A web service based on RESTful API and JSON Schema/JSON Meta Schema to construct knowledge graphs

Adam Agocs
CERN
CH-1211 Geneva 23
Switzerland
Email: adam.agocs@cern.ch

Jean-Marie Le Goff
CERN
CH-1211 Geneva 23
Switzerland
Email: jean-marie.le.goff@cern.ch

Abstract—Data visualisation assists domain experts in understanding their data and helps them make critical decisions. Enhancing their cognitive insight essentially relies on the capability of combining domain-specific semantic information with concepts extracted out of the data and visualizing the resulting networks. Data scientists have the challenge of providing tools able to handle the overall network lifecycle. In this paper, we present how the combination of two powerful technologies namely the REST architecture style and JSON Schema/JSON Meta Schema enable data scientists to use a RESTful web service that permits the construction of knowledge graphs, one of the preferred representations of large and semantically rich networks.

Index Terms—Computer architecture, RESTful API, JSON, JSON Schema, Data validation.

1. Introduction

Collaboration Spotting (CS) [1][2] is a platform designed to support visual analytics of multivariate knowledge graphs built out of data from heterogeneous sources. It offers a novel approach to handle semantic and structural complexity at the interactive visualisation level by enabling users to generate different perspectives of domain-related knowledge graphs, navigate between these perspectives and execute different graph algorithms within them.

Since 2012, CS has been deployed on many pilot projects in various domain-related analysis to demonstrate its capability of enhancing the cognitive insight of humans into the understanding of their data. In particular, CS has been used

- to analyse publications and patents for dental science [3] and for the detection of technology and innovation developments [4],
- for a security-threat analysis [5],
- for a university ranking project in collaboration with the MTA-PE Budapest Ranking Research Group [6] and
- for a neuroscience project in collaboration with the Complex Systems and Computational Neuroscience Group [7] at Wigner RCP.

CS capability of enhancing cognitive insight strongly depends on the construction of domain-independent and semantically rich knowledge graphs. In essence, constructing such graphs calls for an API that enables users to:

1) use descriptors to specify data elements with minimal restrictions,
2) structure data elements in an ontology-like hierarchy,
3) set up data-related control parameters to satisfy the data-driven approach of the platform,
4) validate descriptors by using pre-defined node and edge descriptors,
5) validate data with their descriptors and
6) upload data and descriptors by using well-known and widespread technologies.

To these, one must add requirements that are specific to the interactive visualization and navigation aspects of the CS platform:

7) Use of a graph database (which is a natural choice in support of visual analytics on graphs) such as Neo4j [8] for storing **a)** users' data as knowledge graphs i.e. a 4-element tuples ($G = (V, E, L, \alpha)$) where both vertices and edges are labelled and **b)** the knowledge graph's schema called as reachability graph (please, see [9] for further details).
8) Use of Django [10] as a web framework implemented in Python.
9) Use of JSON [11][12] as a data exchange format between the different layers of the platform.

A REST architecture like web service combined with JSON Schema/JSON Meta Schema [13][14] and Py2neo Python package [15] provides an adequate tool for constructing knowledge graphs in compliance with the above-mentioned requirements.

After the related work emphasising JSON format and enabling technologies, Section 3 presents the architecture of the RESTful web service of the Collaboration Spotting platforms showing how the selected technologies satisfy the requirements and how an extension of the JSON Schema specifications can support an ontology-like hierarchy for the descriptors. Section 4 gives a use-case and some experimental results on the scalability of the API and some comparisons between the single mode and the bulk mode operations. And finally, the paper finishes with Future Work and Conclusion in Section V and Section VI respectively.

2. Related Work

In his PhD dissertation, Roy Fielding described the principled design of the modern web architecture that

leads to the REST architecture style [16]. Since then, REST gained in popularity amongst the API (Application Programming Interface) developers and became the most used approach for developing web services [17][18]. According to programmableWeb.com [18], most of these services have been developed with API using the JSON (JavaScript Object Notation) format to send and receive requests and responses over the HTTP protocol. Since its draft submission, JSON has followed an XML-like path starting as a data exchange format over the Internet to become part of an exchange protocol used by various APIs. The most notable ones being:

- JSON-LD [19] (W3C recommendation [20]), a JSON-based serialisation for handling Linked Data [21] extended with contextual explanations,
- JSON API [22] specifies the communication protocol between clients and servers,
- JSON-RPC [23] a remote procedure calls in JSON, similar to XML-RPC with XML.

