IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 15, 2021, accepted May 15, 2021, date of publication May 20, 2021, date of current version June 1, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3082182

Open Data Consumption Through the Generation of Disposable Web APIs

PALOMA CÁCERES GARCÍA DE MARINA[1], JOSÉ MARÍA CAVERO BARCA[1], CARLOS E. CUESTA[1], MIGUEL ÁNGEL GARRIDO[1], IRENE GARRIGÓS[2], CÉSAR GONZÁLEZ-MORA[2], JOSE-NORBERTO MAZÓN[2], ALMUDENA SIERRA-ALONSO[1], BELÉN VELA[1], AND JOSÉ JACOBO ZUBCOFF[2]

[1]Department of Computer Science and Statistics, Rey Juan Carlos University, 28933 Madrid, Spain
[2]Department of Software and Computing Systems, University of Alicante, 03690 Alicante, Spain

Corresponding author: Irene Garrigós (igarrigos@ua.es)

This work was supported in part by the Access@City coordinated Research Project through the Spanish Ministry of Science, Innovation and Universities under Grant TIN2016-78103-C2-1-R and Grant TIN2016-78103-C2-2-R; in part by the Plataforma intensiva en datos proveedora de servicios inteligentes de movilidad (MoviDA) Project through Rey Juan Carlos University; and in part by the Recolección y publicación de datos abiertos para la reactivación del sector turístico postCOVID-19 (UAPOSTCOVID19-10) Project through the Consejo Social of the University of Alicante. The work of César González-Mora was supported in part by the Generalitat Valenciana, and in part by the European Social Fund under Grant ACIF/2019/044.

ABSTRACT The ever-growing amount of information in today's world has led to the publication of more and more open data, i.e., that which is available in a free and reusable manner, on the Web. Open data is considered highly valuable in situational scenarios, in which thematic data is required for a short life cycle by a small group of consumers with specific needs. In this context, data consumers (developers or data scientists) need mechanisms with which to easily assess whether the data is adequate for their purpose. SPARQL endpoints have become very useful for the consumption of open data, but we argue that its steep learning curve hampers open data reuse in situational scenarios. In order to overcome this pitfall, in this paper, we coin the term disposable Web APIs as an alternative mechanism for the consumption of open data in situational scenarios. Disposable Web APIs are created on-the-fly to be used temporarily by a user to consume open data. In this paper we specifically describe an approach with which to leverage semantic information from data sources so as to automatically generate easy-to-use disposable Web APIs that can be used to access open data in a situational scenario, thus avoiding the complexity and learning curve of SPARQL and the effort of manually processing the data. We have conducted several experiments to discover whether non-experienced users find it easier to use our disposable Web API or a SPARQL endpoint to access open data. The results of the experiments led us to conclude that, in a situational scenario, it is easier and faster to use the Web API than the corresponding SPARQL endpoint in order to consume open data.

INDEX TERMS Disposable Web APIs, open data, semantic annotation, SPARQL.

I. INTRODUCTION

The volume of data on the Web has, with the advent of the open data movement, exploded in the last few years. Open data is published in order to be freely accessible and reusable (without copyright restrictions) and is, therefore, usually considered highly valuable as situational data. Situational data [1] has a narrow focus on a specific area and, often, a short lifespan so as to add value to data owned (and controlled) by a small group of consumers with a unique set of needs. Examples of this might be data scientists who wish to analyze a company's sales with respect to weather conditions within a period of time, or a Web developer who needs to implement a prototype of an envisioned smart city app that employs traffic data.

There are roughly two ways in which to publish open data: (i) open data portals that focus on published tabular-form data (such as CSV files), and (ii) Linked Open Data that allow users to use Semantic Web technologies to access data on the Web in the same way as a database management system is used, i.e., by means of query languages such as SPARQL [2]. The current goal of any open data project is

The associate editor coordinating the review of this manuscript and approving it for publication was M. Anwar Hossain.

to provide a SPARQL endpoint that will enable the powerful and seamless querying of data. However, average open data consumers (such as data scientists or developers) lack the skills required to smoothly use the SPARQL query language and the underlying RDF (Resource Description Framework) data model, since they both have a steep learning curve. Moreover, recent surveys [4] conclude that no usable tool that could be used to support the whole Linked Data consumption process currently exists. Consequently, in a situational scenario, alternative mechanisms that will allow powerful open data consumption without any knowledge of Semantic Web technologies are, therefore, required [3], [5].

Our hypothesis is that, in situational scenarios, it is easier to have a Web API access to open data than a SPARQL endpoint. These Web APIs must specifically allow data consumers to access data easily and rapidly without wasting time processing tabular data sources (from CSV files or from Web sites) or learning how to query a SPARQL endpoint. They have been denominated as disposable Web APIs because they are created on-the-fly to be used temporarily by a user to consume open data as situational data. Moreover, the use of a disposable Web API allows a data consumer (data scientist or developer) to easily assess whether the data is adequate for his/her purpose, thus avoiding the complexity and learning curve of SPARQL and the effort of manually processing the data. However, developing disposable Web APIs requires to enrich tabular data sources with data semantics in the sense of the ''Model for Tabular Data and Metadata on the Web'' developed by the W3C CSV on the Web Working Group[i1] which proposes to add semantic annotations by means of separated metadata file to supplement tabular data. In this paper we, therefore, describe an approach with which to leverage semantic information from data sources so as to automatically generate easy-to-use disposable Web APIs that can be used to access open data in a situational scenario. Our specific contributions are the following:

- A process for the semantic annotation of tabular data sources from the Web.
- A model-driven approach that can be employed to ingest our semantic annotation and generate disposable Web APIs with which to query data sources.
- A controlled experiment carried out in order to compare the process of accessing data from the SPARQL endpoints with that of using our equivalent generated disposable Web APIs.

Throughout this paper, we use a real situational case study (i.e., a running example) based on data obtained from the public transport sector in Madrid (Spain), namely Metro Madrid data, in order to exemplify our approach. A data scientist working at a real estate company was willing to analyze the company's internal data regarding housing rental and incomes in Madrid by considering additional open data related to the accessibility of public transport, which helps create a more inclusive society that provides the same opportunities for all [11]. This would, therefore, allow the real estate company to attain new insights and make informed decisions with which to, e.g., provide adapted solutions to those of its customers with special mobility needs during a short-term marketing campaign. A disposable Web API with which to consume public transport accessibility data for Madrid was, therefore, required.

