Searching for Heterogeneous Geolocated Services via API Federation

Ala Arman, Pierfrancesco Bellini, Paolo Nesi

DISIT Lab, DINFO department, University of Florence
Http://www.disit.org
`<name>.<Surname>.@unifi.it`

Abstract. In the context of Smart City applications, the usage of Smart City APIs, for exposing services and data to web and mobile applications, is quite frequent. Most of the mobile solutions, using the Smart City APIs, are focused on a single city which can expose several services that are contextualized on a single geographic area. In fact, passing from one city/area to another, the users must change applications and services, and consequently, discontinuity problems could occur at the border. This also happens for the lack of interoperability among the Smart City APIs and related operators that may strongly differ, depending on the applicative levels at which they are developed. A large part of the services proposed via Smart City APIs are geo-localized, and as a result, may provide different results according to the GPS coordinates of the client context. In this paper, the *problem of the federation of smart city services is addressed by proposing a solution for federating smart city APIs, related knowledge-base, and ontology*. To this end, a solution to *autonomously federate API services* has been presented together with other requirements (e.g., *efficiency*, *overlapped* and *included* areas of competence, *distributed searches*, *security and privacy*, *scalability*, *interoperability* among different smart city application servers) which are typically neither all satisfied by classical Geographical Information System (GIS) solutions that federate the services at the level of database nor by those based on Internet of Things (IoT) Brokers. The solution is open-source and has been developed in the context of the Snap4City European platform enhancing the former Km4City Ontology and API of the *Sii-Mobility* national project (https://www.snap4city.org). The solution is presently in use in Snap4City federation of Smart City Services in Europe, among several cities/areas including, Florence, Tuscany, Bologna, Helsinki, Antwerp, Valencia, Dubrovnik, and Mostar, just to mention a few.

Keywords: knowledge base, smart city API, smart city services, federation of smart cities, FiWare, IoT Orion Broker

1 Introduction

In the context of Smart Cities, not all cities/areas are becoming smart in the same manner and are smart at the same level because provided services are typically different [1]. In most cases, the cities decide to address only a selection of smart services (e.g., smart parking, smart education, smart gov., smart lighting), and not others, according to their needs and strategies. Therefore, vertical applications have been implemented for years and are not integrated in most of the cases. In the context of Smart City applications (web and mobile), most of the early solutions have been based on GIS and provide standard solutions for distributing geo-localized entities (e.g., maps, shapes), using protocols such as Web Feature Service (WFS), Web Map Service (WMS) [2]. Other solutions provide data via Open Data platforms such as the Comprehensive Knowledge Archive Network (CKAN) [3]. These solutions provide Application Programming Interfaces (APIs) to access a single dataset file as well as a collection of APIs provided by multiple stakeholders in the city/area and the possibility of exchanging these descriptions via harvesting protocols. IoT solutions, based on IoT brokers for the smart city, have been also proposed. For example, FIWARE-based solutions, which expose Next Generation Sensors Initiative (NGSI) REST APIs, are provided to the Web and mobile clients to access the data, typically last values (and not historical values), directly accessing IoT sensor data (and not sophisticated data structures via sematic queries) [4]. A strong push on the usage of Smart City APIs (SCAPIs) for providing and creating data and services for web and mobile applications has been recently observed (e.g., the Knowledge Model for the City (Km4City) API [5] , E015 [6], [7]). Therefore, with the aim of developing smart city solutions, the usage of SCAPIs can be the way to provide smarter applications, considering multiple aggregated data sources and analytics (e.g., weather, reasoning, and predictions on parking, traffic and people flow [8]). On the other hand, most of the SCAPIs services are focused on a single city/area and expose a limited number of contextualized services in the same geographic area (e.g., info-mobility, Point of Interest (POI), routing, smart light, smart parking). In fact, in most cases, passing from one city/area to another, the users must get other applications to get the same services. This also happens due to the lack of interoperability among the SCAPIs at a semantic level that is not standardized and may strongly differ depending on the applicative levels at which they are developed.

In this paper, a solution for federating SCAPIs among geographic areas and contexts is presented. The development of the proposed solution for the smart city federation overcomes the problems of GIS and open data solutions. The main addressed problems are related to the possibility (i) to provide a network of geolocated services without constraining the providers to agree on the service shared with the other providers, (ii) to provide the clients a GIS/IoT list of results services without reporting eventual duplications on overlapped and/or duplicated services, (iii) not to pose limits to the size/shape of the geo-area and of the shared number of services, iv) to avoid addressing the problems of data privacy in a centralized structure, (v) to (or not) decide to join the network of services, (vi) to offer (non-)geolocalized information along with services in the network, and (vii) to provide a scalable and fault-tolerant solution for recovering services. The main cases are depicted in **Fig. 1**, where two regions of services may be overlapped, or one included in another. The services can be present in both (thus they are duplicated) or in one and even shared among them such as a passing road/path from one to another.

Therefore, the main contributions of this paper are: i) the possibility of federating Snap4City/GIS solution with FIWARE solutions based on IoT Orion Broker, and ii) the assessment of performance for the federated solutions and to the access private and public IoT devices. The validation of the presented solution has been performed by considering 4 large areas and smart city services (together with smaller areas) in place covering the Tuscany region of 3.5 million of inhabitants in the center of Italy, north of Italy (Garda area) and Sardegna island, Antwerp and north of Belgium, Helsinki and south of Finland, Spain (Valencia), Occitanie (Pont du Gard), Dubrovnik, and West Greek. To solve the above-mentioned problems, the solution reported in this paper has been completely developed open-source and presently used in a federation of Snap4City (https://www.snap4city.org) Smart City Services and APIs in Europe including federated services in different cities and regions (e.g., Florence, Tuscany region, Helsinki, Antwerp, Dubrovnik, Garda, Valencia, Pont du Gard, Greek) [9].

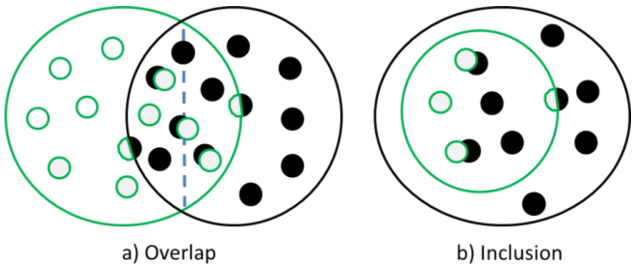


Fig. 1. Overlapped and included areas of services, with duplicated, exclusive, and shared services of two areas.

