International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 06 | June 2020 www.irjet.net p-ISSN: 2395-0072

Survey Paper: Framework of REST APIs

Sujan Y M[1], Dr. Shashidhara H R[2], Dr. Rohini Nagapadma[3]

[1]*Student , Networking and Internet Engineering , Dept. of EC&E, The National Institute of Engineering, Mysuru, Karnataka, India*
[2]*Associate Professor, Dept. of EC&E, The National Institute of Engineering, Mysuru, Karnataka, India*
[3]*Professor, Dept. of EC&E, The National Institute of Engineering, Mysuru, Karnataka, India*

Abstract - *Discussions are a notable idea in administration configuration to depict complex communications between a customer and one or then again various administrations. The REST building style compels the qualities of customers, servers and their connections in REST structures, which truly affects discussions in such frameworks. REST API is mainly used in various industries and are used in cloud computing, Internet of Things, and Micro services. Representational state transfer (REST) is a type of software architecture, which has more flexibility and is widely used in web services. REST determines how the API looks like.*

Key Words: **RESTful web service REST, RESTful, Web Services, API, HTTP, Web, Web Services Analysis, CRUD, HMAC.**

1. INTRODUCTION

Restful API is a simple, flexible and easy maintainable API that is built on the REST architecture. Restful Architecture, proposes the APIs from the client application in a stateless manner. The client can perform different operations using the Restful service. The underlying protocol for REST is HTTP [3]. REST stands for Representational State Transfer. The key abstraction of information in REST is a useful resource. Any records that may be named may be a useful resource: a report or picture, a temporal service, a group of different assets, a non-virtual item (e.g. someone), and so forth. REST uses a useful resource identifier to discover the specific resource worried in an interaction between components. Resources have relationships with different resources and a fixed of techniques or verbs to perform between those resources. Then you may have a group of sources that could interact as a group with one or extra sources or collections.

REST architecture is based on seven properties [9]:

•Performance – different elements interact and describes the ability of the system.
•Scalability - supports large number of elements.
•Simplicity - easy interacting interfaces.
•Modifiability - changing of the components as per the need.
•Visibility - clear communication between components.
•Portability - of the data-filled code.
•Reliability - resistance to fail at system level.

2. RESTFUL KEY ELEMENTS

The key elements of a RESTful implementation are as follows [1]:

Resources – The first key element is the resource. Let us assume the example of a server containing the data of several employees. Let us assume the URL of the web application is http://aaa.bbb.com. So in order to access an employee record resource via REST, one can use the command http://aaa.bbb.com /employee/1 - this command displays the data of the employee 1 that is stored in the web server and fetches the data from the server in different formats such as XML or json.

Request Verbs - The request verbs describes what needs to be done with the data that is obtained from the server. The browser uses a GET request to instruct the system to get or fetch the data from the server. There are different other request verbs that can be used other than GET they are GET PUT POST DELETE So in the example http://aaa.bbb.com/employee/1 , the browser is actually using a GET Verb because it wants to get the details of the employee record.

Request Headers – Headers are the different additional information that is being sent along with the data this headers also contains the different format in which the data needs to be fetched. The headers also consists of the different authorization and authentication methods used.

Request Body - Data is posted with the request. Data is normally sent in the request when a POST request is made to the REST web service. In a POST call, the new data that is sent is added to the server if the data is not present in the server before. Therefore, the request body contains the details of where the new data needs to be added in the web server.

Response Body –Response body is the body or the details of the response that we get from the request that is being sent. Therefore, in our example, if we were to query the web server via the http://aaa.bbb.com /employee/1, the web server might return an XML document with all the details of the employee in the Response Body.

Response Status codes – Response status code are those codes that indicates the status of the response from the web server. There are different codes that indicates the status of each response. For example, the code 201 indicates that the new resource has been created in the web server without error.

REST defines six constraints, which make any web service – a true RESTful API [5].

A. Uniform interface
B. Client–server
C. Stateless
D. Cacheable
E. Layered system
F. Code on demand

A. Uniform Interface

A resource in the system should have only one logical URL, and that should provide a way to fetch related or additional data having uniform interface make it easy to be operated with different operating system and different interfaces.

B. Client-Server

The client and server technique is used to fetch the data in the REST API process. The client sends a request to the server and the server responds back with the data that is stored in it.

C. Stateless

Client-server interacts stateless. The client server architecture that is mainly used in the REST API does not contain the information of the any HTTP request and the history of the calls and the session details.

D. Cacheable

The cache is used in the web service in order to fetch the data from the server in a quick manner so that the transaction speed of the process or the system would be faster and the results could be used in a faster way by the other sources.

E. Layered System

REST uses the layered system architecture. The REST architecture comes in the application layer of the OSI model.