The remainder of the paper is as follows: Section II presents some related work, while Section III describes the process of automatically generating disposable Web APIs using semantic-annotated open data sources. Section IV provides a validation of the proposal by describing an experiment in which data consumption from a Web API and the use of SPARQL were compared, and finally, Section V shows a summary of our main conclusions and future work.

II. RELATED WORK

This section reviews some research works related to how Web APIs are useful as regards providing easier data consumption.

The starting point for our current work was the research described in [11], which was focused on adding semantics to existing data. In the present paper, however, we have used this step as a part of the whole process carried out to make it easier for users to access open data on the Web.

In [6], a simple API, which is used to provide access to the KnowledgeStore, a semantic web storage, is described. It is used by many developers that are unfamiliar with RDF and SPARQL technologies to build web applications that use this data. The developers have to compose a query using simple API methods, which is then converted into a SPARQL query so as to access the endpoint and obtain the data required. However, this API was created manually and is used only to access specific data from specific semantic web storage.

The authors of [19], [20] propose an automatic query-centric API for routine access to Linked Data. In these papers, they present and extend ''grlc'', a generic Linked Data gateway. The papers propose the generation of APIs that provide uniform API access to any Linked Data published in SPARQL endpoints, Linked Data Fragments servers, RDF dumps, or RDFa embedded in HTML pages. Unfortunately, a Github repository containing all possible SPARQL queries to the endpoint is required, signifying that a developer must manually write down all possible queries that might be unfamiliar with RDF and SPARQL queries.

Rather than using SPARQL queries, users can also use GraphQL-LD [21] queries to access RDF data, which are GraphQL queries enhanced with a JSON-LD context. This GraphQL-LD approach is a developer-friendly alternative to SPARQL, but only for experts in GraphQL APIs, signifying that developers must be familiar with this programming language. Instead, users employing our approach are provided with an easy-to-use standard API based on OpenAPI principles.

An approach to help Web developers access Linked data is proposed in [22]. The objective is to transform ontologies into

[1] https://www.w3.org/TR/tabular-data-model/

Ontology-Based APIs such that developers can easily query Linked Data. However, our approach is directly addressed toward developers who wish to access open data available on the Web, signifying that the starting point is not an ontology or an SPARQL endpoint from the Linked Data cloud.

In [7], a RESTful API with which to semantically augment the data that is being published by a sensor is proposed, but it is not useful as regards retrieving data from the semantic web. The authors have manually created a system that transforms a set of inter-linked JSON documents into an RDF graph in order to obtain a semantically-enriched dataset and have the full query capability provided by SPARQL. However, an API that facilitates access to the semantic data without advanced knowledge of the SPARQL query language is still required.

The authors of [8] describe how cities are starting to release their public information and create public data catalogues, but there is still a lack of open APIs, which is restricting the full potential of the open data. The paper in question therefore proposes the development of an open data API, which provides tools that will help users to access the data concerning their own cities.

In [9], a framework called Prov4J is described, which uses Semantic Web tools and standards. It provides to developers a provenance management mechanism with which to build provenance-aware applications. Tracking the origin of information currently plays a fundamental role on the Web because it enables users to determine the suitability and quality of the data. The Prov4J framework includes a Client API that contains key interface methods for a set of provenance queries, thus minimizing the users' interactions with SPARQL. However, this API was created manually for data provenance purposes only.

Finally, mapping strategies have been investigated to a significant extent, and work researching conversion from other formats to RDF has also been carried out for static [23] and live conversion [24]. Other solutions [3], [5] aim to reduce the difficulty involved in using SPARQL and RDF technologies. However, they still lack the support specifically required by data scientists and open data reusers to overcome the difficulty of accessing RDF data using SPARQL. According to experts (as detailed in [3]), the problem does not reside in RDF, and these solutions may not be suitable for non-experts in this Semantic Web environment. It is even more necessary in a situational scenario, where data consumers need to temporarily access data easily and rapidly without spending too much time learning how to query the SPARQL endpoint.

In our approach we, therefore, provide an alternative through the use of standard and easy-to-query APIs rather than SPARQL endpoints that access RDF data.

In summary, there is much research on the creation of Web APIs with which to access data, but there is, to the best of our knowledge, no solution that tackles situational scenarios in which thematic open data is temporarily used for specific needs.

III. OPEN DATA CONSUMPTION PROCESS FOR SITUATIONAL SCENARIOS

The proposed process for open data consumption through the use of automatically generated disposable Web APIs comprises three stages:

1) **Data Input Process:** in this stage, it is necessary to select the Web data from either open data sources or data embedded in the HTML code of a Web site (specifying the Web data source URL), required to satisfy data consumer needs according to a situational scenario. This data must be processed (e.g., by programming a Python script in order to obtain the data by means of web scraping techniques). We would like to point out that we consider, as input of this process, Web data sources with no semantic information.

2) **Semantic Annotation Process:** the input tabular data (e.g. a CSV file) is annotated using a domain-specific ontology, obtaining semantically annotated data. The output of this process is a CSV file with semantic annotations as RDF triples.

3) **Disposable Web APIs Generation Process:** the input of this process is the previously semantically annotated tabular data, from which disposable Web APIs are automatically generated in order to allow the consumption of data in a situational scenario.

The complete open data consumption process is shown in Fig. 1, including the inputs and outputs of each process.

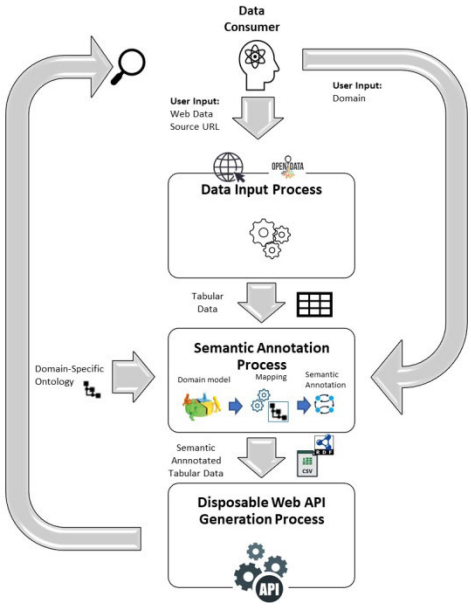

FIGURE 1. Open data consumption process for situational scenarios.