The paper is organized as follows. Section 2 reports and discusses the related works. In Section 3, the requirements identified for federating SCAPIs are presented and analyzed. Section 4 presents the general architecture and solution of the proposed Snap4City federation smart city services which may cover single cities and regions, with area and service overlaps and flexibility. The same section also includes the formal presentation of the operational model of the federation of SCAPI enabling the solution, thus addressing both local and geo-distributed services and information. Section 5 briefly describes the Km4City Ontology and the developed web-based tool for the proposed solution. Section 6 presents the mechanism for propagating the queries of the SCAPI in the federated network, managing the exception, and combining the results. Section 7 provides a description of the validation and the experiments performed for the assessment of the performance of semantic queries and for those regarding IoT devices/sensors to access private/public city entities, via federated queries, by different user roles. Conclusions are drawn in Section 8.

2 Related Work

The first smart city applications have been developed by exploiting technologies of GIS solutions which the federation of services also allows data exchange [2]. The classical GIS interoperability is limited to a 1: 1 exchange of geographical data; for example, exploiting protocols (e.g., WFS, WMS) for the exchange of Maps and geo-elements (e.g., paths, POIs, road elements, road graphs). A Web-service-based software for data discovery, download, visualization, and analysis has been proposed in [10] which included a middle layer, named HIS Central, mediating among clients and data located on distributed GIS-based HydroServers. The solution is then focused on delivering GIS data and not full smart services. In [11], the CityPulse project has been proposed for real-time data stream analysis by exploiting semantic modeling. In [12], a wide review of IoT solutions for smart cities has been presented with current and future research directions. The paper has performed an extensive analysis of literature identifying major keywords and domains of applications, but only marginally addressing smart city interoperability. In the context of Smart Cities, the solutions for managing geographically distributed Big Data to provide Smart Services are more relevant [13], especially when they are capable to work on a large scale (i.e., spreading across sets of cities, intersects with the problem of managing with the Big Data generated by dense and/or extensive environmental monitoring systems [14]).

Other solutions provide data via Open Data platforms (e.g., CKAN [3]) which provides APIs to access the single dataset file and a collection of APIs provided by multiple stakeholders in the city/area and the possibility of exchanging these descriptions via harvesting protocols. A similar approach has been proposed by $E015$ with their collection of SCAPIs and services in the north of Italy [6]. IoT solutions, based on IoT brokers, for the smart city, have been also proposed; for example, FIWARE-based solutions which expose NGSI Rest APIs are provided to web and mobile clients to access data (typically last values and not historical values), directly accessing IoT sensor data (and neither via sophisticated data structures nor semantic queries) [4], [15], and [16]. They mainly move the complexity of the service to the Mobile Apps since the IoT broker only provides data, and thus, the business logic of the application must be elsewhere. Therefore, smart city applications may use those APIs for implementing their logic on the server and client sides. In most cases, mobile and Web Apps need to be updated when a new data type is added.

A review of SCAPIs can be recovered on [17] and [18]. Examples of more complete SCAPIs are: Km4City API on a large range of SCAPIs to search and retrieve information and services [5], on DIMMER for the composition of smart city services exploiting service-oriented architecture [7], CitySDK which provides a set of smart city services without federation and Transport APIs on mobility. An early version of the SCAPI federation has been presented in [19].

3 Requirements and Analysis

In this section, the requirements that should be satisfied by a solution for federating SCAPI are reported and discussed. The solution should be a distributed system of SCAPIs to create a federation of City services. The federated network is conceptually a middleware, based on a set of SCAPI services (provided by Nodes), that is independently offered and maintained by a number of cities/areas. The federation approach should not be confused with a collection of APIs in a common basket to expose them uniformly in terms of definitions. It is, in fact, managed and offered by different organizations that provide the service to a single city/area (e.g., [6], [3], [20], [21]).

Therefore, according to the definition of a federated SCAPI network of Nodes, the services, and in particular, each Node, should satisfy the following advanced requirements.

Req.1. **not permanently replicate data** of other Nodes;

Req.2. support **distributed search** on the network federated Nodes. The computational workload for providing results is distributed among the nodes which can work only on their own data;

Req.3. be of **any size** in terms of geo area and data volume. The geospatial size and shape of each Node may be (i) of any form and multiple connected (so-called multi-polygon). A Node may manage one or more geo-areas even if they are distant and not neighbouring areas, (ii) partially overlapped with other Nodes, (iii) totally included in other Nodes, and (iv) disjoint and even far from each other;

Req.4. **offer a different number/kind of services**. This allows a Node to provide different kinds of services without constraints and to autonomously decide the set of provided /removed services;

Req.5. **contain (non-) georeferenced services.** There may exist services that are generic for a certain Node and not associated with a GPS position (e.g., global service of payment, global service to save the car position);

Req.6. **respond to API** calls in terms of services in a **transparent manner, thus allowing clients to pass from one Node to another**. When federated nodes are geographically contiguous, requested results may take into account both areas and avoid duplications of results;

Req.7. support access control to **prevent access to services by non-authorized users**. Users should be registered and authorized in multiple services/cities to freely access the protected data on both sides when authorization may arrive from a common Single Sign-On (SSO). This feature opens the path for multi-site operators, such as those for parking, car sharing, etc. This requirement implies a set of access rules to assure that the data is accessed only by authorized personnel at which the single Node may grant access independently, considering access authorization and GDPR compliance [22];

Req.8. **join and abandon the network,** without the need for network restructuring, and modifications with an immediate effect when no service reloads or disrupts;

Req.9. **provide query results in real time even in presence of a large number of Nodes**. The implementation should provide support for creating redundant solutions with high resilience and fault-tolerance;

Req.10. provide search query **results in a coherent format** with the expected response of the single services. For example, REST calls in some cases provide responses in JSON, XML, or HTML formats;

Req.11. interoperate with **Nodes based on data services**, for example, IoT Brokers, when they are supported by historical databases and geo-queries (as in FIWARE [4]);

Req.12. support **services based on IoT Devices.** This is possible if the Node is able to manage IoT Devices as an IoT Broker or the Knowledge Base, KB, exploited by the Node, is capable of modelling, indexing, and searching IoT Devices. To this end, the Km4City Ontology, by supporting the Industry 4.0 domain, has been extended to model IoT Devices by including a range of IoT Brokers, protocols, and devices;

Req.13. support the **creation of disjoint federations of Nodes and the presence of independent services** which are not connected to any federation of Nodes.

