API is the acronym for ***Application Programming Interface***, which is a ***software intermediary*** *(mediator)* that allows two applications to talk to each other.

Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you’re using an API.

***HTTP Protocol:*** When you use an application on your mobile phone, the application connects to the Internet and sends request to a server. The server then retrieves that data, interprets it, process it, performs the necessary actions, and sends it back to your phone as a response. 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.

## **API Also Provides Is a Layer of Security**

Your phone’s data is never fully exposed to the server, and likewise the server is never fully exposed to your phone. Instead, each communicates with small packets of data, sharing only that which is necessary

## **The Modern API**

Over the years, what an “API” is has often described as any sort of generic connectivity interface to an application. More recently, however, the modern API has taken on some characteristics that make them extraordinarily valuable and useful:

* Modern APIs ***adhere to standards*** (typically HTTP and REST).
* That are ***developer-friendly***, ***easily accessible*** and ***understood broadly.***
* They are treated more like ***products*** than code.
* They are designed for consumption for ***specific audiences*** (e.g., mobile developers).
* They are documented, and they are versioned in a way that users can have certain expectations of its maintenance and lifecycle.
* Because they are much more standardized, they have a much stronger discipline for ***security and governance***, as well as monitored and managed for performance and scale.
* As any other piece of productized software, the modern API has its own software development lifecycle (SDLC) of designing, testing, building, managing, and versioning.  Also, modern APIs are well documented for consumption and versioning.

# Types of APIs

**Private APIs**

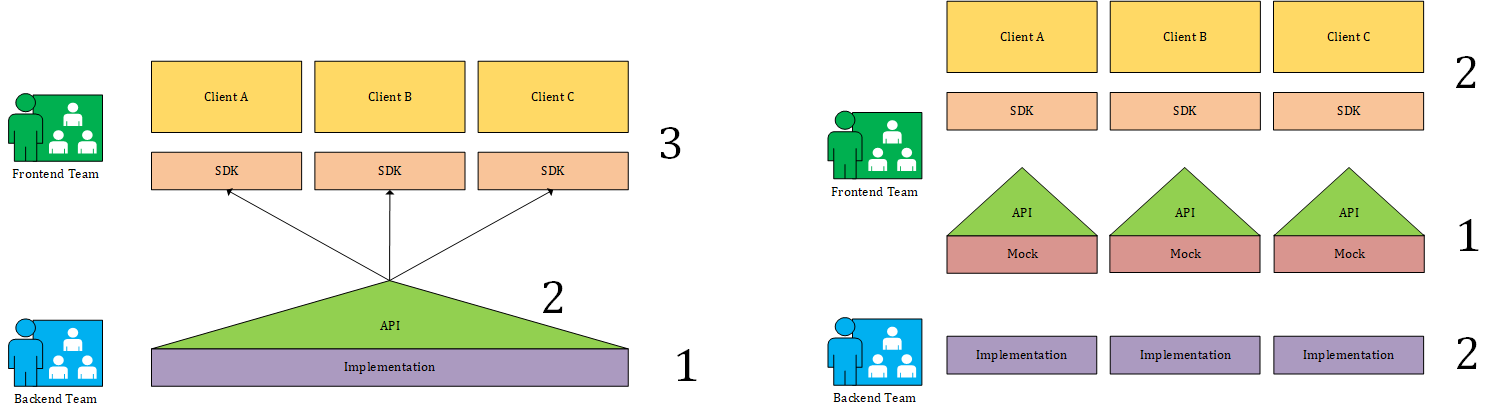
* Internal or Enterprise APIs
* Within the firewall
* API Provider and Consumer own both ends of the pipe
* Example – HR Payroll Service, Travel Management Service
* Advantage - Agility

**Partner APIs**

* That facilitate integration between a business and their Partners.
* Limited Access and Users
* Example – Payment APIs
* Advantage- Collaboration

**Public or Open APIs**

* Open to the Entire Internet
* API Management is major concern
* Example - Google Maps, Facebook, Twitter
* Advantage – Innovation



# Graphical user interface Description automatically generated with medium confidence

# API-first development is a strategy in which first APIs are developed based on requirements and following best practices then the module on top of it (be it a website, mobile application, or a SaaS software)

**Advantage**

Faster App Development | Allows Focusing on the Business Logic | Better Documentation | Inherently supports Micro service Architecture based Applications | Decoupling dependencies | Secure abstraction layer | fast and frequent iterations

# REST API

Based on **HTTP methods**

* **GET** = Retrieve data from a specified resource.
* **POST** = Submit data to be processed to a specified resource.
* **PUT** = Update a specified resource.
* **DELETE** = Delete a specified resource.

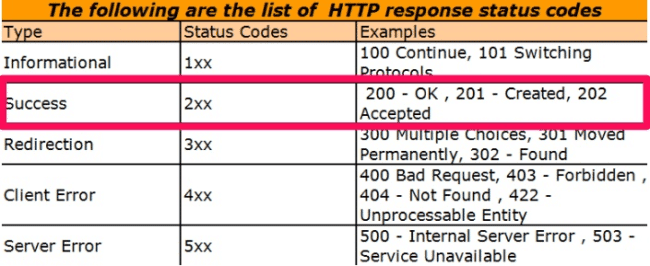
**Rarely used methods.**

* **HEAD** = Same as get but does not return a body.
* **OPTIONS** = Return the supported HTTP methods.
* **PATCH** = Update partial resources.

| **API** | **Description** | **Request body** | **Response body** |
| --- | --- | --- | --- |
| GET /api/todoitems | Get all to-do items | None | Array of to-do items |
| GET /api/todoitems/{id} | Get an item by ID | None | To-do item |
| POST /api/todoitems | Add a new item | To-do item | To-do item |
| PUT /api/todoitems/{id} | Update an existing item | To-do item | None |
| DELETE /api/todoitems/{id} | Delete an item | None | None |
|  |  |  |  |

Graphical user interface

Description automatically generated



Graphical user interface, application

Description automatically generatedGraphical user interface, text

Description automatically generated with medium confidence

# Authentication & API Keys

• Many APIs require an API key.

• Just as a real-world key allows you to access something, an API key grants you access to a particular API.

• Moreover, an API key identifies you to the API, which helps the API provider keep track of how their service is used and prevent unauthorized or malicious activity.

• Some APIs require authentication using a protocol called *OAuth* (OAuth 2.0 ***authorization*** framework).

A screenshot of a computer

Description automatically generated

## Caching in REST APIs

Being [cacheable](https://restfulapi.net/rest-architectural-constraints/" \l "cacheable) is one of the architectural constraints of REST.

* **GET** requests should be **cacheable** by **default** – until a special condition arises. Usually, ***browsers treat all GET requests as cacheable***.
* **POST** requests are **not cacheable** by **default** but can be made cacheable if either an Expires header or a Cache-Control header with a directive, to explicitly allows caching, is added to the response.
* Responses to **PUT** and **DELETE** requests are **not cacheable** at all.