Archives available at journals.mriindia.com

Multidisciplinary Journal of Research in Engineering and Technology

ISSN: 2348-6953

Volume 13 Issue 01, 2026

Ontology-Based Web Services for RESTful Application in Aviation

[1]Praveenkumar A. Patel, [2]Harshada Chougale, [3]Vishal Chavan, [4]Omkar Pawar, [5]Mayuri Satpute, [6]Shivraj Patil

[1]*Assistant Professor, Bharati Vidyapeeth Collage of Engineering, Kolhapur Maharashtra India*

[2,3,4,5,6] *Students, Bharati Vidyapeeth Collage of Engineering, Kolhapur Maharashtra India*

Peer Review Information

Submission: 12 April 2026

Revision: 02 May 2026

Acceptance: 23 May 2026

Keywords

Semantic Web, RESTful APIs, Service Registry, OWL-S, SPARQL, Ontology-Driven Discovery, Quality of Service (QoS), Web Service Invocation, Service Orchestration.

Abstract

The number of REST APIs has grown very fast, which makes it hard for developers to find, connect, and use the right services. Most current systems rely on keyword searches, but these often fail to capture important details about how a service works or how well it performs. This leads to slow and error-prone integration. To solve this problem, this project introduces the Semantic RESTful Service Registry (SRSR). It combines REST APIs with Semantic Web technologies to make services easier to understand for ma chines. Using tools like OWL-S and a custom REST-based ontology, each service is described in a structured and meaningful way, including its functions and performance factors such as availability, response time, and reliability. The system uses RDFLib to manage data and SPARQL to perform intelligent searches based on properties rather than just keywords. It also includes a smart ranking system that selects the best service based on quality factors. Additionally, SRSR can automatically invoke services by matching user inputs with the correct API requests. The results show that this approach improves service discovery accuracy and supports automation. A graph-based visualization feature also helps developers easily understand service relationships and networks.

Introduction

In today's digital world, modernapplications are no longer built as isolated systems. Instead, they rely heavily on integrating multiple external services through Application Programming Interfaces (APIs). For example, a travel booking application may simultaneously interact with airline systems, payment gateways, hotel services, and weather APIs. This interconnected ecosystem enables powerful functionality, but it also introduces significant complexity. As the number of publicly available RESTful APIs has grown into the millions, developers face increasing challenges in discovering, understanding, and integrating the right services. Traditional API discovery methods are largely based on keyword searches and manual documentation review. While these approaches may help locate APIs with relevant names, they fail to capture deeper details such as functionality, input-output compatibility, and performance characteristics. As a result, developers often spend considerable time analyzing documentation, testing endpoints, and resolving integration issues, making the process inefficient and error-prone.

To address these limitations, this project introduces the Semantic RESTful Service Registry (SRSR), a smart framework designed to enhance how APIs are described, discovered, and utilized. The core idea behind SRSR is to move beyond simple keyword-based listings and instead provide machine-readable descriptions that capture the meaning and behavior of services. By incorporating Semantic Web technologies, APIs are enriched with

structured metadata that clearly defines their inputs, outputs, and operational characteristics. At the heart of this approach lies the concept of semantics, which focuses on meaning rather than just textual representation. Using ontology-based modeling, shared concepts such as “temperature,” “location,” or “currency conversion” are defined in a standardized way. When APIs are registered in the system, they are linked to these concepts, ensuring consistency and interoperability across different services. This allows computers, not just humans, to understand and process API functionalities effectively.