Req.14. support for an interactive user interface.

In Section 4, the architecture, Req.1 is satisfied by the solution because different Nodes/servers do not locally copy the data of the other services and only know about the presence of the other services. In fact, the nodes/servers keep their high-level descriptor, in terms of area of competence, as described in Sections 4 and 5. Meeting Reqs. 11, 12, and 13, by the solution, has been also demonstrated in Section 4. Reqs. 2, 3, 4, 8, and 9 are satisfied through validation and performance assessments in Section 7. Reqs. 5 and 6 are discussed in Section 7.1. In addition, Req.7 is satisfied and addressed at the level of single SCAPI, according to security aspects, as described in [22]. The evidence for the satisfaction of Req.10 has been described in Section 7.2. Also, Req. 14 is briefly highlighted in Section 5.1, while the actual implementation is accessible via https://www.snap4city.org and specifically via https://www.snap4city.org/MultiServiceMap/.

4 General Architecture of Federated Services and SCAPIs

To satisfy the above-described requirements, we have designed and implemented the solution reported in **Fig. 2** in terms of its architecture. It is based on a network of federated Nodes exposing services via SCAPI. The Nodes, which are called Super, each of which (i) provides access to services via SCAPI formalism as REST API to client applications of the federated network and (ii) exploits SCAPI of the actual Smart City service providers and of other Supers. According to Req.11 on interoperability, the Nodes/Supers can be on the front of the SCAPI interface of two kinds of smart city servers: (i) *Snap4City/Km4City*, which provides GIS data via WFS/WMS, services via SCAPI, querying the smart city server with SPARQL queries, and exploiting the Km4City Ontology on the RDF store Virtuoso, and (ii) *SSM2ORION* (SuperServiceMap to Orion FIWARE), which converts the call on SCAPI to NGSI V2 Rest Calls, provides data information accessible from FIWARE solution, based on IoT Orion Context Broker, with storage support, using well known tools (e.g., Quantum Leap broker, CrateDB) as described in the following [4]. According to **Fig. 2**, Node (a) manages data of Area 1 by the Km4City RDF store and ServiceMap. Nodes (b) and (c), using the Km4City RDF stores and in a balancing and fault-tolerant approach, share the same geo-Areas $2a$ and $2b$. Node (d) is covering Area 3 with an IoT Orion Broker FIWARE and a related storage. Some of the Nodes manage overlapped areas while Areas $4a$ and $4b$ are managed by an independent service. It is noted that some of the areas covered by Nodes contain multiple disjoint subareas. To avoid having a single point of failure, each Node includes a master of the distributed communication among Supers reporting the lists of Nodes/Supers and providing services to the clients. A central server, with the list of the connected Supers, is made accessible in one or more Web servers for the periodic update of Supers. Each Node/super has a representation of the multi-polygon addressed by the nodes (with their data/services) and thus of their partitioning over the nodes of the federated SCAPI network. In more details, each Node may be registered in the list of Supers with its descriptor of the multi-polygon area of competence.

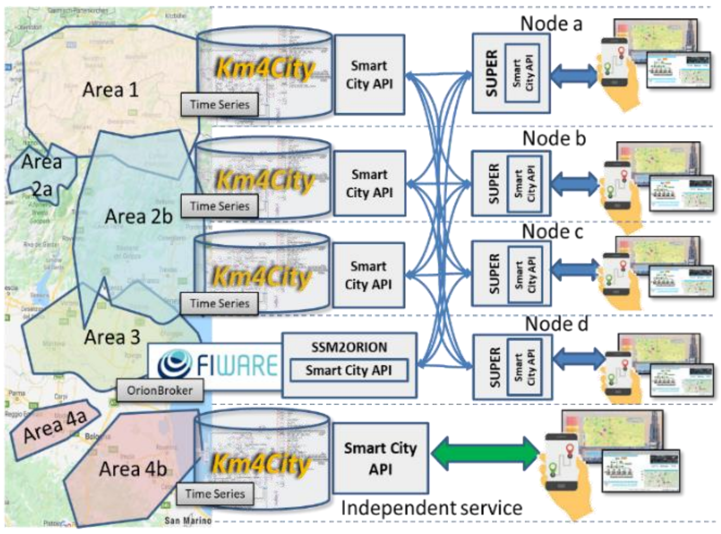


Fig. 2. General Architecture of Super and Federated SCAPI network

The most relevant aspects and complexity of the procedure are mainly related with (i) identification of the Nodes to be involved, according to the requests received, with a complexity that may depend on the size of the node descriptors (number of polygons), (ii) distribution of query in each Super, with a complexity $O(1)$, which is based on the decreasingly sorted execution time of nodes, using a multithreading approach, and (iii) collection of results and their fusion, as the main complexity, to avoid duplicates and compounding the results, with a complexity, depending on the size S of the elements $O(S)$.

In **Fig. 3**, a simplified architecture is reported where the authentication and authorization layer which allows to satisfy Req.1. It is noted that, while any kind of data may be integrated via Data Connectors in Node-Red (including WFS/WMS of GIS via Snap4City tools), IoT Devices can be registered on the IoT Directory of Snap4City and connected via one or more IoT Brokers with different protocols. New IoT Devices must be registered, and each data access must be authorized according to the user authentication/authorization [22]. In the case of Federated Nodes, the authentication/authorization must satisfy the GDPR. In addition, the federation can also be performed at the level of authentication and SSO-SAML (Security Assertion Markup Language) of Key-Cloak. Snap4City is then based on OpenID Connect and JWT Access Token while the role management is performed by using Lightweight Directory Access Protocol (LDAP) federation protocols.