The Semantic Annotation of the Structured Open Data Process (subsection III.A) and the Automatic Generation of Disposable Web APIs (subsection III.B) are described in the following subsections.

A. SEMANTIC ANNOTATION OF STRUCTURED OPEN DATA

In this section, we describe the systematic process employed to generate a semantically annotated dataset from the

structured open data on the Web. This method makes it possible to enrich data by adding new semantics.

The process entails the following steps:

Step 1. First, information contained in the data source and important features of the selected domain must be studied. It is then necessary to create a glossary of terms concerning the terminology used in the data source, describing each one, after which a domain model with which to represent these terms and their relationships must be defined in a domain model by using a UML class diagram.

Step 2. Identify the semantics of the information in order to align them with the terms of a reference vocabulary, that is, a vocabulary that the developer needs to choose (e.g. domain-specific ontology). At this point, when necessary, auxiliary vocabularies can be chosen so as to cover additional properties. The mappings required in order to establish the correspondence between different information elements from the data source (glossary of terms) and the reference vocabulary (domain-specific ontology) are defined by means of a manual inspection of the input data source. We specifically identify the subjects, predicates and objects of a CSV input file and map them onto elements from the reference vocabulary (e.g. a domain-specific ontology, such as MAnto). This mapping is manually defined and added to a configuration file. Once all the mappings have been defined, a custom script is programmed in order to transform the input tabular source data (CSV file) into the semantically annotated data, according to the mappings defined in the configuration file. This data annotation process is described in a detailed manner by means of a case study shown in our previous work [11].

Step 3. If the alignment is possible, then it is necessary to use the original data, and semantically annotate them using the existing terms from the reference vocabulary (domain-specific ontology) by means of the RDF language using the previously defined mappings. If the alignment is not possible, then we should analyze the difference between the original data and the reference vocabulary in order to extend it and, in this case, return to the second step.

Step 4. Integrate the semantic dataset with other sources. This is done by relating elements from the different semantic datasets and discarding duplicated information in the datasets, when necessary. At this point, it is also possible to add new properties to the final annotated dataset. Finally, a single annotated dataset is obtained as the output of Step 4. It is worth mentioning, in the case of integrating with non-semantic data sources, that the semantic annotation process from step 1 to step 3 has to be performed beforehand.

This generic process has been applied to our case study about the public transport domain, and specifically to Metro Madrid.

During the **Data Input Process**, we first selected the open datasets available in the Madrid Public Transport open data portal. We then developed a Python script in order to obtain the Metro Madrid accessibility data by means of web scraping techniques, thus completing our input dataset.

We subsequently semantically annotated the real Metro Madrid public transport company data (by applying aforementioned steps 1, 2 and 3); in this case, there is no previously annotated data and we, therefore, omit step 4.

An explanation of each of the steps in the Semantic Annotation Process applied to our case study is provided as follows.

In **Step 1** of the semantic annotation of structured open data, we created a glossary of terms concerning the terminology used in the data source, describing each one. We then defined the domain model using a UML class diagram with which to represent these terms and their relationships (see Fig. 2).

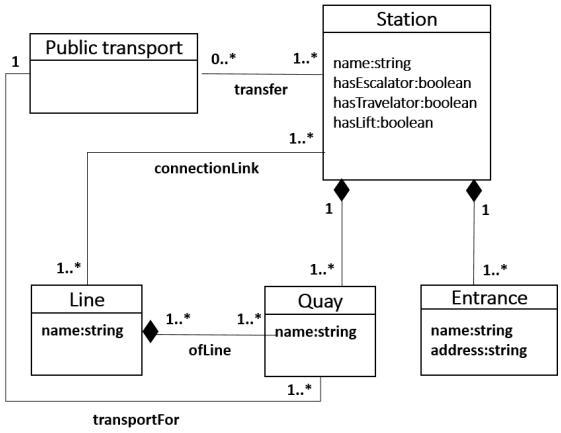

FIGURE 2. Domain model of metro Madrid.

In **Step 2**, we first choose our reference vocabulary, which is denominated as MAnto [12], [13]. MAnto[2] is a light ontology based on the Identification of Fixed Objects in the Public Transport (IFOPT) reference data model (CEN/TC278 2012)[14], included in the European Reference Data Model for Public Transport Information (Transmodel) [15]. The reason for using the MAnto ontology as a reference vocabulary in our case study is twofold: (i) first, we wished to describe a real scenario, and what is more important, (ii) we required a well-known ontology with which to conduct the experiments in a real setting. We then established the mappings between the elements of the Metro Madrid and the chosen reference vocabulary, MAnto. For example, we manually identify Sol as a station (mao:StopPlace subject) and 1, 2 and 3 as lines (mao:Line objects), and there are three connection links (mao:hasConnectionLink properties) that relate the subject (Sol) with the objects (lines 1,2 and 3).

In **Step 3**, we annotated the data semantically using the terms from our reference vocabulary (MAnto ontology) as follows: if the alignment is possible: (a) we first extract the data from the data source by means of a Python scraper; (b) we semantically annotate them (for example, each quay belonging to Line 1 of Metro Madrid has been annotated as `?quay mao:ofLine "1"`) and (c) we reuse terms from

[2]https://github.com/vortic3/LinkedUnifiedDataset/blob/master/MAnto_Lite_ontology.rdf

other vocabularies only when necessary (for example, if the name term exists in the `schema` ontology, we do not create the same term in MAnto -mao:name metadata-, but use that which already exists: in the RDF triple "`43566 sch:name Sol`", we assign the name of Sol to station 43566 using a metadata property that already exists in the schema ontology).

The output of the Semantic Annotation Process will be a CSV file with semantic annotations as RDF triples, which will be used as input to the Disposable Web APIs Generation Process.

B. AUTOMATIC GENERATION OF DISPOSABLE WEB APIs

In this section, a complete model-based transformation process is proposed in order to achieve the automatic generation of a disposable Web API from the previously semantic-annotated data sources (an overview of this process is shown in Fig. 3). The transformation process includes (apart from the RDF annotation using the semantic engine, as explained above), the following steps: (a) a text to model (T2M) transformation from the annotated data source to a specific data model (specific to the source format) that we have defined specifically for tabular data in CSV format; (b) a model to model (M2M) transformation from this data model to an OpenAPI model that we have also defined; (c) a model to text (M2T) transformation from the OpenAPI model to its OpenAPI specification, and finally, (d) a text to text (T2T) transformation from the OpenAPI specification to the Web API that makes it possible to access open data.

