API:

- Application Programming interface
- Not a "visual" interface, It's Programming interface (one computer communicate with other computers)
- Computer use APIs to talk to each other over the internet
- written in any server-side language
- Consumed in: Browsers use JS for their API requests, Servers use any language that runs on that computer

API Security: API keys:

- Passwords to access an API. these are your authentication credentials.

Rest API

- Representational state transfer
- All about transferring the data form server to client or server to server.
- It manages the resources(data), representation of these resources and it passes representational state from one layer to another.
- example: API request->need flight details from different websites- so it will fetch the data from different airline websites
- output: JavaScript object notation-> JSON(lightweight) or xml
- Any rest api implemented over http protocol will be composed of URI which points to exact location of resource in some dbs.
- Path parameter: variable in a URI path that helps in pointing towards specific resource. eg. https://myapp/customer/2
- Query parameter: variable in a URI path that queries/filters through a list of resources. Eg. https://myapp/customers?name=abc

Request body: It is the data sent by the client to the server as part of an HTTP request. This data can be in various formats such as JSON, XML, form data, or other.

Http headers:

- Set of attributes that corresponds to any meta-data associated with API request.

Idempotent: Refers to a property of certain HTTP methods where performing the same operation multiple times has the same effect as performing it once. Idempotent methods are designed to not cause unintended side effects or changes to the system state when invoked repeatedly.

Request Methods : HTTP methods:

A. Post: used to create a resource, to trigger some kind of operation on any existing resource and just post some data for an existing resource which must be used for processing
   - Response code: 201- Resource has been created (resource id returned), 200- Resource has been created (details of resource returned), 204- Resource processed successfully with no content returned.
   - When you sent invalid data in body then 400 code is return
   - 409- conflict response if duplicate data is post
   - Post URIs: can have a path parameter, but is should never have a query parameter.
   - Not idempotent

B. Get: there is no request body for this method. we are using path parameter to fetch the data of one entry.
   - To access list of resource all data without path parameter.
   - To access 10 resources then: https: //localhost:8080/mystore/customer?limit=10 & offset=0 This concept call pagination
   - Idempotent
   - Code: 404- not found, 200: fetched successfully

C. Put/Patch: update resources, also used to create resource (upsert- only if server trusts client)
   - If given resource already available then it will simply update the resource, but if the resource is not available then it will create new resource.
   - For upsert we need to pass id in body not as a parameter.
   - Code: 404-wrong path parameter, 400- wrong payload
   - Put vs patch: put will update all attributes, replaces the whole resource, but patch will only change few attributes.
   - PUT is Idempotent

D. Delete: soft delete (it will store entry of deleted data) and hard delete
   - Idempotent

Request Methods Summary: CRUD OPERATION:

- HTTP GET: Read
- HTTP POST: Create -> Do not go through the standard URL, but use a URL as the endpoint, ask another computer to create a new resource, Returns data about the newly created resource.
- HTTP PUT: Create/Update/ Replace-> Do not go through the standard URL, but use a URL as the endpoint, ask another computer to update an entire resource, if the resource doesn't exist the API might decide to Create the resource, update/replace.
- HTTP DELETE: Delete-> do not go through the standard URL, but use a URL as the endpoint, ask another computer to delete a single resource or a list of resources, use with caution.
- HTTP PATCH: Update/ Modify -> Do not go through the standard URL, but use a URL as the endpoint, ask another computer to update a piece of a resource, are not fully supported by all browsers or frameworks , so we typically fall back on PUT requests, e.g. updating name(Not Supported by Python), partial update/modify, to update small piece of info from whole request e.g. wrong price.

Filtering: to find specific data.

Pagination: response which is divided into multiple pages.

- Limit: The "limit" query parameter specifies the maximum number of items (resources) to be returned in a single response. It controls the page size or batch size of the results. Eg. GET /api/users?limit=5
- Offset: The "offset" query parameter defines the starting position within the dataset from which the results will be retrieved. It is used for pagination, allowing the client to request subsequent pages of results. Eg. GET /api/users?limit=10&offset=40

Asynchronous Rest API:

- request: accepted but not fulfil the request (In progress) - response.

API rate limiting: API rate limiting is a mechanism implemented by API providers to restrict the volume of requests a client or user can send to the API within defined time intervals (e.g., per minute or per hour). The purpose is to mitigate potential misuse, optimize system performance, and uphold equitable utilization of the API's allocated resources.

Headers: are the sort of indication for the metadata of what the response or request consists inside

- key value pair
- categories: Request headers- client is sending key value pair to server to for response in given format. i.e.connection: close
- Response headers: server is giving some additional information about the data that it has send eg. Date: Monday, 09 august 2023
- Payload headers:
- Representation headers: talk about the data that has been transferred from one server to another can be in different formats.
  Custom header: eg. Authorization

Response codes:

Here's a brief explanation of common HTTP status codes used in REST APIs:

Request & Responses (2xx)

- 200: OK – The request was successfully processed.
- 201: Created – A resource was successfully created, typically in response to a POST request.
- 202: Accepted– The request was accepted but is still being processed (often queued).

Redirect Responses (3xx)

- 301: Moved Permanently – The resource has permanently moved to a new URL. You should update the endpoint.
- 302: Found – The resource is temporarily located elsewhere.

Client Errors (4xx)

- 400: Bad Request– The server couldn't process the request, usually due to invalid input or malformed request.

- 401: Unauthorized – Authentication credentials (e.g., API keys) are missing or incorrect.
- 403: Forbidden – The request was understood but is not allowed (e.g., insufficient permissions).
- 404: Not Found– The requested resource could not be found.
- 405: Method Not Allowed – The HTTP method used is not supported for this endpoint (e.g., using POST on a GET-only resource).

Server Errors (5xx)

- 500: Internal Server Error – Something went wrong on the server side, and the request couldn't be processed. Not in your control—contact the server administrator.

Response codes Summary:

- 2xx: success
- 3xx: redirection and others
- 4xx: problem on client side
- 5xx: problem on server side

Examples of some API testing frameworks and tools:

1. **Postman** – User-friendly tool for creating, testing, and automating API requests.
2. **Rest-Assured** – Java-based library for testing REST APIs with simple syntax.
3. **SoapUI** – Comprehensive tool for functional, load, and mock testing of REST and SOAP APIs.
4. **JMeter** – Tool for API performance testing, supporting REST and SOAP.
5. **Karate** – BDD framework for API testing with simple Gherkin syntax and JSON/XML support.