I.J. Information Technology and Computer Science, **2018, 6, 64-71**
Published Online June 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2018.06.07

MASHUP of Linked Data and Web API

Mohammed Amine Belfedhal
EEDIS Laboratory, Djillali Liabes University, Sidi Bel Abbes, Algeria
E-mail: mohammed.belfedhal@gmail.com

Mimoun Malki
LabRI-SBA Laboratory, Higher School of Computer Science, Sidi Bel-Abbes, Algeria
E-mail: m.malki@esi-sba.dz

Received: 24 February 2018; Accepted: 20 March 2018; Published: 08 June 2018

Abstract**—**Mashups are an important way to allow normal users to build their own applications responding to the specific needs of each one. The basic components of mashups are Data and Web APIs especially Restful ones, but it is difficult for an unexperienced user to combine manually APIs with Data. Therefore, there is a need to predefine links between these resources to permit an easy combination. In this paper, we propose a new approach to make Restful Web APIs adhere to Linked Data principles, which facilitate their combination in mashup applications. The advantage of the proposed approach is the fact that it allows integrating linked data with the composition of Restful APIs, It also uses an algorithm to automatically create links between APIs.

Index Terms**—**Mashup, REST, Web API, Linked Data, Linked Web Service.

I. INTRODUCTION

The web has become a combination of large sets of data and services, but several structures, protocols and techniques have made the utilization of these sets of resources difficult because of heterogeneity. To facilitate the use of the resources available on the web we have to transform the existing ones to suite a unified model. In this paper, we are interested in building mashup applications using different resources from different providers on the web, so we propose a framework, which provides a unified vision of web APIs and Linked data.

The multitude of resources in the web gives the user a large choice of data and services to combine which is a good thing, but as a downside, this throng of providers causes a problem due to the fact that each one has his own manners to develop, publish and provide web resources.

To solve this problem many works were done to transform existing resources to linked data services in order to facilitate their utilization and integration. For example, [1-3] extend the description of SOAP web services then establish links between them; while [4] propose descriptions adapted to Restful services based on WSMO-Lite [5]. Meanwhile [6] propose a scripting language to enhance the description of Restful services.

In this work, we propose an approach to give web APIs a unified vision with linked data to allow them to be combined together in mashup applications, so we describe web APIs in a Linked Data style. This description contains useful information such as links to other APIs and possible interactions with Linked Data resources, links between APIs are automatically created using STRIM algorithm, this allows us to have a Linked Web API repository that can be combined using Linked Data principles.

The rest of the paper is organized as follows: the second section contains a quick presentation of fundamentals about linked data and the composition of RESTful services in mashups. The third section talks about related works and efforts in the description, the discovery and the composition of RESTful services. Our contribution is shown by detailing the architecture of the proposed approach in section four. Section five shows how our prototype runs under a specific scenario to illustrate how it does work. Finally, we discuss the results and we give perspectives of future work in section six.

II. BACKGROUND

A. *Linked Data*

The term linked data make reference to a new manner of publishing and interlinking structured data in the web, it is expressed by a set of practices which were introduced in the web architecture note Linked Data [7] of Tim Berners-Lee that initiates the following main principles [7]:

1. Use URIs to name things.
2. Make the look up of those names possible by using HTTP URIs.
3. Using the standards (RDF, SPARQL) to provide useful information when someone looks up a URI.
4. Allow the discovery of more things by including links to other URIs.

Linked data applies the architecture of WorldWide Web [8] to share structured data on the web in order to allow them to be interlinked.

The first linked data principle consists of using URI references to identify web documents, real world objects and abstract concepts [9].

The second principle introduces the use of HTTP protocol, which is a global method to access data and resources online. It relies on the combination of a special identification and an ordinary retrieval mechanism.