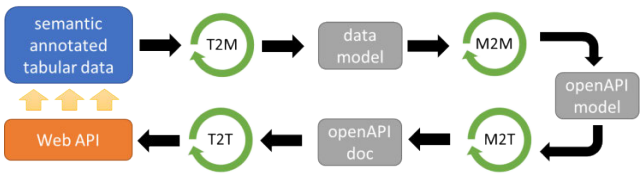

FIGURE 3. Automatic API generation process.

1) FROM DATA SOURCE TO DATA MODEL (T2M)

The first stage of the transformation process is a text to model (T2M) transformation. The data source is converted into the data model in order to represent the data and proceed with the model-based transformation approach. This data model is based on a metamodel based on MOF0F[3] (shown in Fig. 4) which is implemented in the Ecore format from the Eclipse Modeling Framework (EMF).

The input semantic annotated tabular data in our case study consists of: (a) the Madrid metro accessibility data source, which is a CSV file, and (b) the semantic annotations as RDF triples. Our approach processes the CSV by rows and columns, injecting the semantic information into the columns names. The data is analyzed, and a data model with table, row, and cell objects is created. This generated model (as shown in Fig. 5) contains an object 'Table', in which there is a set

[3]https://www.omg.org/mof/

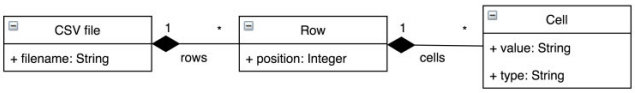

FIGURE 4. CSV datafile metamodel.

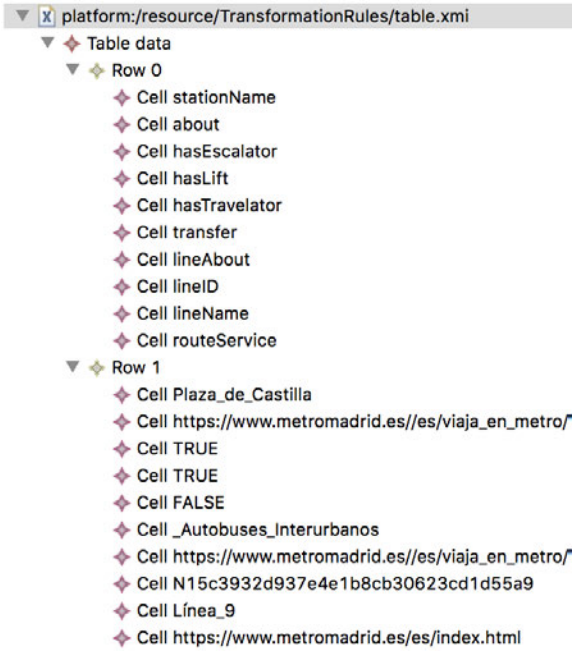

FIGURE 5. Datafile model in XMI format.

of 'Rows' containing many 'Cells'. Each cell in the model contains the information regarding each cell from the CSV file, which is the information concerning metro stations and their accessibility.

2) FROM DATA MODEL TO OpenAPI MODEL (M2M) AND SPECIFICATION (M2T)

Once the data model has been populated from the data file, the second stage involves a model to model (M2M) transformation from the CSV data model to the OpenAPI model, followed by a model to text (M2T) transformation between the OpenAPI model and the OpenAPI specification.

First, an M2M transformation defined in ATL language [16]–[18] is launched. ATL is one of the most widely used model transformation languages currently employed, and is backed by a mature and efficient execution runtime. A set of transformation rules between the data model and the OpenAPI model has, therefore, been defined using the ATL language, as shown in the extract of the code in Fig. 6. The ATL transformation rules start from the Table object defined in its Ecore metamodel, and its rows and cells are used to generate the OpenAPI model and all the different objects contained in its metamodel.

The OpenAPI model generated is in XMI format, as shown in Fig. 7.

It is based on its OpenAPI metamodel in Ecore format (Fig. 8), which has been created by updating an existing

```
-- @atlcompiler emftvm
-- @path Table=/TransformationRules/Table.ecore
-- @path Openapi=/TransformationRules/Openapi.ecore
module Table2Openapi;

create OUT: Openapi from IN: Table;
rule Main {
    from
        s: Table!Table
    using {
            firstTableRow : Sequence(Table!Cell) = s.rows->first().cells;
            lastTableRow : Sequence(Table!Cell) = s.rows->last().cells;
    }
    to
        t: Openapi!API (
            openapi <- '3.0.0',
            info <- openapi_info,
            servers <- Sequence{openapi_servers},
            paths <- Sequence{openapi_basic_path}.union
                (firstTableRow -> collect(e | thisModule.OpenapiPaths(e, s))),
            components <- Sequence{openapi_components}
        ),
        openapi_info: Openapi!Info (
            title <- s.filename,
            version <- '1.0.0',
            description <- 'Obtaining the ' + s.filename
        ),
        openapi_servers: Openapi!Server (
            url <- 'http://www.urlprueba.com/v1'
        ),
```

FIGURE 6. ATL transformation rules.

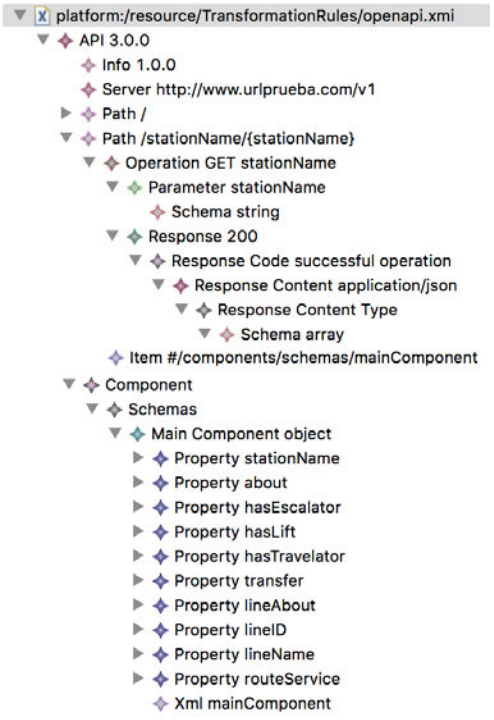

FIGURE 7. OpenAPI model (XMI).

