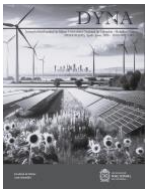
DYNA
http://dyna.medellin.unal.edu.co/



UNIVERSIDAD NACIONAL DE COLOMBIA

A catalog of API Gateway metrics and its quantitative evaluation

Eder dos Santos & Sandra Casas

GISP. Instituto de Tecnología Aplicada, Unidad Académica Río Gallegos, Universidad Nacional de la Patagonia Austral, Río Gallegos, Argentina.
esantos@uarg.unpa.edu.ar, sicasas@uarg.unpa.edu.ar

Received: February 7th, 2025. Received in revised form: April 28th, 2025. Accepted: May 21st, 2025.

Abstract
In the rapidly evolving API Economy, quality of API management software has become a critical concern. This work analyzed the API management industry with a particular focus on the metrics available for various API Gateway products. The primary artifact is a catalog with 59 metrics, compiled from global industry reports and technical documentation of 68 leading API Gateway products. The Design Science Research (DSR) approach was adopted to design the catalog, and metrics were categorized based on clear definitions in recent scientific literature. Secondly, a quantitative analysis method was proposed and systematically performed. Findings indicate that the metrics mainly focus on latency, response time, API performance, error capturing, and traffic monitoring. Features such as caching, resource utilization, and system health were scarcely addressed by the examined products. The proposed artifacts provide an objective foundation for a deeper understanding of API Management software quality, and lay the groundwork for future research.

Keywords: API management; API gateway; design science research; software engineering; software metrics; software quality.

Un catálogo de métricas de API Gateway y su evaluación cuantitativa

Resumen
En la Economía de las API, la calidad del software de administración de API se ha convertido en una preocupación crítica. Este trabajo analizó productos de administración de API con un enfoque particular en las métricas disponibles en productos API Gateway. Se obtuvo un catálogo de 59 métricas, compiladas y categorizadas a partir de la literatura vigente, reportes globales y documentación técnica de 68 API Gateways. Se adoptó el enfoque Design Science Research (DSR) para el diseño. Adicionalmente, se propuso y se implementó un método de análisis cuantitativo. Los hallazgos indican que las métricas se centran principalmente en latencia, tiempo de respuesta, rendimiento, captura de errores y monitoreo del tráfico. Características como almacenamiento en caché, utilización de recursos y estado del sistema fueron escasamente abordadas. Los artefactos propuestos proporcionan una base objetiva para comprender más profundamente la calidad del software de administración de API, y sientan bases para futuras investigaciones.

Palabras clave: administración de API; API Gateway; Design Science Research; ingeniería de software; métricas de software; calidad de software.

1. Introduction

In recent years, the distribution models for information systems have been moving towards Everything-as-a-Service (XaaS) [1] paradigms. These paradigms are based primarily on microservice architectures that provide a flexible framework [2], enabling organizations to efficiently distribute their information systems into a highly scalable set of services. Within such a framework, APIs are essential in streamlining the development process and fostering interoperability and collaboration between multiple software applications and platforms [3,4]. As a result, effective quality management over API Management products has emerged as a critical aspect for organizations striving to harness the full potential of their digital assets and remain competitive in fast-paced market environments. The global API Management market grew by 13. 7\% and reached 3.3 billion USD in 2023[1].

At the core of API Management products lies the API Gateway. The API Gateway acts as a centrally managed, API-oriented control service that encapsulates, exposes, secures, and manages back-end data and services as RESTful APIs, which provides a framework

How to cite: Santos E., and Casas, S., A catalog of API gateway metrics and Its quantitative evaluation. DYNA, 92(237), pp. 106-114, April - June, 2025.
[1] Source: https://www.gartner.com/en/documents/5594559

for establishing a facade on top of these services. API Gateway's main features may include security, traffic management, service and data mapping, caching, load balancing, error reporting, and performance monitoring, among others. API Gateways provide a broad set of metrics that enable the production of business-value reports and user-friendly dashboards aiming at helping to understand who uses available APIs and how they are utilized [3].

Current industry reports[2] indicate that there are 68 API Gateway offerings available nowadays. In such a competitive landscape, it is essential to understand how quality is measured and what metrics these products provide to achieve it. However, research in this domain is still emerging. A systematic mapping study conducted in [5] indicates that only 20 studies addressed a handful of quality characteristics among a small selection of available API Management software products.

The main aim of this work is to develop the third step towards an expressive, platform-neutral API Gateway quality model that can be used to create open tools for API Management quality characterization, measurement, and evaluation. The main contributions of this paper include a) a catalog of 59 metrics collected from eight industry-leading API Gateway products; b) the specification of a flexible, domain-neutral quantitative analysis method based on scoring and comparison, adapted from [6]; c) an analysis based on the described method, which highlighted trends and gaps in the metrics provided by the examined products; and iv) an open dataset that allows to replicate and improve this research.