The SRSR framework also introduces an advanced discovery mechanism powered by semantic querying. Instead of relying on basic keyword searches, users can perform intelligent queries based on desired inputs, outputs, and service behavior. This significantly improves the accuracy of service discovery and reduces the time required to find suitable APIs. Another important feature of the system is its Quality of Service (QoS)– aware ranking mechanism. Not all APIs perform equally— some may be faster, more reliable, or more available than others. The registry evaluates these factors and ranks services accordingly, enabling developers to choose the most efficient and dependable options for their applications. In addition, SRSR supports dynamic service invocation, allowing users to test and execute APIs directly within the platform. A graph- based visualization component further enhances usability by providing a clear representation of relationships between services, providers, and categories. In conclusion, the Semantic RESTful Service Registry represents a significant step toward building a smarter and more organized web ecosystem. By enabling meaningful, machine-readable API descriptions and intelligent discovery, it reduces development effort, improves integration efficiency, and lays the foundation for automated service orchestration in future applications**.**

Key Contribution of Research

This research presents several important contributions toward improving API discovery, integration, and automation using Semantic Web technologies:

- **Development of Semantic RESTful Service Registry (SRSR):**

A novel framework that integrates RESTful APIs with Semantic Web concepts, enabling machine-readable service descriptions instead of traditional keyword-based listings.

- **Ontology-Based API Modelling:**

Introduction of a customized ontology that extends standard semantic models (such as OWL-S) to represent RESTful services, including their inputs, outputs, and operational behaviour in a structured and meaningful way.

- **Enhanced Semantic Service Discovery:**

Implementation of an intelligent discovery mechanism using SPARQL queries, allowing users to search APIs based on functional requirements rather than simple keywords, improving accuracy and relevance.

- **Quality of Service (QoS)–Aware Ranking Mechanism:**

Design of a multi-criteria scoring system that evaluates APIs based on performance attributes such as availability, response time, and reliability, helping users select the best service.

- **Automated Service Invocation:**

A dynamic execution module that maps semantic descriptions to actual HTTP requests, enabling automatic API invocation without manual configuration.

- **Integration of RDFLib and Knowledge Graph Representation:**

Efficient management of semantic data using RDFLib, along with a graph-based visualization layer that helps users understand relationshipsbetween services, providers, and categories.

- **Improved Discovery Accuracy and Reduced Integration Effort:**

Experimental validation demonstrating that the proposed approach significantly reduces manual effort and improves the precision of API discovery compared to conventional methods.

- **Foundation for Autonomous Service Orchestration:**

Establishes a basis for future systems where applications can automatically discover, select, and integrate services without human intervention.

Literature Review

[1] D. Martin et al., "Bringing semantics to web services with OWL-S," *World Wide Web*, vol. 10, no. 3, pp. 243–277, Sep. 2007. This foundational paper introduces OWL-S, a set of ontologies for describing service capabilities (Profile), internal logic (Process), and interaction details (Grounding). It is a primary reference for the project’s use of OWL-S as the underlying semantic model for service registration. The authors argue that formal semantics are essential for moving from manual service integration to automated discovery and invocation.

[2] E. Al-Masri and Q. Mahmoud, "QoS-based discovery and ranking of web services," in *Proc. 16th Int. Conf. World Wide Web (WWW)*, 2007, pp. 1279–1280. This research addressesthe

challenge of picking the "best" service from a list of functionally identical candidates. It proposes a ranking mechanism based on Quality of Service (QoS), such as response time and availability. This directly informs the QoS- aware ranking algorithm implemented in this project's Discovery Service.

[3] L. D. Ngan and K. Rajaraman, "Semantic Web service discovery: State-of-the-art and research challenges," *Pers. Ubiquitous Comput.*, vol. 17, no. 1, pp. 1–19, 2013. This comprehensive survey reviews the evolution of semantic web service discovery. It highlights the shift from keyword-based UDDI registries to modern ontology-driven approaches. The paper identifies key challenges in scalability and precision that the Semantic API Registry aims to address through SPARQL- based querying.

[4] A. Sheth, K. Gomadam, and J. Lathem, "SA-REST: Semantically Annotated RESTful Services," *IEEE Internet Comput.*, vol. 11, no. 4, pp. 58–61, Jul. 2007. SA-REST introduces a method for adding semantic annotations to RESTful services using RDFa. It bridges the gap between traditional SOAP-based semantic services and the modern REST architectural style. This project builds upon the principles of SA-REST by allowing users to map REST endpoints to specific semantic concepts.

