, 2018. p. 797 – 802. Disponível em: <https://dl:acm:org/doi/pdf/10:1145/3184558:3188743>.