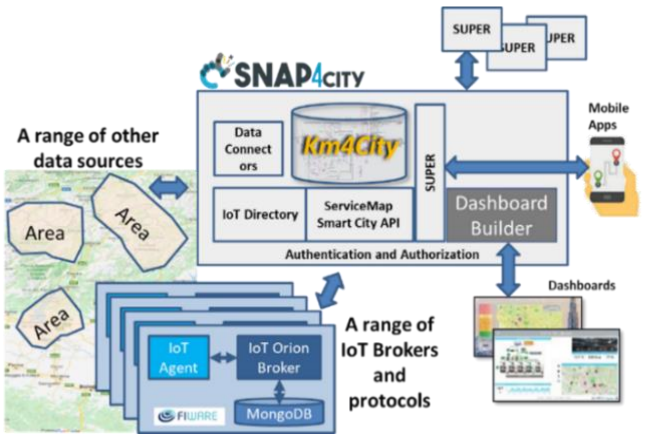


Fig. 3. Snap4City solution integrated with Federated Nodes

5 Km4City Data Model and User Interface

Km4City Ontology [23] is based on several vocabularies (e.g., DCTERMS, FOAF, SCHEMA.org, WGS84_pos) [24]. It addresses several domains namely, mobility and transport, energy, health, economy, and key performance indicators (KPIs), just to mention a few while modelling concepts such as POIs, road structure, and civic numbers. The Km4City Ontology I, based on 10 macro-classes/areas (e.g., Places, Administration, POIs). These concepts and relations may model a wide spectrum of applications (e.g., indoor routing, the automatic building of context-rich synoptics). Regarding the modeling of Data Analytics results, the Km4City ontology provides, for example, the Predictions macro-class together with a set of associated concepts (e.g., BusStopForecast) which enables the representation of the expected arrival/departure time of a given ride at a given stop [19].

5.1 User Interface Exploiting SCAPI: SuperServiceMap

In this section, we present details on the designed and developed a web-based tool to perform the queries via a graphic user interface, called SuperServiceMap, which is freely accessible via www.snap4city.org and depicted in **Fig. 4**. Using the selector in the center of the tool, the user can define to connect the user interface to a Super as well as to one of the SCAPI services of the single organization/node (e.g., Firenze, Helsinki, Valencia) which 18 of them are present now. The menu on the right comprises two tabs. One is dedicated to categories (e.g., Accommodation) and sub-categories (e.g., Camping, Farm, Hostel), associated with Regular Services, while the other contains categories (e.g., Area) and sub-categories (e.g., Gardens, Sports_facility) regarding Transversal Services. The user can select, in addition to the possibility of instant searching for a service category or a sub-category, a set of them, including Regular or Transversal. Also, it is possible to limit the search results, using N. Results, or filter them, using the Text Search and Search Range (e.g., 100 (m)) together with the Service Model and Value Type [25] options.

The menu on the lower left is used to show the weather information (e.g., minimum/maximum temperature) in a region. The menu on the upper left is used to obtain information on Public Transportation, Address Search, and Events along with the possibility of text service search in a desired area. Considering the Public Transportation tab, it is possible to detect the current position of vehicles, using the associated Agency (and Line). The user can also see a Route (or all of them) of a Line and a Stop/station (or all of them) of a Route, together with detailed information on stops/stations. It is possible to Search Path, considering the Route Via (e.g., car, public_transport) and Start date&time options, and Search Geometry between two selected points. The interface also allows performing text search when constraining the search around a point or in a given area.

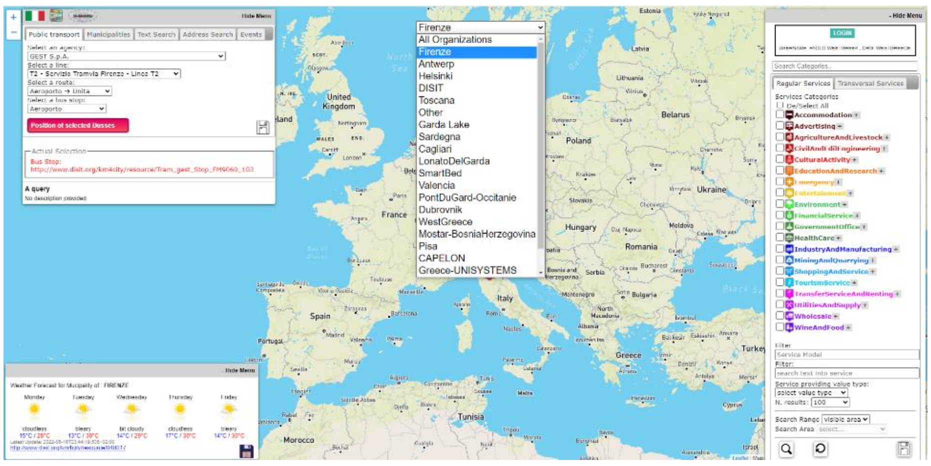


Fig. 4. SuperServiceMap tool (https://www.snap4city.org/MultiServiceMap/)

6 Federation of Enabled SCAPI via Network of Supers

In this subsection, the actions needed to enable the federation of Nodes are described by starting from the limitations and the improvements performed on former SCAPI as it was presented in [17]. The full list and semantic of input arguments for API is available on https://www.km4city.org/swagger/external/index.html .

6.1 Enhancing Smart City API on Geo-distributed Services

The extension of the SCAPI has been performed on specific query and data types that may be shared over different areas managed by different nodes and API services in the network. In more detail, since an area can be overlapped with other areas, some of the geo-located elements can be completely located/duplicated in multiple areas, partially located into multiple areas (e.g., a national across multiple provinces or regions).

Therefore, the most affected services are those related to Service discovery and information family of APIs, that depending of values received from input arguments, performs a search for services (i) close to a given position (identified by a GPS location or service), (ii) within the boundaries of a bounding box described through the geospatial coordinates of its vertexes, (iii) within a shape of a given geographic area (which can be defined as a Well Known Text (WKT) shape format), with an arbitrary shape or the shape of a city, a province, etc., or iv) through the submission of a full-text to be matched in addition to or without geo restrictions. The key parameter through which the geographical area of interest is delimited is called Selection. It can be (i) a Point plus distance, (ii) two points, or (iii) a shape for which builds a geographic area.

More complex entities are those including a Path or a Shape in the Selection. For example, the Bus Lines may pass into the area of Selection even if they do not present any Stop into the area. Therefore, because of certain Selection, one could be interested to retrieve the Lists of Lines, Routes, Stops, etc. that are different from a geographical position or area. Among the most complex queries, those along paths need special attention (e.g., cycling path, bus-lines, routing paths). The identification of Services along a path implies to search for services that are close to the path within a certain distance. Otherwise, the perfect match on the path would be too selective. Therefore, the search along a path must be transformed to a search into a closed polyline which has the Selected path as mean point by point.