[5] S. Ben Mokhtar et al., "EASY: Efficient semAntic Service discoverY in pervasive computing environments with QoS and context support," *J. Syst. Softw.*, vol. 81, no. 5, pp. 693–708, May 2008. The EASY framework focuses on efficient discovery in dynamic environments. It combines functional matching with QoS and context-aware filtering. The project adopts a similar multi-layered approach to matching, where initial semantic filters are followed by composite QoS scoring.

[6] M. Maleshkova, M. Kopecký, and J. Kosek, "hRESTS: An HTML Microformat for Describing RESTful Web Services," in *Proc. 2008 IEEE/WIC/ACM Int. Conf. Web Intel.*, 2008, pp. 119–125. This paper proposes hRESTS, a lightweight microformat for marking up RESTful service documentation. It is relevant to this project's "Registration" phase, where structured HTML or JSON descriptions (like OpenAPI) are transformed into machine-readable RDF triples.

[7] T. Vitvar et al., "WSMO-Lite: Lightweight Semantic Annotations for Services on the Web," *IEEE Internet Comput.*, vol. 12, no. 2, pp. 11–17, Mar. 2008. WSMO-Lite presents a lightweight alternative to the complex WSMO framework. It advocates for simple semantic annotations that focus on service classifications and I/O types. This "lightweight" philosophy is reflected in the project's user interface, which simplifies the registration of semantic metadata.

[8] J. Kuster et al., "SPARQL-based Matchmaking of Semantic Web Services," in *Proc. 4th Int. Conf. Semantic Web and Digit. Libraries (ICSD)*, 2010. This paper details how SPARQL can be used as a primary engine for service matchmaking. By storing service descriptions in an RDF triple store, complex discovery queries can be executed with standard tools. This approach is the core logic behind the SPARQL Engine module in this project.

[9] J. R. V. Dantas et al., "Semantic Web services discovery adopting SERIN," in *Proc. 2015 IEEE Int. Conf. Comput. Intel. Commun. Netw. (CICN)*, 2015, pp. 586–590. The SERIN framework provides a modern implementation path for RESTful semantic discovery. It emphasizes the use of shared vocabularies to improve interoperability between different service providers, a key goal of this project's ontology management system.

[10] A. Verborgh et al., "Semantic Describing RESTful Services through Linked Data," in *Proc. 5th Workshop on Linked Data on the Web (LDOW)*, 2012. This work exploresthe integration of RESTful services with the principles of Linked Data. It argues that by treating APIs as linked resources, discovery engines can navigate the "Web of Services" automatically. The knowledge graph visualization in this project is inspired by this linked data approach.

Table 1: Literature review table

Ref. No.	Authors & Year	Title	Key Contribution	Relevance to Project	Limitations
[1]	D. Martin et al., 2007	Bringing Semantics to Web Services with OWL-S	Defines OWL-S ontology for semantic service description	Core semantic foundation used in project	Complex to implement; mainly designed for SOAP-based services, not REST
[2]	E. Al-Masri & Q. Mahmoud, 2007	QoS-based Discovery and Ranking	Introduces QoS-based ranking using metrics like response time	Basis for QoS ranking in discovery module	Focuses only on QoS; ignores semantic matching of services

