**REST APIs:**

* An API is an application programming interface. It is a set of rules that allow programs to talk to each other.
* The developer creates the API on the server and allows the client to talk to it.

REST determines how the API looks like. It stands for “Representational State Transfer”. It is a set of rules that developers follow when they create their API. One of these rules states that you should be able to get a piece of data (called a resource) when you link to a specific URL.

Each **URL** is called a **request** while the **data sent back** to you is called a **response**.

There’s a high chance you came across the term “REST API” if you’ve thought about getting data from another source on the internet, such as Twitter or Github. But what is a REST API? What can it do for you? How do you use it?

What Is A REST API?

Let’s say you’re trying to find videos about Batman on Youtube. You open up Youtube, type “Batman” into a search field, hit enter, and you see a list of videos about Batman. A REST API works in a similar way. You search for something, and you get a list of results back from the service you’re requesting from.

**The Anatomy of a Request:**

It’s important to know that a request is made up of four things:

* The endpoint (URI- REST APIs use URI to address resources).
* The method (Resource methods).
* The headers
* The data (or body)

1. **The endpoint** (or route) is the url you request for. It follows this structure:

root-endpoint/?

The root-endpoint is the starting point of the API you’re requesting from. The root-endpoint of Github’s API is https://api.github.com while the root-endpoint Twitter’s API is https://api.twitter.com.

The path determines the resource you’re requesting for. Think of it like an automatic answering machine that asks you to press 1 for a service, press 2 for another service, 3 for yet another service and so on.

/users/:username/repos

Any colons (:) on a path denotes a variable.

Technically, query parameters are not part of the REST architecture, but you’ll see lots of APIs use them. Query parameters give you the option to modify your request with key-value pairs. They always begin with a question mark (?). Each parameter pair is then separated with an ampersand (&), like this:

?query1=value1&query2=value2

**JSON:**

JSON (JavaScript Object Notation) a common format for sending and requesting data through a REST API.

A JSON object looks like a JavaScript Object. In JSON, each property and value must be wrapped with double quotation marks.

1. **The** **Resource Methods:**

The method is the type of request you send to the server. You can choose from these five types below:

GET

POST

PUT

PATCH

DELETE

These methods provide meaning for the request you’re making. They are used to perform four possible actions: Create, Read, Update and Delete (CRUD).

Method Name Request Meaning

**GET**

This request is used to **get a resource** from **a server**. If you perform a `GET` request, the **server looks for the data you requested** and **sends it back to you**. In other words, a `GET` request performs a `READ` operation. This is the default request method.

**POST**

This request is used to **create a new resource** on a **server**. If you perform a `POST` request, the **server creates a new entry** in the **database** and **tells you whether the creation is successful**. In other words, a `POST` request performs an `CREATE` operation.

**PUT and PATCH**

These two requests are used to **update a resource** on a **server**. If you perform a `PUT` or `PATCH` request, the **server updates an entry** in the **database** and **tells you whether the update is successful**. In other words, a `PUT` or `PATCH` request performs an `UPDATE` operation.

**DELETE**

This request is used to **delete a resource** from a **server**. If you perform a `DELETE` request, the **server deletes an entry** in the **database** and **tells you whether the deletion is successful**. In other words, a `DELETE` request performs a `DELETE` operation.

1. **The Headers:**

Headers are used to **provide information** **to both the client and server**. It can be used for many purposes, such as **authentication** and **providing information** about the **body content**.

HTTP Headers are property-value pairs that are separated by a colon.

1. **The Data (Or “Body”):**

The data (sometimes called “body” or “message”) contains information you want to be sent to the server. This option is only used with POST, PUT, PATCH or DELETE requests.

**Authentication:**

You wouldn’t allow anyone to access your bank account without your permission, would you? On the same line of thought, developers put measures in place to ensure you perform actions only when you’re authorized to do. This prevents others from impersonating you.

Since POST, PUT, PATCH and DELETE requests alter the database, developers almost always put them behind an authentication wall. In some cases, a GET request also requires authentication (like when you access your bank account to check your current balance, for example).

**HTTP Status Codes and Error Messages**

Some of the messages you’ve received earlier, like “Requires authentication” and “Problems parsing JSON” are error messages. They only appear when something is wrong with your request. HTTP status codes let you tell the status of the response quickly. The range from 100+ to 500+.

In general, the numbers follow the following rules:

200+ means the request has succeeded.

300+ means the request is redirected to another URL

400+ means an error that originates from the client has occurred

500+ means an error that originates from the server has occurred

**API Versions:**

Developers update their APIs from time to time. Sometimes, the API can change so much that the developer decides to upgrade their API to another version. If this happens, and your application breaks, it’s usually because you’ve written code for an older API, but your request points to the newer API.

You can request for a specific API version in two ways. Which way you choose depends on how the API is written.

These two ways are:

* Directly in the endpoint
* In a request header

Twitter, for example, uses the first method. At the time of writing, Twitter’s API is at version 1.1, which is evident through its endpoint:

On the web, there are two main ways to authenticate yourself:

* With a username and password (also called basic authentication)
* With a secret token

The secret token method includes oAuth, which lets you to authenticate yourself with social media networks like Github, Google, Twitter, Facebook, etc

**Web API Request/Response Data Formats**:( html, xml, plain text, json, jpeg, pdf etc).

Media Type (aka MIME type) specifies the format of the data as **type/subtype**

e.g. text/html, text/xml, application/json, image/jpeg stc. text/plain

In **HTTP request**, **MIMI type** is specified in the **request header** using **Accept** and **Content-Type** attribute.

The **Accept header attribute** specifies the **format of response data which the client expects,** and the **Content-Type header attribute** specifies the **format of the data in the request body** so that request can parse it into appropriate format.

For example:

if a **client wants response data** in **JSON format** the it will **send** following **GET HTTP request** with **Accept header** to the Web API.

**HTTP Get Request:**

GET http://localhost:604464/api/student HTTP/1.1

User-Agent: Fiddler

Host: localhost:1234

Accept: application/json

The same way, if the **client includes JSON data** in the **request body to send** it to the receiver then it will send following **POST HTTP request** with **content-Type header** with JSON data in the body.

**HTTP POST Request:**

POST http://localhost:604464/api/student?age=15 HTTP/1.1

User-Agent: Fiddler

Host: localhost:60464

Content-Type: application/json

Content-Length: 13

{

id:1,

name:'chakra'

}