ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

Representational State Transfer API Life Cycle Management

Madhuri Surendrakumar Warvante, Dr. R. B. Ingle

M. E Student, Department of Computer Engineering, Pune Institute of Computer Technology Pune, Maharashtra, India

Dean and HOD, Department of Computer Engineering, Pune Institute of Computer Technology Pune, Maharashtra, India

ABSTRACT: REST (Representational State Transfer) is stateless, client-server, cacheable communications protocol. REST uses HTTP for all four CRUD operations. In REST each data element is a resource, addressed by a URI. Media type is the name of a specific format or schema of a representation. Within an organization all API should be consistent and should have easy access. While designing REST API all information should be stored at one place for standardization. For that there will be one repository for Mime-type and Swagger documents. Developers will consume REST APIs and they need documentation, for that Developers Portal is required. It is useful for user to get all the information at one place. It will use repository created earlier. There should be some automation which will check implemented REST API's matches with Documented REST API's. All above steps are called as REST API life cycle management tool. By using this tools it will be easier to integrate with other tools.

KEYWORDS: Mime-types, Open API, REST API, swagger.

I. INTRODUCTION

Representational State Transfer (REST) [5] relies on a stateless (no state is saved), client-server (request/response), cacheable communications protocol and in virtually all cases, the HTTP protocol is used [5]. REST is used to design networked applications. The idea is that, rather than using complex mechanisms such as CORBA, RPC or SOAP to connect between machines, simple HTTP is used to make calls between machines. If we used SOAP or CORBA then there is requirement for client side to use the same because there is tight coupling. In many ways, the World Wide Web can be viewed as a REST-based architecture [12] because it is based on HTTP. RESTful applications use HTTP requests to post data (create and/or update), read data (e.g., make queries), and delete data [14]. Thus, REST uses HTTP for all four CRUD (Create/Read/Update/Delete) operations. In REST two software can be integrated. In REST each data element is addressed by a URI(Uniform Resource Identifier) and it a resource.REST was defined by Roy Thomas Fielding in his 2000 PhD dissertation "Architectural Styles and the Design of Network-based Software Architectures [6]". We can manage APIs throughout the entire lifecycle of the API, from planning and design, to development, operations and maintenance of it [12].

A. *RESTful API:*

Representational State Transfer is an architectural style helps to achieve fast performance, reliability and the ability to scale. In REST reusable components can be managed or updated without affecting whole system [9].

B. *RESTful API Lifecycle:*

RESTful APIs that enable you to integrate their tools into your own development lifecycle.We can use RESTful APIs to writescripts or code that programmatically deploy REST API web services, or that migrate REST API services from one environmentto another, as part of a larger automated process that also deploys or migrates other applications [13]. As shown in fig.1 RESTful API lifecycle have different states.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

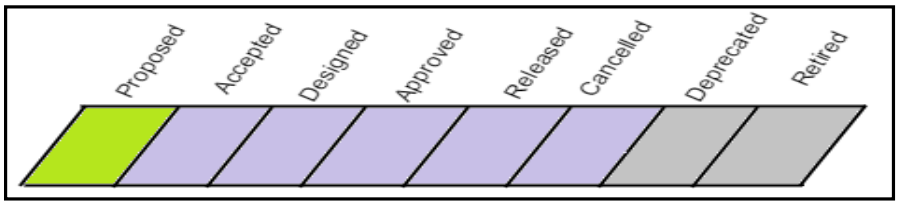

Fig.1Common REST API Service Development Lifecycle image source: [13]

- Proposed: This state shows new API or version of new API being proposed. A new proposed API is presented to the API Governance Board, and they will evaluate it from a product and technical perspective. The board will decide whether to reject or approve the proposed API.
- Accepted: This state shows API is reviewed by the API Governance Board and approved to move further.
- Designed: The API goes to design and its specifications are saved into a specifications document. After design is completed, the API will then be presented to the architecture review for final approval.
- Approved: Once it passes the architecture design review, the API service is approved, and authorized for production release.
- Released: The API Service is live in Production.
- Cancelled: API are cancelled either by the API Governance Board or by the author(s) depending on the life cycle stage when the decision is made.
- Deprecated: When new version of API has been deployed or when the product is no longer part of company portfolio then An API is deprecated.
- Retired: A retired API is no longer deployed in the production environment and is no longer in use.