Queries to identify Shortest Route/Paths to compute the best route from a starting point to a destination through a modal or multimodal routing provide results only when both the start and the destination parameters are geospatial coordinates belonging to the same service Node/area. However, to get the results, it is also possible to provide the URI of the service where they locate, and/or the URI of the service where they wish to go. In this case, the following are checked: (i) the Node managing them (via cache or via request) and (ii) if there exists a Node that matches both the required source and destination; otherwise, an error is returned. The error management allows to split the problem in distinct routing queries towards the corresponding Node services.

The results of the queries are services expressed as a list of ServiceURI (services are stored as resources in the RDF store and identified through a URI that also is a Uniform Resource Locator (URL) from a Linked Data perspective). On the contrary, if it is not possible to identify the corresponding Super managing the entity, all nodes must be queried. To reduce this effort, the pairs ServiceURI and Node ID are cached for shortening time of future requests.

6.2 Exceptions in the Selection of Nodes to be Queried by the Supers

With the aim of creating fault-tolerant redundant solutions, it possible to have different Super largely covering the same area. To maximize performance or making a priori balance on services, it is possible to define a priority, among the different Supers/nodes, based on the query/areas or on services or exclude some of them in specific cases. Nodes with the same priority are queried in random order. Thus, the above-described distribution of queries based on the area of competence, based on the API type and/or output format, can be overwritten by specific rules for each Super,. This mechanism can be used to redirect the requests to a specific node non-geolocated services as planned, according to Req.5.

6.3 Merging Node Results of any Format

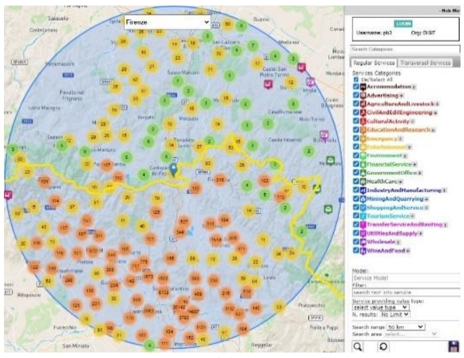
The results of Snap4City SCAPI can be provided in JSON or HTML [17], according to the request performed. When the result of a call is requested to be in JSON, a resulting message can be easily merged. On the contrary, if the result is requested in HTML, a final full Web page is provided (e.g., a set of geolocated services, each of which is represented by a clickable pin drawn on a map) with relative URLs to JavaScript, style sheets, images, etc. can be found. Also, the Super may need to combine multiple results by merging data of different Nodes with possibly different base URLs. Therefore, the HTML received from the Nodes must be first parsed, and then, relative URLs identified and replaced with full URLs by simply prefixing them with the base URL of the Node that has produced the Web page. Remarkably, operating this way, artifacts, located on a Node, are directly addressed and embedded in a Web Page that is provided by the Super, operating the server-side on the original Node, which leads to managing the Cross-Origin Resource Sharing (CORS) [26].

7 Performance Assessment and Validation

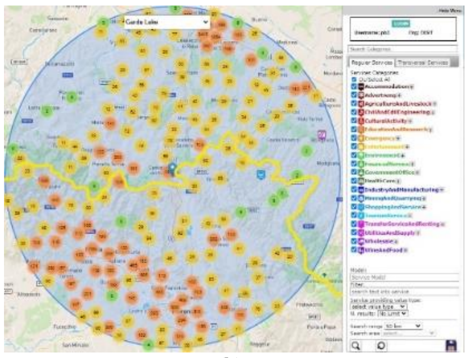
This section describes the results of the validation and performance assessment of the proposed solution for managing distributed geo services when addressing the problems derived from the overlap and inclusion of different areas and their geo services (e.g., locations, paths, shapes), as depicted in **Fig. 1**. Considering both scenarios, we compared the performance of results when direct and federated queries are formulated. The queries have been performed by searching for (i) all regular services (i.e., single points as POIs), including 20 categories and 540 sub-categories, and (ii) a regular service TransferServiceAndRenting with 47 associated sub-categories (e.g., Tramline, Urban_bus). For the assessment, we focused on measuring the Response Time (RT) and Number of Results (NoR), by identifying circular areas with different diameters, ranging from 1 to 100 (Km).

7.1 Case (a) for Overlapped Areas

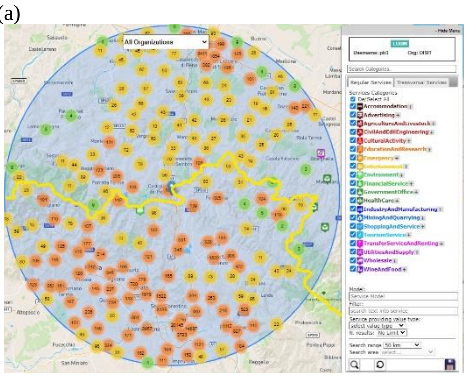
This case has been assessed in real conditions at the border of two adjacent regions, with two overlapping KBs, in terms of services modeled by Km4City (as tenants on Snap4City.org platform), namely, Florence, and Garda Lake. **Fig. 5** reports the query results, all presented in clusters, when searching for all regular services, including all categories and sub-categories, when the radius is equal to 50 (km), and the center has been placed on the center of the overlapped area, at the border (in yellow) of two provinces of Florence and Bologna.



(a)



(b)



(c)

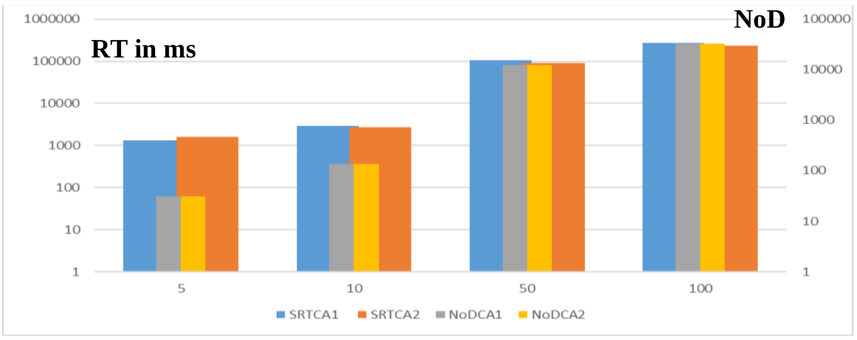
Fig. 3. Results for query on all Regular Services on an overlapped area (case (a) of Fig. 1), when search radius is equal to $50\ (km)$, only Firenze KB (a), only Garda Lake KB (b)**,** and full query results on the federated network (c) (GPS Center location (Lat,Lon): 44.122657449923224, 11.212175488471985).

