Automatic API Generation to Access Open Data

César González-Mora, Irene Garrigós, Jose-Norberto Mazón, José Jacobo Zubcoff

WaKe Research, University of Alicante, Spain
{cgmora, igarrigos, jnmazon, jose.zubcoff}@ua.es

Abstract. Nowadays, the amount and variety of available open datasets is growing more and more. However, the access to this open data is not as easy as it should be, so that it is needed a platform to facilitate this access. In order to provide easier access to different data which is not machine readable, this research proposes an automatic API generator, which creates Application Programming Interfaces to allow developers to query the desired data. The generated APIs will be used by developers to easily create applications that use open data in order to help the citizens solve their daily-life problems.

Keywords: automatic API generator, open data, data access, open data API.

1 Introduction

Worldwide governments and organizations are generating open data increasingly, in order to encourage the reuse of public sector information to solve citizens' daily-life problems. This open data is the data that have the suitable licences to be accessed, reused and redistributed freely by anyone [1]. However, the open data available is difficult to access and reuse, because in most cases they cannot be directly processed by computers [2]. Therefore, from the point of view of the open data reuser, an important issue is that it is commonly assumed the direct use of the data, through the use of complex query languages as SPARQL [3] within a Linked Open Data scenario, or through very limited interfaces. An alternative, which is becoming more popular, consists of exporting these data by defining open Application Programming Interfaces (APIs) [4], which allows data consumers to build their own applications to access the data. Their biggest risk is the need for development teams to know in depth the variety of APIs used and be able to combine them properly. In order to facilitate this work, different API integration platforms are currently being defined [5]. The effort required to develop this type of architecture is considerable because of the multiple APIs that developers must handle. So that, it is necessary to use techniques that help reducing both: the development effort and its later maintenance. This fact motivates that our proposal is based on automatic generation processes.

There are a variety of related works that deal with the topic of open data and accessing it using APIs. In [2] it is detailed how open data is mostly not readily usable by citizens, so it is important to facilitate the access to encourage software developers to use the open data, which is essential in smart cities. Also, in [6] it is described 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. So that the article proposes the development of an open data API which provides tools that will help the general public to access the data about their own cities. Other work [7] describes a simple API which is used to provide access to the *KnowledgeStore*, a semantic web storage which is only accessible by complex query languages.

Existing research argues that open data is crucial, but it is also important to bring easy access to the open data reusers to create added-value products and services, i.e., software developers. Some papers deal with the creation of APIs to facilitate the access to the data, but this process is not done automatically, as we propose. The automatic generation of APIs to access the data without dealing with different formats that are complex to process will give the opportunity to use the obtained data for many topics. Due to the importance of transport in our society, it will be used for the running example of the proposal.

The outline of the paper is as follows. Section 2 introduces our approach from a conceptual point of view illustrated by a simple case study, including implementation details, and in Section 3 we discuss about the strength of the approach and present future work.

2 Automatic API generation

In this section, we present our proposal, an automatic generator to facilitate the access to data through APIs. In Figure 1, it is shown the overall generation process. Taking a dataset as the input, the automatic generator creates an API to help open data reusers querying the data easily.

This process is divided in two stages. In the first stage, the definition and documentation of the API is generated from the input data file. This API definition and documentation is represented by a JSON file, which is generated according to the standards of Swagger [8] because it helps us to design, build, document and test the API. At the second stage, it is generated the API code to return the data queried by the users. The automatic generator, with the help of the Swagger Codegen tool from the Swagger platform, generates the API code represented by a Server in NodeJS [9], a simple and efficient runtime environment for network applications.

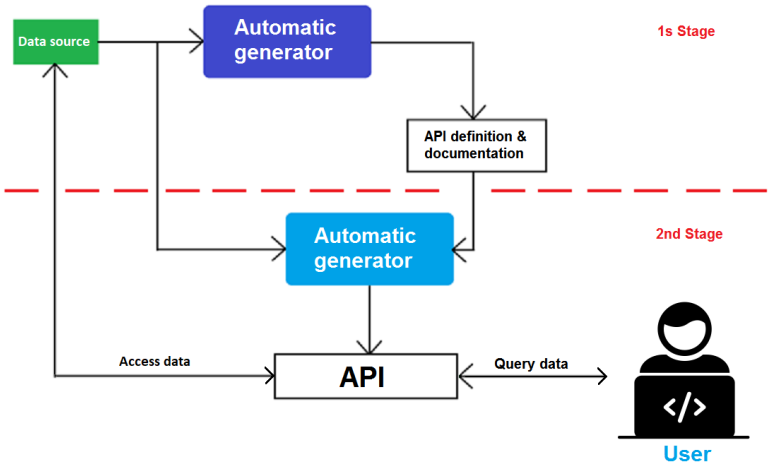

Fig 1. Automatic API generation process.

For demonstration purposes, we use a London underground dataset from the data.gov.uk open data portal, which contains interesting information about the mobility in London [10]. This dataset has been summed up in a single CSV file,

containing comma-separated information about the lifts available in the stations of the London underground. The running example, including the dataset used and all the auto-generated files, is publicly available online[1].

As it is shown in Figure 2, from the data file the automatic generator obtains the names of the columns, the data types and other needed data to build the JSON file that represents the API basic information. The name of each column is used in the API definition as the parameters of the API main method. Also, some information for documentation, which may not include the data file, can be asked to the user.

