What is an API? A Simple Guide for Non-Techies

Once upon a time, before a pandemic took over the world, there was a daily opportunity for cross-departmental engagement: lunch.

Back then, you gathered team members from various departments and walked to a nearby restaurant. Maybe you talked about the latest TV show revival, or you finally ask your developer colleague what is an API.

When you got to your table and a waiter stopped by to give you menus, maybe your developer colleague said, the waiter is an API.

Enter an awkward moment of silence.

Embed:

https://giphy.com/gifs/Staatsloterij-dancing-dansen-staatsloterij-30pNlKLmRWyUN17Tdq

But then your developer colleague explained.

What is an API?

The restaurant analogy has been going around the industry for about a decade for a good reason: It makes more sense than answering "What is an API?" with "Application Programming Interface."

- **You are the API customer.** The one sending a request for service or data.
- **The menu is the documentation.** It tells you what you can ask of the API, or your waiter.
- **The kitchen is the API server.** It supplies the service or data, yet it only has certain sets of data. If you go to a vegan restaurant, you can't ask for a meat dish

(and often vice versa). It won't be on the menu, or in the documentation. If you try to order it anyway, you'll get an error code, because the kitchen can't deliver it.

- **Your waiter – the API – is a data messenger.** You can't go into the kitchen and order food directly from the chef. You have to send a request through the waiter. The waiter transports the request (what you want to eat) from you to kitchen, then transports the data or service (your food, when it's ready) from the kitchen to you.
- **Then there are the secret ingredients.** You can get certain datasets, like an answer to whether a dish includes onion. However, you won't be able to discover the secret ingredients that make potatoes better here. You also won't be able to get the chef's home address. The kitchen, as the API server, will serve you while maintaining confidentiality, security and compliance.

But how does it look like in actual application integrations?

How APIs are Used in App Integrations: A Simple Example

Every once in a while, you'll sign up to a new app or software, and you'll need to create a new user from scratch. To save you the trouble, the app might suggest you use your Facebook credentials. A click of a button, and you're in.

- **The new app is the API customer.** It's the one requesting service from Facebook – hey, let me verify this person's identity and know basic information, like her name.
- **Facebook has developed documentation,** which clarifies what can be integrated and shared, what can't, and under which conditions.
- **Facebook itself is the API server.** It provides the data-based service – to help the app sign you up quickly.
- **The API allows the communication between Facebook and the new app.** It transfers the new app's request to Facebook, and the ability to sign you up quickly back to the new app.

- **There are secret ingredients here, too.** Facebook won't share all your data, like your password, with the new app. Facebook will only share what was determined in its documentation.

That said, the answer to "what is an API" is a bit more complex, since there's more than one type of API.

REST API vs. Real-Time API

Let's discuss two of the key API types – REST API and real-time API. What's the difference?

REST API

You might be thinking, a REST API has got to be what you do after the team lunch, right?

That would make sense, but actually, a REST API is what happens *during* lunch. It's "restful" because it's a relatively simple operation.

Basically, REST stands for Representational State Transfer. That means that a server responds to four types of requests:

- **Create** a new element in the code, such as adding a new user. Think of a team member that got delayed at the office and is joining you mid-lunch. The waiter brings her or him an extra chair.
- **Read** the information that's available. When you request the menu, it's like requesting a server to show you a list of data points.
- **Update** the code, say, when you request that onion be taken off the dish.
- **Delete** items from the code. Like when you decide to skip that extra drink after all, because you need to get back to work. You ask the waiter to delete it from the order's code.

These requests – called CRUD for short – are easy and fast to update. Hence, the name REST API.

Real-Time API

A real-time API is similar, yet gets updated automatically. Think of that time you ordered a taxi instead of walking to a restaurant, and saw the taxi's progression toward your office across a map. A real-time API transfers data from its server (Google Maps, GPS) to the destination app (your taxi provider) every 100 milliseconds or less.

Browsing a site while waiting for the taxi?

Searching for lunch deals?

A website could integrate real time promos from multiple restaurant servers through APIs. Therefore, every time a promo gets offered in any of the restaurants' apps, it will automatically show up on this one site you're checking.

Apps are Better Together

The best way to understand "what is an API" is to view it as communication between two applications. Just like in human communication, apps need to speak the same language and ensure clarity.

We've all been in situations where we said one thing (sent an API request), and the other person (the API server) heard something else entirely. That's why apps have documentation. And like in other healthy relationships, apps have boundaries. For apps, it means they protect their customers' data.

Ultimately, it's our abilities to rely on each other's strengths and lift each other up that make this world better. An API helps apps do the same.