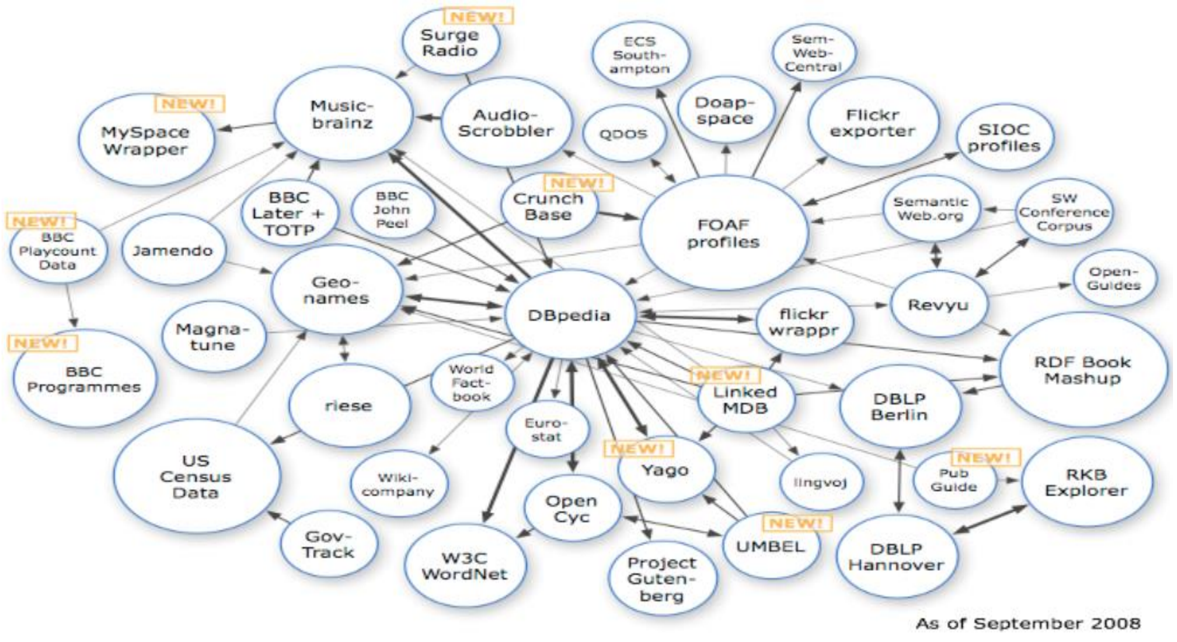


Fig.1. Linked Data

The third linked data principle imposes the use of Resource Description Framework (RDF) which is a simple graph-based data model essentially designed to be used in the context of the Web [10] for the publication of structured data.

The use of hyperlinks to connect web documents as well as any other type of things is imposed by the fourth Linked Data principle.

The Linked Data (Figure1), by definition [11], links the instances of multiple sources.

B. *Restful Services*

REST (REpresentational State Transfer) is the architectural style of the web [12]. It is essentially about using a range of design constraints that web services follow in order to preserve the good properties of the web. REST uses nouns and verbs for readability. The client sends a request to get a response about the corresponding resource.

A RESTful Web Service is a service that coheres with the REST principles in his interface and accessing mechanism. It is used over HTTP protocol and the resources are identified with URIs. The URI should be human-readable and easily identifiable and understandable. For example, a restful resource for a costumer can be identified as

“http://www.example.com/customer”

REST protocol imposes also stateless interactions between the client and the server so every request is treated independently from the previous ones.

The client has enough information about resources and the actions that he is authorized to perform on them thanks to the associated metadata. To use resources the client would send HTTP requests and receive HTTP responses.

Some of the requests used in REST are as follows [13]:

- GET: The GET request retrieves a representation of a resource from the server to the client
- POST: The POST request creates a resource on the server based on the representation given by the client.
- PUT: The PUT request is used to update or create a reference to a resource on the server
- DELETE: The DELETE request is used to delete a resource on the server
- HEAD: The HEAD requests checks for a resource without retrieving it.

Every resource can be provided in multiple representations allowing the client to choose the most suitable format of the information.

C. *Mashup*

Commonly we define a mashup as a web application that integrates data, application logic, and pieces of user interfaces (UIs) [14]. It gives the possibility to a simple web user to use existing web resources to create his own web application in accordance with his needs and preferences.

« Most mashups do more than simply integrate services and content. Sites that do mashups typically add value. They benefit users in a way that's different and better than the individual services they leverage »[15]

A mashup can combine several types of resources [16] it integrates web services and different types of API as well as data available on the web.

Five main benefits of using mashups that were identified in [17] are:

1. Provide an easier way to share information among organizations.
2. Permit an efficient reutilization of data resources and applications developed previously.
3. Allow analysis, reporting, information collection, and decision support,
4. Stimulate users of information systems to take part in the novel process of building information systems responding to the needs of each one.
5. Allow users to collaborate by sharing useful information.

REST (Representational State Transfer) is the architectural style of web services used in mashup applications, it uses URIs (Uniform Resource Identifiers) to identify web resources. The access to web-based data and rich user-interface controls is guaranteed by using AJAX.

III. Related Works

In order to increase the level of automation in the discovery and the composition of web services, several works were made to extend the classic description of web services by enhancing it with semantic capabilities to form Semantic Web Services by using ontologies [1,2] and Rule-based descriptions [3]. An example of such works is the one described in [4], where the authors proposed an approach to apply SWS technologies on RESTful web services by using WSMO-Lite [5] and [18], which proposes a graph-based web service composition framework.