The rest of the paper is structured as follows. In section IIrelated work on REST and REST API is given. In section III proposed system for REST API life cycle management with block diagram is given. Pseudo code is described in Section IV. Section V describes Simulation Results. Section VI closes the paper with conclusion and future work.

II. RELATED WORK

REST is a term coined by Roy Fielding to describe an architecture style of networked systems. Roy T. Fielding defines REST in [6] as coordinated set of architectural constraints that attempts to minimize latency and network communication while at the same time maximizing the independence and scalability of component implementations. REST enables the caching and reuse of interactions, dynamic substitutability of components, and processing of actions by intermediaries, thereby meeting the needs of an Internet-scale distributed hypermedia system.

HTTP defines a set of methods, in [7] they used GET, PUT, POST and DELETE. Additional methods like HEADER and OPTIONS can be used to employ improved caching (HEADER), and return a service description (explanation of available methods etc.) of the given URI (OPTIONS).

According to The Definitive Guide to API Management [1] Ebook by apigee An API management solution needs to include at least thesecapabilities: Developer portal, API gateway and API lifecycle management. API lifecycle management to manage the process of designing, developing, publishing, deploying, and versioning APIs.

A Comprehensive Solution for API Management, an Oracle White Paper [2] describes create, annotate, publish, Portal, Deploy these steps for managing API's. In addition to these steps we can provide security, performance, and scalability.

ORACLE API MANAGEMENT [4] provides end to-end lifecycle management of APIs. Through a centralized repository, policy enforcement, and tracking of key performance indicators, the solution provides the foundation required to deliver business value. There are no ways for validating any combination for available solutions.

Developing RESTful Web Services with Webmachine [3] uses HTTP methods and conditional GET.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

III. PROPOSED SYSTEM

Now a days REST has become industry standard, Delivering standard and consistent API with quality is challenge for an organization. For that some tooling is required. This section is divided into number of steps, these are as following.

1. Central Repository
2. Developers portal
3. Validation
4. Comment / Review automation

A. *Central Repository:*

Swagger is a specification for documenting REST API [15] [11]. For easy access we stored all swagger documentat one place and meta data of all swagger document into one relation. We have provided GET, PUT, POST and DELETE http methods.

- If user want to post a new swagger document then content of his swagger document should be provided in body of POST request. This content will be validated against swagger schema version 2. If it's a valid swagger document then only the next step is executed otherwise exception is thrown. Then it will check the entry for given base path is there in repository or not if entry is there then it will check if the content of swagger for given base pathare same or not, if contents are same then it will do nothing else it will create new revisionId(revisionId is creation timestamp). If there is no entry for given basepath then it will create new entry. Then the swagger document is stored into repository.
- GET is used to get valid swagger document.User can get metadata information or user can get whole swagger document into json or yaml format depending on media-type. There are different API's for GET request based on user's requirements.
- User can replace meta information of open API or he can replace whole swagger document by using PUT request. PUT is somewhat similar to POST method. In put it will first check if the entry for swagger document given by basepath is present into repository or not, if it is present then it will check content of swagger document, if content are same then it will do nothing, if content's are different then it will create new entry given by this basepath with new revisionId, if there is no entry for given basepath then it will create new entry.
- User can delete open API by providing basepath in URI. It will delete all versionIds and all revisionIds for given basepath.

B. *Developer's portal:*

This contains design and development of website so that it will be easy for a user to get all the information at one place. Developer’s portal uses repository created earlier. There are two portals one is internal and another one is external portal. External user/developer can give query to external portal and get efficient information. Reviewer will review all the apis. Reviewer will decide the status of api whether it should be public or private. if status is public then only external portal can access it. Private apis will be hidden from external portal. This will provide security. Which provides authentication and authorization. Product manager is responsible to make apis public

C. *Validation:*

Validation can be done

- Before posting swagger document to the repository (content validation)
- Implemented rest api will be validated as per documented rest apis

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

D. ***Review comments automation:***

Developer or Reviewer can add comment or review. External user or developer can't add comment or review. Status of comment will be update automatically. This system will handle orphaned comments.

