The smartAPI ecosystem for making web APIs FAIR

Shima Dastgheib[1], Trish Whetzel[2], Amrapali Zaveri[1,3], Cyrus Afrasiabi[4], Pedro Assis[5], Paul Avillach[6], Kathleen Jagodnik[7], Gabor Korodi[6], Marcin Pilarczyk[8], Jeff De Pons[9], Stephan Schürer[10], Raymond Terryn[10], Ruben Verborgh[11], Chunlei Wu⋆[4], and Michel Dumontier⋆[1,3]

[1] Stanford Center for Biomedical Informatics Research, Stanford University, United States
[2] T2 Labs, Sunnyvale, CA, United States
[3] Institute of Data Science, Maastricht University, Maastricht, The Netherlands
[4] The Scripps Research Institute, CA, United States
[5] Department of Genetics, Stanford University, United States
[6] Harvard Medical School, Boston, Massachusetts, United States
[7] Icahn School of Medicine at Mount Sinai, New York, New York, United States
[8] University of Cincinnati, Cincinnati, Ohio, United States
[9] Medical College of Wisconsin, Milwaukee, Wisconsin, United States
[10] University of Miami, Miller School of Medicine, Florida, United States
[11] IDLab, Dep. of Electronics and Information Systems, Ghent University – imec, Belgium

Abstract. We present the implementation of the smartAPI ecosystem to facilitate the creation of FAIR (**F**indable, **A**ccessible, **I**nteroperable, **R**eusable) APIs.

1 Introduction

The emergence of "big data" has accelerated the growth of API space, and hence an excellent opportunity has now arisen to build an intelligent network of Web APIs that empowers a given application to automatically discover and link suitable APIs. However, even semantically relevant APIs are documented independently and isolated from other APIs, and the API descriptions are minimally shared and reused [1]. smartAPI fills this gap by creating API descriptions that are accessible and reusable, and annotating them with explicit knowledge about the structure and datatypes of the API parameters and responses in order to achieve more findable and interoperable APIs.

In our previous work [3], we identified the metadata elements that are crucial to the description of Web APIs and subsequently developed the smartAPI metadata specification available at `https://websmartapi.github.io/smartapi_specification` using the FAIR principles (Findable, Accessible, Interoperable, Reusable) [2]. In this paper, we present the implementation of the smartAPI ecosystem.

We provided a comprehensive review of the challenges related to the existing systems in API life-cycle management in [3]. The Swagger editor[12] has the largest and most active community among API editors. It uses the OpenAPI specification[13], which defines a standard, language-agnostic interface to HTTP APIs in JSON format. Swagger's auto-completion functionality suggests predefined metadata; however, it does not offer a means to reuse/share annotations from/with other API documents.

⋆ Corresponding authors

[12] `https://swagger.io/swagger-editor/`

[13] `https://github.com/OAI/OpenAPI-Specification`

In contrast, smartAPI not only enables authoring and annotating API documents, but also contributes towards improved visibility, interoperability, and reusability of Web APIs. Rather than building an API editor from scratch, we extended the Swagger editor to fulfill the smartAPI objectives (Section 2).

2 Implementation

Figure 1 shows an overview of the smartAPI ecosystem. The smartAPI editor, available at `http://smart-api.info/editor/`, is an extension of the Swagger editor. The panel on the left enables creating a new or editing an existing API document. The preview panel on the right renders the API documentation and allows users to interact with the API (e.g. test an operation), while still editing it. The API document is validated against the smartAPI specification, which is an extended version of the OpenAPI specification. Each API document can be published into the smartAPI registry[14] by clicking the Publish button we added to the editor. All of the published API documents are indexed, and along with the indexed collection from `identifiers.org`, are provided through an HTTP API[15]. For a given metadata element, the editor sends HTTP requests to this API and provides an auto-suggestion list plus the usage frequency of each value.

The source code, documentation, and live demo are available at `https://github.com/WebsmartAPI/smartAPI-editor`. The GitHub repository is archived at `http://doi.org/10.5281/zenodo.580097`.