Table 1 reports the results of the performance assessment for the described Case (a) scenario. For example, when the search radius is equal to $50\ (km)$, the NoR for Regular Services, considering Florence and Garda Lake regions, are respectively equal to 80546 and 23213 entities. The number of results for the same search via Super is equal to 91483 entities which is slightly different than the sum of results ($80546 + 23213 = 103759$) due to the presence of 12276 duplicated data (NoD, Number of Duplicates). For example, bus-stops of lines that connecting Florence and Bologna/Garda Lake regions must be available in both areas. Regarding the RT, it can be observed that, the Super has a better performance when the two regions are considered separately. For example, when searching for all Regular services and the search radius is equal to $50\ (km)$, the response time, is $104640\ (ms)$, using the Super is which less than the sum ($87600\ (ms) +\ 78660\ (ms)$), and more than the max of the two RTs, Max_NoR, ($87600, 78660$). It is noted that the RT for super also includes the time for eliminating the duplications.

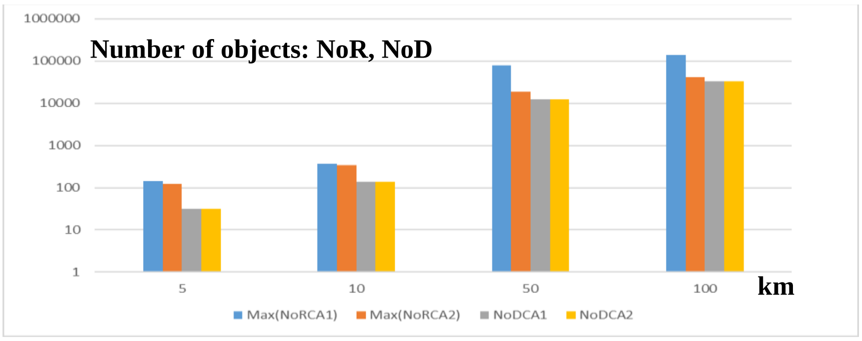
Table 1. PERFORMANCE RESULTS FOR THE CASE (A) OF Fig. 1. (OVERLAPPED REGIONS): SINGLE NODES (FLORENCE, GARD LAKE) VS. SUPER.

Query/kind	Search Ra-dius (km)	Node/Organization/Region				Super	
		Florence		Garda Lake			
		NoR	RT (ms)	NoR	RT (ms)	NoR	RT (ms)
Get Regular Services (case $A1$)	1	0	1578	11	3160	11	3050
	5	47	1592	143	1106	159	1334
	10	260	1622	362	2570	485	2940
	50	80546	87600	23213	78660	91483	104640
	100	139279	270180	48259	191580	154774	271500
Get TransferService-AndRenting service (case $A2$)	1	0	1661	11	3100	11	3100
	5	31	1437	125	1128	125	1576
	10	150	1623	339	2650	352	2670
	50	18720	78900	17856	79140	24281	90360
	100	40163	212160	42393	189900	49769	236040

Considering **Fig. 6,** it can be observed that the performance of the solution is very good in terms of RT if the query at the border is in the range of 10 (km) (which is very realistic for the smart city and rural services, in Europe), even in the presence of a high number of duplications (NoDs) and a very high NoR. When the range of the query is very large (e.g., $50 - 100$ (km)), the number of resulting objects and duplications may be very high and thus the RT to get all results from Super is still comparable with respect to the RT of the service providing the majority of the results.



(a)

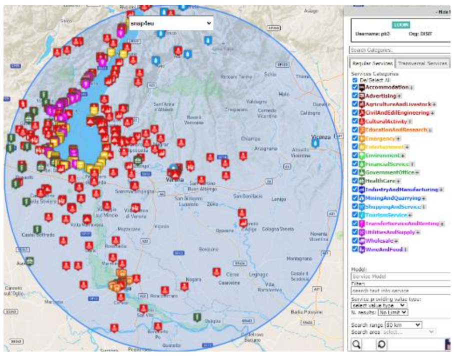


(b)

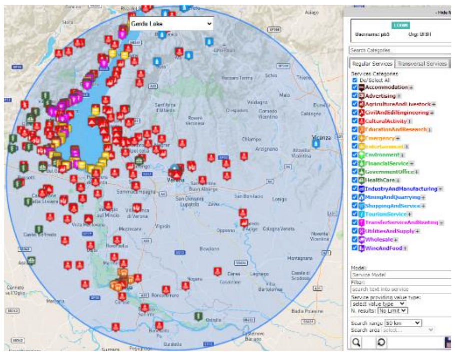
Fig. 6. Comparative trends to get regular (case $A1$) and transversal services (case $A2$) in overlapped areas w.r.t. to the range in km. (a) reports in Log Scales the trends of RT for super case A1 and case $A2$ respectively, $SRTCA1$, and $SRTCA2$, where on the right the data regarding the NoD; (b) reports the comparison of in Log scale of the trend of the NoR and NoD for Cases $A1$ and $A2$ respectively**.**

7.2 Case (B) for Inclusion Areas

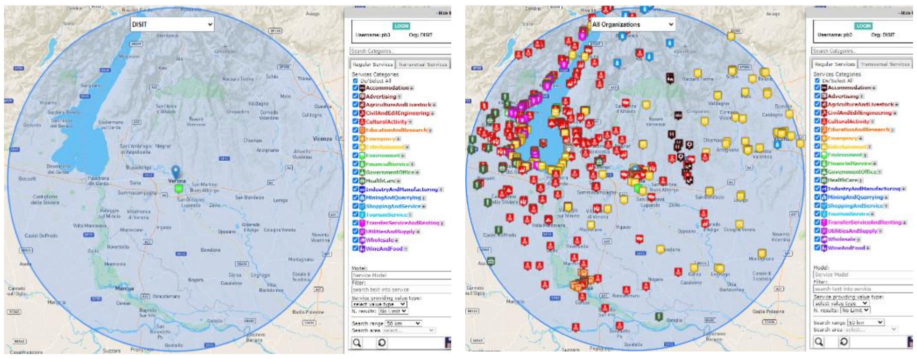
Considering different search radiuses, this case has been also addressed in real conditions by examining **three** regions namely: Garda Lake, Snap4eu, and DISIT (managed by different organizations), where the KBs of Snap4eu and DISIT, in terms of services modelled by Km4City (as tenants on Snap4City.org platform), is included in the KB of Garda Lake. **Fig. 7** shows an example of the described scenario when the search radius is equal to 100 (km).