- Let, $t_p$be the time required to post the swagger documents,
- t_v be the time required to validate the swagger document,
- t_s be the time required to store the swagger document,
- t_r be the time required to review the swagger document,
- t_{v1} be the time required to validate implemented REST API as per documented REST API,
- And t_e be the error

Then the whole REST API life cycle will take T time and it is given by,

$$T= t_p+ t_v + t_s + t_r + t_{v1} + t_e \quad \text{eq. (1)}$$

- Developer's portal is a website which will give all relevant information related to user's query.If server is down then user have to wait till it's working.

$$\text{Mean Time Between Failures} = \frac{total\ up\ time}{number\ of\ breakdowns} \quad \text{eq. (2)}$$

$$\text{Mean Time To Repair} = \frac{total\ down\ time}{number\ of\ breakdowns} \quad \text{eq. (3)}$$

- "Mean Time" means, the average time.
- "Mean Time Between Failures" is the average time elapsed from one failure to the next.
- "Mean Time To Repair" is the average time that it takes to repair something after a failure.
- We can also measure performance in terms of response time, memory required,etc.

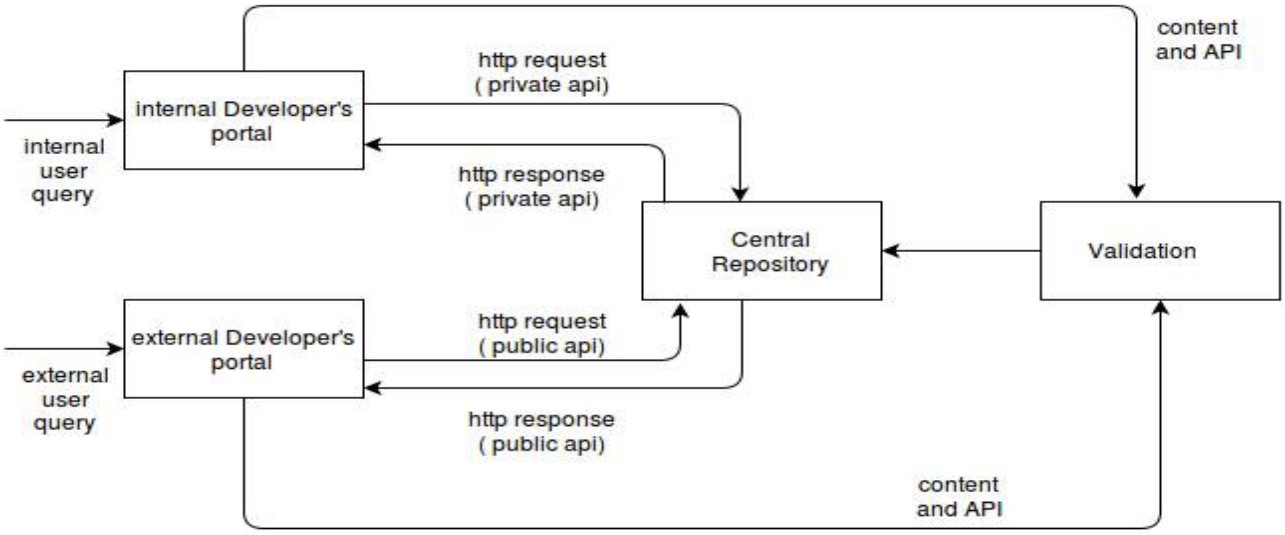

fig. 2 Proposed System: REST API life cycle management

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

IV. PSEUDO CODE

Pseudo code for post request:
Step 1: First user needs to give swagger document's content into body of post request.
Step 2: If(empty) throw badRequestException
Step 3: Else call validate()
Step 4: If valid then check entry for given basepath is already present into repository,if yes then check contents are same or not if contents are same then do nothing else add new revisionId and call file service else throw badRequestException. If entry is not there for given basepath then add new entry.
Step 5: File service will save file into repository.
Step 6: Return metadata of file.

V. RESULTS

The proposed system is implemented in java. In fig. 3 result for POST, GET, PUT, DELETE APIS are shown.