This paper is structured as follows: Section 2 introduces the methodologies employed in designing the catalog artifact and its quantitative analysis method. Section 3 presents the API Gateway products included in this study, as well as the catalog of metrics and its classification. Next, Section 4 discusses the empirical evaluation of the catalog, and additional insights are provided in Section 5. Finally, concluding remarks and insights into future research are encompassed in Section 6.

2. Methods

The study presented herein was conducted during August and October 2024 and represents a primary investigation in which we analyzed 68 API Gateway products. This work followed the DSR methodology [7], a systematic approach to building and evaluating artifacts such as models, frameworks, and software tools. It emphasizes the iterative cycle of designing, implementing, and refining these artifacts while rigorously assessing their effectiveness and utility in real-world settings. The DSR activity phases conducted in this study are summarized as follows:

1. Problem Identification and Motivation: The lack of a tailored quality model for API Gateway in-depth evaluation.
2. Objectives of a Solution: To design a comprehensive and integrated framework driven by metrics to measure, assess and improve API Gateway software quality.
3. Design and Development: To conduct an extensive review of 68 API Gateway products. Build a catalog of metrics.
4. Demonstration: To present the catalog of metrics; to perform a descriptive analysis over a set of eight real-world API Gateways.
5. Evaluation: Interpretation by employing a quantitative analysis model.
6. Communication: To elaborate and publish this paper and a public dataset.

2.1 DSR development

2.1.1 Research problem identification and motivation

Despite the proliferation of software quality models in the literature, there is a notable gap regarding API management software quality models and metamodels [5]. Additionally, existing research primarily focuses on benchmarking a limited set of quality characteristics across a few API Management platforms. This void motivated the development of the present work.

2.1.2 Objectives of a solution

This work proposes designing and developing a catalog of API Gateway metrics as a comprehensive, platform-neutral, flexible, scalable, and integrated framework that facilitates the characterization and measurement of software quality.

2.1.3 Design and development of the artifact

This study aimed to identify, characterize, and classify metrics provided by industry-leading API Gateway products. To achieve this, we conducted an iterative six-step bottom-up process, which consisted of the following activities:

D1. Initial metrics collection:

During this activity, several approaches were reviewed from the current literature. We prioritized works that (i) provided clear and operational definitions of metrics, (ii) were widely cited or recognized as foundational in the domain, and (iii) offered complementary perspectives or categorization schemes. Based on these criteria, definitions from [3, 8, 9] were adopted and a preliminary set of 31 metrics, grouped by categories, was established.

D2. Initial products compilation:

This step involved identifying API Gateway products from global reports on the API "state of the market".

D3. Quality criteria definition:

Two objective quality criteria were defined, namely: a) visibility among different API landscapes - API vendors identified within the three industry reports were included in this study; and b) integration with analytics third-party software - products that contain integration plugins for widely used analytics and dashboard platforms such as Grafana, Prometheus and Datadog were added to this work.

D4. Technical Documentation Analysis:

This step involved a detailed review of the technical documentation for each software product to identify the metrics provided. Due to limitations of scope and space, this study did not aim at exploring measurement practices within the scientific literature.

D5. Data Extraction:

We created three data extraction forms. The first was designed

[2] The industry reports mentioned in this section are further discussed in section 3.

to collect basic information regarding all identified products and their vendors and included the first quality criterion as outlined in D3. Subsequent forms retrieved data for the API Gateways that met the quality criteria. The second form captured the company activity information, and the third form compiled the complete list of identified metrics from the selected products.

D6. Grouping Metrics:

Based on their definitions, metrics were grouped into categories according to the classification scheme proposed in [3].

2.1.4. Demonstration

The metrics catalog introduced in this paper contains data retrieved from a real-world scenario, as described in D5. Therefore, this phase involved presenting the artifact and showcasing the final data obtained. Section 3 details this step-in depth.

2.1.5. Evaluation

This step assesses the effectiveness and relevance of the artifact by employing an adapted version of the analytical model proposed by [6]. The collected data was quantified and used to benchmark the API Gateway products by scoring and comparison. Section 4 encompasses the results of this step.

2.1.6. Communication

This paper articulates the findings and insights from developing and evaluating the metrics catalog. This document also discusses implications, identifies gaps, and outlines future research avenues to enhance the present work. Also, the data collected during this was deposited in a publicly available repository.

3. Results

3.1 API Gateway offerings

An initial set of 68 API Gateway products was identified through the comprehensive review of industry reports conducted during the activity D2. Table 1 summarizes the total number of API management vendors and API Gateway products identified in each report. This overview establishes the context of API Gateway offerings and introduces the results that follow.

Table 1.
Summary of API Management Global Reports reviewed in this study.