OpenAPI metamodel2F[4] from Swagger 2.0 to OpenAPI 3.0 specification.

The definition and documentation of the Web API are represented by a JSON file (Fig. 9) in accordance with the Swagger standards3F,[5] because this helps design, build, document and test the API. The API specification JSON file is, therefore, directly generated from the OpenAPI model in XMI format by using a model to text (M2T) transformation. It consists of a straightforward transformation, since the

[4]https://github.com/SOM-Research/APIDiscoverer/tree/master/metamodel
[5]https://swagger.io

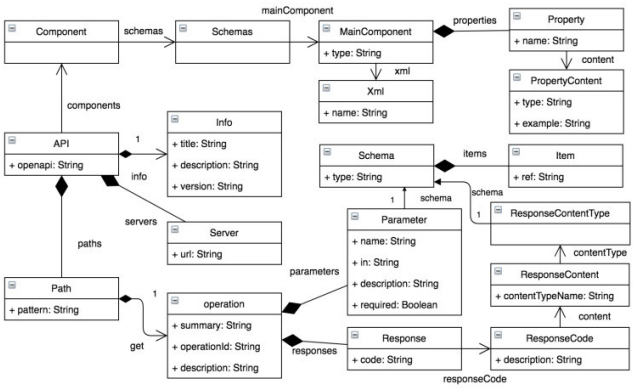

FIGURE 8. OpenAPI metamodel.

```
"openapi":"3.0.0",
"info":{
   "description":"Obtaining the metro stops",
   "version":"1.0.0",
   "title":"stopsmetro"
},
"servers":[
   "https://wake.dlsi.ua.es/madrid/metro"
],
"paths":{
   "/":{ },
   "/stationName/{stationName}":{ },
   "/about/{about}":{ },
   "/hasEscalator/{hasEscalator}":{ },
   "/hasLift/{hasLift}":{ },
   "/hasTravelator/{hasTravelator}":{ },
   "/lineAbout/{lineAbout}":{ },
   "/lineID/{lineID}":{ },
   "/lineName/{lineName}":{ },
   "/routeService/{routeService}":{ },
   "/transfer/{transfer}":{ }
}
```

FIGURE 9. Extract of OpenAPI JSON file.

OpenAPI model has elements that are equivalent to the API specification, but in a different structure because a different format is used. We, therefore, perform this M2T transformation programmatically from the OpenAPI model in XMI format to the OpenAPI specification of the API, which is in JSON format. It is easy to perform the transformation because we have constructed the OpenAPI model according to the OpenAPI specification, such that each object from the model corresponds to a JSON object in the file. For example, each path of the API is transformed from the "Path" object in the OpenAPI model to the "paths" object in the JSON file (e.g., Path /stationName/{stationName} in Figure 7 is transformed in the element "/stationName/{stationName}" from attribute path in JSON file of Figure 8). The complete API specification in Swagger 2.0 format is available online.[6]

3) FROM OpenAPI SPECIFICATION TO WEB API

In this stage, the complete Web API is generated from its OpenAPI specification. The automatic process creates the

[6]https://wake.dlsi.ua.es/madrid/api-docs

API code represented by a server in NodeJS4F,[7] a simple and efficient runtime environment for network applications. The automatic generation process is accomplished with the help of the Swagger Codegen tool, which creates the structure of the server with the files and folders required. It also manages the calls to the API and redirects them to the corresponding method in NodeJS code. The automatic generator then completes the server with the features required to return the data requested, which are retrieved from the data source.

This generated Web API has been published in an online server, thus allowing open data reusers to query the data with the parameters desired to filter the information. The queries to the Web API are specified in the OpenAPI documentation, whose simplicity makes it possible for them to be executed by non-experts in query languages in a situational scenario. The API available online contains a method with which to query the metro stops filtered by many parameters. When a user queries this information, the list of metro stops is returned, including the stops that fulfill the specified query parameters. For instance, a query that requests the metro stops that are be accessible via a lift and escalator is: https://wake.dlsi.ua.es/madrid/metro/?hasLift=TRUE&hasEscalator=TRUE

The Web API will respond to this request with the required information in JSON format, because it is easy for humans to read and write and it is easy for machines to parse and generate5F.[8] The result obtained from the query example is the data concerning metro stops that are accessible by lift and escalator. In this case, many stations fulfill these conditions and we consequently show only the example of the ''Islas Filipinas'' station. The extract of the API output in JSON format, therefore, contains information regarding the stations filtered as:

```
{
    "stationName": " Islas Filipina ",
    "about":
 "https://www.metromadrid.es//es/
viaja_en_metro/red_de_metro/estaciones/
IslasFilipinas.html",
    "hasEscalator": "TRUE",
    "hasLift": "TRUE",
    "hasTravelator": "FALSE",
    "transfer": "",
"lineAbout":
 "https://www.metromadrid.es//es/
viaja_en_metro/red_de_metro/lineas_y
_horarios/linea07.html",
    "lineID": "N0d6e5c75dec24ce9a67f74
     c725894bd0",
    "lineName": "Línea 7",
    "routeService": "https://www.
     metromadrid.es/es/index.html"}
```

[7]https://nodejs.org
[8]https://www.json.org/

IV. VALIDATION

In order to validate our approach, a controlled experiment was conducted in which we compared two means of consuming open data: (i) by using the SPARQL access point of RDFs or (ii) by using our equivalent generated disposable Web API.

A. DESCRIPTION OF THE EXPERIMENT

The objective of the experiment was to study whether there were any differences as regards consuming data via the Web API vs. via SPARQL queries. In order to conduct the experiment we, therefore, planned to carry out a set of surveys (concerning Web APIs and SPARQL) with Data Science Master's Degree students at the Universidad Rey Juan Carlos (Madrid, Spain). One group of 15 users was instructed in the use of APIs, while the other group of 17 students was instructed in the use of SPARQL, such that each of the groups would answer the surveys by using the API or SPARQL, respectively. It is worth noting that all the participants had similar previous knowledge of the technologies used in the experiment, and they were also simultaneously instructed in the use of APIs and SPARQL. Each survey, i.e., that concerning the use of the Web API and the other concerning the use of SPARQL, had to be answered by the participants using the correct queries in each case.

The surveys were composed of five data requests (using a SPARQL endpoint of the corresponding Web API), i.e., the participants were given five queries and requested to find the right data. We also made a note of the time it took to answer each question, in addition to whether the query was correct and the number of attempts made until the correct query was attained. These variables helped us assess how difficult it was for the users to obtain the correct data and whether the Web API access (generated by using the approach proposed in this paper) is better than the use of the corresponding SPARQL endpoint.