JSON-to-RDF/XML converters provide an indirect means to valid native JSON documents. To the best of our knowledge, JSON Schema is the only format that enables users to define the syntax of a JSON document. Python supports the fourth draft version of this format, but a more recent version is available (the seventh one). Although JSON schema format is still in the process of standardisation, the number of JSON Schema-based theoretical and practical results is growing. In particular, this is the case for the schema's formal definition [24]) and SDMX-JSON data retrieval of OECD [25] datasets.

The JSON key features that are the validation rules (JSON Meta Schema) and the code to validate these rules made it possible to develop node and edge validators for Collaboration Spotting. Since 2004, OWL [26] has become a W3C supported standard for query languages on ontologies. OWL and ontologies for domain-specific knowledge representations are key technologies in various research areas such as data retrieval, data mining, machine learning and data visualisation. The criteria of Section 1, does not require the creation of an OWL-equivalent language built upon JSON schema. Applying the Amann and Fundulakis mathematical definition [27] of an ontology combined with a graph database supporting a labelled property graph data model will be sufficient for the construction of knowledge graphs compliant with the above requirements. In Amann and Fundulaki's work, an ontology is expressed as a triplet, $\mathcal{O} = (C, R, isa)$, where 1) C is a set of concepts, 2) R is a set of binary typed roles between these concepts and 3) isa is a set of inheritance ("is a") relationship between them.

3. Architecture

As indicated above, the CS RESTful API is built upon three main technology elements, namely the Django REST framework, the Python implementation of JSON Schema/JSON Meta Schema and Py2neo Python package for the Neo4j graph database. These technologies address the user requirements of Section 1 as follows:

1) Descriptors in JSON Schema format can describe user data,
4) the validation of descriptors in JSON schema requires modifications of the JSON Meta Schema such that each descriptor either describes a graph node or a graph edge in the knowledge graph,
5) JSON documents containing user data are validated by using descriptors written in the JSON Schema format,
6) Django REST framework is a powerful and flexible toolkit for building Web APIs that is compliant with the REST architecture style,
7) Py2neo is a client library for working with Neo4j,
8) Django REST framework can easily be combined with Django Web framework and
9) Django REST framework and JSON Schema/JSON Meta Schema support JSON format.

Essentially all the user requirements are addressed with the exception of requirements 2 and 3.

3.1. Descriptor

Requirement 2 requires an extension of JSON schema specifications in the form of additional keywords (see Table 1) in order to support the correspondence between descriptors and concepts together with the inheritance mechanism as requested by Amann and Fundulakis in their definition of an ontology [27]. More precisely, additional keywords are needed to specify that:

- a set of concepts (definition of C) corresponds to a set of descriptors,
- edge descriptors can specify binary-typed roles (definition of R) between node descriptors (i.e. concepts), and
- the use of keyword parents in node descriptors characterises the inheritance relationship (definition of isa), see Example 1.

```
{
  "$schema": "http://localhost:8000/schemas/
      validators/node_validator.json#",
  "id": "http://localhost:8000/schemas/ranking/
      he.json#",
  "title" : "HE",
  "type" : "object",
  "properties" : {
  "id": {"$ref": "../basic/basic_definitions.
      json# /definitions/id"},
  "name" : {"type": "string", "maxLength": 100
      0,  "minLength": 1}
},
  "additionalProperties" : {"$ref": "../basic/
      basic_definitions.json#/definitions/
      default_property"},
  "required": ["id", "name"],
  "parents" : ["institute"],
  "graph_element" : "node"
}
```

Example 1. JSON Schema for the "High Education" node in the Ranking project [6]

Two restrictions on keyword values apply when writing descriptors:

- The value of keyword *$schema* must be either *node_validator.json* or *edge_validator.json* and
- keyword value *required* must contain values *id* and *name*.

settings is an additional keyword that is needed to address Requirement 3. *settings* is a list of key-value pairs where each pair makes a connection between a pre-defined function (key) and its attribute (value).

