RESTFUL API

Have you ever thought how the data is being available according to our needs.

It is through the web servers that provides the data whenever required.

The client requests the server for a required information and then the server sends back the response to the client.

**NEED FOR REST API**

You have to use many methods to get the required information from the server even if the data is returned in a simple format i.e.JSON or XML format .

It would be good for a single information to be retrieved but when you are continuously  requesting for a data  then we have to look for so much of methods.

It becomes a tedious process.

To avoid such kind of scenarios what came into picture is the rest api.

**REST** (Representational State Transfer)

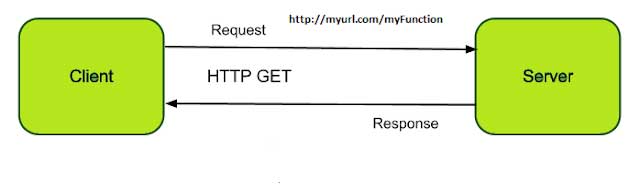
Representational – Clients possess the information necessary to identify,modify,or delete a resource.

State – All resource state information is stored on the client.

Transfer – Client state is passed from client to the service through HTTP.

It creates an object of the data requested by the client and then send the values of the object In response to the user .i.e.the state of the object is sent to the client.

Each and every time the object is not created we are just passing the state of the object that’s where the term representational state comes.



**DEFINITION**

Representational State Transfer or REST is an architectural style as well as an approach for communications purpose that is often used in various web services development.

**RESTAPI**

API stands for Application Programming Interface.

It uses HTTP requests to GET,PUT,POST,DELETE data.

**FEATURES OF RESTFUL API**

1.Simpler than SOAP.

2.It has a proper documentation.

3.It gives you proper error messages.

**PRINCIPLES OF RESTFUL API**

1.**Stateless**

When the requests are sent from a client to the server it contains all the information that is required for the server to understand so it can be a part of a URL or query string parameters or even headers.

2.**Client-Server**

Client Server principle assumes a disconnected system of the server and the client.

It enhances the scalability of the system components.

3.**Uniform Interface**

To obtain uniformity REST has defined four interface constraints.They are

1.Resouce identification

2.Resource manipulation

3.Self descriptive messages

4.hypermedia

4.**Cacheable**

To provide better performance the applications are made cacheable.

It is done by labeling the respone from the server as cacheable or non-cacheable either implicitly or explicitly.

If the response is defined as cacheable then the data can be reused for equivalent responses in future and it also helps in reuse of steal data.

5.**Layered System**

It means that client cannot assume a direct connection to the server and so hardware or software intermediates will exsists between client and the server.

This improves scalability.

6.**Code on Demand**(optional)

The server can temporary extend the client by transferring the logic from the server to the client and the client can execute the logic.

**METHODS OF REST API**

We can Create a resource ,Read a resource,Update a resource and Delete a resource.

Resource is nothing but a client wants to know or maybe the data that the client is looking for.

