International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 10 Issue: 07 | July 2023 www.irjet.net p-ISSN: 2395-0072

A Deep Dive into REST API Framework Survey

R SUNIL[1], Dr. SHOBHA G[2]

[1]Student, Department of Computer Science and Engineering
[2]Professor, Department of Computer Science and Engineering
R V COLLEGE OF ENGINEERING, Bengaluru, Karnataka, India

Abstract - In Modern Day, Communication between Devices is the key to run a successful business. There are many frameworks available to implement communication between devices. In this paper we introduce REST API Framework and compare it with other frameworks like SOAP. The attributes of clients, servers, and their connections are constrained by the REST framework, which significantly influences talks between them. This paper provides details of authentication of client application using OAuth 2 and securing REST API using SSL Certificates. REST API is used in multiple industries including medical industry, cloud computing, micro services, websites using information of other websites like comparing hotel prices, Internet of Things and many more. Representational State Transfer (REST) provides more elasticity and is widely used in web services.

Key Words - Representational State Transfer, Application Programming Interface, Hypertext Transfer Protocol, OAuth 2, Secure Sockets Layer, RESTful, Web Services.

1. INTRODUCTION

REST is a framework that specifies a set of rules to be followed while designing API's. REST API is a straightforward and flexible method of accessing web services without any processing. REST requires less bandwidth and is more suitable for internet use. It is often preferred to the more reliable Simple Object Access Protocol (SOAP) technology where the latter consumes more bandwidth and highly complex. Also, API calls are not cached in SOAP [1]. It is utilized to obtain or provide data from an online service. The client can perform various operations by utilizing the REST service. The major advantage of REST is that it is easier to document them which can be referenced by developers to write programs [2]. The protocol used for communication between devices in REST framework is Hypertext Transfer Protocol (HTTP) [3]. HTTP is an application layer protocol which can be used for distributed and hypermedia systems.

The foundational concept of a RESTful API is a resource. Any record that can be called can be as a resource: a report, an image, a time service, a set of different resources, etc. REST uses a uniform resource identifier to locate a specific resource associated with an interaction between components [4]. Resources can have relationships with different resources and certain techniques or actions must be performed between these resources. A resource which exists individually is called a singleton resource. Same type of resources can be grouped into collections. Collections are themselves a resource. A resource has data associated with it. In this paper, we introduce constraints on REST API, properties of REST API, key elements of REST API, interaction between client server, Authentication of REST API, Securing REST API using SSL Certificates.

2. Constraints on REST API

Constraints on REST API ensure that the API endpoints and the communication between the client and server are based on REST framework. There are six constraints on REST API which are as follows:

A. Statelessness: Server does not store any session data. It means all the necessary information that a server needs to understand with respect to a particular resource has to be contained in the client request.

B. Cacheability: When a Server sends a response to the client in its response it should indicate that whether the response can be cached or not and for how much duration the responses can be cached at the client side. This reduces network latency.

C. Uniform Interface: There are many different devices like smart phones, laptops, IoT devices, etc. Each of these devices can act as a client. This constraint specifies that any device can interact with the Server through the same uniform interface [5].

There are 4 key elements of Uniform Interface Constraint:

- Identification of Resources. Example: In the URI "api/products", we are identifying the "products" resource.
- Manipulation of resources through representations: The resource held at the server can be modified by the client using the representation that the client has which is JSON format.
- Self-descriptive messages for each request: The client needs to include all the details so that the server can understand the nature of the request.

D. Layered System: It allows an architecture to be composed of multiple or hierarchical layers. Each layer performs a specific task and only knows about the input from and output to the intermediate layer. This reduces the amount of complexity that can be put in a single layer.

E. Client-Server: A client requests resources which is internal to the server. A server holds the resources and is not connected to the user interface. Client is not aware of the business logic where as the server is not aware of the user interface.

F. Code on Demand: This constraint is optional. The server can send executable code along with the data to the client. The client can execute the code on behalf of the server.

