REST API NOTES

REST stands for **Representational State Transfer.**

The client makes a request to the server through an Http request. The server will have a whole bunch of APIs which are services that it can for expose its clients to be able to tap into. So, building an API is like building a menu of things that our server can respond with.

REST is essentially an **architectural style** just as you have different buildings of different styles. REST is not the only style for designing APIs. The other really popular style before REST was **SOAP** and there is also **GraphQL** or **FALCOR** etc. but the gold standard for web API is REST.

REST came out as a product of Roy Fielding's PHD at the University of California. It proposes a set of rules that web developers could follow when they are building APIs and he came up with this idea that all websites across the web would use the same structure for building their APIs. If every web API was built using the same common guiding principles then it would be so easy for everybody to work together and use different APIs quickly, easily and efficiently.

There are a lot of rules that an API has to follow to be RESTful but the two most important ones are **using the HTTP request verbs** and **to use a specific pattern of routes and endpoint URLs**.
HTTP Request Verbs: **GET**, **POST**, **PUT**, **PATCH** and **DELETE**
These verbs are very similar to the CRUD operations that we carry out on a database.

Fist verb is **GET** and it is basically the same as **READ**. Every single time we want our server to serve up some resource, we've been using **app.get()** and then we pass a callback that responds to the request and sends back a result. If the request involved something that relates to our database, then that's the equivalent of searching through our database and returning the data as a result.

Next verb is **POST** which corresponds to **CREATE** in CRUD. So whenever we create a form on our website, we use **app.post()** and then we have our callback with our request and response and when data is posted to the server, we create an entry in our database and save that data for later. In this case the request will contain the data and the response will simply be success or maybe an error code.

PUT and **PATCH** both update our database. So we might have **app.put()** or **app.patch()** and these both go into the database and update some pieces of data. When you are sending a patch request to the server, you are only sending the piece of data that needs and when you are sending a put request, you replace the entire data.

DELETE is that same as delete in CRUD and it just deletes a piece of data in our database. You can do this by using **app.delete()**.

So now that we’ve looked at HTTP request verbs and we can see how they are used, the next thing to talk about is the specific pattern of routes and endpoints that you have to use.

So just as you go on a safari and there are different routes to see different animals, we can specify specific routes or URLs in order to access certain resources. In order for our APIs to be restful we have to follow a specific pattern of endpoints.

So, for example our API is Wikipedia and in our database, we have a bunch of articles. Now the route for articles applies to all the articles. Now if we created a route for articles then when a client makes a get request to /articles, it should fetch all the articles. When we make a post request to /articles route then it should create a single new article and add it to our database of articles and when we make a delete request to /articles, then it would delete all the articles in our database. RESTful routing also has rules for individual resources. Say within all the articles we have some specific articles. For instance, we have an article on Jack Bauer, then if the client targets /articles/jack-bauer then they made a get request that would fetch the specific article on Jack Bauer from our database. You could also use patch to update that specific article on Jack Bauer and you can delete the specific article as well.