The scripting language S that was proposed in [6] offers a definition of resources used in interactions between RESTful web services, in this approach the calculations during the discovery process are parallelized to ensure a good performance.

ReLL[19] is an Hypermedia property based approach to describe RESTful Web services. It considers a RESTful service as a set of interlinked resources described in a human readable language. A representation of a resource may contain links to other resource representations. In this approach, Petri Nets are used as a technique to describe the machine-client navigation.

LRDD [20] is an approach proposed by IETF[1] to describe Web resources that are identified by URIs. The resource descriptor link is indicated in the link field of the HTTP header. The descriptor contains a machine-readable information allowing better interactions between the resources.

RESTdesc [21] is a lightweight discovery process, which is based on Link headers, HTTP OPTIONS verb, and URI templates for description of RESTful Services. It returns information about the resources in *Notation3*[2] syntax that expresses the current resource interaction possibilities.

RESTdoc [22] relies on HTML links in order to point the descriptions of other resources. The links help to construct a graph by identifying the resources. The descriptions of services are augmented with semantic annotations to obtain machine-readable descriptions; the descriptions are then used to link related services enabling automatic discovery and composition.

LRA [23] proposes a middleware that provides execution plans for SPARQL queries invoking web APIs, and takes care of linking and combining their responses based on semantic annotation from linked data ontologies.

Authors of [24] propose a reconstitution of the existing Linked Data Platform to give a solution for communication needs of geographically distributed resources.

HTTP vocabulary in RDF [25] is a W3C working draft, which purpose is to indicate the HTTP protocol in RDF format. Its goal is to connect the REST paradigm with the concepts of the semantic web. It introduces a range of RDF classes and properties that aim to represent the HTTP specifications in form of concepts.

Among the approaches to facilitate the composition of RESTful web services we mention [26] that extends BPEL in order to support HTTP operations on RESTful resources to ensure their composition.

Work presented in [27] is another approach that proposes a semantic model for REST services. The proposed model is a formal definition based on a combination of pi-calculus [28] and approaches to triple space computing [29], which allows a rigorous description of resources and services.

Authors of [30], propose a framework to compose RESTful services by adding links to other RESTful resources in the resource description.

A linked web API dataset is developed in [31] to provide information about Web APIs, mashups and mashup developers based on the ProgrammableWeb.com[3].

The work presented in [32] is an extension of the linked USDL model to support different Web API datasets using linked data principles.

A semantic approach to discover and maintain links between RDF data based on the description models used by the providers of these resources is presented in [33].

In [34] the authors went further to track the provenance of the data resources and to dissimulate it in URIs.

IV. Proposed Approach

In this paper we propose an approach to allow the combination of Restful web APIs and Linked Data in a mashup application by providing a framework who unifies the vision of Linked Data with web APIs by

[1] Internet Engineering Task Force

[2] An RDF Syntax: http://www.w3.org/TeamSubmission/n3/s

[3] http://programmableweb.com

creating Linked Web APIs and providing CRUD methods to manipulate linked data.

To make linked Web API we have to make them join Linked Data principles, we use URIs over HTTP protocol to access APIs, provide an RDF description for each API so that it can provide useful information concerning the API, And then put external links to other API in the description to facilitate their integration, we also add links to external linked data resources to facilitate their utilization by web APIs.

To get this done we associate a descriptor to each Restful web API, each descriptor contains a number of information concerning the category, allowed operations, possible links with other web APIs and possible links with Linked Data.

A. *Linked Web API*

To create a linked web API repository, we made them subject to linked data principles by doing the following actions:

- Use URIs as names for Restful web APIs.
- Use HTTP URIs to allow people to look up those names.
- When someone looks up a URI, it provides useful information using standards (RDF, SPARQL)
- Include links to other URIs so that they can discover other APIs

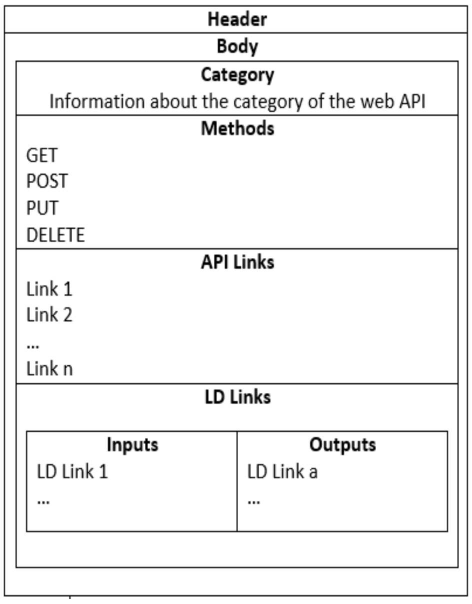


Fig.2. Web API Descriptor

In order to exploit a descriptor of a Web API, the user has to send a HTTP GET request with the address of the resource.