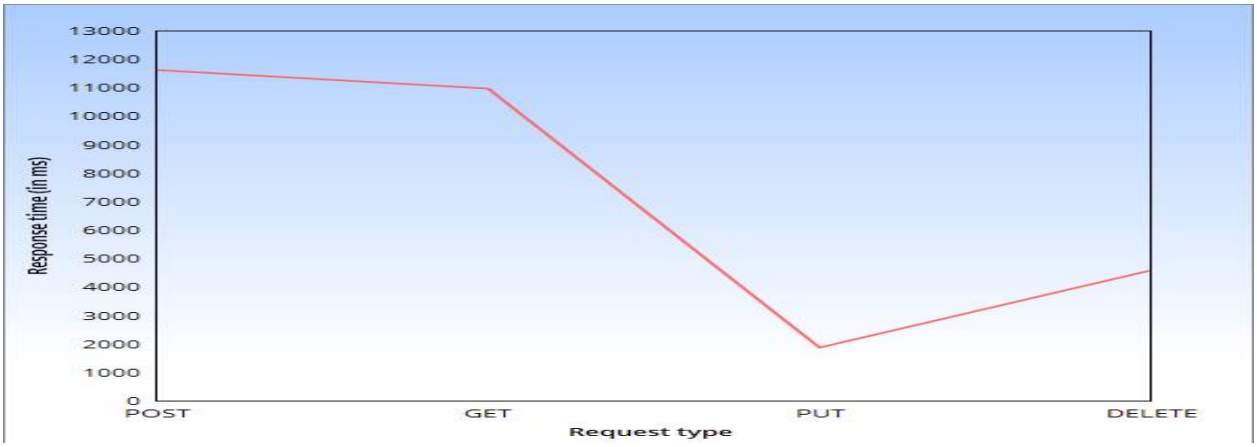

Fig.3. Response Time (in ms) for each method

VI. CONCLUSION AND FUTURE WORK

Due to some advantages of REST over SOAP it is convenient to use REST. REST uses all Http methods for all four CRUD operations. To maintain single version of truthstore all swagger document at one place. Central repository helps to easily accessswagger documents. Developers will consume REST APIs and they will need documentation forthat developer's portal is required.

The future work can be optimization of steps required to manage REST APIs. If user is not bothered about security then one can remove security from REST API management. Likewise organization can customize these steps as per user's requirement.

VII. ACKNOWLEDGEMENT

We would like to thank Mr.Amit Babhale for providing guideline. We would like to thank Mr.Arvind Jagtap for his valuable support. We would like to thank SAS Research and Development, Pune for giving opportunity to work on this project. We would like to thank PICT college staff for supporting us.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

REFERENCES

1. Bruno costa, Paulo F. Pires, Flavia V Delicato, Paulo Merson, “Evaluating REST architectures - Approach, tooling guidelines”, Journal of System and Software Volume 112, February 2016, Pages 156-180.
2. Helen Whelan, “The Definitive Guide to API Management, apigee”, 20 July 2015.
3. David Baum “A Comprehensive Solution for API Management”, oracle white paper, 1 Oct 2015.
4. Justin Sheehy, Steve Vinoski, “Developing RESTful Web Services with Webmachine”, IEEE Computer Society, MARCH/APRIL 2010.
5. David Baum, “ORACLE API MANAGEMENT”, oracle white paper, 1 Oct 2015.
6. Michael Jak l, “Representational State Transfer”, University of Technology Vienna,2006.
7. Florian Haupt, Frank Leymann, Cesare Pautasso, “A conversation based approachfor modeling REST APIs”, Institute of Architecture of Application Systems, 2016.
8. Megan Lunde, Richard May,"API Governance", https:projects.tmforum.org/wiki/display/TP/API + Governance, 08 Jul 2016.
9. Chris Haddad, “Apply API Governance to RESTful Service APIs using WSO2 Governance Registry and WSO2 API Manager”, http://www.slideshare.net/wso2.org/apply-api-governance-to-restful-service-apis-using-wso2-governance-registry-and-wso2-api-manager, 2011.
10. Roberto Medrano, “The Reality of API Lifecycle Management”, https://blog.akana.com/api-lifecycle-management/, June 9, 2013.
11. Lisa Greene, “RESTful API”, BMC Cloud Lifecycle Management 4.6, 22 oct 2016.
12. Guy Levin, "REST API, API Driven Development, Development Lifecycle", http://blog.restcase.com/development-lifecycle-of-an-api - service, 20 DECEMBER 2015.