TABLE 1. EXTRA KEYWORDS FOR DESCRIPTORS

Key	Data type	Scope	Description
graph_element	"node" | "edge"	each descriptor	Mandatory, defines the role of the descriptor
parents	array of references	node descriptors	Optional, defines *ISA* connections between node descriptors
direction	"double" | "single"	edge descriptors	Mandatory, defines whether the edge descriptor is directed or undirected
source_label	reference	edge descriptors	Mandatory, labels the source nodes
target_label	reference	edge descriptors	Mandatory, labels the target nodes

3.2. Main processes and modules

Figure 1 shows a simplified architecture model of the CS RESTful API with its three main processes *create project*, *upload descriptors* and *upload data*. These processes make use of the following supporting modules:

- **RESTful Interface** uses Django REST Framework. This module is responsible for handling and parsing every RESTful message from clients and sending back response messages,
- **Neo4j Interface** maps the RESTful calls to Neo4js Cypher queries.
- **JSON Schema/Meta Schema Validator** module validates user descriptors and data in JSON format according to pre-defined descriptors and JSON Meta Schema,
- **Project** module generates a bulk descriptor for each user-defined ones and handles single and bulk versions of descriptors. See the High Education descriptor of Example 1 related to the Ranking project and its bulk version in Example 2. Users can validate and upload multiple graph elements in one bulk.
- **Project Manager** module supports project creation and selection and forwards messages between the RESTful Interface and a given project.

Figure 2 and 3 show the UML Activity Diagrams for the upload descriptor and upload data processes respectively. The source code found in Collaboration Spotting GitHub repository [28] follows these diagrams.
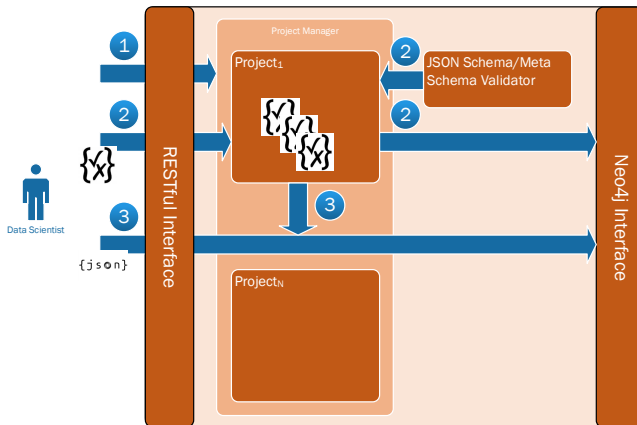

Figure 1. Simplified architecture model of Collaboration Spotting's API with the three main processes 1) create projects, 2) upload descriptors and 3) upload data.

```
"$schema": "http://localhost:8000/schemas/
    validators/node_validator.json#",
"id": "http://localhost:8000/schemas/ranking/
    bulk_he.json#",
"title" : "Bulk HE",
"definitions" : {
  "node" : {"$ref": "./he.json#"}
},
"type" : "array",
"items": {"$ref" : "#/definitions/node"}
}
```

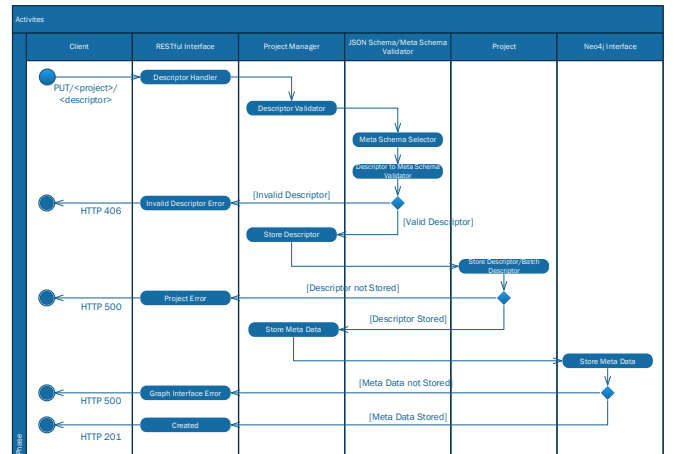Example 2. Automatically generated bulk version of descriptor "High Education" in the Ranking project.