F. Code on Demand

The server can extend the functionality of the client on the requirement of the user and can add the functionality on the run time.

3. CLIENT SERVER

The client-server architecture is also termed as a network-computing structure because every request and their associated replies are distributed over a network. [4]. In the client server mode the client and the server interacts in a very meaningful way in order to process the data the client server architecture is a similar model to master and slave system. In the client server mode the client sends the request to the main or the central repository server the server processes the data, searches the data in the server memory, and sends the data back to the client for it to use. The client and the server can be connection less or connection oriented. There are different scheduling methods in order to process the data from the client. The client server architecture - client sends the data to the server, and the server manages hosts and delivers the data to the client.

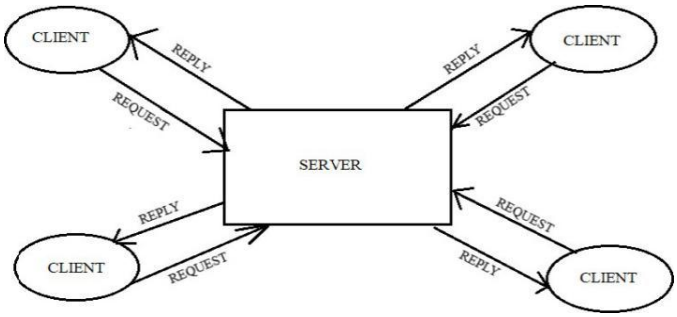

Fig-1: Client Server Architecture

4. RESTFUL CONVERSATIONS

A Restful discussion commonly means many correspondence or collaboration exercises between at least two members in the framework. Concerning REST, we center on the correspondence between a customer and a Restful Web API, i.e. a lot of resources. Discussions between a customer and many resources, what we call Restful discussion, can be described as follows.

There are two sorts of individuals in a Restful conversation. Clients are communicating with an API to fulfill a specific target. Resources are the structure squares of each Soothing Web API; they give a uniform interface enabling to get and adjust their state. A participation between a client and an API may realize the creation or deletion of its advantages, or in the recuperation and update of the depiction of its benefits.

The correspondence natives used in a conversation are given by the uniform interface of the REST plan. In the occasion of APIs using the HTTP convention, every correspondence is begun by the client and involves an interest followed by a response message. Along with the benefit identifier, every sales message joins the HTTP activity word (e.g., GET, PUT, POST, DELETE) [2] characterizing the movement to be performed on the advantage.

5. HTTP METHODS FOR RESTFUL SERVICES

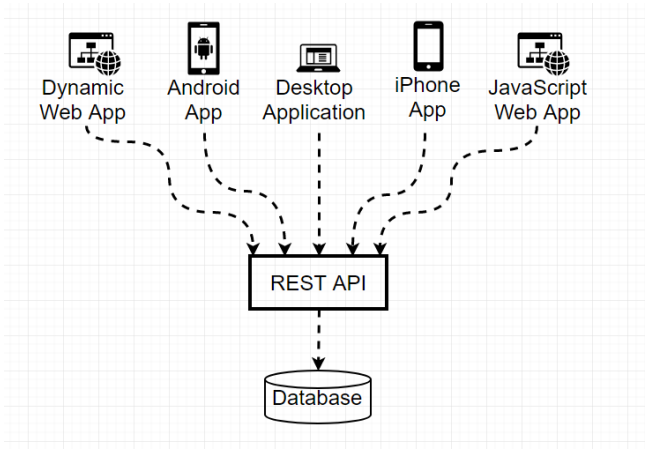

Fig- 2: RESTAPI Overview

The four main HTTP methods can be mapped to CRUD (Create, read, update and delete) operations as follows [6]:

A. GET.
B. PUT.
C. POST.
D. DELETE.
E. PATCH.

A.GET

The GET method is to retrieve the requested information from the server and is identified by the Request-URL. If the Requesting-URL refers to a data-processing process, then the data is returned by the requesting entity as the response, unless that text happens to be the output of the process. Use GET requests to retrieve resource representation/information only – and not to modify it in any way. It is said to be safe method because it does not cause any changes to the resources stored.

Ex: HTTP GET http://www.aaabbbccc.com/users/123.

B.PUT

Use PUT APIs is primarily used to update existing resource (if the resource does not exist, then API may decide to create a new resource or not). If a new resource has been created by the PUT API, the origin server MUST inform the user agent via the HTTP response code 201 (Created) response and if an existing resource is modified, either the 200 (OK) or 204 (No Content) response codes should be sent to indicate successful completion of the request.

Ex: HTTP PUT http://www.aaabbbccc.com/users/123.