Report	API Management Vendors	API Gateway Products
API State of the Market (Sep 2024)	71	62
Magic Quadrant for API Management (Oct 2023)	19	19
State of the API (Jun 2023)	20	20

Source: Own elaboration.

Table 2.
API Gateway products included in the catalog development.

Product	Vendor	Since
AWS API Gateway	Amazon	2006
Amplify API Gateway	Axway	2001
API Management & Gateway	Boomi	2000
Gravitee API Gateway	Gravitee.io	2015
IBM API Connect	IBM	2013
Kong API Gateway	Kong	2017
Tyk API Gateway	Tyk	2014
WSO2 API Gateway	WSO2	2005

Source: Own elaboration.

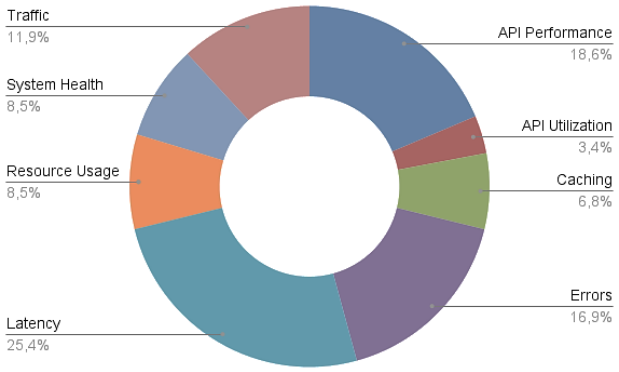


Figure 1. Metrics of the Catalog per each category.
Source: Authors' own data.

After applying the quality criteria described in D3, we selected eight products for developing the catalog since they met all inclusion requirements. Table 2 presents this subset and displays the product name, the vendor's name, and the product version (latest in all cases). The column Since was included to indicate the vendors' experience in the domain, and contains the year in which each company started to develop its first software product line related to API Management or previously-related technologies, such as web services.

3.2 Preliminary metrics catalog

A total of 59 metrics were identified during the analysis. These metrics were grouped according to the criteria outlined in D6. Fig. 1 illustrates the total numbers of metrics identified for each category, while Fig. 2 summarizes the total number of metrics for each API Gateway included in this study.

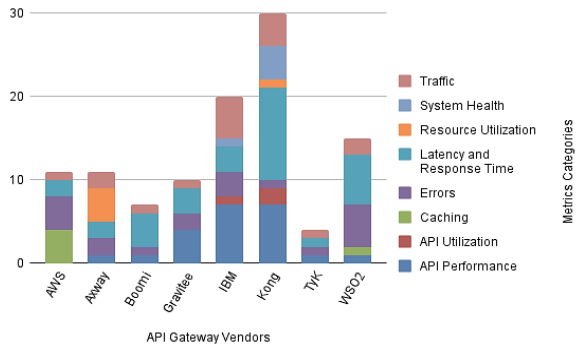


Figure 2. Metrics per each API Gateway.
Source: Authors' own data.

Table 3.
Traffic metrics.

Metric	Gateways
T.1.1 Total APIs Counts all the APIs that are deployed and being managed within the API Gateway.	1
T.1.2 Total number of Products Total number of products available in the API Gateway, which includes collections of APIs and usage plans (e.g. rate-limited plans).	1
T.1.3 Total API Requests Number of incoming requests processed by the API Gateway over a given period.	8
T.2.1 Total data transferred The amount of data that has been transferred through the API Gateway over a given time period. Includes both incoming and outgoing traffic.	2
T.2.2 Total inbound data The total volume of data received by the API Gateway from clients during a specified time period.	2
T.2.3 Total outbound data The total volume of data sent from the API Gateway to clients during a specified time period.	2
T.4.1 Throttling Errors The number of failing requests due to exceeding rate limits or throttling rules set for APIs or subscriptions.	1

Source: Own elaboration.

Next, the metrics are presented to the groupings as described in D6.

3.2.1 Traffic monitoring

Traffic metrics primarily focus on the volume and frequency of requests. These metrics capture the flow and volume of data transmitted through an API or network over a specified period, thus providing insights into user engagement, network activity, and usage patterns. Consequently, traffic metrics help to understand how much data is processed, its source, and its destination. Table 3 synthesizes the identified metrics within this category.

3.2.2 System health

Table 4 depicts the metrics encompassed in this category. System health metrics assess the overall operational status and connectivity of the API Gateway and its dependencies, such as upstream services and back-end databases.

Table 4.
System Health metrics.

Metric	Gateways
SH.1.1 Database Reachability Status A gauge type with a value of 0 or 1, which represents whether the database can be reached by the API Gateway (or not).	1
SH.2.1 Upstream Health Status Reflects the health status of upstream targets.	1
SH.3.1 Active Reading Connections Current number of connections where the Gateway is reading request headers.	1
SH.3.2 Idle Waiting Connections Current number of idle client connections waiting for requests to be processed.	1
SH.3.3 Active Writing Connections Current number of connections where the Gateway is writing responses back to clients.	1

Source: Own elaboration.

