REST API:

What is API?

- An application program interface (API) is a set of routines, protocols, and tools for building software applications.
- Basically, an API specifies how software components should interact.
- A good API makes it easier to develop a program by providing all the building blocks. A programmer then puts the blocks together.
- When you use an application on your mobile phone, the application connects to the Internet and sends data to a server.
- The server then retrieves that data, interprets it, performs the necessary actions and sends it back to your phone.
- The application then interprets that data and presents you with the information you wanted in a readable way. This is what an API is - all of this happens via API.

What is REST?

- REpresentational State Transfer (REST) is an architectural style that defines a set of constraints and properties based on HTTP.
- REST API lets you interact with Parse from anything that can send an HTTP request.
- An API can be considered “**RESTful**” if it has the following features (main ones):
    - Client–server – The client handles the front end the server handles the backend and can both be replaced independently of each other.
    - Stateless – No client data is stored on the server between requests and session state is stored on the client.
    - Cacheable – Clients can cache response (just like browsers caching static elements of a web page) to improve performance.
- The main advantages of REST over SOAP (Simple Object Access Protocol) are:
    - REST permits many different data formats whereas SOAP only permits XML.
    - REST has better performance and scalability. REST reads can be cached, SOAP based reads cannot be cached.

Rest methods:

- The primary or most-commonly-used HTTP verbs (or methods, as they are properly called) are POST, GET, PUT, PATCH, and DELETE.
- These correspond to create, read, update, and delete (or CRUD) operations, respectively. There are a number of other verbs, too, but are utilized less frequently.

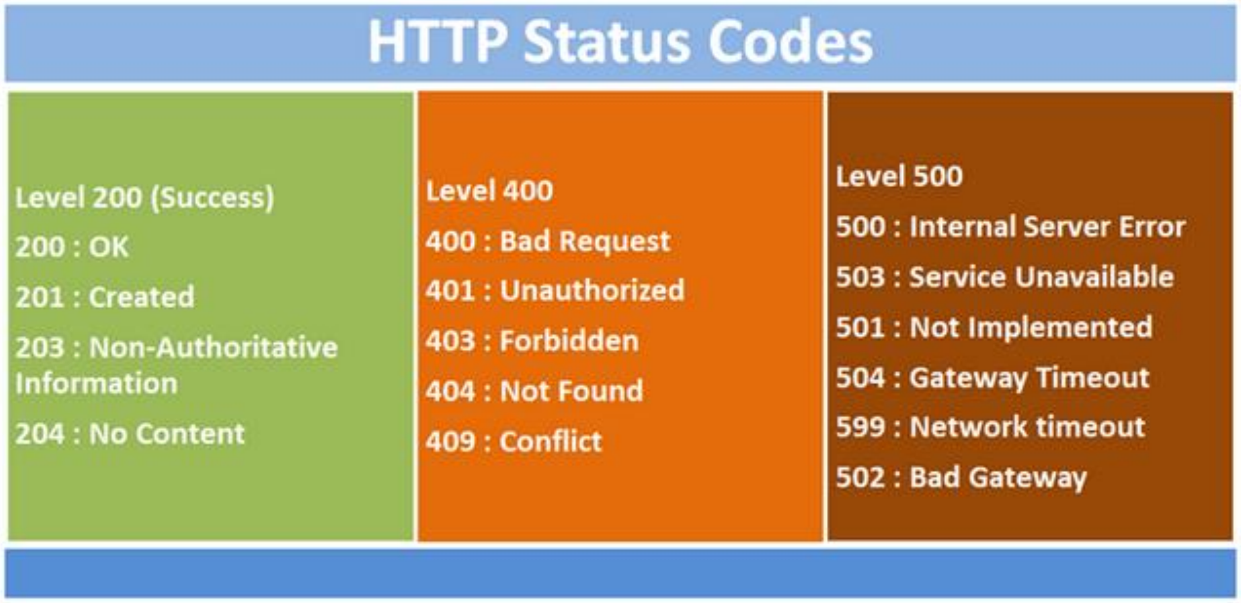
- **Post -**The POST verb is most-often utilized to **create** new resources. In particular, it's used to create subordinate resources. That is, subordinate to some other (e.g. parent) resource. In other words, when creating a new resource, POST to the parent and the service takes care of associating the new resource with the parent, assigning an ID (new resource URI), etc.

- **Get -** The HTTP GET method is used to **read** (or retrieve) a representation of a resource. In the “happy” (or non-error) path, GET returns a representation in XML or JSON and an HTTP response code of 200 (OK). In an error case, it most often returns a 404 (NOT FOUND) or 400 (BAD REQUEST).

- **Put -** PUT is most-often utilized for **update** capabilities, PUT-ing to a known resource URI with the request body containing the newly-updated representation of the original resource.