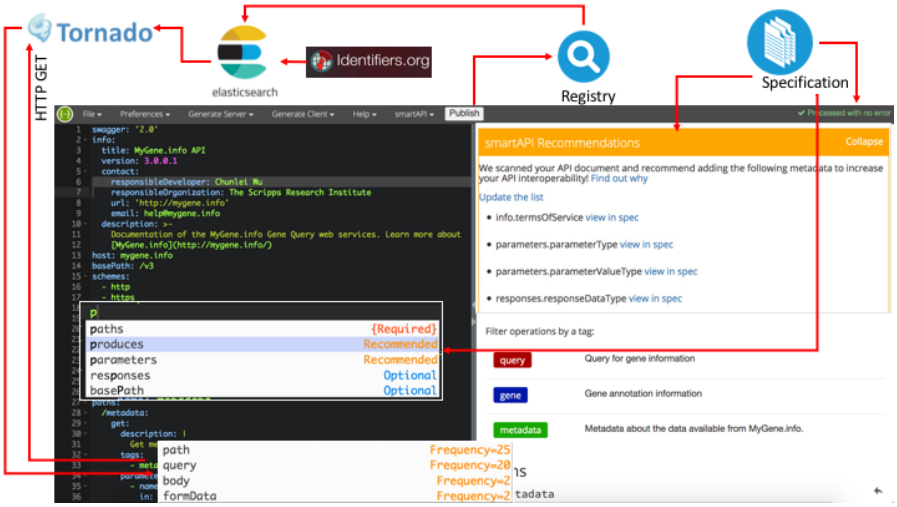


Fig. 1: An overview of the smartAPI ecosystem.

We extended the Swagger editor as follows to fulfill the potential of smartAPI:
Reuse of API descriptions. We extended the auto-completion functionality of the

[14] `http://smart-api.info/registry/`
[15] `http://smart-api.info/api/metadata/all`

Swagger editor, by suggesting not only the list of predefined metadata and values, but also the values retrieved from the indexed API documents previously created and saved in the registry, along with the usage frequency. The conformance level (Required, Recommended, or Optional) of the suggested metadata is also provided (Figure 1).
Semantic annotation of API description. To annotate the *parameterValueType* and *responseDataType* metadata elements, the auto-suggestion list also includes persistent semantic URIs from `identifiers.org`, which enables the referencing of data for the scientific community. While it is possible to extend the semantic annotation to other metadata elements, we have focused on the API's parameters and responses, which can be considered as the inputs and outputs of the API, respectively.

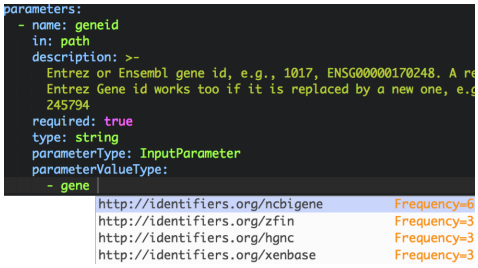
i Semantic annotation of API parameters is done by annotating *parameterValueType* with semantic identifiers. We used Elasticsearch to build a suggestion service that takes user search terms from the editor (e.g. hgnc), and uses HTTP requests to GET the list of matching identifiers (e.g. `http://identifiers.org/hgnc.symbol`). As shown in Figure 2a, this list is appended to the auto-suggestion list retrieved from the previously used identifiers.
ii Semantic annotation of API responses is done by annotating *responseDataType*. We developed the smartAPI profiler, a web application for automatic annotation of web services. We integrated the profiler within the editor. Adding the *responseDataType* metadata for a specific operation triggers the profiler (Figure 3), which then asks the user to enter example API response data (e.g. `http://mygene.info/v3/gene/1017`). This response data is then recursively traversed to provide a `keypath/value` pair where the keypath consists of one or more labels concatenated together and the value is either a single value or list of strings. The resource annotation is provided by comparing the keypath labels to resource names and synonyms from `identifiers.org`. In cases where a match is not found, an example value for the keypath is then compared against resource identifier patterns from `identifiers.org`, and resulting matches are displayed as suggested annotations. The user may also add his own resource annotation if one does not exist. The annotated API response data is stored in the *responseDataType* element as an array of YAML objects (Figure 2b).