C.POST

Use POST APIs to create new resources, i.e., the data is strictly created in the server .Talking strictly in terms of REST; POST methods are used to create a new resource into the collection of resources. Ideally, if a resource has been created on the origin server, the response SHOULD be HTTP response code 201 (Created) and contain an entity which describes the status of the request and refers to the new resource, and a Location header.

Ex: HTTP POST http://www.aaabbbccc.com/users/123.

D.DELETE

DELETE APIs are utilized to DELETE assets. DELETE activities are idempotent. On the off chance that you DELETE an asset, it is expelled from the assortment of assets. Over and again calling DELETE API on that asset will not change the result – nevertheless, calling DELETE on an asset a subsequent time will restore a 404 (NOT FOUND) since it was at that point evacuated.

Ex: HTTP DELETE http://www.aaabbbccc.com/clients/123.

E.PATCH

HTTP PATCH requests are to make partial update on a resource. If you see, PUT requests also modify a resource entity so to make clearer – PATCH method is the correct choice for partially updating an existing resource and PUT should only be used if you're replacing a resource in its entirety.

Ex: HTTP PATCH /users/1

[{"op":"replace","patch":"/email","value":"new.aaa@bbb.org"}]

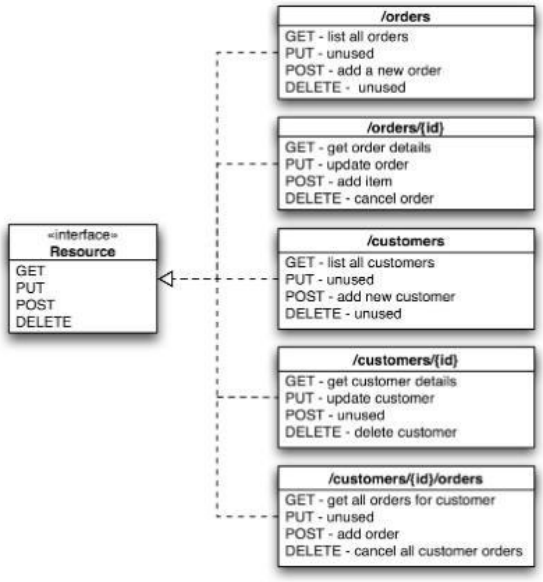

Fig-3: HTTP Methods

6. AUTHORIZATION OF REST API

The difference between authentication and authorization is important in understanding how RESTful APIs are working and why connection attempts are either accepted or denied:

- Authentication is the verification of the credentials of the connection attempt. This process consists of sending the credentials from the remote access client to the remote access server in an either plaintext or encrypted form by using an authentication protocol.

- Authorization is the check that the connection endeavor is permitted. Authorization happens after successful confirmation.

6.1 Basic Authorization

In this method, the sender places a username: password into the request header. The username and password are encoded with Base64, this encrypted value is the cipher that is converted using Base 64 and this encrypted cipher prevents the different attacks and is safe for transmission. APIs that use Basic Auth will also use HTTPS, which means the message content will be encrypted within the HTTP transport protocol. When the API server receives the message, it decrypts the message and examines the header. After decoding the string and analyzing the username and password, it then decides whether to accept or reject the request.

6.2 *HMAC (Hash-based message authorization code)*

HMAC stands for Hash-based message authorization code, is a stronger type of authentication, more common in financial APIs, and is used in banks. With HMAC, both the sender and receiver know a secret key that no one else does.i.e. it uses symmetric encryption. The sender creates a message based on some system properties. The message is then encoded by the secret key and passed through a secure hashing algorithm. The resulting value, referred to as a signature, is placed in the request header. When the receiver (the API server) receives the request, it takes the same system properties and uses the secret key (which only the requester and API server know) and SHA to generate the same string. If the string matches the signature in the request header, it accepts the request. If the strings do not match, then the request is rejected.

6.3 OAuth (2.0)

OAuth is an authentication protocol that allows a user a third- party application access to their information on another site. [7]The most common implementations of OAuth use one or both of these tokens instead:

- **Access token**: The Access tokens are those tokens that are sent along the header and REST\api key.
- **Refresh token:** optionally part of an OAuth flow, these refresh tokens retrieve the new tokens if the access tokens are expired. OAuth2 combines Authentication and Authorization to allow more sophisticated scope and validity control.

