

NAO: 3306

30-09-2025

IN-MEXICO PROGRAM BACKEND
DEVELOPER CERTIFICATION

PLAN

Server and Database
Commands

Valeria Anel Duque Salgado

Contents:

Endpoints: URLs used to access different API functions.....	2
-How API endpoints work:	2
API endpoints, end to end:	2
Authentication methods: How to obtain and use access keys or tokens.....	2
How does user authentication work?	2
Query parameters: Options to filter and customize searches.....	3
Response formats: How the returned data is structured.	4
Usage limits: Restrictions on the number of requests you can make.	5
Code examples: Demonstrations of how to use the API in different programming languages.....	6

Endpoints: URLs used to access different API functions.

An API endpoint is the location where an API receives requests for data and functionality. For most services, these endpoints are URLs, just like the ones you use to navigate to a website.

API endpoints are critical to the internet's API ecosystem. By exposing endpoints via APIs, and facilitating access to them via API requests, organizations around the world can share their applications' data and functionality.

-How API endpoints work:

APIs usually facilitate access to functionalities or resources within a server or cluster of servers, which might include text, images, videos, or audio files. To access that functionality, or any of those resources, a client would need to know the relevant API endpoint on the server and then send its request there by specifying a URL.

API endpoints, end to end:

Once a client sends a request, it must go through the following sequence of steps before it reaches the endpoint and receives a response:

- The client sends a request, in the form of a URL, to the API server.
- The API server authenticates the request.
- The API server validates the request's input data.
- The API server responds to the request either by delivering the resource to the client, or by sending back an error code that explains why the request could not be fulfilled.

Authentication methods: How to obtain and use access keys or tokens.

Authentication is not the same as authorization. Authentication determines *who* the user is. Authorization determines *what* they're allowed to do once they've been authenticated, like viewing sensitive data, editing settings, or accessing admin-only dashboards.

How does user authentication work?

The user provides credentials, like a password, biometric scan, or security token, and the system verifies those credentials against a trusted source before granting access.

At its core, user authentication is a handshake between a person and a system.










Here's an oversimplified version of an authentication flow:

1. User inputs credentials (e.g., email and password).
2. The system sends a request to an authentication service or identity provider (IdP).
3. The IdP checks the credentials against a user directory (like LDAP or Active

Directory).

4. If the credentials are valid, the IdP returns a token or session confirmation.
5. The user gains access to the application or resource.

Types of authentication methods:

Bad: Password	Good: Password and...	Better: Password and...	Best: Passwordless
123456 qwerty password iloveyou Password1	 SMS  Voice	 Authenticator (Push Notifications)  Software Tokens OTP  Hardware Tokens OTP (Preview)	 Authenticator (Phone Sign-in)  Window Hello  FIDO2 security key  Certificates

Query parameters: Options to filter and customize searches.

Query parameters are a defined set of parameters (key-value pair) attached to the end of a URL used to provide additional information to a web server when making requests. They are an important part of the URL that define specific content or actions based on the data being passed. Parameters give you the flexibility to dynamically change the output of your queries depending on their value, and can be used for:

- Changing the argument values for particular transforms and data source functions.
- Inputs in custom functions.

The following is an example:

[https://example.com/path?name=Branch&products=\[Journeys,Email,Universal%20Ads\]](https://example.com/path?name=Branch&products=[Journeys,Email,Universal%20Ads])

In this example, there are two query parameters:

1. 'name' with the value "Branch"
2. 'products' with the value "[Journeys,Email, Universal%20Ads]"

These parameters can be used to instruct the web server how to process the request, such as customizing the page based on the 'name' field or filtering products based on the 'products' list.

<https://www.google.com/search?q=chatgpt>

?q=chatgpt es el query parameter.

q → la clave (query, o búsqueda).

chatgpt → el valor (lo que buscas).

<https://www.amazon.com/s?k=laptop&brand=asus&price=1000-2000>

?k=laptop → palabra clave buscada.

&brand=asus → filtra por marca.

&price=1000-2000 → filtra por rango de precio.

What are query parameters used for?