Table 5.
Caching metrics.

Metric	Gateways
C.1.1 Total Cache Hits Number of requests that were successfully served from the cache, rather than requiring fresh retrieval from the back-end.	1
C.1.2 Cache Hit Rate % of requests served from the cache compared to the overall total of requests.	2
C.2.1 Total Cache Misses Number of requests that required retrieval from the back-end system (since they could not be served from the cache).	1
C.2.2 Cache Miss Rate % of cache misses compared to the overall total of requests.	1

Source: Own elaboration.

Table 6.
API Utilization metrics.

Metric	Gateways
AU.1.1 Active inbound data The total volume of inbound data currently being processed by the API Gateway. Reflects the real-time data received from clients.	2
AU.1.2 Active outbound data The total volume of data currently being sent from the API Gateway to clients. Reflects the real-time data being transmitted in response to client requests.	1

Source: Own elaboration.

3.2.3 Caching

This category included metrics that evaluate the efficiency and effectiveness of the caching mechanism in the API Gateway, as described in Table 5. These measures reflect the role of caching efficiency and its impact on the system's overall performance. Therefore, they help provide insights into how effectively cached data can improve response times and reduce back-end load.

3.2.4 API utilization

API Utilization metrics are summarized in Table 6. Rather than cumulative or average data over time, the metrics in this category typically provide snapshots of real-time workload and resource usage by the API Gateway.

Table 7.
System Health metrics.

Metric	Gateways
SH.1.1 Database Reachability Status A gauge type with a value of 0 or 1, which represents whether the database can be reached by the API Gateway (or not).	1
SH.2.1 Upstream Health Status Reflects the health status of upstream targets.	1
SH.3.1 Active Reading Connections Current number of connections where the Gateway is reading request headers.	1
SH.3.2 Idle Waiting Connections Current number of idle client connections waiting for requests to be processed.	1
SH.3.3 Active Writing Connections Current number of connections where the Gateway is writing responses back to clients.	1

Source: Own elaboration.

3.2.5 System health

Table 7 depicts the metrics encompassed in this category. System health metrics assess the overall operational status and connectivity of the API Gateway and its dependencies, such as upstream services and back-end databases.

3.2.6 Latency and response time

Latency and Response Time Metrics are summarized in Table 8. These measures evaluate the time taken for an API Gateway to process incoming requests and relay responses, providing insights into the overall efficiency of the API Gateway and its interactions with upstream services. Various aspects can be analyzed, including service-level agreements (SLAs), performance issues, traffic patterns and response behaviors.

3.2.7 Resource utilization

This category focuses on monitoring and assessing how the API Gateway uses the underlying resources, such as compute power and memory, to handle API requests and traffic. It helps ensure that the infrastructure effectively supports API workloads and that resources are not over-utilized or under-utilized. Therefore, these measures provide insights to optimize system performance, resource usage, and scalability. Metrics are summarized in Table 9.

Table 8.
Latency and Response metrics.

Metric	Gateways
RT.1.1 Internal Processing Time (IPT) The total time taken to route a request through the API Gateway and execute all configured plugins.	1
RT.1.2 IPT (90th percentile) IPT measured at the 90th percentile.	1
RT.1.3 IPT (95th percentile) IPT measured at the 95th percentile.	1
RT.1.4 IPT (99th percentile) IPT measured at the 99th percentile.	1
RT.2.1 Total Response Time (TRT) The total time taken from when the API Gateway receives a request to when it returns a response to the client, this including processing time by the API Gateway and upstream services.	6
RT.2.2 Maximum Response Time The peak response time recorded for API calls.	3
RT.2.3 Minimum Response Time. The shortest recorded response time for API calls.	2
RT.2.4 Standard Deviation of Response Time The standard deviation of response times.	1
RT.2.5 TRT (50th Percentile) TRT measured at the 50th percentile.	1
RT.2.6 TRT (90th Percentile) TRT measured at the 90th percentile.	1
RT.2.7 TRT (95th Percentile) TRT measured at the 95th percentile.	5
RT.2.8 TRT (99th Percentile) TRT measured at the 99th percentile.	2
RT.3.1 Upstream Response Time (UpRT) The time taken for the API Gateway to relay a request to the back-end service and receive a response.	4
RT.3.2 UpRT (90th Percentile) UpRT measured at the 90th percentile.	1
RT.3.3 UpRT (95th Percentile) UpRT measured at the 95th percentile.	2

Source: Own elaboration.

Table 9.
Resource Utilization metrics.