The specific queries were related to the Metro Madrid transport data (i.e., the running example used throughout this paper):

1. Set of stations on a given subway line (Line 1).
2. Subway lines that pass through a certain station (''Canal'').
3. Stations that have correspondence with other means of transportation.
4. Correspondence of a station (''Plaza de Castilla'') with a specific means of transport (''Cercanías Renfe'').
5. Determining whether a given station had an escalator, elevator or moving walkway.

All of these questions were marked as having a basic level of complexity (considered as simple), with the exception of the last (considered as complex), because, unlike the others, it was necessary to use the SPARQL FILTER command to correctly perform the query. These simple and complex queries allowed us to assess whether our approach for generating Web APIs improves SPARQL data access in any particular way.

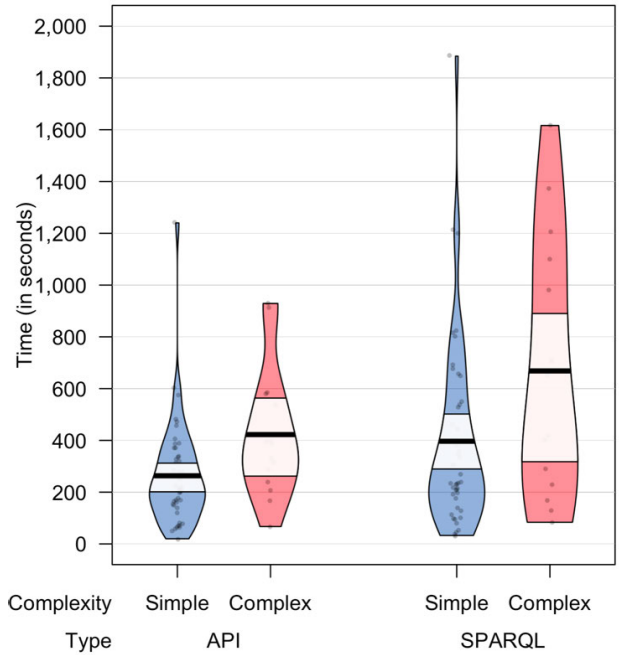

FIGURE 10. Time by type and complexity.

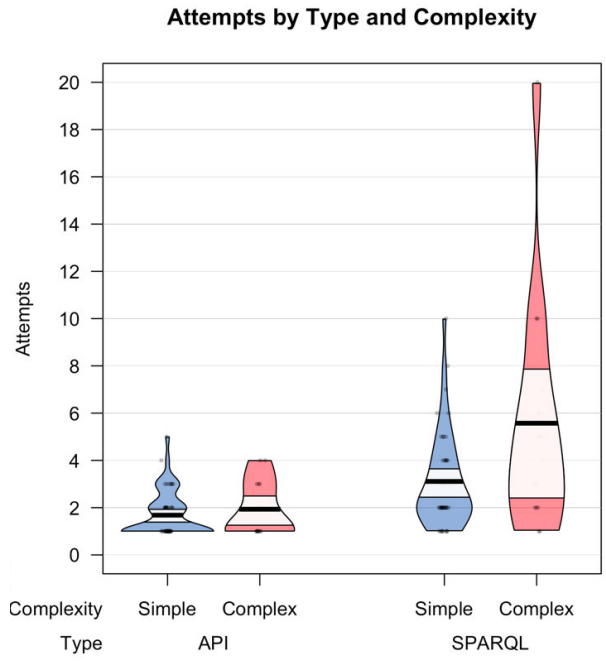

FIGURE 11. Attempts by type and complexity.

The analysis was focused on studying the time it took the participants to answer the query and the number of attempts needed to find the correct answer. We used a two-factor ANOVA with interaction to analyze the effect on the response time for each type of tool (API/SPARQL) and the complexity (simple/complex). The time variable was transformed with the double square root in order to comply with homocedasticity. The number of attempts was analyzed by using Poisson's GLM approach, taking into account the type and complexity factors.

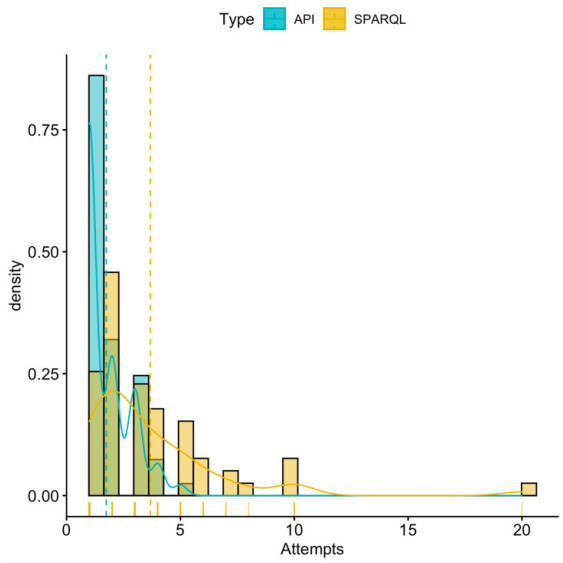

FIGURE 12. Histogram for attempts by type.

B. RESULTS

The results obtained from the analysis of the time that the participants took to respond with the correct answers have a different pattern according to whether they were attained using API or SPARQL (Fig. 10). No interaction between the two factors was detected ($F1,118$, p-value $= 0.8450$), signifying that the pattern observed in both types is the same: more complex queries take more time, regardless of the type of approach (SPARQL or Web APIs). However, both factors separately had significant differences in their levels. Users spent more response time when using SPARQL than when using Web APIs for both simple and complex queries ($F1,118$, p-value $= 0.0018$). The complexity was also significant: the simpler queries required less time than the complex ones ($F1,118$, p-value $= 0.0324$).

The number of attempts was higher when using SPARQL (p-value $=$ 1.42e-10) (Fig. 11). It was also higher for the more complex queries (p-value $= 0.0001$). In the case of SPARQL, there were up to 20 attempts, the average being around 4 attempts (Fig. 12). In the case of API, the average number of attempts was about 2.

It is, therefore, possible to conclude that the use of Web API attains better results than the use of SPARQL as regards both the number of attempts and the response time.