- API requests: Mobile apps communicate with APIs, and developers can use API query parameters to filter, sort, or customize the data received by the app.
- Deep linking: Query parameters can be used in deep linking to pass information to an app to open a specific in-app screen.
- Search: For mobile apps that have search functionality, query parameters can be used to pass search queries to the app's server or API.
- Tracking and attribution: Many mobile apps use query parameters to track user interactions and measure campaign effectiveness.

Response formats: How the returned data is structured.

When you make a request to a server (for example, asking an API for data), the **response format** defines how the server structures and delivers the data back to you. It's basically the *"language"* or *"layout"* of the returned data.

The server usually tells you the format in the HTTP header Content-Type.

Common Response Formats

1. **HTML (HyperText Markup Language)**
 - Example:
 - `<html>`
 - `<body>`
 - `<h1>Hello World</h1>`
 - `</body>`
 - `</html>`
 - Used for web pages that browsers display.
2. **JSON (JavaScript Object Notation)**
 - Example:
 - `{`
 - `"name": "Valeria",`
 - `"age": 25,`
 - `"country": "Mexico"`
 - `}`
 - Very common in APIs. Lightweight, easy for machines and humans.

- Content-Type: application/json
- 3. **XML (eXtensible Markup Language)**
 - Example:
 - <person>
 - <name>Valeria</name>
 - <age>25</age>
 - <country>Mexico</country>
 - </person>
 - Older format, but still used in enterprise systems.
 - Content-Type: application/xml
- 4. **Plain Text**
 - Example:
 - Hello World
 - Simple text response.
 - Content-Type: text/plain
- 5. **CSV (Comma-Separated Values)**
 - Example:
 - name,age,country
 - Valeria,25,Mexico
 - Good for spreadsheets and tabular data.
 - Content-Type: text/csv
- 6. **Binary / File formats**
 - Example: Images, PDFs, videos.
 - Content-Type: image/png, application/pdf, etc.

Usage limits: Restrictions on the number of requests you can make:

Usage limits are **restrictions set by an API or service** on the number of requests you can make in a given period of time.

They exist to:

- Prevent **overloading** the server.
- Ensure **fair use** across all users.
- Protect against **abuse** (like bots or spam).

How They Work

Usually, usage limits are defined by:

1. **Number of requests** → e.g., 100 requests per minute.
2. **Time window** → per second, per minute, per hour, or per day.
3. **Scope** →
 - Per **user** (your account).
 - Per **API key** (the token you use).
 - Per **IP address** (your device's network).