The WADL description file of the resource contains a link to the description file as shown in Figure3.

The descriptors are expressed in RDF and contain useful metadata corresponding to the web API as shown in Figure2, the header indicates the URI of the actual resource and in the body, we can find three important fields:

- Methods: contains information about the CRUD methods applicable on the resource
- API Links: contains external links to other web APIs
- LD Links: contains the possible inputs and outputs, which the API can have from external Linked Data resources, these information are joined with the methods that can be performed on the resource.

The web API descriptors help to discover APIs by means of their category, and then to detect links allowing to combine them together.

Figure4 represents an example of a description file for hotel reservation service; the part 1 contains information about possible methods, part 2 contains information about links with other services, and part 3 contains information about links with linked data.

B. *Exploitation of Linked Data*

The descriptor contains also information about linked data susceptive to be linked to the API as an input or an output. If there is no API suitable to be combined then the framework makes a research in the linked data to find a method (query) appropriate to be combined with this API.

The framework offers also a number of functions that ensure providing Linked Data in a Restful web API style. It implements CRUD methods applicable on linked data to create, modify, show or delete them to allow the manipulation of linked Data like Restful web API, each method is implemented by means of a SPARQL query, this gives us a set of Restful resources and Linked Data which can be manipulated by REST methods (POST, GET, DELETE, PUT) for web APIs and (LD_POST, LD_GET, LD_DELETE, LD_PUT) for Linked Data resources. In addition, the final user does not have to write SPARQL queries.

C. *Detection of Links between APIs*

In our framework we use STRIM (STRing based algorithm for Instance Matching) that we developed in [11] which gave very interesting results in ontology alignment in OEAI 2015, we use the same principle to detect links between the resources in order to facilitate the creation of the descriptors.

The algorithm consists of three phases:

Extraction and normalization

The algorithm extracts from each resource a set of information and properties then NLP techniques are used to normalize information in order to keep exclusively useful information, In particular, three pre-processing steps are performed: (1) Case conversion, (2) Stemming Lemmatization and (3) Stop words elimination.

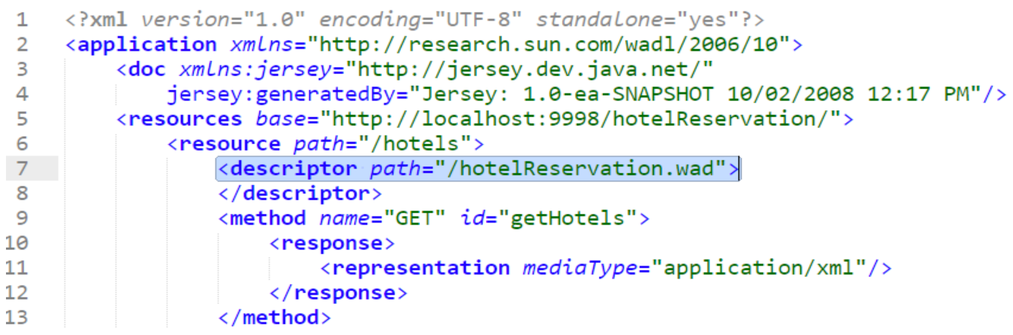
Similarity calculation

In this phase, the system uses edit distance as a String matcher to calculate similarity between normalized information and refers to the maximum similarity values to count the number of similar information between two resources.

Identification

Finally, in order to determine the correspondences between resources, we apply a filter on maximum counter values.

The selected correspondences are added to linked web API descriptors.



```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<application xmlns="http://research.sun.com/wadl/2006/10">
    <doc xmlns:jersey="http://jersey.dev.java.net/"
        jersey:generatedBy="Jersey: 1.0-ea-SNAPSHOT 10/02/2008 12:17 PM"/>
    <resources base="http://localhost:9998/hotelReservation/">
        <resource path="/hotels">
            <descriptor path="/hotelReservation.wad">
            </descriptor>
            <method name="GET" id="getHotels">
                <response>
                    <representation mediaType="application/xml"/>
                </response>
            </method>
```