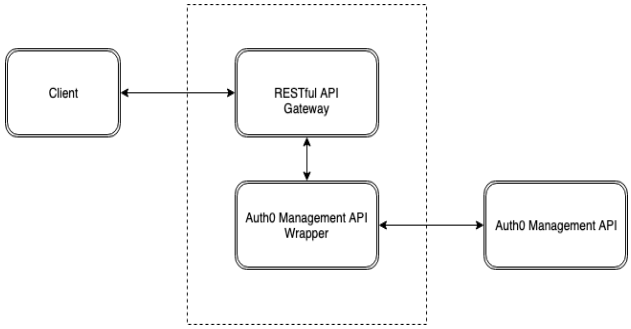

Fig- 4: OAuth Authorization

OAuth 2.0 provides several popular flows suitable for different types of API clients:

- **Authorization code** – The Authorization code are those codes that is mainly used to authorize the transaction and to maintain the data confidentiality. These codes are mainly used for financial transactions.

- **Implicit** – This stream requires the customer to recover an entrance token straightforwardly. It is helpful in situations when the client's qualifications cannot be put away in the customer code since the outsider can effortlessly get to them. It is appropriate for web, work area, and versatile applications that do exclude any server segment.

- **Resource owner password** – Resource owned passwords are those passwords that mainly contains the adding the credential of the owner and the password. These resources are protected requires the authorization.

- **Client Credentials** – The client credentials are requested by the server in order for the client to connect to the server the server manages the list of client's username and passwords and matches them according to the data that is being sent by the client.

- **Authorization Server**-The server that authenticates the Resource Owner, and issues Access Tokens after getting proper authorization. In this case,Auth0.

6.4 Working of OAuth

- The first website connects or interacts with the second website using OAuth protocol in order for the connection between them [8].
- The second website then generates or creates the password, access tokens and the different secret keys that are used to authenticate to the first website and these keys that area generated by the second website is shared among the different users in order for the authentication.

- The first site then issues the secret key and the authentication code to the client end user software.
- The client's software presents the request token and secret to their authorization provider.
- If not already authenticated to the authorization provider, the client then authenticates. After authentication, the client is asked to approve the authorization transaction to the second website.
- The user approves (or their software silently approves) a particular transaction type at the first website.
- The user is given an approved access token (notice it is no longer a request token).
- The user gives the approved access token to the first website.
- The first website gives the access token to the second website as proof of authentication on behalf of the user.
- The second website lets the first website access their site on behalf of the user.
- The user sees a successfully completed transaction occurring.
- OAuth is not the first authentication/authorization system to work this way on behalf of the end-user. In fact, many authentication systems, notably Kerberos, work similarly. What is special about OAuth is its ability to work across the web and its wide adoption. It succeeded with adoption rates where previous attempts failed.

7. CONCLUSIONS

Restful engine is essentially used to create lightweight, quick, versatile and simple to keep up web benefits that regularly use http as methods for correspondence. In this paper, we introduced many of REST API's designers utilized for making a site or some other framework. In This paper, we have additionally introduced the approval procedures like OAuth Authorization. Web is the most well-known stage for business, numerous organizations are setting their organizations on the web and Ecommerce is the most mainstream among them. Subsequently, the framework is created to manufacture an adaptable REST API for Ecommerce to serve any frontend customer. Programming interface ease of use. REST API documentation device designers can use our discoveries to improve reusable apparatus support for programming engineers.

REFERENCES

[1] Neumann, A., Laranjeiro, N., & Bernardino, J. (2018). An Analysis of Public REST Web Service APIs. IEEE Transactions on Services Computing, 1–1. J. Clerk Maxwell, a Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] Li, L., & Chou, W. (2015). Designing Large Scale REST APIs Based on REST Chart. 2015 IEEE International Conference on Web Services.

[3] Li, L., & Chou, W. (2011). Design and Describe REST API without Violating REST: A Petri Net Based Approach. 2011 IEEE International Conference on Web Services.

[4] Djossou, C. (n.d.). A client-server based architecture for communication between expert systems. Proceedings ICCI `92: Fourth International Conference on Computing and information

[5] Haupt, F., Leymann, F., & Pautasso, C. (2015). A Conversation Based Approach for Modeling REST APIs. 2015 12th Working IEEE/IFIP Conference on Software Architecture.

[6] Haupt, F., Leymann, F., Scherer, A., & Vukojevic- Haupt, K. (2017). A Framework for the Structural Analysis of REST APIs. 2017 IEEE International Conference on Software Architecture (ICSA).

[7] Yang, F., & Manoharan, S. (2013). A security analysis of the OAuth protocol. 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM).

[8] Noureddine, M., & Bashroush, R. (2011). A provisioning model towards OAuth 2.0 performance optimization. 2011 IEEE 10th International Conference on Cybernetic Intelligent Systems (CIS).

[9] Stoudenmier, S., & Olmsted, A. (2017). Efficient retrieval of information from hierarchical REST requests. 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST).

[10] Solapurkar, P. (2016). Building secure healthcare services using OAuth 2.0 and JSON web token in IOT cloud scenario. 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I).