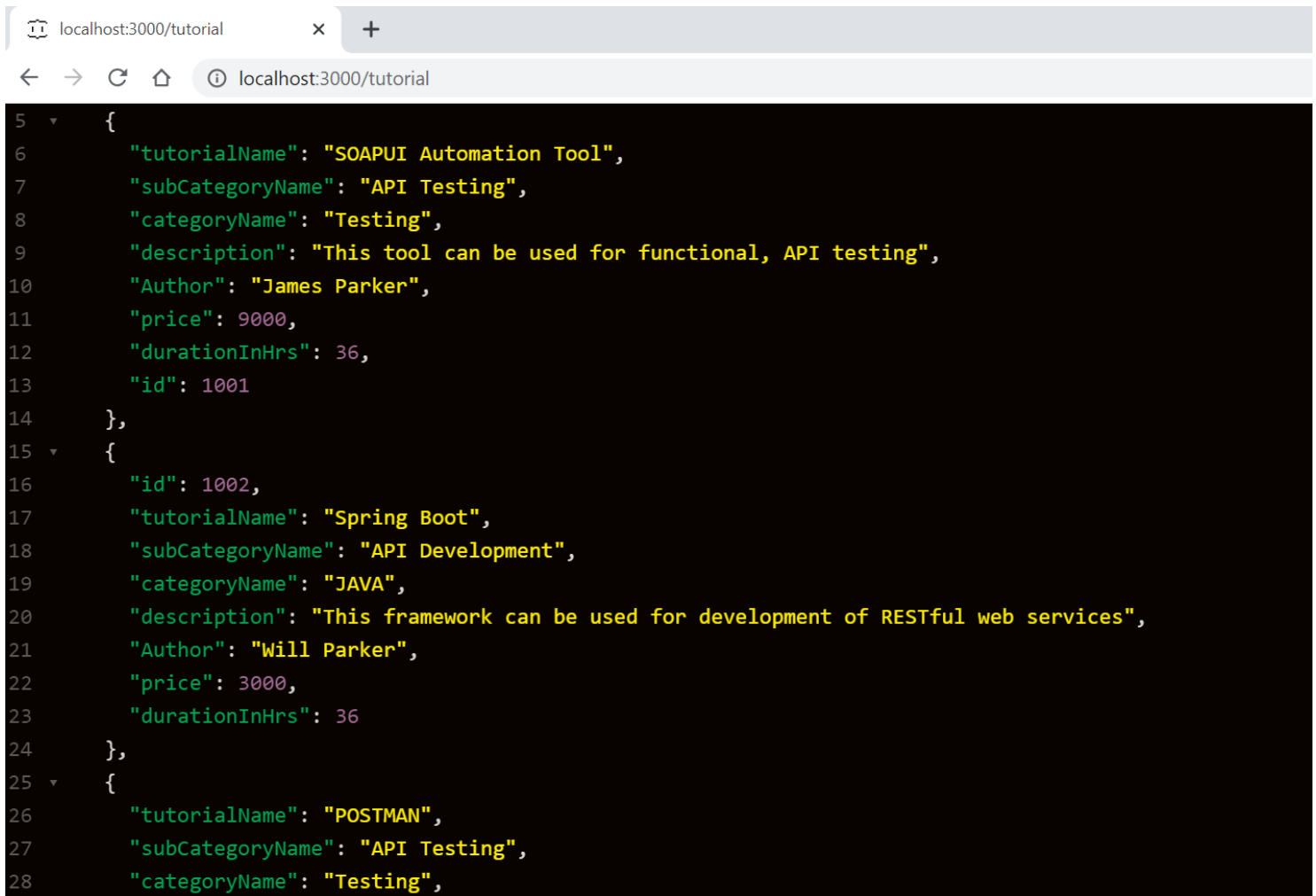
Examples:

- **Twitter API (X)**: might allow 900 requests every 15 minutes for standard accounts.
- **GitHub API**: limits unauthenticated users to 60 requests per hour, but authenticated users get 5,000 per hour.
- **Google Maps API**: for free accounts, about 2,500 requests per day.

What Happens If You Exceed the Limit?:

- The server may return an **HTTP error code** like:
 - 429 Too Many Requests → means you hit the limit.
- Sometimes the response includes a **Retry-After** header telling you how long to wait before trying again.

Code examples: Demonstrations of how to use the API in different programming languages.



The screenshot shows a web browser window with the address bar displaying 'localhost:3000/tutorial'. The browser's developer tools are open, showing a JSON response from the API. The JSON is a list of three tutorial objects. The first object is for 'SOAPUI Automation Tool', the second for 'Spring Boot', and the third for 'POSTMAN'. Each object contains fields for 'tutorialName', 'subCategoryName', 'categoryName', 'description', 'Author', 'price', 'durationInHrs', and 'id'.

```
5 {
6   "tutorialName": "SOAPUI Automation Tool",
7   "subCategoryName": "API Testing",
8   "categoryName": "Testing",
9   "description": "This tool can be used for functional, API testing",
10  "Author": "James Parker",
11  "price": 9000,
12  "durationInHrs": 36,
13  "id": 1001
14 },
15 {
16   "id": 1002,
17   "tutorialName": "Spring Boot",
18   "subCategoryName": "API Development",
19   "categoryName": "JAVA",
20   "description": "This framework can be used for development of RESTful web services",
21   "Author": "Will Parker",
22   "price": 3000,
23   "durationInHrs": 36
24 },
25 {
26   "tutorialName": "POSTMAN",
27   "subCategoryName": "API Testing",
28   "categoryName": "Testing",
```