- **Delete -** DELETE is pretty easy to understand. It is used to **delete** a resource identified by a URI.

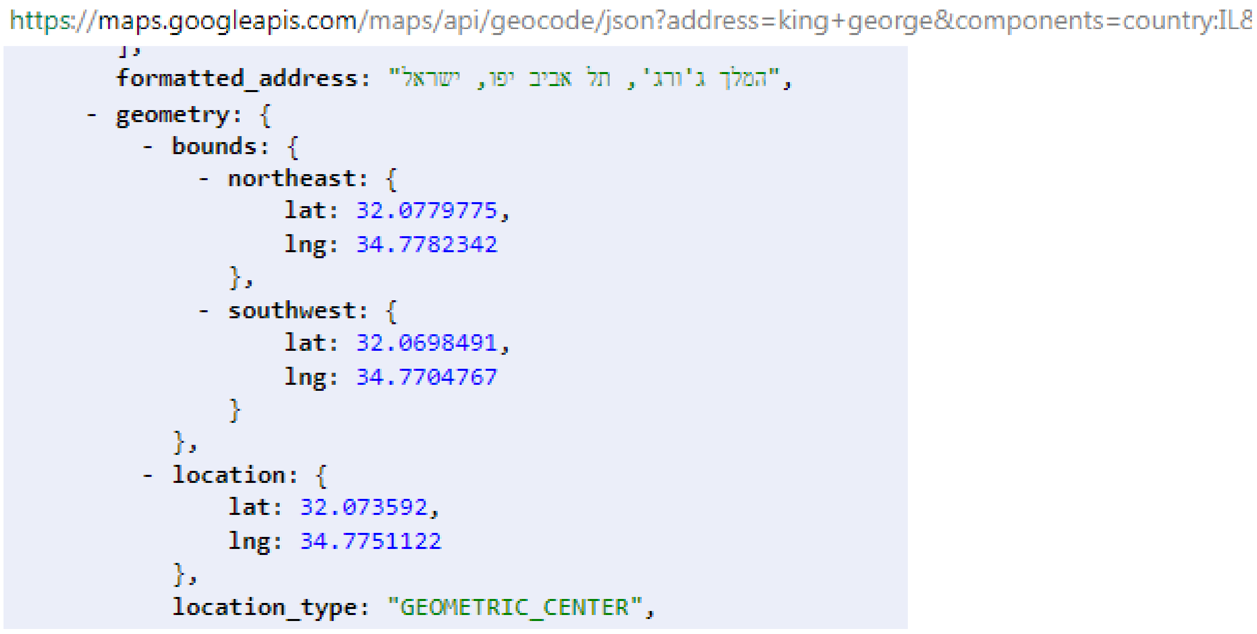
HTTP codes:



HTTP Status Codes

Level 200 (Success)	Level 400	Level 500
200 : OK	400 : Bad Request	500 : Internal Server Error
201 : Created	401 : Unauthorized	503 : Service Unavailable
203 : Non-Authoritative Information	403 : Forbidden	501 : Not Implemented
204 : No Content	404 : Not Found	504 : Gateway Timeout
	409 : Conflict	599 : Network timeout
		502 : Bad Gateway

HTTP request-response example:



https://maps.googleapis.com/maps/api/geocode/json?address=king+george&components=country:IL&

```
    ],
    formatted_address: "המלך ג'ורג', תל אביב יפו, ישראל",
  - geometry: {
      - bounds: {
          - northeast: {
                lat: 32.0779775,
                lng: 34.7782342
            },
          - southwest: {
                lat: 32.0698491,
                lng: 34.7704767
            }
        },
      - location: {
            lat: 32.073592,
            lng: 34.7751122
        },
        location_type: "GEOMETRIC_CENTER",
```

JSON

- JSON stands for JavaScript Object Notation
- JSON is a lightweight data-interchange format
- JSON is "self-describing" and easy to understand
- JSON is language independent.
- Since the JSON format is text only, it can easily be sent to and from a server, and used as a data format by any programming language.
- JavaScript has a built in function to convert a string, written in JSON format, into native JavaScript objects: **JSON.parse()**
- So, if you receive data from a server, in JSON format, you can use it like any other JavaScript object.

- JSON syntax is derived from JavaScript object notation syntax:
    - Data is in name/value pairs
    - Data is separated by commas
    - Curly braces hold objects
    - Square brackets hold arrays

Json parsing using Json module

```
import urllib.request, json

# Opening a web request

with urllib.request.urlopen(

"https://free.currencyconverterapi.com/api/v5/convert?q=USD_ILS&compact
=n") as url:

# Decoding response to str

   data = json.loads(url.read().decode()) # Decoding a web request

# Parsing results

   results = data['results']

   USD_ILS = results['USD_ILS']

   val = USD_ILS['val']

   print(val)
```