Metric	Gateways
RU.1.1 CPU Utilization Percentage % of CPU resources utilized by the API Gateway instance.	1
RU.2.1 Disk Utilization Percentage % of disk space used by the API Gateway instance.	1
RU.3.1 Memory Usage in the Last Hour The average amount of memory utilized by the API Gateway instance over the past hour.	1
RU.3.2 Maximum Memory Utilization The peak memory usage recorded by the API Gateway instance during its operation.	1
RU.3.3 Memory Usage by Node The amount of memory being used by each individual node within a distributed API Gateway architecture.	1

Source: Own elaboration.

3.2.8 API Performance

API performance metrics, documented in Table 10, focus on the efficiency and responsiveness of API responses, aiding in evaluating the API's operational performance. These metrics allow for monitoring the overall reliability of the API service.

3.2.9 Errors

Portrayed in Table 11, the metrics included in this category measure all types of failures and issues that occur during API interactions such as client-side and server-side errors. These metrics help identify issues that may affect user experience or system functionality.

Table 10.
API Performance metrics.

Metric	Gateways
AP.1.1 Informational Response Rate % of responses that fall within the 1xx status coded (informational responses) over all received requests.	2
AP.2.1 Successful Response Rate % of responses that fall within the 2xx status codes (successful responses) over all received requests.	2
AP.2.2 Total Successful Requests Number of API requests that received a successful response, typically indicated by a 2xx status code.	3
AP.3.1 Redirection Response Rate % of responses that fall within the 3xx status codes (redirection responses) over all received requests.	2
AP.4.1 Request Throughput The total number of API requests processed by the API Gateway per unit of time (typically requests per second).	4
AP.5.1 Average inbound data The average volume of data received per specified time unit during a given period.	2
AP.5.2 Maximum inbound data The largest volume of data received in a single specified time unit during a given period.	2
AP.5.3 Minimum inbound data The smallest volume of data received in a single specified time unit during the given period.	2
AP.6.1 Average outbound data The average volume of outbound data (in bytes) sent per specified time unit during a given period.	1
AP.6.2 Maximum outbound data The largest volume of data sent to clients in a single specified time unit during a given period.	1
AP.6.3 Minimum outbound data The smallest volume of data sent to clients in a single specified time unit during a given period.	1

Source: Own elaboration.

Table 11.
Errors metrics.

Metric	Gateways
ER.1.1 Total Error Count Total number of failed API calls (both client and server errors) during a given period.	4
ER.1.2 Overall Error Rate % of failed transactions (including all 4xx and 5xx responses) during a specified time interval.	3
ER.2.1 Client Error Count Total number of client-side errors (4xx responses) logged within a specified period.	1
ER.2.2 Client Error Rate % of client-side errors relative to all responses in a given period.	3
ER.3.1 Server Error Count Total number of server-side errors (5xx responses) recorded in a specified time frame.	1
ER.3.2 Server Error Rate % of server-side errors compared to all responses over a designated period.	3
ER.4.1 Total Other Errors Count of all other error types not classified as client or server errors, including mediation errors and resource-not-found errors.	1
ER.4.2 Exception Count Raw count of transactions completed with an "exception" status.	1
ER.4.3 Total Authentication Errors Total number of authentication-related errors, which includes expired, missing, or invalid credentials.	1
ER.4.4 Total Target Connection Errors Total number of back-end connection errors. Typically includes timeouts and other back-end issues.	1

Source: Own elaboration.

4. Validation

Each metric sourced in this study represents a distinct element of the proposed catalog. Hence, using mutually exclusive set theory, it was possible to quantify the metrics available for each API Gateway product and assign a score, both globally and by category. In this framework, the catalog (C) serves as the universal set of metrics, *Mp* represents the metrics, and the API Gateway's benchmark score (S) is figured by comparing the metrics of the product against the comprehensive set of metrics in the catalog, as illustrated in (1):

$$S = \frac{\sum(M_p)}{\sum(C)} where M_p \in C \quad (1)$$

This quantitative method provides a detailed assessment through both internal and external comparison, yielding an understandable benchmark score. Appraisal schema is outlined as follows: metrics present on the API Gateway are compared to the metrics catalog, with each metric present marked as '1', and each metric missing marked as '0'. Metrics within the same category are summed to generate a category score.

Table 12 displays the scores for each API Gateway. The total of metrics from the universal set was included in row Catalog to facilitate comparison.

Table 12.
API Gateway metric raw scores (total of metrics).

API Gateway	Product Raw Benchmark Score (S)
AWS API Gateway	11
Amplify API Gateway	11
API Management & Gateway	7
Gravitee API Gateway	10
IBM API Connect	20
Kong API Gateway	30
Tyk API Gateway	4
WSO2 API Gateway	15
Catalog	59

Source: Own elaboration.