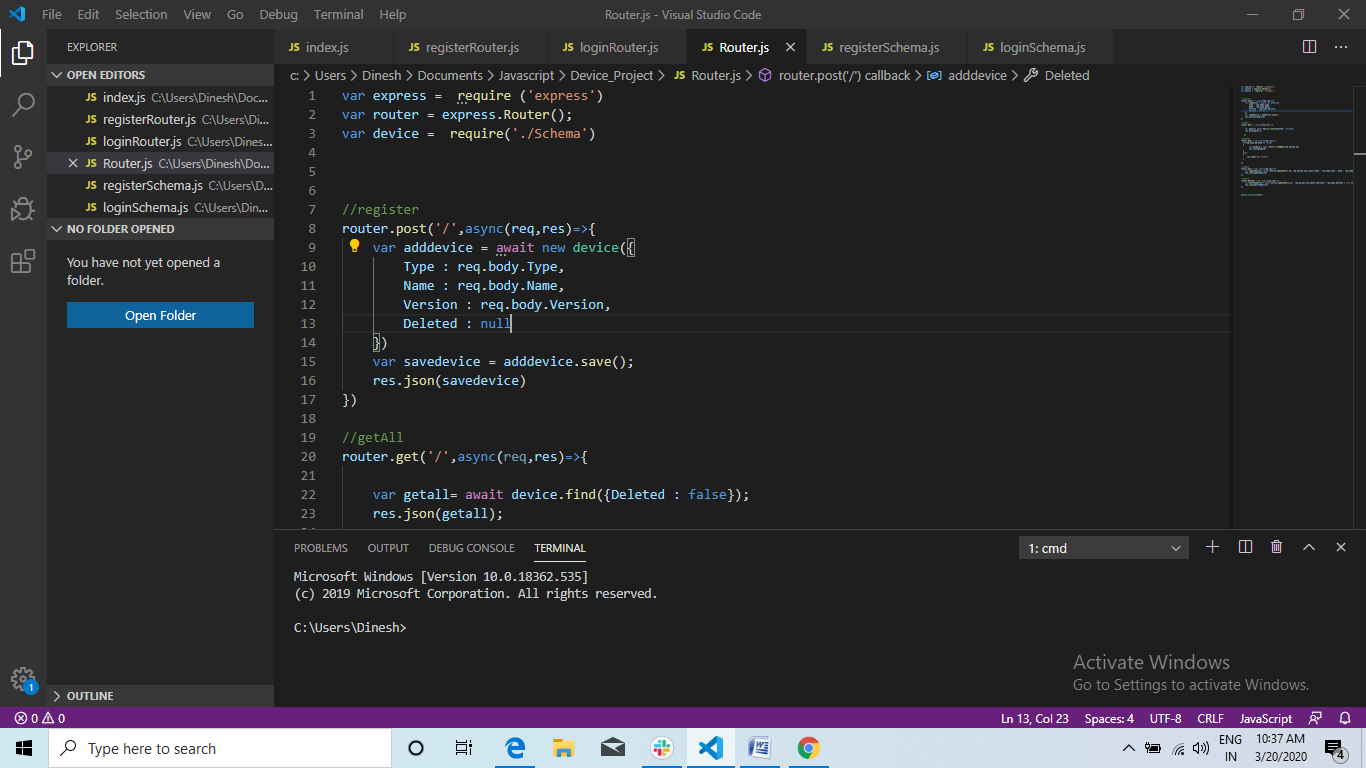
To do these actions i.e.to create,read,update,delete we can use the HTTP methods which are nothing but the methods of REST API.

For a creating a resource we can use the POST method,for reading a resource we can use the GET method,for updating a resource we can use the PUT method,and for deleting a resource we can use the DELETE method.