Figure 2. UML Activity Diagram of the *upload descriptor* process
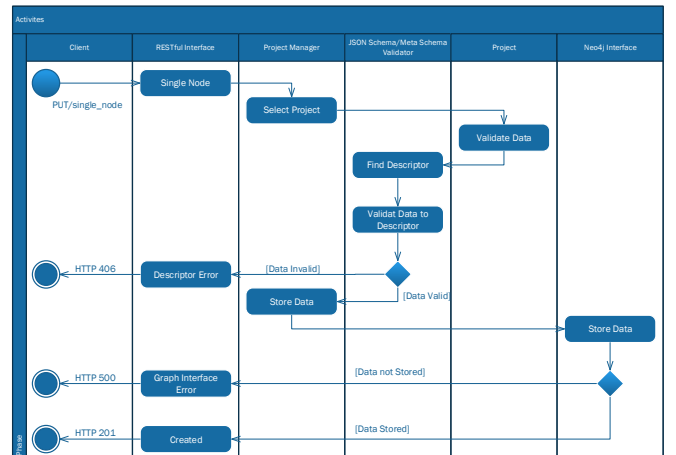

Figure 3. UML Activity Diagram of the *upload data* process

{

4. Use-case

The MTA-PE Ranking Research Group provided a collection of anonymised student travels in the framework of the European Commissions Erasmus programme during 2008-9 to 2013-14. The Ranking Research Group edited the data in order to merge different time-intervals and selected the relevent attributes for their analysis. A small sample of the results is shown in Table 2. Figure 4 gives a simplified data model of the knowledge graph resulting from uploading in Neo4j the data from the Ranking Research Group. One can note that the node of Example 1-2 appears in as a node labelled HE with its connecting relationships in the data model.

The measurements address the performance of the CS RESTful API when uploading single and bulk inserts. These measurements have been done on an HP Z440 workstation (Intel Xeon E5-1620 v3, 3,50 GHz processor, 32 GB DDR4 RAM and 512 GB SSD drive), using the community version of Neo4j 2.3.1 with -Xmx8192m JVM settings.

Figure 5 shows the uploading time of a single JSON document, the descriptor being pre-uploaded. The variations observed depend on the writing time of the JSON document in the Neo4j database. Time, which is directly related to the size of the previously uploaded data in the *upload data* process as can be seen in the activity diagram of Figure 3. Fig. 6 plots the uploading time of single and bulk inserts. It shows that the bulk insert is approximately 163 times faster than the single insert, confirming the clear advantage of using bulk inserts.
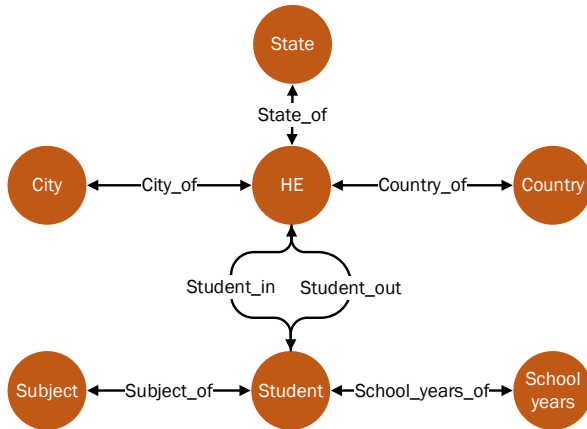

Figure 4. The simplified database model for the Ranking project. The *parent* label of the descriptor in Example 1 has been omitted.