3. Properties of REST API

REST API has several properties which make it a preferred choice over other methods of communication. In this paper we explain seven important properties which are as follows:

A. Performance: To the end users, it means getting immediate results for their request. Example: Getting list of hotel prices on different websites [6].

B. Scalability: It means consistently provide web service irrespective of the increase or decrease in the number of clients or end users.

C. Simplicity: It provides a Uniform interactive interface.

D. Modifiability: New changes can be incorporated in the REST Architecture to suite the changing needs of the client [7].

E. Visibility: It provides a clear communication between system components through Uniform Interface constraint.

F. Portability: Components can be moved from and deployed to other location by moving program code or data [8].

G. Reliability: It provides more reliability by avoiding single point of failure and using fail over mechanisms to quickly recover the state of the system.

4. Interaction between Client and Server

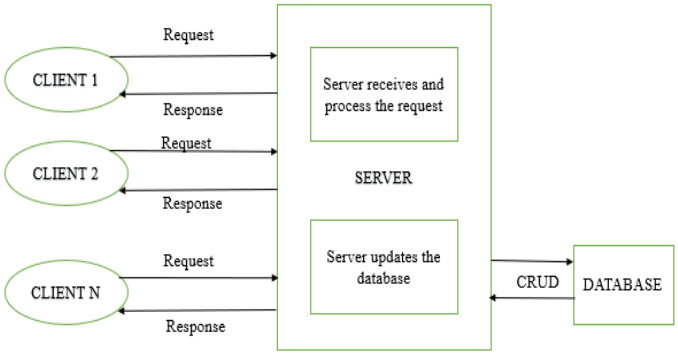

Fig -1: Client Server Architecture

A system that hosts, provides, and manages the majority of the resources and services that the client requests is known as a client-server architecture as shown in fig 1. This approach involves the delivery of all requests and services across a network.

The client is a device which requests for resources. A server is a device which holds resources. This contraint specifies that client and server should be seperated. The principle used in this is the principle of seperation of concerns [9]. Client does not know anything about the business logic. The Server is not aware of the Client UI.

In the client - server architecture the client sends the date along with the associated data to the server. The server verifies the information provided in the meta data including parameters passed in the Request Headers. The server checks if the client is authorized to receive the information. If yes, the server processes the request and returns a HTTP response in the prescribed format to the client along with Status Code.

5. KEY ELEMENTS of REST Framework

There are several key elements of REST Framework. They are as follows:

Resources: The basic element of REST Architecture is a resource. Consider the example of an e-commerce website.

The Rest Server contains details of all the products. Each product can be accessed by its product id. The URL of the web application is http://hostname:<port number>/resource [10]. The client can access the product by calling the URL http://mywebsite/products/1 or the client can also search for the product using queries.

Example: "http://mywebsite/products/search?q=SmartPhone". The server sends the result to client in XML or JSON format.

HTTP Verbs: HTTP Verbs are used to specify the type of Request of the client to the Server [11]. There are 9 different HTTP Verbs.

GET: It is used to get information from the server in a specified format.

Example: GET http://mywebsite/products/1

POST: It is used to add a record in the database by sending this information to the Server in appropriate format.

Example: POST http://mywebsite/products

Request Body:{

id = 10, productName = "Watch", brand = "ABC" }

PUT: It is used to update a record in the database by sending this information to the Server in appropriate format.

Unlike in POST, in the PUT method only the attributes which have to be changed are specified in the Request Body.

Example: PUT http://mywebsite/products

Request Body: {

id=10,

productName = "SmartWatch",

brand = "ABC" }

DELETE: It is used to delete a record or all the records of a particular resource.

Example:

DELETE http://mywebsite/products/10 deletes the product with id =10

DELETE http://mywebsite/products, deletes the products resource.

Http Headers: Http headers allow the client or server send additional information through request and response messages.

Request Headers: Request headers contain information about the resource to be obtained from the Server.

Response Headers: Response headers specifies additional information about the response like its location or about the server providing it.

Request body: Request body contains information in a prescribed format usually JSON format which can be added as a record in the database by the server or update existing records in the database by the server.