[3]	L. D. Ngan & K. Rajaraman, 2013	Semantic Web Service Discovery Survey	Reviews evolution of semantic discovery approaches	Identifies gaps addressed by this project	Does not provide implementation or practical solution
[4]	A. Sheth et al., 2007	SA-REST	Adds semantic annotations to REST services using RDFa	Inspires REST semantic annotation approach	Limited tooling support and adoption; manual annotation required
[5]	S. Ben Mokhtar et al., 2008	EASY Framework	Combines semantic, QoS, and context-aware discovery	Influences multi-layer discovery strategy	High computational complexity; less scalable for large datasets
[6]	M. Maleshkova et al., 2008	hRESTS Microformat	Provides lightweight method to describe REST APIs	Helps in structuring API descriptions	Limited expressiveness; not fully semantic like OWL/RDF
[7]	T. Vitvar et al., 2008	WSMO-Lite	Lightweight semantic annotation model	Inspires simplified user interface for annotation	Less detailed compared to full semantic models like OWL-S
[8]	J. Kuster et al., 2010	SPARQL-based Matchmaking	Uses SPARQL for semantic service discovery	Core logic for SPARQL query engine	Requires well-structured RDF data; performance issues at scale
[9]	J. R. V. Dantas et al., 2015	SERIN Framework	Focuses on interoperability using shared vocabularies	Supports ontology-based interoperability	Limited real-world adoption; lacks advanced QoS integration
[10]	A. Verborgh et al., 2012	Linked Data for REST Services	Treats APIs as linked data resources	Inspires knowledge graph visualization	Complexity in implementation; requires strong linked-data infrastructure

or booking systems. But even though APIs are widely used, finding and using the right one is still difficult and time-consuming. The main problem is that current API systems only describe technical details like URLs and data formats, but they do not explain the actual meaning of what the API does. Because of this, searching for APIs using keywords often gives confusing results, and developers have to manually read long documentation to understand how each API works. This process is slow, error-prone, and not practical for large-scale systems. Also, computers cannot automatically find and use APIs because there are no proper semantic descriptions. Another issue is that developers cannot easily compare APIs based on performance factors like speed, reliability, or availability. On top of that, relationships between different services are not clearly visible, making the system harder to understand. This project aims to solve these problems by creating a Semantic RESTful Service Registry, which adds meaning to APIs, improves search accuracy, enables automatic usage, and provides a simple, efficient way to discover and use the best services.

Research Gap

Even though Semantic Web Services have been studied for many years, there are still some important problems that arenot solved. Most old semantic systems like OWL-S and WSMO were made for SOAP-based services, but today almost all APIs use REST. So, these old models don't fit well with modern systems. Another issue is that many semantic approaches are too complex. Developers need to understand RDF, OWL, and other advanced concepts, which makes it hard to use in real projects. There is a gap between research ideas and what developers can easily apply in practice. Also, current systems usually treat API discovery, ranking, and usage as separate tasks. But in real life, developers want everything in one place—find the API, check its performance, and use it directly. There is also a lack of tools to clearly show how different APIs are connected. Most systems don't provide visual understanding, making it harder to

explore services. Finally, many existing methods are slow because they use heavy processing techniques. They are not suitable for real-time use. This project solves these problems by creating a simple, fast, and integrated system that works well with modern REST APIs and is easy for developers to use.

Problem Statement

In today's world, most applications depend on REST APIs to connect with different services like payments, weather,

Proposed System

The proposed system, called the Semantic RESTful Service Registry (SRSR), is designed to make API discovery, selection, and usage faster and smarter by adding meaning (semantics) to REST APIs. It follows a three-layer architecture. The Presentation Layer provides a user-friendly web dashboard built with Flask, where users can register APIs, search for services, visualize connections, and test APIs. The Service Layer acts as the core logic of the system and includes components like the Ontology Manager, which handles semantic data, the Conversion Engine, which converts OpenAPI files into RDF format, the Discovery and Ranking Unit, which finds APIs using SPARQL queries and ranks them based on Quality of Service (QoS), and the Invocation Engine, which automatically executes API requests. The Data Layer stores all semantic information in an RDF triple store using Turtle files. The system also connects to external REST APIs to fetch real-time responses. Overall, this architecture creates a complete and intelligent platform where users can register, discover, evaluate, and use APIs efficiently in one place.