5. Future Work

As mentioned in the introduction, JSON Schema is not a standard format yet. We will strive to support the new version of the standard as soon as it becomes available. The replacement of simple JSON documents with ones written in JSON-LD could be an alternative solution. It would enable to combine the expressive power of JSON-LD with the verification power of JSON Schema. But the development of the validation code required for JSON-LD
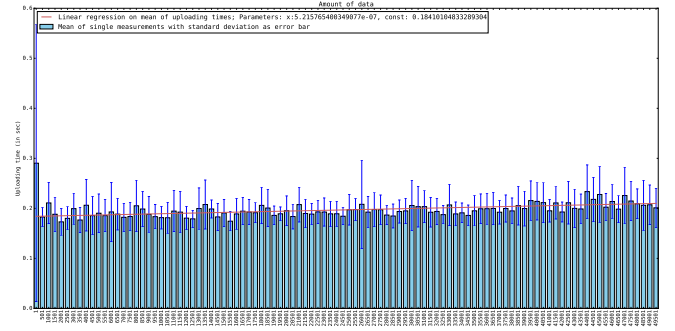

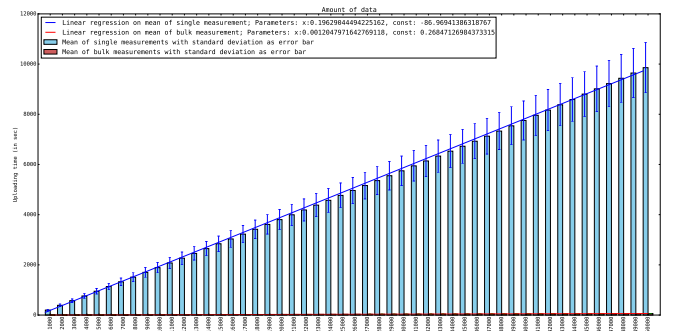Figure 5. Uploading time of single JSON documents

Figure 6. Comparison of single and bulk inserts

would call for a substantial amount of work in comparison with the gain in expressiveness. Indeed, the implementation of the ontology-like hierarchy in the data model is sufficient to construct the knowledge graphs supported by Collaboration Spotting.

6. Conclusion

In this paper, we present the architecture of a RESTful web service based on the combination of the JSON-based REST architecture style with JSON Schema/JSON Meta Schema that permits the construction of knowledge graphs. With this service, users of the Collaboration Spotting platform are able in rather unique way and with few restrictions, to create the elements of their knowledge graph(s) and validate the uploaded data togheter with these elements by using JSON Schema. A small extension of the JSON Meta Schema, made it possible to extend user-created descriptors to support ontology concepts together with an appropriate validation of these descriptors.

References

[1] A. Agocs, D. Dardanis, R. Forster, J.-M. Le Goff, X. Ouvrard, and A. Rattinger, "Collaboration Spotting: A Visual Analytics Platform

TABLE 2. ERASMUS STUDENT DATA

from he	from he country	to he	to he country	subject	year	distance	direction
D KONSTAN01	DE	K BATH01	UK	3	2008-09	907	296,286297877918
D KONSTAN01	DE	F PARIS007	FR	4	2008-09	501	283,942344291399
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

to Assist Knowledge Discovery," *ERCIM NEWS*, vol. 111, pp. 46–47, Oct 2017.

[2] Collaboration Spotting Developer Team. (2018, March) Collaboration Spotting.

[3] E. Leonardi, A. Agocs, S. Fragkiskos, N. Kasfikis, J.-M. Le Goff, M. Cristalli, V. Luzzi, and A. Polimeni, "Collaboration Spotting for dental science," *Minerva Stomatologica*, vol. 63, no. 9, pp. 295–306, September 2014.

[4] G. Joanny, A. Agocs, S. Fragkiskos, N. Kasfikis, J.-M. Le Goff, and O. Eulaerts, "Monitoring of Technological Development - Detection of Events in Technology Landscapes through Scientometric Network Analysis," in *ISSI*, 2015.