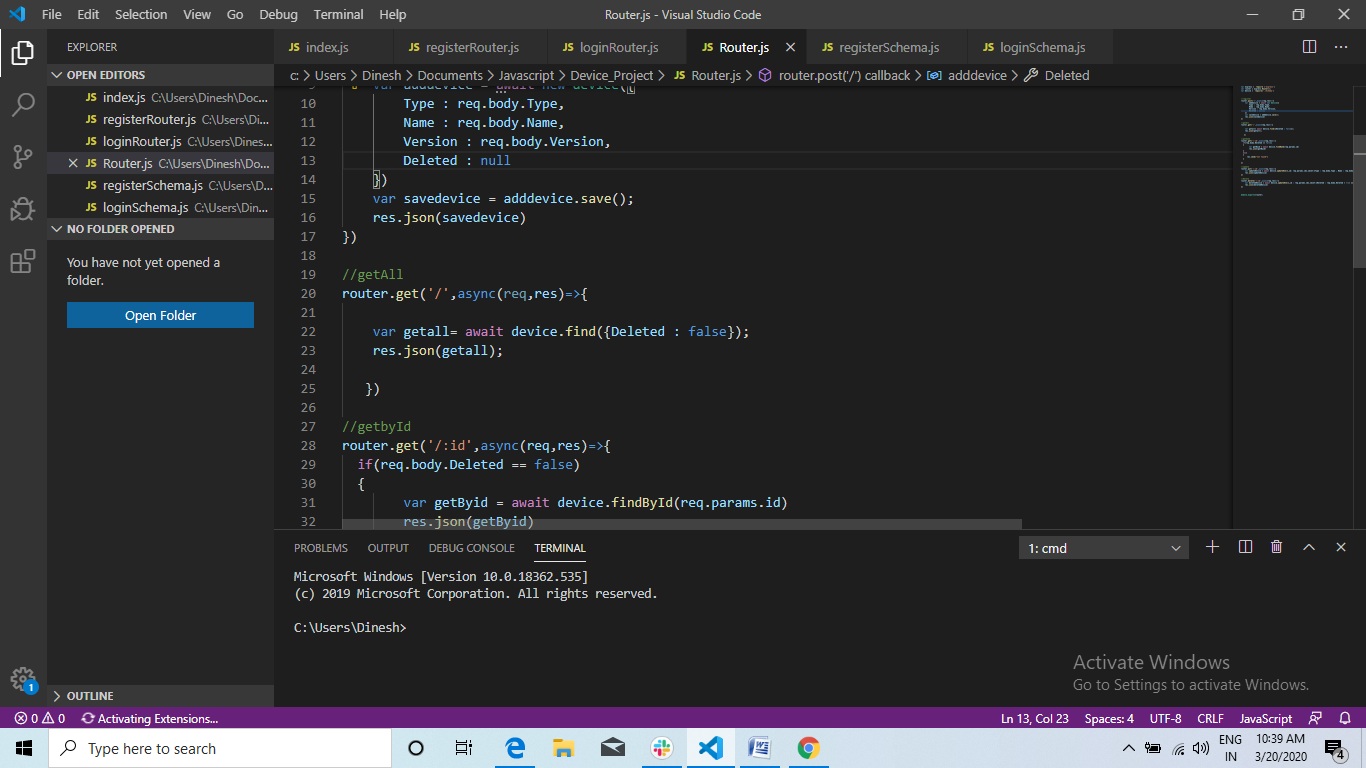


Examples:

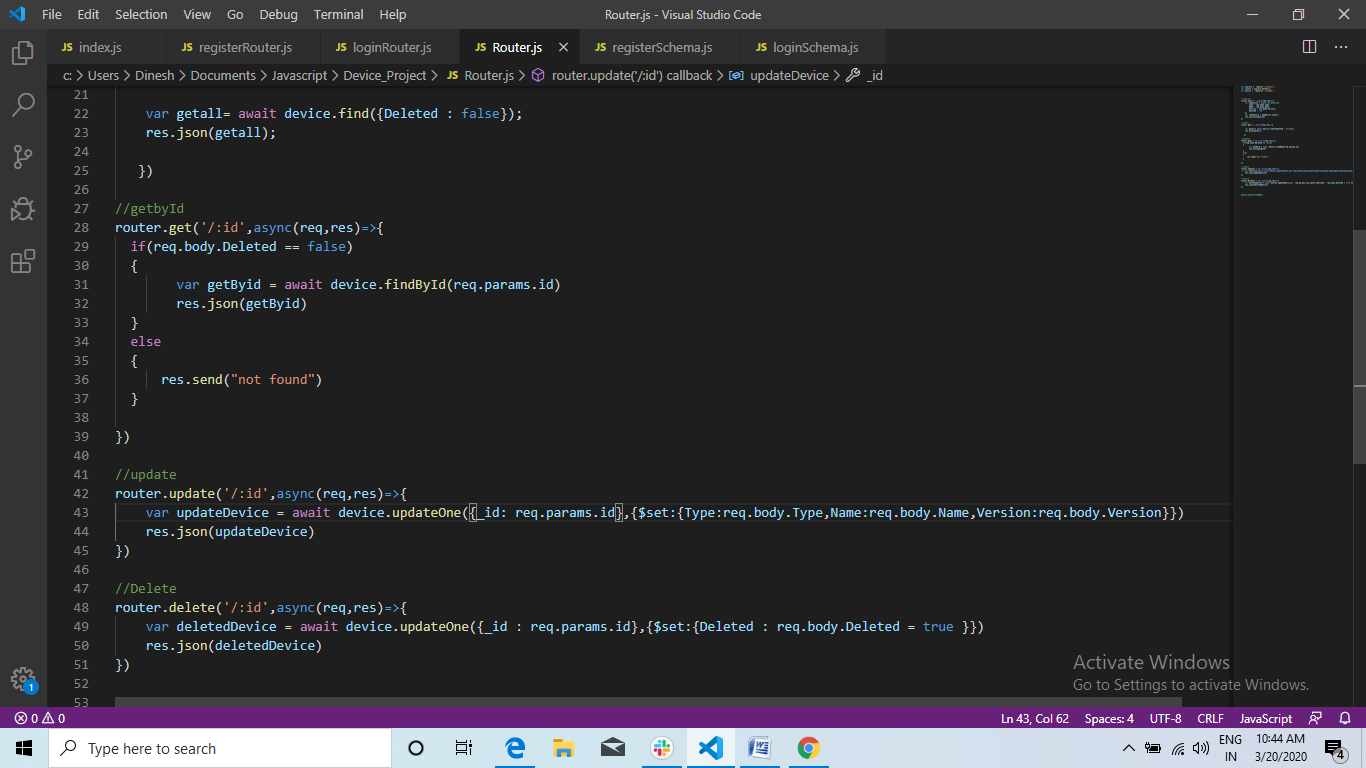
**CREATE METHOD**



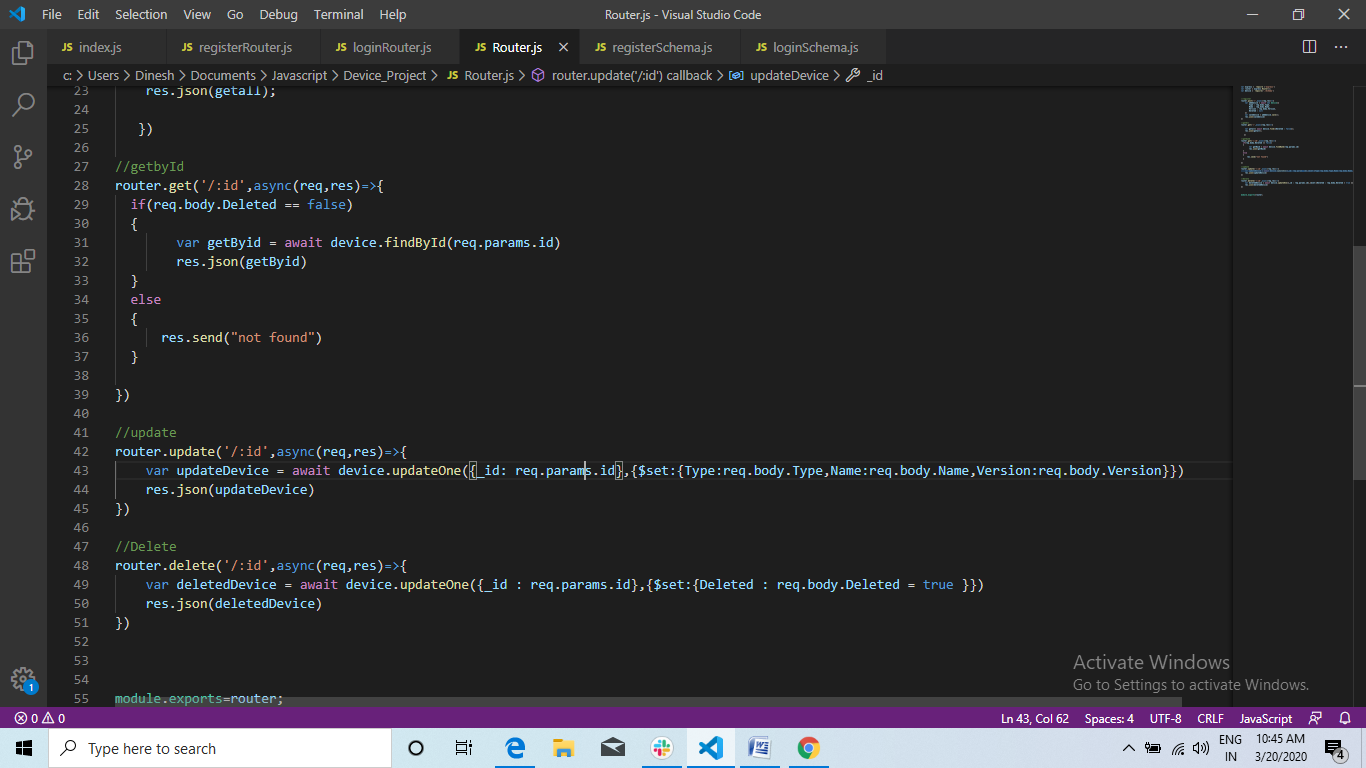
**READ METHOD**



**UPDATE METHOD**



**DELETE METHOD**



**RESTFUL API STANDARDS**

API’s are designed around resources which are any kind of object, data, or service that can be accessed by the client.

API’s built on HTTP, the uniform interface includes using **standard** HTTP verbs to perform operations on resources.

The most common operations are GET, POST, PUT, PATCH, and DELETE

**DIFFERENCE BETWEEN SOAP AND REST**

|  |  |
| --- | --- |
| SOAP | REST |
| Protocol | Architectural style |
| Javascript can call SOAP,but it is difficult to implement | Easy to call from Javascript |
| Performance is not great | Performance is great |
| Cannot be cached | Can be cached |

**-DINESH KUMAR.D**

**Msc Information Technology**