**REST API in 4 minutes -** [Rasanga Lakshith](https://medium.com/@rasangalakshith38?source=post_page-----a7e8ac7f895--------------------------------) May 14, 2022

Diagram

Description automatically generated

REST is stands for **Representational State Transfer**. Well well well… I know that does not make any sense to you. Today I will teach you what actually it is and how it works with a real world example in 240 seconds. Let’s dive in.

First of all we should be aware that the REST is not HTTP and it is not a protocol. It is all about communicating. Restful web service is a service that uses REST APIs to communicate. We’ll get back to that later. Most of the applications that we are using today is based on client-server architecture. The app itself is the client or the frontend. Under the hood it needs to talk to a server or backend to get or store data. This communication happens using the HTTP protocol. Yeah that exact same protocol that powers our web. So on the server(backend) we expose bunch of services that are accessible via the HTTP protocol. Client can then directly call the services by sending HTTP requests. This is where REST comes into the picture.

We use simple HTTP Protocol principles to enable create, read, update and delete data. We call all these operations as CRUD operations. For those CRUD operations HTTP has given us its own **HTTP methods** as **POST**for create, **GET**for read, **PUT**for update and **DELETE**for delete. Now let’s explore this convention using a real world example.

Let’s say that you and I work for an ice-cream shop🍦. We are going to make a web application to show the flavors of ice cream that are in the stock. And we enable to actually make updates to those flavors. Here’s how our system architecture will look like.

Diagram

Description automatically generated

So how would our REST APIs look like? We have an **endpoint** which might look like this: **http://icecream.com/api/flavors**

let’s break that apart a bit. “**api**” in the endpoint signifies that this is the **API portion** of the endpoint. This is not compulsory but you see lot of companies follow this convention to expose their restful services. Pretty straightforward there. **Flavors**is actually what is known as **resource.**So the main building blocks or parts of the REST API are, the request that send from the client to the server and the response that is received back from the server. Here’s how a request will look like.

Diagram

Description automatically generated

Request

API key or some authentication data will be stored in the header section of the request. Now let’s play with our example scenarios.

Let’s say we want to display what is currently in the stock. So how our REST API request will look like? Well, we have GET as our HTTP method. Because we are actually wanting to get those flavors. And then the endpoint is /api/flavors. After sending this REST API request to the server, it will respond with a array of flavors in a JSON format.

Diagram

Description automatically generated

REST API for GET request

Yummy right? But then, ops, mint chocolate is so popular that it actually runs out for the day. Now the store is scrambling and we want to replace that flavor with another one. So we chose chocolate. Which is a great choice ha? As this is an update we are going to use PUT as our HTTP method in our REST API. This is how our REST API request and respond look like,

Diagram

Description automatically generated

REST API for PUT request

A good news. The store just received a shipment of a brand new experimental flavor called “choco chip”. Now we want to create this new flavor to our shop. What is the HTTP method we are going to use here? Yeah, simply the POST.

Diagram

Description automatically generated

REST API for POST request

In response we see this new ID of 2 was created and the flavor is choco chip. That’s pretty much of it.

Before concluding the article, here are some best practices when making REST APIs,

1. Use nouns over verbs
2. Use plural nouns.
3. Represent sub resources.
4. Use HTTP status codes for errors.
5. Use links to other related resource where needed.
6. Provide additional functionalities like filtering, pagination, sorting and field selection via query parameters.
7. Version the API.
8. Always expose via HTTPS.