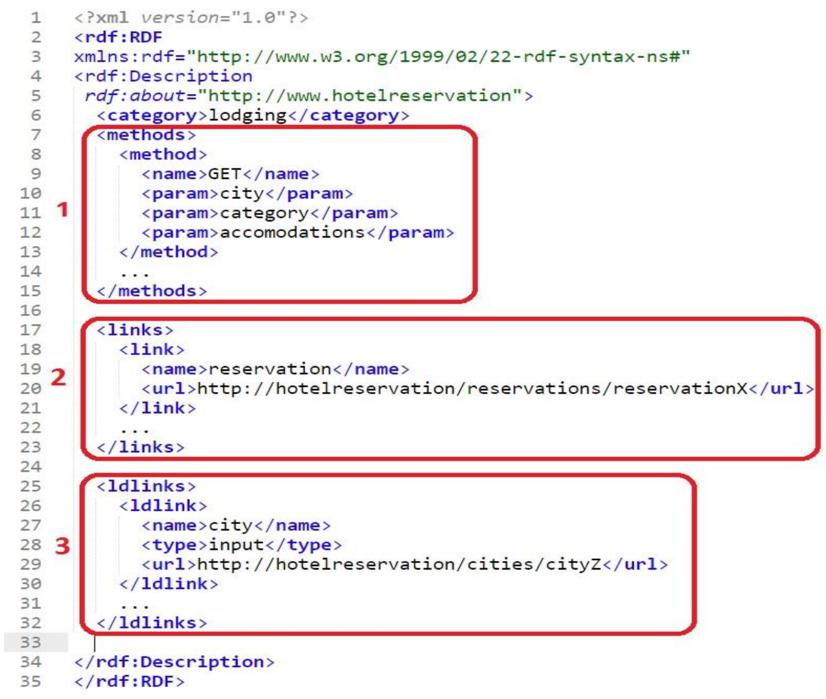
Fig.3. WADL file example

In order to test our framework, we used ISLab Instance Matching Benchmark to create a set of API descriptors and then we applied our algorithm to detect correspondences between APIs. The ISLab benchmark consists of four tasks: Value Transformations, Structural Transformations, Logical Transformations and a combination of the previous transformations.

Table 1 shows the results obtained after evaluating our matching algorithm, we can see that it had very good results especially in F-measure and Recall.

Table 1. Results for ISLab benchmark

Track	Precision	F-measure	Recall
Value Transformations	90%	95%	99%
Structural Transformation	98%	98%	99%
Logical Transformation	91%	94%	99%
Combination	91%	94%	99%



```xml
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
<rdf:Description
 rdf:about="http://www.hotelreservation">
  <category>lodging</category>
  <methods>
    <method>
      <name>GET</name>
      <param>city</param>
      <param>category</param>
      <param>accomodations</param>
    </method>
    ...
  </methods>

  <links>
    <link>
      <name>reservation</name>
      <url>http://hotelreservation/reservations/reservationX</url>
    </link>
    ...
  </links>

  <ldlinks>
    <ldlink>
      <name>city</name>
      <type>input</type>
      <url>http://hotelreservation/cities/cityZ</url>
    </ldlink>
    ...
  </ldlinks>

</rdf:Description>
</rdf:RDF>
```

Fig.4. Web API Descriptor File

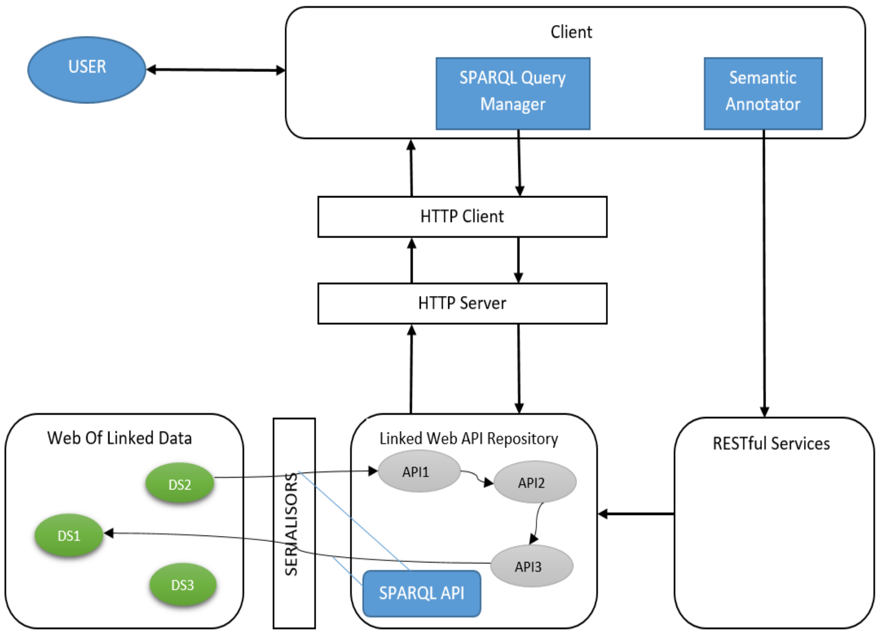


Fig.5. General Architecture

D. *General Architecture*

The figure 5 shows the general architecture of our approach, the main components are:

Client Application: This application forms the interface between the user and the components he wants to combine to create a composite application responding to his needs.