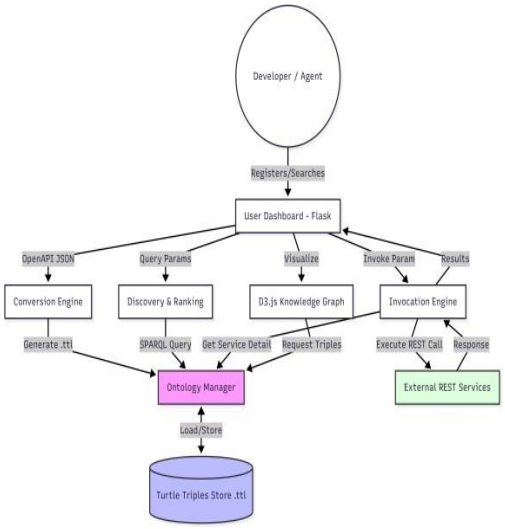

Figure 1: Proposed system architecture

Experimental result and discussion

Table 2: Service Model Registry

Service Name	Provider (Actor)	Category	Endpoint Path	Is Free?
OpenAI Chat API	OpenAI	AI	/v1/chat/completions	No
Exchange Rate API	ExchangeRate-API	Finance	/v6/latest/	Yes
IP Geolocation API	IP Geolocation	Utilities	/ipgeo	Yes
NewsAPI	NewsAPI	News	/v2/everything	No
OpenWeatherMap	OpenWeather	Weather	/data/2.5/weather	Yes
REST Countries	REST Countries	Reference	/v3.1/all	Yes

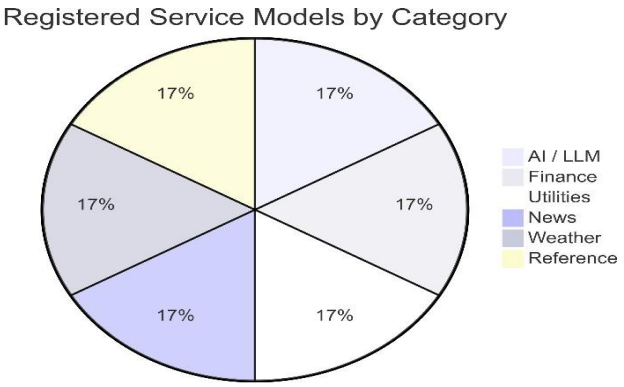

Figure 2: Registered Services Model by Category

This figure 2 shows the distribution of registered services across different categories in the system. Each category— AI/LLM, Finance, Utilities, News, Weather, and Reference— occupies an equal portion of 17% in the pie chart. This indicates that the system maintains a balanced collection of services without giving priority to any single category. Such equal distribution ensures that users can access a wide variety of APIs from different domains. It also improves the efficiency of service discovery by providing diverse options and making the registry more organized and user-friendly.

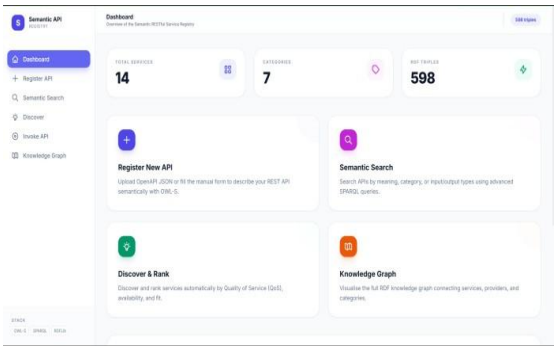

Figure 3: Semantic RESTful Service Registry

This figure 3 shows the main dashboard of the Semantic RESTful Service Registry system. It gives a quick overview of the platform, including total services (14), categories (7), and total triples (598), which represent stored semantic data. The left sidebar provides easy navigation options like Dashboard, Register API, Semantic Search, Discover, Invoke API, and Knowledge Graph. In the center, key features are shown as cards: users can register new APIs, perform semantic searches using SPARQL, discover and rank services based onquality, and view relationships using the knowledge graph. Overall, this dashboard makes the system simple, organized, and easy to use.