C. DISCUSSION

When employing Web APIs, the users generally responded with correct answers in less time than when employing SPARQL. If we consider the complexity of the queries, the more complex the query was, the more time they spent resolving them by using SPARQL. The response time for complex queries using SPARQL has greater variance than the other cases.

With regard to the number of attempts, the pattern is similar: the more complex the query, the longer it took, regardless

of using Web APIs or SPARQL. In addition, the number of attempts almost doubles when using SPARQL, despite the complexity of the query. This is particularly relevant in the case of situational scenarios, in which the data lifetime is very short. We do not normally know the structure of situational data, signifying that it will be more complex to manage.

The results of the experiment have allowed us to verify our previous hypothesis: that it is easier and faster in a situational scenario to use a Web API than the corresponding SPARQL endpoint in order to consume open data. Using disposable Web APIs, therefore, makes sense if the data to be consumed has a short lifespan and the objective is to solving specific one-time problems. When compared to a SPARQL endpoint, this could, therefore, be more useful in other more stable scenarios.

V. CONCLUSION

In this paper, we present an approach whose objective is to generate disposable Web APIS for open data consumption. Disposable Web APIs will likely be used to consume situational data, i.e., those data that have a narrow focus on a specific area and, often, a short lifespan in order to add value to data owned (and controlled) by a small group of consumers with a unique set of needs. This was the case in our situational scenario, in which a data scientist working at a real estate company was willing to analyze its internal data regarding housing rental and incomes in Madrid by adding open data from the Madrid public transport so as to attain new insights and make informed decisions in order to provide adapted solutions to customers with special mobility needs. Our approach first proposes a process with which to acquire structured open data. A semantic annotation process is then carried out to obtain semantically annotated data (i.e., RDF triples). Finally, this annotated data is used as a starting point for a model-driven process that automatically generates the disposable Web API for open data consumption in a situational scenario.

Our approach makes use of some semantic information, but we do not rely on SPARQL endpoints. Indeed, we have also conducted a controlled experiment to show that our Web APIs approach is more convenient for accessing data than a SPARQL endpoint.

One piece of immediate future work is that of carrying out more detailed experimentation in order to consider a wider range of situational scenarios. Our long-term future work consists of extending our disposable Web APIs generation process in order to consider open data consumers' personalization requirements.

ACKNOWLEDGMENT

The authors would like to thank to those students who agreed to participate in the experiment.

REFERENCES

[1] A. Abelló, J. Darmont, L. Etcheverry, M. Golfarelli, J.-N. Mazón, F. Naumann, T. Pedersen, S. B. Rizzi, J. Trujillo, P. Vassiliadis, and G. Vossen, "Fusion cubes: Towards self-service business intelligence," *Int. J. Data Warehousing Mining*, vol. 9, no. 2, pp. 66–88, Apr. 2013.

[2] B. DuCharme, *Learning SPARQL: Querying and updating with SPARQL 1.1*. Newton, MA, UAS: O'Reilly Media, 2013.

[3] D. Booth, C. G. Chute, H. Glaser, and H. Solbrig, "Toward easier RDF," in *Proc. W3C Workshop Standardization Graph Data*, Berlin, Germany, 2013, pp. 1–5.

[4] J. Klímek, P. Škoda, and M. Neäaský, "Survey of tools for linked data consumption," *Semantic Web*, vol. 10, no. 4, pp. 665–720, May 2019.

[5] P. Lisena, A. Meroáo-Peáuela, T. Kuhn, and R. Troncy, "Easy Web API development with SPARQL transformer," in *Proc. Int. Semantic Web Conf.* Cham, Switzerland: Springer, 2019, pp. 454–470.

[6] I. M. S. Hopkinson and M. Rospocher, "A simple API to the knowledge-store," in *Proc. Int. Conf. Developers*, Oct. 2014, pp. 1–5.

[7] H. Müller, L. Cabral, A. Morshed, and Y. Shu, "From RESTful to SPARQL: A case study on generating semantic sensor data," in *Proc. 6th Int. Conf. Semantic Sensor Netw.*, vol. 1063, Oct. 2013, pp. 51–66.

[8] M. Rittenbruch, M. Foth, R. Robinson, and D. Filonik, "Program your city: Designing an urban integrated open data API," in *Proc. Conf., Open Helsinki-Embedding Design Life*, 2012, pp. 24–28.

[9] A. Freitas, A. Legendre, S. O'Riain, and E. Curry, "Prov4j: A semantic Web framework for generic provenance management," in *Proc. 2nd Int. Workshop Role Semantic Provenance Manage. (SWPM*, 2010, pp. 17–22.

[10] J. Cabot. (2018). *Open Data for All: An API-Based Approach Interested Modeling Languages*. [Online]. Available: https://modeling-languages.com/open-data-for-all-api/

[11] P. Cáceres, A. Sierra-Alonso, B. Vela, J. M. Cavero, M. A. Garrido, and C. E. Cuesta, "Adding semantics to enrich public transport and accessibility data from the Web," *Open J. Web Technol.*, vol. 7, no. 1, pp. 1–18, 2020.

[12] P. Ceres, A. Sierra-Alonso, B. Vela, J. M. Cavero, and C. E. Cuesta, "Towards smart public transport data: A specific process to generate datasets containing public transport accessibility information," in *Proc. 3rd Int. Conf. Universal Accessibility Internet Things Smart Environments*, Rome, Italy, 2018, pp. 66–71.

[13] P. Cáceres, A. Sierra-Alonso, C. E. Cuesta, B. Vela, and J. M. Cavero, "Modelling and linking accessibility data in the public bus network," *J. Universal Comput. Sci.*, vol. 21, no. 6, pp. 777–795, 2015.

[14] CEN/TC 278. (2012). *Intelligent Transport Systems—Public Transport—Identification of Fixed Objects In Public Transport*. [Online]. Available: http://www.normes-donnees-tc.org/wp-content/uploads/2014/05/IFOPT-FR.pdf

[15] Transmodel. (2016). *Transmodel, Road Transport and Traffic Telematics*. [Online]. Available: http://www.transmodel.org/en/cadre1.html

[16] F. Jouault, F. Allilaire, J. Bázivin, and I. Kurtev, "ATL: A model transformation tool," *Sci. Comput. Program.*, vol. 72, pp. 1–2, May 2008, doi: 10.1016/j.scico.2007.08.002.