Response body: A response body is the data sent to the client by the server in the appropriate format.

HTTP Status Codes:

HTTP status code provide basic information to the client [12]. It is useful for debugging where the error has occurred. The most important HTTP status codes are:

A. Successful Responses (20x):

200 OK: The request was successful.

201 Created: A new resource was created

202 Accepted: The request was received by the Server but not yet acted upon because another process or server will handle the request.

203 Non-Authoritative Information: It means that the returned metadata is not exactly the same as is available from the original server, but is collected from a third-party copy.

204 No Content: This means the Server has no content to return to the client. This is issued in cases of a DELETE request by the client.

After successfully deleting the records the client sends 204 Status Code as response to the client.

B. Client Error Responses(40x):

400 Bad Request: The server will not process the request due to error in data sent by the client.

401 Unauthorized: The client must authenticate itself to receive the requested reponse.

403 Forbidden: The server does not provide the requested resource as the client does not have the necessary access rights.

404 Not Found: The Server cannot find the requested resource.

C. Server Error Responses (50x):

500 Internal Server Error: The Server has encountered a problem and does not how to handle it.

501 Not Implemented: The request method is not provided by the server and cannot be handled.

502 Bad Gateway: The server while acting as a proxy received an invalid response from an upstream server.

6. Authentication of REST API

Authentication of API end points provides points which enables the user to sign up, log in, log out, access API's and more. Various identity protocols are available for authentication of REST API. Some of them are OpenID Connect, OAuth 2.0 and SAML. The Authentication is provided over HTTPS.

Authentication of REST API using OAuth access token is preferred because it works well and is easier to implement.

There are five types of flows mentioned in the OAuth 2.0 specification. They are:

A. Authorization Flow – Client sends an authorization request to the Auth Server. Auth Server requests the user for their approval for the client to access the information it requests [13]. If user provides the approval Auth Server provides authentication code to the client. Client sends Access Token request along with the Authorization code to the Auth Server. Auth Server provides the client with access token to the client to access the information requested before.

B. Implicit Flow - Client sends an authorization request to the Auth Server. Auth Server requests the user for their approval for the client to access the information it requests. If user provides the approval Auth Server provides authentication token to the client. After the user gives their approval, Auth Server redirects the user-agent to the redirect URI provided by the Client that contains a URI fragment containing the access token. The client sends a script to the user agent that can extract the access token. User-agent runs the script and returns the token to the client.

C. Password Flow - In this flow, the user will provide their credentials to the Client and it will produce these credentials and request an Access Token from the Auth Server.

D. Client Credentials - The client forwards its credentials (Client ID, Client Secret) in order to receive the access token and authorize itself to access its own account.

7. Securing REST API using SSL Certificates

Secure Sockets Layer (SSL) Certificates are used to secure REST API endpoints. It can be done through the following steps:

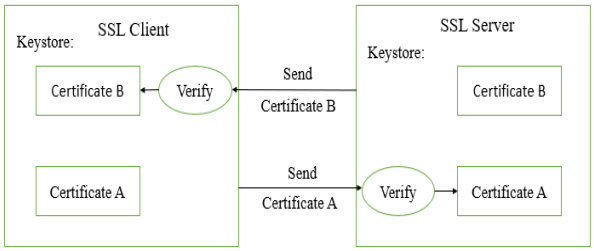

Fig -2: SSL used to secure REST APIs

A. Get or create the certificates. One can also create a self-signed certificate as shown in fig 2.

B. Create a keystore using those certificates.

C. Place the keystore in the application classpath (resources folder).

D. Create a custom REST Template which will fetch the keystore.

E. Call the protected REST API using the custom REST Template.

8. CONCLUSIONS

Restful web service is the most widely used service to communicate between various devices as it is light weight, provides flexibility in designing the APIs, provides stateless communication and easily scalable. In addition to that this paper introduces use of OAuth 2.0 Authentication for authenticating the Client Applications and use of SSL

certificates for securing the REST API's. This provides secure communication between client and server. Another advantage of using RESTful web services is that REST APIs are easier to document. REST APIs are extensively used in e-commerce applications [14], inventory management and monitoring of devices like medical devices [15] where logs are collected from the device through REST based communication.