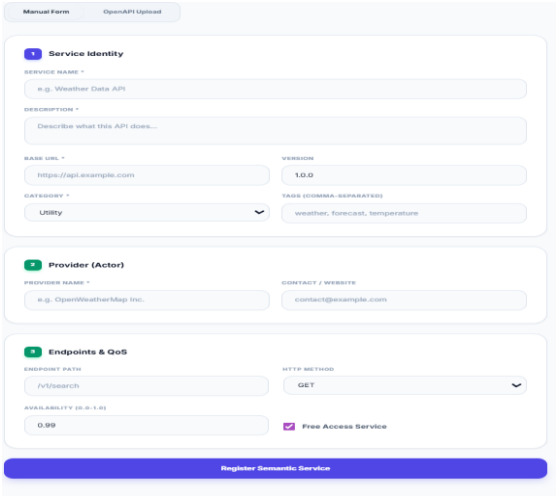

Figure 4: API registration form

This figure 4 shows the API registration form in the Semantic RESTful Service Registry system. It is used to add new APIs with proper semantic details. The form is divided into three sections. The first section, Service Identity, collects basic information like service name, description, base URL, version, category, and tags. The second section, Provider (Actor), includes details about the API provider such as name and contact information. The third section, Endpoints & QoS, captures technical details like endpoint path, HTTP method, availability, and whether the service is free. This structured form helps in accurate registration and better service discovery.

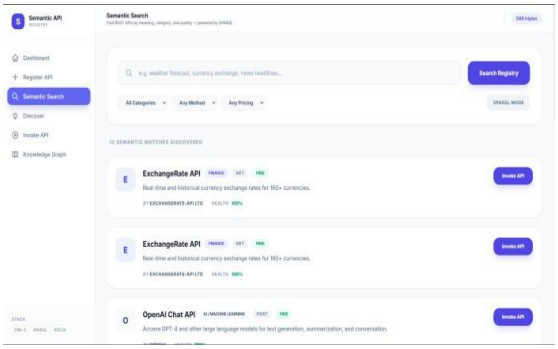

Figure 5: Semantic Search page

This figure shows the Semantic Search page of the system. It allows users to find APIs based on meaning rather than just keywords. At the top, there is a search bar where users can enter queries like weather, currency exchange, or news. Filters such as category, method, and pricing help refine the search results. The system uses SPARQL-based semantic queries to provide accurate matches. Below, the results display relevant APIs like ExchangeRate API and OpenAI Chat API, along with short descriptions and options to invoke them. This feature improves search accuracy and helps users quickly find suitable services.

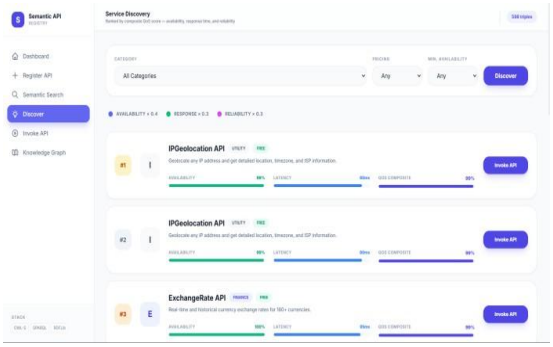

Figure 6: Service Discovery and Ranking page

This figure 6 shows the Service Discovery and Ranking page of the system. It helps users find the best APIs based on performance factors. At

the top, filters like category, pricing, and availability allow users to refine results. The system ranks APIs using Quality of Service (QoS) metrics such as availability, response time (latency), and reliability. Each API card displays these metrics with visual bars, making comparison easy. For example, APIs like IPGeolocation and ExchangeRate are shown with their performance scores. Users can directly invoke APIs from this page. This feature helps users quickly select the most efficient and reliable service.