[17] J. S. Cuadrado, E. Guerra, and J. de Lara, "AnATLyzer: An advanced IDE for ATL model transformations," in *Proc. 40th Int. Conf. Softw. Eng., Companion Proceeedings*, May 2018, pp. 85–88, doi: 10.1145/3183440.3183479.

[18] A. Srai, F. Guerouate, N. Berbiche, and H. Drissi, "An MDA approach for the development of data warehouses from relational databases using ATL transformation language," *Int. J. Appl. Eng. Res.*, vol. 12, pp. 3532–3538, 2017.

[19] A. Meroáo-Peuela and R. Hoekstra, "GRLC makes Github taste like," in *Proc. Eur. Semantic Web Conf.*, 2016, pp. 342–353.

[20] A. Meroáo-Peáuela and R. Hoekstra, "Automatic Query-centric API for routine access to linked data," in *Proc. Int. Semantic Web Conf.*, 2017, pp. 334–349.

[21] R. Taelman, S. M. Vander, and R. Verborgh, "GraphQL-LD: Linked data Querying with GraphQL," in *Proc. 17th Int. Semantic Web Conf.*, 2018, pp. 1–4.

[22] D. Garijo and M. Osorio, "OBA: An ontology-based framework for creating REST APIs for knowledge graphs," in *Proc. Int. Semantic Web Conf.* Cham, Switzerland: Springer, Nov. 2020, pp. 48–64.

[23] A. Dimou, S. M. Vander, P. Colpaert, R. Verborgh, E. Mannens, and R. Van de Walle, "RML: A generic language for integrated RDF mappings of heterogeneous data," in *Proc. LDOW*, Jan. 2014, pp. 1–5.

[24] M. Lefranáois, A. Zimmermann, and N. Bakerally, "A SPARQL extension for generating RDF from heterogeneous formats," in *Proc. Eur. Semantic Web Conf.* Cham, Switzerland: Springer, May 2017, pp. 35–50.

PALOMA CÁCERES GARCÍA DE MARINA received the M.Sc. degree in computer science from the Polytechnic University of Madrid, Spain, in 1993, and the Ph.D. degree in computer science from Rey Juan Carlos University, Madrid, Spain, in 2006. She is currently an Associate Professor with Rey Juan Carlos University. Her research interests include software engineering, Web engineering, model-driven development, data science, open and linked data, the semantic Web, and transport and accessibility. She is the author or coauthor of several national and international articles related to these areas.

JOSÉ MARÍA CAVERO BARCA received the M.Sc. degree in computer science from the Polytechnic University of Madrid and the Ph.D. degree in computer science from Rey Juan Carlos University. He is currently an Associate Professor with the School of Computer Science Engineering, Rey Juan Carlos University. His research interests include databases, open data, transport, accessibility, and scientometrics.

CARLOS E. CUESTA received the Ph.D. degree in information technologies from the University of Valladolid, in 2002. He is currently an Associate Professor of software engineering with Rey Juan Carlos University, Madrid, Spain. He has had work published in premier conferences and journals, such as the *International Journal of Information Technology & Decision Making* and *Future Generation Computer Systems*. His main research interest includes software architecture, including such topics as self-adaptive systems and systems-of-systems, and with relation to concurrency theory, formal methods, distributed systems, or data engineering. He has been the Program Chair of the Sixth and 12th European Conferences on Software Architecture (ECSA 2012 & 2018).

MIGUEL ÁNGEL GARRIDO received the M.Sc. degree in computer science from Rey Juan Carlos University, Madrid, Spain, in 2014, where he is currently pursuing the Ph.D. degree in computer science. For ten years, he has worked as a computer technician at several entities. He is also an Assistant Professor with Rey Juan Carlos University. His research interests include Web engineering, model driven development, semantic Web, linked open data, transport, and accessibility.

IRENE GARRIGÓS received the Ph.D. degree. She is currently an Associate Professor with the Department of Software and Computing Systems and the Head of the Web and Knowledge Research Group, University of Alicante, Spain. Her research interests include open data, Web augmentation, Web modeling languages, and personalization and application programming interfaces.

CÉSAR GONZÁLEZ-MORA is currently pursuing the Ph.D. degree with the Web and Knowledge Research Group, Department of Software, University of Alicante, Spain. His research interests include open data, Web augmentation, semantic Web, and application programming interfaces. His work is funded by a contract with the Generalitat Valenciana of Spain and the European Social Fund for predoctoral training.

JOSE-NORBERTO MAZÓN received the Ph.D. degree. He is currently an Associate Professor with the Department of Software and Computing Systems, University of Alicante, Spain. He is also the Chair of the Torrevieja Venue at the University of Alicante. He is the author of more than 100 scientific works published in international conferences and journals. His research interests include open data, business intelligence in the big data scenario, the design of data-intensive Web applications, smart cities, and smart tourism destinations.

ALMUDENA SIERRA-ALONSO received the Ph.D. degree in computer science from the Universidad Politécnica de Madrid, in 2000. She is currently an Associate Professor with the Department of Computer Science and Statistics, Rey Juan Carlos University. She is the author or coauthor of several articles related to software engineering, the semantic web, and teaching in engineering education (in areas, such as operating systems and software engineering). Her research interests include software engineering, data science, open and linked data, the semantic Web, and transport and accessibility.

BELÉN VELA received the M.Sc. degree in computer science from Carlos III University and the Ph.D. degree in computer science from Rey Juan Carlos University. She is currently an Associate Professor with the Department of Computing Science, Computer Architecture, Programming Languages and Systems and Statistics and Operative Investigation, Rey Juan Carlos University, Madrid, Spain. She also leads the Vortic3 Research Group. She has participated in and led several research projects and has had numerous articles published in prestigious journals and conferences. Her research interests include software engineering, information system engineering, data science, open data, DB (NoSQL), model driven development, transport, accessibility, and scientometrics.

JOSÉ JACOBO ZUBCOFF received the Ph.D. degree. He has a wide range of teaching and research experience in the field of statistics, and data mining and its application to biology. He has more than 100 publications in which he has dealt with obtaining knowledge from a data source. He has carried out research in various fields of science, such as computing, biology, medicine, education, and social sciences. He has additionally directed and participated in more than 20 competitive public projects financed by the Ministry of Economy and Competitiveness, the Generalitat Valenciana, the University of Alicante, and European and private projects, all of which have contributed to his knowledge of data analysis, data mining, and the attempt to democratize knowledge.