TechnoReady In-México

2025

Technical Report

Challenge 3

DIEGO GONZÁLEZ MIRANDA

NAO ID: 3338

September 30, 2025

Backend Developer Certification

Server and Database Commands

Table of contents

Endpoints

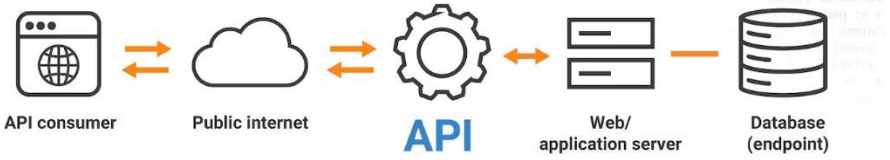
Endpoints are the URLs that allow a user or an application to communicate with the API to access different functions. Each endpoint has a specific purpose, such as searching for articles, retrieving author profiles, or obtaining citations of publications.

In SerpApi, the main endpoints for Google Scholar are:

Endpoint	**Function**
/search.json?engine=google_scholar	Performs academic article searches using keywords or phrases.
/search.json?engine=google_scholar_profiles	Allows searching for author profiles in Google Scholar and retrieves basic information for each researcher.
/search.json?engine=google_scholar_author	Retrieves detailed information about a specific author, including articles and metrics.
/search.json?engine=google_scholar_cite	Returns citations of an article in different bibliographic formats, such as APA, MLA, or Chicago.

Example usage:

https://serpapi.com/search.json?engine=google_scholar&q=machine+learning&api_key=YOUR_API_KEY



Authentication Methods

Authentication allows the API to identify and validate the user making requests. This ensures that only authorized users can access the services.

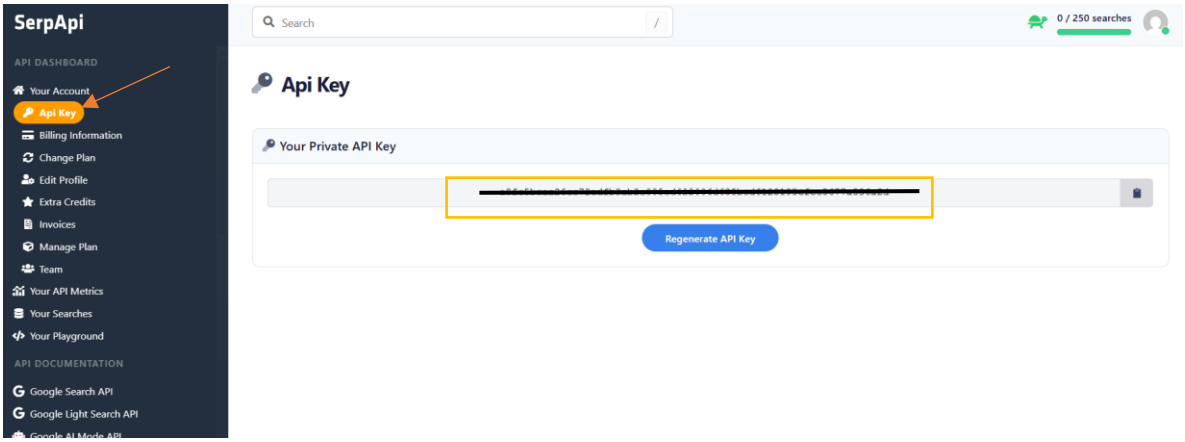
In SerpApi, the only authentication method is the API Key, which is a unique string associated with each account. Temporary tokens, or username/password credentials are not used.

How to obtain and use the API Key:

1. Create a free account on SerpApi.
2. Access the user dashboard and copy the assigned API key.
3. Include the API Key in every request using the api_key parameter.

Example:

https://serpapi.com/search.json?engine=google_scholar&q=deep+learning&api_key=YOUR_API_KEY



SerpAPI and API key created

Query Parameters

Query parameters allow customizing and filtering search results according to user needs. This helps obtain more precise and relevant results.

Parameter	Description	Example
q	Keyword or phrase that defines the topic of articles.	q=deep+learning
hl	Language of the results to receive.	hl=en
num	Number of results per page, maximum 20.	num=10
start	Index from which to start pagination.	start=10
as_ylo	Minimum publication year of articles.	as_ylo=2018

as_yhi	Maximum publication year of articles.	as_yhi=2023

These parameters allow, for example, searching for articles published within a certain period, in a specific language, or controlling the number of results returned per page.

Response Formats

The API response is delivered in JSON format, a standard for structured data that is easy to use in different programming languages.