(a)



(b)



(c) (d)

Fig. 7. Results for query on Regular Services on an inclusion area (case (b) of Fig. 1), when search radius is equal to 50 (km), considering, only snap4eu KB (a), only Garda Lake KB (b), DISIT (c), and full query results on the federated network (d).

We compared the results when the queries are performed on the federated network (Super). **Table 2** reports the validation and performance assessment results for the above-mentioned scenario. As on can see, for example, when the search radius is equal to 100 (km), the NoR for Regular Services, considering Snap4eu, Garda Lake, and DISIT regions, are respectively equal to 231, 4385, and 2086. The NoR for the same search, the Super find 4621 results which is much smaller than the sum (231 + 4385 + 2086) of the results coming from the single organizations: Snap4eu, Garda Lake, and DISIT. Regarding the RT, we can see that, using the Super provides a response time comparable with the larger service. For example, when searching Regular services and the search radius is equal to 50 (km), the RT for super is 440 (ms), which is comparable with 322 (ms) presented by the GardaLake service which provided the majority of the results.

Table 2. PERFORMANCE RESULTS FOR THE CASE (B) OF INCLUSION REGIONS (SNAP4EU, GARD LAKE, AND DISIT): SINGLE NODES VS SUPER.

Query/kind	Search Radius (km)	Node/Organization/Region						Super	
		Snap4eu		Garda Lake		DISIT			
		NoR	RT (ms)	NoR	RT (ms)	NoR	RT (ms)	NoR	RT (ms)
Get all Regular Services (case B1)	1	11	64	9	1541	0	81	20	1513
	5	19	76	16	1880	1	944	36	2140
	10	25	65	22	3220	1	646	48	3310
	50	125	186	430	322	1	127	556	440
	100	231	197	4385	12900	2086	12425	4621	12760

Get Transfer-ServiceAndRent-ing service (case B2)	1	0	61	0	1705	0	117	0	1813
	5	0	62	0	1816	0	1326	0	1770
	10	0	62	0	3290	0	688	0	3090
	50	11	66	57	993	0	160	68	1103
	100	11	75	3102	13820	2081	12790	3113	13980

8 Conclusions

This paper presented a solution for federating Smart CityAPIs that provide data and services. The solution aims to satisfy requirements that presently cannot be met by traditional solutions based on GIS or IoT Brokers. The proposed solution, while avoids migrating data, provides federation at level of APIs, involves nodes of any size (in terms of geo area and number of content/entities), and combines them autonomously so that each of them may (not) decide to join an area/content. It also leaves the possibility of having different kind of services, enables the movements among federated areas, respects privacies according to GDPR, and combines services with IoT Brokers NGSI of FiWare and GIS via WFS/WMS, by using Snap4City tools. Moreover, the proposed solution provides scalable performance in data access even in the case of private devices, etc. It finally allows the creation of separate clusters of federated APIs and standalone solutions. The main results include the formalization of decisions for the propagation of queries, optimizing the composition of results in an efficient manner, the possibility of federating Snap4City solutions with native FiWare solutions based on IoT Orion Broker, the assessment of performance for the federated solution, and access private and public IoT devices. To design, implementation, and validation of the solution, we enhanced the former Km4City Smart City API and ontology to improve the semantic queries that can be overlapped among different areas. We also developed an adapter from/to NGSI V2 of IoT Brokers FiWare and Smart City API. The validation has shown that the solution is scalable and viable in terms of performance by removing the duplications at a reasonable expense, for urban and rural areas, even in presence of a large number of duplications. When the range of the query is very large (e.g., $50 - 100\ (km)$), the number of resulting objects is high, while the response time of federated solution is still comparable with respect to the response time of the service providing the majority of the results.

Future work on this research line should address the new emerging protocol of NGSI-LD that is going to bring semantic descriptions into the messages. It is presently based on a non-consolidated but still interesting set of data models. Therefore, the connection of FiWare brokers based on NGSI-LD could allow to extend some of the semantic reasoning also to those nodes.

ACKNOWLEDGMENTS

The authors would like to thank the European Union's Horizon 2020 research and innovation program for funding the Select4Cities PCP project (supported within the Snap4City framework) under the grant agreement No. 688196, and all the companies

and partners involved. Snap4City and Km4City are 100% open-source technologies and the platform of DISIT Lab can be accessed at https://www.snap4city.org.