Provide recommendations. The smartAPI editor scans the API document and provides a list of 'Recommended' and 'Optional' metadata to improve the API interoperability.

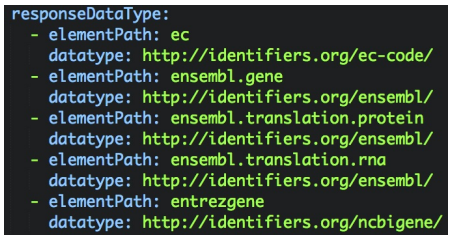
We developed a searchable API registry[14] to store, index, and view API documents. An API document can be published in the registry directly from the smartAPI editor. The smartAPI registry provides faceted search and displays the list of published APIs along with some details, including the lists of their input/output semantic annotations.

3 Conclusion and Future Work

The smartAPI ecosystem aims to make Web APIs FAIR, through development of (i) a specification and an editor making API descriptions more **R**eusable and **A**ccessible, (ii) semantic annotation tools for making APIs more **I**nteroperable, and (iii) a searchable registry for publishing and indexing API descriptions, to achieve more **F**indable and **R**eusable APIs. As the next steps, we will implement smartAPI compliant with openAPI[13]



(a) identifiers.org URIs are suggested to semantically annotate *parameterValueType*.



(b) *responseDataType* is automatically annotated by the profiler within the editor (Fig. 3).

Fig. 2: Semantic annotation of API parameters and responses.

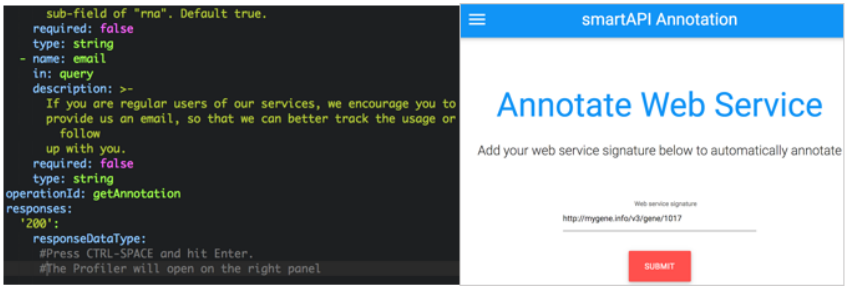


Fig. 3: The profiler opens within the editor, once the user wants to annotate response data.

version 3. Moreover, smartAPI will leverage JSON-LD to provide semantically annotated content that can be treated as Linked Data (`http://hdl.handle.net/10421/7478`).

4 Acknowledgments

The smartAPI pilot project was funded as a supplement to CEDAR (U41HG000131521). MyGene.info and MyVariant.info are supported by U01HG008473 (from NHGRI). The LINCS Data Portal is supported by grant U54HL127624 awarded by the National Heart, Lung, and Blood Institute through funds provided by the trans-NIH LINCS Program `http://www.lincsproject.org/` and the trans-NIH Big Data to Knowledge (BD2K) initiative `https://commonfund.nih.gov/bd2k`.

References

1. Verborgh, R., Dumontier, M.: A Web API ecosystem through feature-based reuse. arXiv preprint arXiv:1609.07108 (2016)
2. Wilkinson, M.D., Dumontier, M., Aalbersberg, I.J., et al.: The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data 2 (2016), `http://www.nature.com/articles/sdata201618`
3. Zaveri, A., Dastgheib, S., Wu, C., Whetzel, T., Verborgh, R., Avillach, P., Korodi, G., Terryn, R., Jagodnik, K., Assis, P., et al.: smartAPI: Towards a More Intelligent Network of Web APIs. In: European Semantic Web Conference. pp. 154–169. Springer (2017)