[5] (2018, February) United Nations Interregional Crime and Justice Research Institute. [Online]. Available: http://www.unicri.it/

[6] (2018, February) MTA-PE Budapest Ranking Research Group. [Online]. Available: http://www.gtk.uni-pannon.hu/mta-pe-budapest-ranking-research-group

[7] (2018, February) Complex Systems and Computational Neuroscience Group. [Online]. Available: http://cneuro.rmki.kfki.hu/

[8] Neo4j, *The Neo4j Manual*, November 2015, Release 2.3.2.

[9] A. Agocs, D. Dardanis, J.-M. L. Goff, and D. Proios, "Interactive graph query language for multidimensional data in collaboration spotting visual analytics framework," *arXiv preprint arXiv:1712.04202*, 2017.

[10] *Django Web framework - Reference Manual*, Django Software Foundation, April 2015, release 1.8. [Online]. Available: https://docs.djangoproject.com/en/1.8/ref/

[11] *The JSON Data Interchange Format*, 1st ed. ECMA International, October 2013. [Online]. Available: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

[12] T. Bray, "The JavaScript Object Notation (JSON) Data Interchange Format," IETF RFC 7158, Oct. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7158.txt

[13] (2018, March) JSON Schema. [Online]. Available: http://json-schema.org/

[14] K. Zyp, G. Court, and F. Galiegue, "JSON Schema: core definitions and terminology," Internet Engineering Task Force, Internet-Draft draft-zyp-json-schema-04, Aug. 2013, work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-zyp-json-schema-04

[15] Nigel Small. (2018, March) The Py2neo 2.0 Handbook. [Online]. Available: http://py2neo.org/2.0/index.html

[16] R. T. Fielding and R. N. Taylor, *Architectural styles and the design of network-based software architectures*. University of California, Irvine Doctoral dissertation, 2000, vol. 7.

[17] T. Vitvar and J. Musser, "ProgrammableWeb.com: Statistics, trends, and best practices," in *Keynote of the Web APIs and Service Mashups Workshop at the European Conference on Web Services*, 2010.

[18] Wendell Santos. (2017, November) Which API Types and Architectural Styles are Most Used? [Online]. Available: https://www.programmableweb.com/news/which-api-types-and-architectural-styles-are-most-used/research/2017/11/26

[19] Lanthaler, Markus and Gütl, Christian, "On Using JSON-LD to Create Evolvable RESTful Services," in *Proceedings of the Third International Workshop on RESTful Design*, ser. WS-REST '12. New York, NY, USA: ACM, 2012, pp. 25–32. [Online]. Available: http://doi.acm.org/10.1145/2307819.2307827

[20] M. Sporny, D. Longley, G. Kellogg, M. Lanthaler, and N. Lindström, "A JSON-based Serialization for Linked Data," January 2014. [Online]. Available: https://www.w3.org/TR/2014/REC-json-ld-20140116/

[21] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data-the story so far," *International journal on semantic web and information systems*, vol. 5, no. 3, pp. 1–22, 2009.

[22] S. Klabnik, Y. Katz, D. Gebhardt, T. Kellen, and E. Resnick. (2018, March) JSON API. [Online]. Available: http://jsonapi.org/

[23] JSON-RPC Working Group. (2018, March) JSON-RPC. [Online]. Available: http://www.jsonrpc.org/

[24] F. Pezoa, J. L. Reutter, F. Suarez, M. Ugarte, and D. Vrgoč, "Foundations of JSON Schema," in *Proceedings of the 25th International Conference on World Wide Web*, ser. WWW '16. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2016, pp. 263–273. [Online]. Available: https://doi.org/10.1145/2872427.2883029

[25] Statistical Data and Metadata eXchange Community. (2018, March) API documentation (SDMX-JSON). [Online]. Available: https://data.oecd.org/api/sdmx-json-documentation/

[26] (2004, February) OWL Web Ontology Language. [Online]. Available: https://www.w3.org/TR/owl-features/

[27] B. Amann and I. Fundulaki, "Integrating ontologies and thesauri to build rdf schemas," in *International Conference on Theory and Practice of Digital Libraries*. Springer, 1999, pp. 234–253.

[28] Adam Agocs. (2018, March) Collaboration Spotting - RESTful API Repository. GitHub repository. [Online]. Available: https://github.com/AdamAgocs/REST_Collspotting