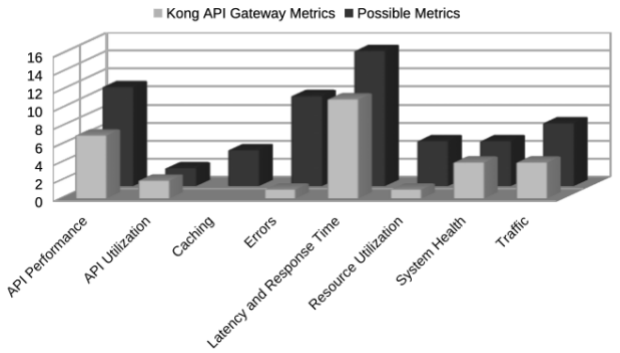


Figure 3. Comparison between Kong API Gateway metrics and the catalog. Source: Authors' own data.

To further facilitate a comprehensive visual comparison, Fig. 3 presents the raw score achieved by Kong API Gateway alongside the catalog of metrics. In the three-dimensional space, the front gray category score for all metrics in the API Gateway can be visually benchmarked against the darker catalog of metrics for each category, highlighting key gaps. This approach can similarly be applied to the overall API Gateway score (*S*).

The relative contribution of a category, derived from its metrics, is characterized by the maximum number of metrics present within that category, with '10' used to standardize the metrics into a comparable subset solution. As a result, a normalization function is implemented as follows (2):

$$F_j = \frac{10}{x_j} \sum_{i=1}^{x_j} c_{ij} \quad (2)$$

In order to compare different products, category scores were collated into a product level (P_{jk}) normalized benchmark score (3), derived with (x_{jk}) as the maximum number of metrics present in each category *j* of the API Gateway (*k*) and y_k as the maximum number of metrics in this API Gateway.

$$P_{jk} = \sum_{j=1}^{y_k} \frac{10}{x_{jk}} \sum_{i=1}^{x_{jk}} c_{ijk} \quad (3)$$

Table 13.
API Gateway metric normalized scores.

API Gateway	Product normalized benchmark score (P)
AWS API Gateway	16,76
Amplify API Gateway	15,10
API Management & Gateway	6,00
Gravitee API Gateway	9,06
IBM API Connect	25,51
Kong API Gateway	40,41
Tyk API Gateway	4,00
WSO2 API Gateway	15,27

Source: Own elaboration.

Table 13 displays the product level normalized scores of each API Gateways. The values within the table indicate the normalized benchmark score for each product.

5. Discussion

The field of API management remains underexplored in scientific research. In [10], a systematic literature review was performed to identify practices and capabilities in API Management. The authors compiled 24 unique definitions, along with 114 practices and 39 capabilities. A recent systematic mapping study [5] highlights a limited body of work, primarily focusing on a small set of quality attributes assessed using benchmarking tools. In terms of Web API quality modeling, several authors have made contributions. [11] proposed a maturity model for API management from an organizational perspective, while [12] introduced a quality model with three attributes and their quantitative evaluation in real-world applications. Additionally, [13] presented a GQM-based catalog of metrics to assess API usability. Despite these contributions, there remains a significant gap in research on software quality within the domain of API management. This gap motivated the development of our study, which aims to create an integrated framework to characterize, measure, assess, and improve the product quality.

There exist several systematic literature reviews and mappings that study different aspects of API lifecycle. These works provide valuable perspectives by exploring various aspects of API Management and API quality, such as documentation [14-16], usability [17-19], security vulnerabilities [20], architectural patterns [21], API evolution [22], and API research [23].

The outcomes of this study demonstrate that API Gateway tools provide metrics that align with several established methods related to system monitoring. These methods offer a comprehensive perspective on system reliability and responsiveness. For instance, the Utilization-Saturation-Errors (USE) method [9] and the Four Golden Signals [8] are designed to monitor the performance and health of APIs by defining metrics that identify performance issues, including systemic bottlenecks concerning utilization, saturation, errors, latency, response time, and traffic. In this context, our findings indicate that Errors, Latency, Response Time, and Traffic are prominently addressed features among the various API Gateway offerings analyzed in this study. However, metrics related to saturation were absent from the compiled data. Saturation is a critical concern that encompasses various aspects linked to Service Level Agreements (SLAs) and influences the business objectives of both software vendors and users. This void suggests a development opportunity for vendors to focus on developing saturation-based strategies to enhance their products.

In alignment with the previous findings, most of the tools examined also provided metrics in the category of API Performance. On the other hand, API Utilization, Caching, Resource Utilization, and System Health metrics were notably scarce. This disparity suggests that vendors prioritize specific aspects of API Management over others, potentially missing relevant features that could enhance overall product effectiveness. Addressing these gaps could lead to more robust API management solutions and better alignment with user needs and expectations.

The quantitative analysis method developed in this work revealed that Kong achieved the highest product normalized score, while Tyk, Boomi, and Gravitee.io presented the lowest scores. However, this disparity is not definitive enough to determine which product is inherently "better" than others. In this sense, this work presents two implications. On the one hand, low product scores may signify the absence of metrics that may add significant business value if addressed; for instance, caching may be a critical factor under different scenarios, although most products did not specify any metrics in this domain. On the other hand, examining high product scores against industry benchmarks can be crucial to confirm the appropriateness of each metric.