Semantic Annotator: It allows the user to annotate Restful services in order to create links between services to facilitate their combination.

SPARQL query manager: The interface allows the user to exploit information about the linked web APIs by using simple methods, which implements SPARQL queries, so the user does not need to write SPARQL queries.

Linked Web API Repository: It contains a set of web APIs interconnected between them thanks to the descriptors edited by the semantic annotator.

SPARQL API: In order to use Linked Data, the SPARQL API transforms simple CRUD methods used by the APIs to SPARQL queries.

E. *Scenario*

To illustrate our approach we will take the scenario of hotel reservation:

In the following, we present the resources that we implemented in our system and their URIs:

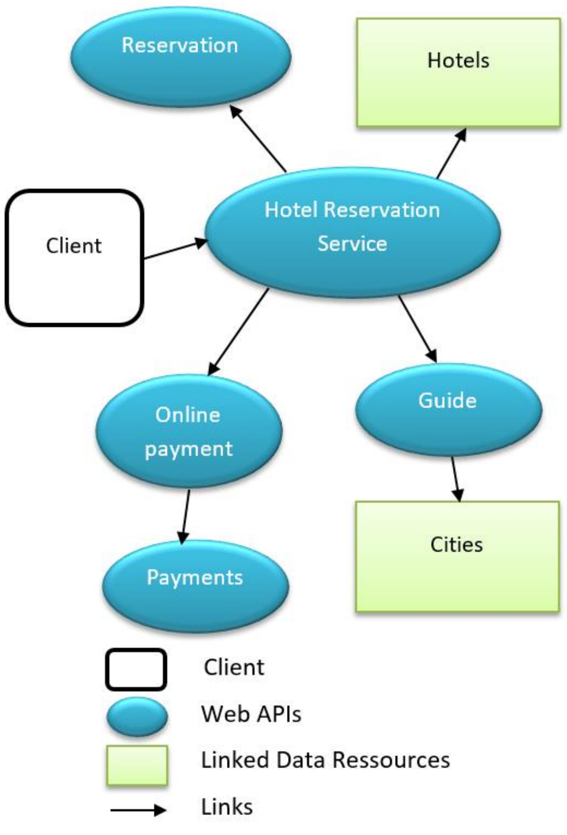


Fig.6. Service and Linked Data discovery

- *Hotel/hotelreservation/hotels/hotelT*
- *Reservation/hotelreservation/ reservations/reservationX*
- *Payment/hotelreservation/ payments/paymentY*
- *City/hotelreservation/cities/cityZ*

The client machine will search for the Hotel Reservation Service which will be the entry point for this system, so when sending the GET request for Hotel Reservation Service, the response will contain the links to other APIs: Reservation, Guide and Online Payment Service and a link to a Liked Data resource: Hotels as an input.

To choose a hotel the user has to choose a city first, so he will invoke the Guide service. A GET request to Guide will return a link of a Linked Data resource, which is Cities as an input.

The user will browse the list of cities to choose a destination, once this choice made, the Guide service will return the name of the city to the Hotel Reservation Service, which can take input from a list of Hotels.

A LD_GET method is executed on Hotels Linked Data resource to browse the hotels available in the chosen city, the user makes his choice, so the Hotel Reservation Service has the necessary input to make a Reservation, a POST operation is executed on Reservation to record the new reservation with a 'False' value of the attribute PAID.

To accomplish the payment, the payment service is invoked, and then a POST operation is executed on Payments to add a new payment recording.

Once the payment executed, a PUT is operated on Reservation to change the value of PAID attribute from 'False' to 'True'.