REFERENCES

[1] L. Mouhibbi, M. Elhozmari and A. Ettalbi, "Sorting and persisting REST and SOAP client for MaaS based architecture," 2018 6th International Conference on Multimedia Computing and Systems (ICMCS), Rabat, Morocco, 2018, pp. 1-5, doi: 10.1109/ICMCS.2018.8525963.

[2] S. M. Sohan, F. Maurer, C. Anslow and M. P. Robillard, "A study of the effectiveness of usage examples in REST API documentation", IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), pp. 53-61, Raleigh, NC, USA, 2017.

[3] C. V. Phung, J. Dizdarevic and A. Jukan, "An Experimental Study of Network Coded REST HTTP in Dynamic IoT Systems," IEEE International Conference on Communications (ICC), Dublin, Ireland, 2020, pp. 1-6, doi: 10.1109/ICC40277.2020.9149026.

[4] A. Neumann, N. Laranjeiro and J. Bernardino, "An Analysis of Public REST Web Service APIs," in IEEE Transactions on Services Computing, vol. 14, no. 4, pp. 957-970, July-Aug. 2021, doi: 10.1109/TSC.2018.2847344.

[5] G. Jia-di and W. Zhi-li, "Modeling Language Design and Mapping Rules for REST Interface," 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2019, pp. 1-6, doi: 10.1109/ICSESS47205.2019.9040766.

[6] J. Vihervaara and T. Alapaholuoma, "The impact of HTTP/2 on the service efficiency of e-commerce websites," 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 2018, pp. 1317-1321, doi: 10.23919/MIPRO.2018.8400238.

[7] B. Costa, P. F. Pires, F. C. Delicato and P. Merson, "Evaluating a Representational State Transfer (REST) Architecture: What is the Impact of REST in My Architecture?", IEEE/IFIP Conference on Software Architecture, pp. 105-114 Sydney, NSW, Australia, 2014.

[8] R. Kazi and R. Deters, "RESTful dissemination of healthcare data in mobile digital ecosystem", 7th IEEE International Conference on Digital Ecosystems and Technologies (DEST), pp. 78-83, Menlo Park, CA, USA, 2013.

[9] L. Li and W. Chou, "Design and Describe REST API without Violating REST: A Petri Net Based Approach", IEEE International Conference on Web Services, pp. 508-515, Washington, DC, USA, 2011.

[10] Pettenati, M.C., Ciofi, L., Pirri, F., Giuli, D."Towards a RESTful Architecture for Managing a Global Distributed Interlinked Data-Content-Information Space", The Future Internet, 2011, Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-20898-0_6.

[11] J. Min and Y. Lee, "An Experimental View on Fairness between HTTP/1.1 and HTTP/2," 2019 International Conference on Information Networking (ICOIN), Kuala Lumpur, Malaysia, 2019, pp. 399-401, doi: 10.1109/ICOIN.2019.8718119.

[12] Finamore, A., Varvello, M., Papagiannaki, K. (2017), "Mind the Gap Between HTTP and HTTPS in Mobile Networks." Passive and Active Measurement 2017, vol 10176. Springer, Cham. https://doi.org/10.1007/978-3-319-54328-4_16.

[13] P. Solapurkar, "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), Greater Noida, India, 2016, pp. 99-104, doi: 10.1109/IC3I.2016.7917942.

[14] Y. Zhao and X. Wan, "The Design of Embedded Web System based on REST Architecture," 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chengdu, China, 2019, pp. 99-103, doi: 10.1109/IAEAC47372.2019.8997929.

[15] K. Kulsresth, S. Shasheesh, C. Mishra and K. P. Arjun, "Covid 19 Tracker Using REST API Android App," 2021 Fourth International Conference on Computational Intelligence and Communication Technologies (CCICT), Sonepat, India, 2021, pp. 105-109, doi: 10.1109/CCICT53244.2021.00031.