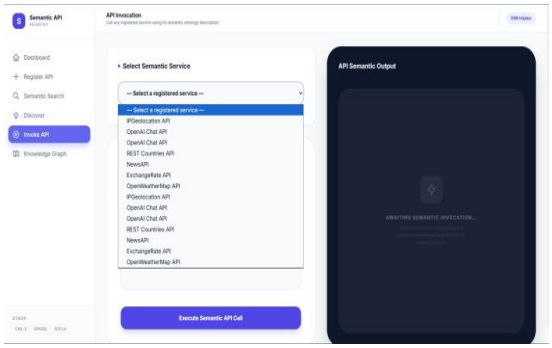

Figure 7: API Invocation page

This figure 7 shows the API Invocation page of the system. It allows users to execute APIs directly using their semantic descriptions. On the left side, users can select a registered service from a dropdown list, such as IPGeolocation API, OpenAI Chat API, or ExchangeRate API. After selecting a service, users can click the "Execute Semantic API Call" button to run it. On the right side, the output panel displays the response of the API after execution. This feature removes the need for manual API testing and makes it easier to use services directly within the system.

Figure 8: The Knowledge Graph visualization

This figure 8 shows the Knowledge Graph visualization of the system. It displays how different services, providers, categories, and endpoints are connected to each other. Each node in the graph represents an element like an API, provider, or category, and the lines (edges) show their relationships. For example, an API is linked to its provider and its category, making the structure easy to understand. The graph helps users visually explore how services are organized and related. This improves understanding of the system and makes it easier to analyze connections between different APIs.

Conclusion

The Semantic RESTful Service Registry (SRSR) solves the common problems developers face when finding and using APIs. It combines REST APIs with Semantic Web technologies like OWL-S and SPARQL to make APIs smarter and easier to understand. Instead of just searching by keywords, this system helps users find APIs based on what they actually do. It also ranks APIs based on performance factors like speed and reliability, helping developers choose the best option. Features like automatic API execution and visual graphs make the system easy to use and understand. Overall, this project makes API usage faster, more accurate, and less manual. It also creates a strong base for future systems where software can automatically find and use APIs without human effort, leading to a smarter and more connected web.

References

D. Martin et al., "Bringing Semantics to Web Services with OWL-S," *World Wide Web*, vol. 10, no. 3, pp. 243–277, Sep. 2007. DOI: 10.1007/s11280-007-0033-x

E. Al-Masri and Q. H. Mahmoud, "QoS-Based Discovery and Ranking of Web Services," in *Proc. 16th Int. Conf. World Wide Web (WWW)*, 2007, pp. 1279–1280.

L. D. Ngan and K. Rajaraman, "Semantic Web Service Discovery: State-of-the-Art and Research Challenges," *Personal and Ubiquitous Computing*, vol. 17, no. 1, pp. 1–19, 2013. DOI: 10.1007/s00779-012-0609-z

A. Sheth, K. Gomadam, and J. Lathem, "SA-REST: Semantically Annotated RESTful Services," *IEEE Internet Computing*, vol. 11, no. 4, pp. 58–61, Jul. 2007.

S. Ben Mokhtar et al., "EASY: Efficient semAntic Service discoverY in Pervasive Computing Environments with QoS and Context Support," *Journal of Systems and Software*, vol. 81, no. 5, pp. 693–708, May 2008. DOI: 10.1016/j.jss.2007.07.030

M. Maleshkova, M. Kopecký, and J. Kosek, "hRESTS: An HTML Microformat for Describing RESTful Web Services," in *Proc. IEEE/WIC/ACM*

Int. Conf. Web Intelligence, 2008, pp. 119–125. DOI: 10.1109/WIIAT.2008.379

T. Vitvar et al., "WSMO-Lite: Lightweight Semantic Annotations for Services on the Web," *IEEE Internet Computing*, vol. 12, no. 2, pp. 11–17, Mar. 2008.

J. Kuster et al., "SPARQL-Based Matchmaking of Semantic Web Services," in *Proc. 4th Int. Conf. Semantic Web and Digital Libraries (ICSD)*, 2010.

J. R. V. Dantas et al., "Semantic Web Services Discovery Adopting SERIN," in *Proc. IEEE Int. Conf. Computational Intelligence and Communication Networks (CICN)*, 2015, pp. 586–590.

A. Verborgh et al., "Semantic Describing RESTful Services through Linked Data," in *Proc. 5th Workshop on Linked Data on the Web (LDOW)*, 2012.