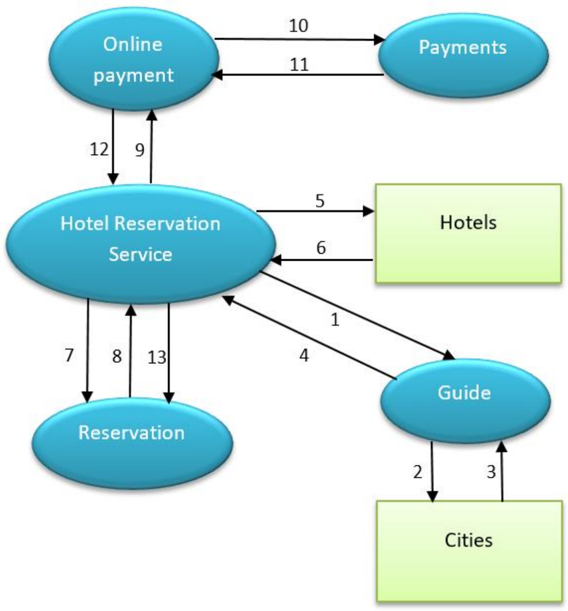


Fig.7. Scenario running

V. Conclusion

In this paper, we proposed an approach to allow the combination of Restful web APIs and Linked data in mashup applications. We validated the feasibility of our approach by implementing a framework, which helps the user to build his mashup application. Our approach relies on the RDF description of Restful Web API that contains links to other APIs and interactions with Linked Data resources. Links between APIs are automatically detected using STRIM algorithm. The framework also offers an interface to guide the user in the choice of the APIs to be selected and a set of methods to manipulate Linked Data. However, the process of composition requests a high interaction with the user, so we have as a perspective to automate the process throw semantic enrichment or other types of reasoning capabilities.

References

[1] R. Studer, S. Grimm, A. Abecker, (eds.): Semantic Web Services: Concepts, Technologies, and Applications. Springer (2007)

[2] D. Fensel, H. Lausen, A. Polleres, de Bruijn, J., Stollberg, M., Roman, D., Domingue, J.: Enabling Semantic Web Services: The Web Service Modeling Ontology. Springer (2006)

[3] J. Cardoso, A. Sheth: SemanticWeb Services, Processes and Applications. Springer (2006)

[4] J. Kopecky, T. Vitvar, D. Fensel: Microwsmo: Semantic description of restful services. Tech. rep., WSMO Working Group (2008).

[5] T. Vitvar, J. Kopecky, D. Fensel: WSMO-Lite:Lightweight Semantic Descriptions for Services on the Web. In: 5th IEEE European Conference onWeb Services (ECOWS 2007), pp. 77–86.

[6] D. Bonetta, A. Peternier, C. Pautasso, W. Binder : S: a Scripting Language for High-Performance RESTful Web Services, Proc. of the 17 th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP 2012), New Orleans, USA, pp. 97-106, February 2012.

[7] T. Berners-Lee. Linked Data - DesignIssues, 2006. http://www.w3.org/DesignIssues/ LinkedData.html. 7,26,82.

[8] I. Jacobs, N. Walsh. Architecture of the World Wide Web, Volume One, 2004.http://www.w3.org/TR/webarch/. 7,9

[9] T. Heath and C. Bizer. Linked Data: Evolving the Web into a Global Data Space (1st edition), volume 1 of Synthesis Lectures on the Semantic Web: Theory and Technology, , 1:1, 1-136. Morgan & Claypool, 2011.

[10] G. Klyne, J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax - W3C Recommendation, 2004. http://www.w3.org/TR/rdf-concepts/. 8,15,17

[11] A. Khiat , M. Benaissa , M. Belfedhal, STRIM Results for OAEI2015 (Ontology Alignment Evaluation Initiative) Instance Matching Evaluation. The Tenth International Workshop on Ontology Matching OM-15 collocated with the 14th International Semantic Web Conference ISWC-2015.