In validating our work, we employed a quantitative analysis method that effectively corroborated the catalog of metrics we developed. This method was theoretically grounded, as it integrated a mutually exclusive set theory with the four principles of causation theory, following the framework established in [6]. To further enhance the robustness of our findings, we meticulously reviewed three industry reports, minimizing the risk of bias and ensuring a well-rounded perspective on market offerings. Additionally, we exclusively relied on official sources of technical documentation, which helped guarantee the accuracy of the compiled data. Together, these approaches contribute to the overall validity and reliability of our study's conclusions.

6. Conclusions and future work

In this paper, we systematically analyzed 68 API Gateway offerings and developed a comprehensive catalog of 59 metrics using the DSR approach. These metrics were categorized based on existing literature, providing a structured set-theory framework. To validate this catalog, we implemented a quantitative analysis method that involved scoring and comparing a final set of eight products. Our findings indicate that current tools focus on critical aspects such as latency, response time, API performance, error capturing, and traffic monitoring. However, we also observed that other important factors - such as caching, saturation, and overall system health - are scarcely addressed in these applications. This disparity highlights potential gaps

in the current API Gateway landscape.

In this paper, we systematically analyzed 68 API Gateway offerings and developed a comprehensive catalog of 59 metrics using the DSR approach. These metrics were categorized based on existing literature, providing a structured set-theory framework that organizations can use to guide the evaluation and selection of API Gateway tools. To validate this catalog, we implemented a quantitative analysis method involving the scoring and comparison of a final set of eight products, demonstrating the practical applicability of our framework in real-world assessments.

Practical implications: Our findings indicate that current tools primarily focus on critical aspects such as latency, response time, API performance, error capturing, and traffic monitoring — factors that are essential for ensuring baseline operational quality. However, we also observed that other important aspects — such as caching mechanisms, saturation management, and comprehensive system health monitoring — are seldom addressed. This disparity highlights actionable gaps in the current API Gateway landscape, signaling areas where tool providers can innovate and improve. Consequently, this work serves as a guide for both adopters and developers aiming to enhance API infrastructure resilience and effectiveness. Finally, this work also offers a practical benchmarking methodology that can be adopted in other software quality domains, such as performance evaluation of service meshes, observability platforms, or other infrastructure-level tools. By providing a replicable and structured approach, the methodology supports broader efforts to assess, compare, and improve software quality across diverse technological contexts.

As with any study, this work has limitations that can guide future research in various directions. a) The primary source for consultation and analysis was the technical documentation of each tool, which may have concealed unidentified metrics. Future research could enhance this by exploring deployed tools to validate the provided metrics. b) Employing different research techniques to uncover black-box metrics could yield valuable insights into the current research. c) While metrics focus on the quantitative dimension of software quality, future efforts might expand this catalog into a tailored quality model, creating a flexible and robust framework that encompasses conceptual, operational, and quantitative levels of software quality. d) Furthermore, externally validating the developed artifacts through techniques such as surveys with practitioners and expert judgment could strengthen the findings of this research. e) Finally, as a flexible framework, this work could be broadened to include API management platforms, providing a more comprehensive understanding of the API ecosystem and its management practices.

References

[1] Duan, Y., et al. Everything as a service (XaaS) on the cloud: origins, current and future trends. En 2015 IEEE 8th International Conference on Cloud Computing. IEEE, 2015. p. 621-628.

[2] Gamez-Diaz, A, Fernandez, P., and Ruiz-Cortes, A. An analysis of RESTful APIs offerings in the industry. International Conference on Service-Oriented Computing. Cham: Springer International Publishing, 2017. pp. 589-604.

[3] De, B. API Management: An Architect's Guide to Developing and Managing APIs for Your Organization. Apress Berkeley, 2023.

[4] Preibisch, S. API Development: a practical guide for business implementation success. New York: Springer Science+Business Media, 2018.

[5] dos Santos, E. and Casas, S., Unveiling quality in API management: A systematic mapping study. Proceedings of L Latin American Computer Conference, (CLEI), 2024, pp 1-10.

[6] Cassidy, L. and Hamilton, J. design science research approach to website benchmarking, Benchmarking: An International Journal 23(5), pp. 1054-1075, 2016. DOI: https://doi.org/10.1108/BIJ-07-2014-0064

[7] Peffers, K., et al. design science research methodology for information systems research, Journal of Management Information Systems, 24(3), pp. 45-77, 2007. DOI: https://doi.org/10.2753/MIS0742-1222240302

[8] Beyer, B., et al. Site reliability engineering: How Google runs production systems. Sebastopol: O'Reilly Media, Inc., 2016.