REFRENCES

1. Hernández-Muñoz, J.M., Vercher, J.B., Muñoz, L., Galache, J.A., Presser, M., Hernández Gómez, L.A., Pettersson, J.: Smart Cities at the Forefront of the Future Internet. In: Domingue, J., Galis, A., Gavras, A., Zahariadis, T., Lambert, D., Cleary, F., Daras, P., Krco, S., Müller, H., Li, M.-S., Schaffers, H., Lotz, V., Alvarez, F., Stiller, B., Karnouskos, S., Avessta, S., and Nilsson, M. (eds.) The Future Internet. pp. 447–462. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20898-0_32.
2. Cinquini, L., Crichton, D., Mattmann, C., Harney, J., Shipman, G., Wang, F., Ananthakrishnan, R., Miller, N., Denvil, S., Morgan, M., Pobre, Z., Bell, G.M., Doutriaux, C., Drach, R., Williams, D., Kershaw, P., Pascoe, S., Gonzalez, E., Fiore, S., Schweitzer, R.: The Earth System Grid Federation: An Open Infrastructure for Access to Distributed Geospatial Data. Future Generation Computer Systems. 36, 400–417 (2014). https://doi.org/10.1016/j.future.2013.07.002.
3. Herrera-Cubides, J.F., Gaona-García, P.A., Gordillo Orjuela, K.: A View of the Web of Data. Case Study: Use of Services CKAN. Ingeniería. 22, 46–64 (2017). https://doi.org/10.14483/udistrital.jour.reving.2017.1.a07.
4. IoT Orion Broker FiWare with Persistence, https://github.com/FIWARE/tutorials.Time-Series-Data, last accessed 2022/02/25.
5. Badii, C., Bellini, P., Cenni, D., Difino, A., Nesi, P., Paolucci, M.: Analysis and assessment of a Knowledge based Smart City Architecture Providing Service APIs. Future Generation Computer Systems. 75, 14–29 (2017). https://doi.org/10.1016/j.future.2017.05.001.
6. Zuccalà, M., Verga, E.S.: Enabling Energy Smart Cities through Urban Sharing Ecosystems. Energy Procedia. 111, 826–835 (2017). https://doi.org/10.1016/j.egypro.2017.03.245.
7. Krylovskiy, A., Jahn, M., Patti, E.: Designing a Smart City Internet of Things Platform with Microservice Architecture. In: 2015 3rd International Conference on Future Internet of Things and Cloud. pp. 25–30 (2015). https://doi.org/10.1109/FiCloud.2015.55.
8. Arman, A., Bellini, P., Nesi, P., Paolucci, M.: Analyzing Public Transportation Offer wrt Mobility Demand. In: Proceedings of the 1st ACM International Workshop on Technology Enablers and Innovative Applications for Smart Cities and Communities. pp. 30–37 (2019).
9. Badii, C., Belay, E.G., Bellini, P., Cenni, d, Marazzini, M., Mesiti, M., Nesi, P., Pantaleo, G., Paolucci, M., Valtolina, S., Soderi, M., Zaza, I.: Snap4City: A Scalable IOT/IOE Platform for Developing Smart City Applications. In: 2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). pp. 2109–2116 (2018). https://doi.org/10.1109/SmartWorld.2018.00353.
10. Ames, D.P., Horsburgh, J.S., Cao, Y., Kadlec, J., Whiteaker, T., Valentine, D.: HydroDesktop: Web services-based Software for Hydrologic Data Discovery, download, visualization, and analysis. Environmental Modelling & Software. 37, 146–156 (2012). https://doi.org/10.1016/j.envsoft.2012.03.013.
11. Kolozali, Ş., Bermudez-Edo, M., Farajidavar, N., Barnaghi, P., Gao, F., Intizar Ali, M., Mileo, A., Fischer, M., Iggena, T., Kuemper, D., Tonjes, R.: Observing the Pulse of a City: A Smart City Framework for Real-Time Discovery, Federation, and Aggregation of Data Streams. IEEE Internet of Things Journal. 6, 2651–2668 (2019). https://doi.org/10.1109/JIOT.2018.2872606.

12. González-Zamar, M.-D., Abad-Segura, E., Vázquez-Cano, E., López-Meneses, E.: IoT Technology Applications-Based Smart Cities: Research Analysis. Electronics. 9, 1246 (2020). https://doi.org/10.3390/electronics9081246.
13. Anttiroiko, A.-V., Valkama, P., Bailey, S.J.: Smart Cities in the new Service Economy: Building Platforms for Smart Services. AI & society. 29, 323–334 (2014).
14. Vitolo, C., Elkhatib, Y., Reusser, D., Macleod, C.J.A., Buytaert, W.: Web technologies for Environmental Big Data. Environmental Modelling & Software. 63, 185–198 (2015). https://doi.org/10.1016/j.envsoft.2014.10.007.
15. Latre, S., Leroux, P., Coenen, T., Braem, B., Ballon, P., Demeester, P.: City of Things: An Integrated and Multi-technology Testbed for IoT Smart City Experiments. In: 2016 IEEE International Smart Cities Conference (ISC2). pp. 1–8 (2016). https://doi.org/10.1109/ISC2.2016.7580875.
16. Salhofer, P., Buchsbaum, J., Janusch, M.: Building a Fiware Smart City Platform. In: Proceedings of the 52nd Hawaii International Conference on System Sciences (2019).
17. Nesi, P., Badii, C., Bellini, P., Cenni, D., Martelli, G., Paolucci, M.: Km4City Smart City API: An Integrated Support for Mobility Services. In: 2016 IEEE International Conference on Smart Computing (SMARTCOMP). pp. 1–8 (2016). https://doi.org/10.1109/SMARTCOMP.2016.7501702.
18. Namiot, D., Sneps-Sneppe, M.: On Software Standards for Smart Cities: API or DPI. In: Proceedings of the 2014 ITU kaleidoscope academic conference: Living in a converged world - Impossible without standards? pp. 169–174 (2014). https://doi.org/10.1109/Kaleidoscope.2014.6858494.
19. Bellini, P., Nesi, D., Nesi, P., Soderi, M.: Federation of Smart City Services via APIs. In: 2020 IEEE International Conference on Smart Computing (SMARTCOMP). pp. 356–361 (2020). https://doi.org/10.1109/SMARTCOMP50058.2020.00077.
20. Soto, J.Á.C., Werner-Kytölä, O., Jahn, M., Pullmann, J., Bonino, D., Pastrone, C., Spirito, M.: Towards a Federation of Smart City Services. In: International Conference on Recent Advances in Computer Systems. pp. 163–168. Atlantis Press (2015).
21. Bonino, D., Alizo, M.T.D., Alapetite, A., Gilbert, T., Axling, M., Udsen, H., Soto, J.A.C., Spirito, M.: ALMANAC: Internet of Things for Smart Cities. In: 2015 3rd International Conference on Future Internet of Things and Cloud. pp. 309–316 (2015). https://doi.org/10.1109/FiCloud.2015.32.
22. Badii, C., Bellini, P., Difino, A., Nesi, P.: Smart City IoT Platform Respecting GDPR Privacy and Security Aspects. IEEE Access. 8, 23601–23623 (2020). https://doi.org/10.1109/ACCESS.2020.2968741.
23. Km4City ontology, https://www.snap4city.org/drupal/node/19, https://www.snap4city.org/download/video/DISIT-km4city-City-Ontology-eng-v5-1.pdf, last accessed 2022/02/25.
24. Bellini, P., Benigni, M., Billero, R., Nesi, P., Rauch, N.: Km4City Ontology Building vs Data Harvesting and Cleaning for Smart-city Services. Journal of Visual Languages & Computing. 25, 827–839 (2014). https://doi.org/10.1016/j.jvlc.2014.10.023.
25. Arman, A., Bellini, P., Bologna, D., Nesi, P., Pantaleo, G., Paolucci, M.: Automating IoT Data Ingestion Enabling Visual Representation. Sensors. 21, 8429 (2021). https://doi.org/10.3390/s21248429.
26. doc: RFC 6454: The Web Origin Concept, https://www.hjp.at/doc/rfc/rfc6454.html, last accessed 2022/02/25.