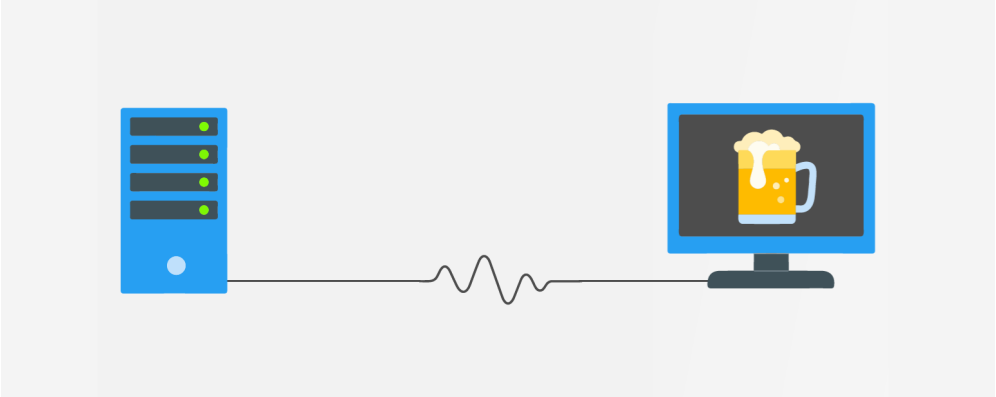
Petr Gazarov Follow
Software developer at PolicyGenius https://github.com/petrgazarov
Aug 13, 2016 · 5 min read

What is an API? In English, please.



Artwork by Phillip Blackowl. Spanish translation. Portuguese translation.

Before I learned software development, API sounded like a kind of beer.

Today I use the term so often that I have in fact recently tried to order an API at a bar.

The bartender's response was to throw a 404: resource not found.

I meet lots of people, both working in tech and elsewhere, who have a rather vague or incorrect idea about what this fairly common term means.

Technically, API stands for **Application Programming Interface**. At some point or another, most large companies have built APIs for their customers, or for internal use.

But how do you explain API in plain English? And is there a broader meaning than the one used in development and business? First, let's pull back and look at how the web itself works.

WWW and remote servers

When I think about the Web, I imagine a large network of connected *servers*.

Every page on the internet is stored somewhere on a remote server. A remote server is not so mystical after all—it's just a part of a remotely located computer that is optimized to process requests.

To put things in perspective, you can spin up a server on your laptop capable of serving an entire website to the Web (in fact, a *local* server is what engineers use to develop websites before releasing them to the public).

When you type www.facebook.com into your browser, a request goes out to Facebook's remote server. Once your browser receives the response, it interprets the code and displays the page.

To the browser, also known as the *client*, Facebook's server is an API. This means that every time you visit a page on the Web, you interact with some remote server's API.

An API isn't the same as the remote server—rather **it is the part of the server that receives requests and sends responses**.

APIs as a way to serve your customers

You've probably heard of companies packaging APIs as products. For example, Weather Underground sells access to its weather data API.

Example scenario: Your small business's website has a form used to sign clients up for appointments. You want to give your clients the ability to automatically create a Google calendar event with the details for that appointment.

API use: The idea is to have your website's server talk directly to Google's server with a request to create an event with the given details. Your server would then receive Google's response, process it, and send back relevant information to the browser, such as a confirmation message to the user.

Alternatively, your browser can often send an API request directly to Google's server bypassing your server.

How is this Google Calendar's API different from the API of any other remote server out there?

In technical terms, the difference is the format of the request and the response.

To render the whole web page, your browser expects a response in *HTML,* which contains presentational code, while Google Calendar's API call would just return the data—likely in a format like *JSON*.

If your website's server is making the API request, then your website's server is the client (similar to your browser being the client when you use it to navigate to a website).

From your users perspective, APIs allow them to complete the action without leaving your website.

Most modern websites consume at least some third-party APIs.

Many problems already have a third-party solution, be it in the form of a library or service. It's often just easier and more reliable to use an existing solution.

It's not uncommon for development teams to break up their application into multiple servers that talk to each other via APIs. The servers that perform helper functions for the main application server are commonly referred to as *microservices.*

To summarize, when a company offers an API to their customers, it just means that they've built a set of dedicated URLs that return pure data responses—meaning **the responses won't contain the kind of presentational overhead that you would expect in a graphical user interface like a website**.

Can you make these requests with your browser? Often, yes. Since the actual HTTP transmission happens in text, your browser will always

do the best it can to display the response.

For example, you can access GitHub's API directly with your browser without even needing an access token. Here's the JSON response you get when you visit a GitHub user's API route in your browser (https://api.github.com/users/petrgazarov):


```
{
  "login": "petrgazarov",
  "id": 5581195,
  "avatar_url":
"https://avatars.githubusercontent.com/u/5581195?v=3",
  "gravatar_id": "",
  "url": "https://api.github.com/users/petrgazarov",
  "html_url": "https://github.com/petrgazarov",
  "followers_url":
"https://api.github.com/users/petrgazarov/followers",
  "following_url":
"https://api.github.com/users/petrgazarov/following{/other_u
ser}",
  "gists_url":
"https://api.github.com/users/petrgazarov/gists{/gist_id}",
  "starred_url":
"https://api.github.com/users/petrgazarov/starred{/owner}
{/repo}",
  "subscriptions_url":
"https://api.github.com/users/petrgazarov/subscriptions",
  "organizations_url":
"https://api.github.com/users/petrgazarov/orgs",
  "repos_url":
"https://api.github.com/users/petrgazarov/repos",
  "events_url":
"https://api.github.com/users/petrgazarov/events{/privacy}",
  "received_events_url":
"https://api.github.com/users/petrgazarov/received_events",
  "type": "User",
  "site_admin": false,
  "name": "Petr Gazarov",
  "company": "PolicyGenius",
  "blog": "http://petrgazarov.com/",
  "location": "NYC",
  "email": "petrgazarov@gmail.com",
  "hireable": null,
  "bio": null,
  "public_repos": 23,
  "public_gists": 0,
  "followers": 7,
  "following": 14,
  "created_at": "2013-10-01T00:33:23Z",
  "updated_at": "2016-08-02T05:44:01Z"
}
```

The browser seems to have done just fine displaying a JSON response. A JSON response like this is ready for use in your code. It's easy to extract data from this text. Then you can do whatever you want with the data.

A is for "Application"

To close off, let's throw in a couple more examples of APIs.

"Application" can refer to many things. Here are some of them in the context of API:

1. A piece of software with a distinct function.
2. The whole server, the whole app, or just a small part of an app.

Basically any piece of software that can be distinctively separated from its environment, can be an "A" in API, and will probably also have some sort of API.

Let's say you're using a third-party library in your code. Once incorporated into your code, a library becomes part of your overall app. Being a distinct piece of software, the library would likely have an API which allows it to interact with the rest of your code.

Here's another example: In **Object Oriented Design**, code is organized into objects. Your application may have hundreds of objects defined that can interact with one another.

Each object has an API—a set of *public* methods and properties that it uses to interact with other objects in your application.

An object may also have inner logic that is *private,* meaning that it's hidden from the outside scope (and not an API).

From what we have covered, I hope you take away the broader meaning of API as well as the more common uses of the term today.

. . .