[9] Gregg, B. Thinking methodically about performance, Communication of the ACM, 56 (2), pp. 45-51, 2013. DOI: https://doi.org/10.1145/2408776.2408791

[10] Mathijssen, M., Overeem, M., and Jansen, S. Identification of practices and capabilities in API management: a systematic literature review. arXiv preprint, 2020. DOI: https://doi.org/10.48550/arXiv.2006.10481

[11] Overeem, M., Mathijssen, M., and Jansen, S. API-m-FAMM: A focus area maturity model for API Management, Information and Software Technology, 147, 2022. DOI: https://doi.org/10.1016/j.infsof.2015.03.007

[12] Yamamoto, R., et al, A quality model and its quantitative evaluation method for web APIs. Proceedings of 5th Asia-Pacific Software Engineering Conference (APSEC), 2018. pp. 598-607. DOI: https://doi.org/10.1109/APSEC.2018.00075

[13] Machini, A. and Casas, S., A preliminary GQM model to evaluate web API usability, Proceedings of Jornadas Argentinas de Informática (JAIIO), 2024. pp 1-13.

[14] Nybom, K., Ashraf, A. and Porres, I., Systematic Mapping Study on API Documentation Generation Approaches. 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE, Prague, 2018. pp 462–469. DOI: https://doi.org/10.1109/SEAA.2018.00081

[15] Casas, S., et al, Uses and applications of the OpenAPI/Swagger specification: a systematic mapping of the literature. 40th International Conference of the Chilean Computer Science Society (SCCC), IEEE, La Serena, Chile, 2021. pp 1–8, DOI: https://doi.org/10.1109/SCCC54552.2021.9650408

[16] Cummaudo, A., Vasa, R. and Grundy, J. What should I document? A preliminary systematic mapping study into API documentation knowledge. ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). IEEE, Porto de Galinhas, 2019. pp 1–6, DOI: https://doi.org/10.1109/ESEM.2019.8870148

[17] Mosqueira-Rey, E., et al. A systematic approach to API usability: Taxonomy-derived criteria and a case study, Information and Software Technology, 97, pp. 46-63, 2018. DOI: https://doi.org/10.1016/j.infsof.2017.12.010

[18] Rauf, I., Troubitsyna, E. and Porres, I. A systematic mapping study of API usability evaluation methods, Computer Science Review, 33, pp. 49-68, 2019. DOI: https://doi.org/10.1016/j.cosrev.2019.05.001

[19] Burns, C. et al, Usable results from the field of API usability: A systematic mapping and further analysis. 2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), IEEE, Innsbruck, 2012, pp 179–182, DOI: https://doi.org/10.1109/VLHCC.2012.6344511

[20] Diaz-Rojas, J. A., et al, Web API Security Vulnerabilities and Mitigation Mechanisms: A Systematic Mapping Study. 9th International Conference in Software Engineering Research and Innovation (CONISOFT), IEEE, San Diego, CA, USA, 2021. pp 207–218, DOI: https://doi.org/10.1109/CONISOFT52520.2021.00036

[21] Taibi, D., Lenarduzzi, V. and Pahl, C., Architectural Patterns for Microservices: A Systematic Mapping Study, Proceedings of the 8th International Conference on Cloud Computing and Services Science - (CLOSER), 2018. INSTICC. SciTePress, Funchal, pp 221–232, DOI: https://doi.org/10.5220/0006798302210232

[22] Koci, R., et al, Classification of Changes in API Evolution. IEEE 23rd International Enterprise Distributed Object Computing Conference (EDOC). IEEE, Paris, France, 2019. pp 243–249, DOI: https://doi.org/10.1109/EDOC.2019.00037

[23] Ofoeda, J., Boateng, R. and Effah, J. Application Programming Interface (API) Research: A Review of the Past to Inform the Future, International

Journal of Enterprise Information Systems, 15(3), pp. 76-95, 2019. DOI: https://doi.org/10.4018/IJEIS.2019070105

E. dos Santos, received the BSc. in Informatics in 2007, from the Universidade Católica do Salvador, Bahia, Brazil. Sp. in Techonlogy Management in 2015, form the Universidad Nacional de la Patagonia Austral, Argentina, and is current a PhD student in Applied Sciences at the Universidad Nacional de la Patagonia Austral. Argentina. From 2008 to date, he worked as a teacher and researcher at Universidad Nacional de la Patagonia Austral, and consulting companies within the software development. Currently, he is an Adjunct Professor in the School of Systems and Informatics and a member of the Institute of Applied Technology, Universidad Nacional de la Patagonia Austral. His research interests include: software quality, web development and API management.
ORCID: 0000-0001-6729-0303

S. Casas, is an associate professor at the Institute of Applied Technology of the National University of Southern Patagonia, Argentina. PhD in 2008 from the University of Vigo, Spain. Her research interests include the study and application of software development improvement techniques.
ORCID: 0000-0002-8289-6132