[12] R. T. Fielding. Architectural styles and the design of network-based software architectures. Ph.D. Dissertation, University of California, Irvine, 2000.
[13] M. Kalali and B. Mehta. Developing RESTful Services with JAX-RS 2.0, WebSockets, and JSON. Birmingham: Packt Publishing, 2013.
[14] F. Daniel, J. Yu, B. Benatallah, F. Casati, M. Matera, and R. Saint-Paul. Understanding UI Integration: A survey of problems, technologies. Internet Computing, 11(3):59{66, May/June 2007.
[15] E. Ort, S. Brydon, M. Basler Mashup Styles, Part 1: Server-Side Mashups .Oracle: Sun Developer Network, May 2007.
[16] L. Clarkin, J. Holmes, Enterprise Mashup, The Architecture Journal, MSDN Architecture Center, 2008
[17] A. Majchrzak, J.T. Maloney. (2008) *Enterprise Mashups: What Do They Mean for CIOs?* Chicago, IL: Society for Information Management Advanced Practices Council, 2009.
[18] P. Rodriguez Mier, C. Pedrinaci, M. Lama, M. Mucientes, An Integrated Semantic Web Service Discovery and Composition Framework, IEEE Transactions on Services Computing 9(4), 537-550, 2016.
[19] R. Alarcón, E. Wilde, J. Bellido: Hypermedia-Driven RESTful Service Composition. In: Maximilien, E.M., Rossi, G., Yuan, S.T., Ludwig, H., Fantinato, M. (eds.) ICSOC Workshops. Lecture Notes in Computer Science, vol. 6568, pp. 111–120 (2010).
[20] Internet Engineering Task Force (IETF): LRDD Internet Draft. https://tools.ietf.org/html/draft-hammer- discovery-06
[21] R. Verborgh, T. Steiner, D.V. Deursen, J.D. Roo, R.V De Walle, J.G. Vallés: Description and Interaction of RESTful Services for Automatic Discovery and Execution. In: Proceedings of the International Workshop on Advanced Future Multimedia Services, Future Technology Research Association International (FTRA), 2011.
[22] D. John, M.S. Rajasree: RESTDoc: Describe, Discover and Compose RESTful Semantic Web Services using Annotated Documentations. International Journal of Web & Semantic Technology (IJWesT) Vol.4, No.1, January 2013.
[23] D. Serrano, E. Stroulia, D. Lau, Linked REST APIs: A Middleware for Semantic REST API Integration, Web Services (ICWS), 2017 IEEE International Conference on, 138-145, 2017.
[24] H. Ferguson, C. Vardeman, J. Nabrzyski: Linked data platform for building cloud-based smart applications and connecting API access points with data discovery techniques. Big Data, IEEE International Conference on, 3016-3025, 2016.
[25] J. Koch, C.A. Velasco, P. Ackermann: HTTP Vocabulary in RDF 1.0. W3C Working Draft,, http://www.w3.org/TR/HTTP-in-RDF10/
[26] C. Pautasso: RESTful Web service composition with BPEL for REST. Data Knowl. Eng. 68(9), 851–866 (2009)
[27] A.G. Hernandez, M.N.M Garcıa: A formal definition of restful semantic web services. In: WS-REST. pp. 39–45 (2010).
[28] R. Milner: The polyadic pi-calculus. In: CONCUR (1992)
[29] D. Fensel: Triple-space computing: Semantic web services based on persistent publication of information. In: INTELLCOMM. pp. 43–53 (2004).
[30] M. Bennara, M. Mrissa, Y. Amghar, An Approach for Composing RESTful Linked Services on the Web. In: 23rd International Wide Web Conference, WWW 2014, pp. 977-982 (2014)
[31] M. Dojchinovski, T. Vitvar, Linked Web APIs Dataset: Web APIs meet Linked Data, Semantic Web 1, 1-5, IOS Press, 2015.
[32] S. Omelkova, P. Kungas, A Linked Data Model for Web API-s, Perspectives in Business Informatics Research: 14th International Conference, Proceedings 48-63, 2015.
[33] Fatima Ardjani, Djelloul Bouchiha, Mimoun Malki,"An Approach for Discovering and Maintaining Links in RDF Linked Data", International Journal of Modern Education and Computer Science(IJMECS), Vol.9, No.3, pp.56-63, 2017.DOI: 10.5815/ijmecs.2017.03.07
[34] Kumar Sharma, Ujjal Marjit, Utpal Biswas,"PTSLGA: A Provenance Tracking System for Linked Data Generating Application", IJITCS, vol.7, no.4, pp.87-93, 2015. DOI: 10.5815/ijitcs.2015.04.10

Authors' Profiles

Mohammed Amine Belfedhal received M.Sc in computer science from Djilali Liabes University (Algeria) in 2011. He is pursuing PhD in Computer Science and Engineering from Djilali Liabes University. He is presently working as Research Scientist in Enterprise mashups and Cloud Computing, web service engineering and linked data. He is a member of EEDIS Laboratory.

Mimoun Malki is currently a full professor at the Higher School of Computer Science of Sidi Bel-Abbes. He is the head of the LabRI-SBA Laboratory. His research interests include Databases, Information Systems Interoperability, Ontology Engineering, Semantic Web Services, Linked Data Services, Web Reengineering, Enterprise Mashup and Cloud Computing. He has published papers in well-known journals (IEEE Transactions on Knowledge and Data Engineering, World Wide Web Springer, etc.)

How to cite this paper: Mohammed Amine Belfedhal, Mimoun Malki, "MASHUP of Linked Data and Web API", International Journal of Information Technology and Computer Science(IJITCS), Vol.10, No.6, pp.64-71, 2018. DOI: 10.5815/ijitcs.2018.06.07