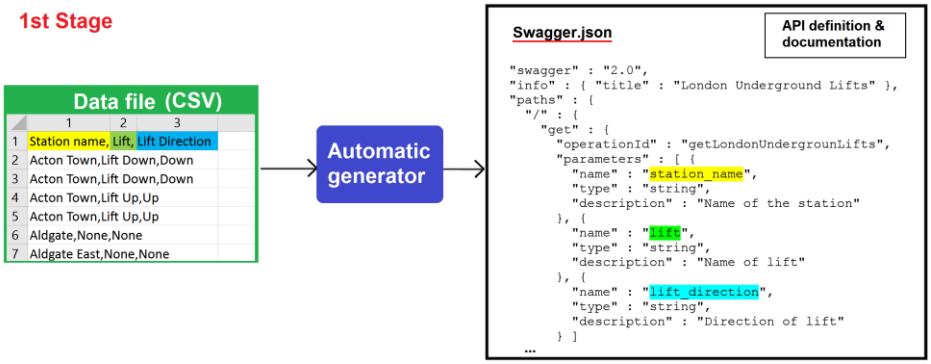

Fig 2. Stage 1 of the automatic API generation process.

In the second stage of the automatic generation process, which is shown in Figure 3, the Swagger Codegen tool generates the NodeJS Server skeleton from the JSON file. It facilitates the creation of the server by creating the different files and folders needed. After that, the automatic generator creates the needed code for the NodeJS Server.

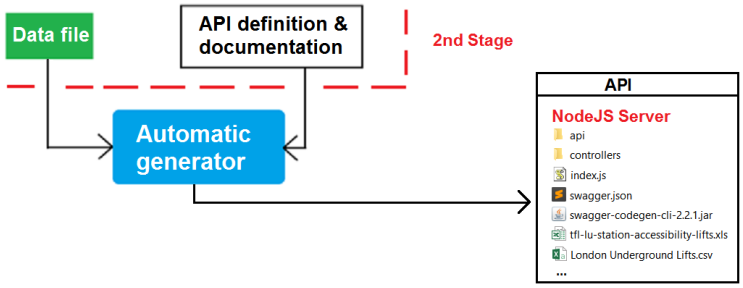

Fig 3. Stage 2 of the automatic API generation process.

Finally, the API must be published in an online server so that users can query the data with the desired parameters to filter the information.

The query used for this example requests the lifts of the London underground, filtered by the station name 'Acton Town' and by the direction 'Up' of the lift: `localhost:8080/cgmora12/lifts/1.0.0/?station_name=Acton Town&lift_direction=Up`

[1] https://github.com/cgmora12/liftsAPI

The results obtained from the query are the data from the Acton Town station which include as equipment a lift in the up direction: `{"csv":[["Acton Town", "Lift Up", "Up"],["Acton Town", "Lift Up", "Up"]]}`

3 Conclusions and Future Work

Nowadays, there is a lot of public information (such as transport data) which is available online, but it is difficult to find, to read and to search for a specific topic. The automatic generation of customized APIs from existing open data, is a powerful mechanism to make open data easily available for the citizen. Therefore, instead of querying the data manually in the data file, the automatic generator allows us to look up any information and filter it only by querying the API with simple requests.

In this research, it is implemented an automatic API generator from data files in a concrete format with specific data, but as a future work, it will be extended to work with different formats and tested with other case studies. Moreover, the option of customizing the APIs by requirements will be added, e.g. the developer will be able to obtain an API according to the requirements of data that are important to its clients.

Acknowledgements

This work has been funded by the National Foundation for Research, Technology and Development and the project TIN2016-78103-C2-2-R of the Spanish Ministry of Economy, Industry and Competitiveness.

References

1. Group, O. (2018). Open Definition 2.1 - Open Definition - Defining Open in Open Data, Open Content and Open Knowledge. http://opendefinition.org/od/2.1/en
2. Weerakkody, V., Irani, Z., Kapoor, K., Sivarajah, U., & Dwivedi, Y. K. (2017). Open data and its usability: an empirical view from the Citizen's perspective. Information Systems Frontiers, 19(2), 285-300.
3. Prud, E., & Seaborne, A. (2006). SPARQL query language for RDF.
4. Espinha, T.; Zaidman, A.; Gross, H.E. (2015). Web API growing pains: Loosely coupled yet strongly tied. Journal of systems and software, 100:27-43.
5. Clark, K.J. (2015) Integration Architecture: Comparing web APIs with Service-Oriented Architecture and Enterprise Application Integration. IBM developerWorks.
6. Rittenbruch, M., Foth, M., Robinson, R., & Filonik, D. (2012). Program your city: designing an urban integrated open data API. In Proceedings of Cumulus Helsinki 2012 Conference: Open Helsinki–Embedding Design in Life (pp. 24-28). Cumulus International Association of Universities and Colleges of Art, Design and Media.
7. Hopkinson, I., Maude, S., & Rospocher, M. (2014, October). A simple api to the knowledgestore. In Proceedings of the 2014 International Conference on Developers-Volume 1268 (pp. 7-12). CEUR-WS. org.
8. Swagger. (2018). World's Most Popular API Framework | Swagger. https://swagger.io
9. Foundation, N. (2018). Node.js. https://nodejs.org
10. Data.gov.uk. (2018). Transport Accessibility Data. https://data.gov.uk/dataset/transport-accessibility-data