Response structure:

- search_metadata: Information about the search, such as execution time and reference URL.
- search_parameters: Parameters used to generate the results.
- organic_results: List of articles with key information such as title, authors, year, and link.

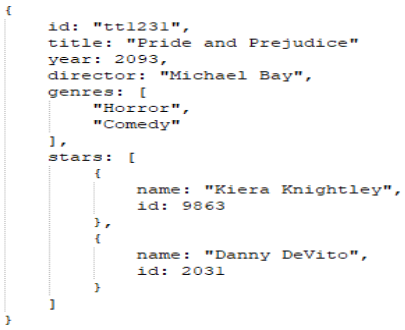
Simplified example:


```
{
  "organic_results": [
    {
      "title": "Deep Learning",
      "authors": "Y LeCun, Y Bengio, G Hinton",
      "year": "2015",
      "link": "https://scholar.google.com/scholar?cluster=123456",
      "cited_by": {
        "count": 12345,
        "link": "https://scholar.google.com/scholar?cites=123456"
      }
    }
  ]
}
```


Important elements:

- title: Article title.

- authors: List of authors.
- year: Year of publication.
- link: Direct URL to the article.
- cited_by: Number of citations and link to documents citing the article.



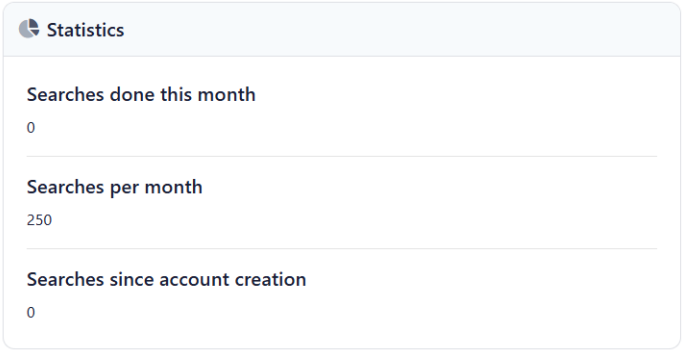
JSON response example

Usage Limits

SerpApi's free plan has limits that restrict the number of requests to prevent service abuse.

Plan	Request Limit	Notes
Free	250 requests/month	Sufficient for testing and learning purposes.

If the limit is exceeded, the API returns an error message indicating that the maximum number of allowed requests has been reached.



Code Examples

The following examples show how to use the Google Scholar API in four different languages, demonstrating how to make requests and process results.

Java

import java.net.URI;

```
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class GoogleScholarExample {
    public static void main(String[] args) {
        String apiKey = "YOUR_API_KEY";
        String query = "machine learning";
        String url = "https://serpapi.com/search.json?engine=google_scholar&q=" + query + "&api_key=" + apiKey;

        HttpClient client = HttpClient.newHttpClient();
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create(url))
                .GET()
                .build();

        try {
            HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());
            System.out.println(response.body());
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Description:
This Java code builds a URL with the query and API Key, sends a GET request to the Google Scholar API, and receives a JSON response. The results are printed to the console.

Python

```
from serpapi import GoogleSearch


params = {
    "engine": "google_scholar",
    "q": "machine learning",
    "api_key": "YOUR_API_KEY"
}


search = GoogleSearch(params)
results = search.get_dict()
for article in results["organic_results"]:
    print(article["title"])
```

Description:
In Python, a dictionary is created with the search parameters and API Key. GoogleSearch automatically sends the request and returns the results, which are then looped through to print the title of each article.

JavaScript (Node.js)

```
const SerpApi = require('google-search-results-nodejs');
const search = new SerpApi.GoogleSearch("YOUR_API_KEY");


const params = {
    engine: "google_scholar",
    q: "machine learning"
};


search.json(params, (data) => {
    data.organic_results.forEach(article => console.log(article.title));
});
```

Description:
This Node.js example uses the official SerpApi library. A search object is created with the query and API Key. The search.json() function sends the request and executes a callback that loops through the results, printing the titles of the articles.

cURL

```
curl
"https://serpapi.com/search.json?engine=google_scholar&q=machine+learning&api_key=YOUR_API_KEY"
```

Description:
With cURL, the URL is built directly with the query and API Key. Running this command in the terminal returns the JSON response immediately. It is useful for quick testing or simple shell scripts.

Conclusion

The Google Scholar API via SerpApi allows automated access to reliable and structured information. Its specific endpoints, API Key authentication, customizable search parameters, and JSON responses facilitate integration into applications and research projects.

The free plan is sufficient for learning and testing purposes. In summary, SerpApi optimizes the information retrieval process, saves time, and improves productivity for researchers and developers.