Becalos Techno Ready Digital NAO TECMILENIO

Javier David Barraza Ureña

NAO ID: 3303

September 30th of 2025

In-Mexico Program Backend Developer Certification Septiembre - Diciembre 2025

Server and Database Commands

| Sprint 1 | Develop | Technical Report

Table of contents:

This report documents the essential aspects of the Google Scholar API provided by SerpApi, a service for extracting academic results from Google Scholar in a structured way. It is based on the official SerpApi documentation (updated to 2025). The API is ideal for applications that require searching for publications, citations, and author profiles.

Endpoints:

Endpoints are the base URLs for accessing API functions. SerpApi uses a unified endpoint for most search engines, including Google Scholar.

- **Base URL:** https://serpapi.com/search

- **Main Endpoint for Google Scholar Searches:** /search?engine=google_scholar
  - **Example of a complete query:** https://serpapi.com/search?engine=google_scholar&q=python+programming&api_key=YOUR_API_KEY

- **Other Related Endpoints:**
  - **Google Scholar Author:** /search?engine=google_scholar_author (retrieves author-specific data).
  - **Google Scholar Cite:** /search?engine=google_scholar_cite (for specific citations).
  - **Google Scholar Profiles:** /search?engine=google_scholar_profiles (for profile searches).

- **Method:** Always **GET** (simple requests without body).
- **Interactive Demo:** Use the SerpApi playground at serpapi.com/playground to try it out live.

Authentication Methods:

Authentication is simple and based on a private API key.

- **Main Method:** Use the api_key parameter in each GET request.
  - **Example:** &api_key=your_secret_key_here

- **How to Obtain the Key:**
  1. Sign up at serpapi.com and select the free plan.
  2. Go to the dashboard > API Keys > Generate a new key.

- **Secure Use:** Store the key in environment variables (e.g., process.env.SERPAPI_KEY in Node.js) or secrets in GitHub. Do not expose it in source code.

- **Common Errors:** If the key is invalid, you will receive a 401 Unauthorized error.

Query Parameters

Query parameters allow you to filter and customize searches. They are optional except for those that are required. They are added as query strings (e.g., ?param1=value&param2=value).

Parameter	Type	Required	Description	Example
engine	String	Yes	Specifies the search engine; must be google_scholar.	engine=google_scholar
Q	String	Yes (unless using cites)	Search query; supports advanced operators like author:, source:.	q=python+programming+author:guido+van+rossum
api_key	String	Yes	Authentication key for API access.	api_key=abcSECRET123
Hl	String	No	Interface language (2-letter code).	hl=en (English)
Lr	String	No	Restricts results to specific languages (e.g., `lang_xx	lang_yy`).
as_ylo / as_yhi	Integer	No	Publication year range (start/end).	as_ylo=2020&as_yhi=2025
start	Integer	No	Pagination offset (e.g., 0 for first page).	start=10
Num	Integer	No	Number of results per page (1–20, default=10).	num=20
cites	String	No	Article ID for “Cited by” searches.	cites=1275980731835430123
cluster	String	No	Article ID for “All versions” searches.	cluster=1275980731835430123
no_cache	Boolean	No	Forces fresh results, bypassing 1-hour cache.	no_cache=true
output	String	No	Response format: json (default) or html.	output=json

Notes: Maximum 20 results per page; use pagination (*start, num*) for additional results. Full parameter list available at https://serpapi.com/google-scholar-api

Response Formats

Responses are returned in JSON format by default, structured for easy parsing. HTML output is available but less common.


```
{
 "search_metadata": {
  "id": "unique_search_id",
  "status": "Success", // or "Processing", "Error"
  "json_endpoint": "url_to_json_response",
  "created_at": "2025-09-30T20:16:00Z",
  "processed_at": "2025-09-30T20:16:01Z",
  "google_scholar_url": "https://scholar.google.com/scholar?q=python+programming",
  "raw_html_file": "url_to_html" // if output=html
 },
 "search_parameters": { /* Echoes query parameters */ },
 "search_information": {
  "organic_results_count": 10,
  "total_time_taken": 0.5 // in seconds
 },
 "organic_results": [
  {
   "position": 1,
   "title": "Python Programming: An Introduction to Computer Science",
   "link": "https://scholar.google.com/...",
   "authors": ["John Zelle"],
   "publication_info": "Book, 2010",
   "cited_by": { "value": 1200, "link": "https://scholar.google.com/..." },
   "year": 2010,
```

```
    "snippet": "This book introduces computer science using Python...",
    "resources": [ /* Links to PDFs or other resources */ ]
   }
  ],
  "profiles": [ /* Optional: related author profiles */ ]
}
```


- **Alternative Format:** output=html returns raw HTML (useful for debugging).
- **Error Handling:** If "status": "Error", check the error field (e.g., {"code": "RateLimitExceeded", "message": "Too many requests"}).
- **Optimization:** Use json_restrictor (paid plans) to limit response fields.

Usage Limits

SerpApi enforces limits to prevent abuse, varying by subscription plan.

- **Free Plan:**
  - 100 successful searches per month.
  - Cached searches (within 1 hour) and failed searches do not count toward the limit.

- **General Limits:**
  - Rate Limit: Approximately 1,000 searches/hour in the free plan; higher in paid plans (up to 100,000/hour for enterprise).
  - Results per Request: Maximum 20 results per call; use pagination for more.
  - Cache: Identical searches within 1 hour are served from cache (free).
  - Monitoring: Track usage via the SerpApi dashboard. Exceeding limits results in a 429 Too Many Requests error.

- **Best Practices:** Distribute requests evenly to avoid rate limits. Upgrade to Developer Plan (5,000 searches/month) for larger projects.

Code Examples:

Below are examples in multiple programming languages to perform a basic Google Scholar search (query: "python programming"). Replace YOUR_API_KEY with your SerpApi key.

- Python (using request):

```python
import requests

params = {
    "engine": "google_scholar",
    "q": "python programming",
    "api_key": "YOUR_API_KEY"
}

response = requests.get("https://serpapi.com/search", params=params)
data = response.json()

if data["search_metadata"]["status"] == "Success":
    for result in data["organic_results"]:
        print(f"Title: {result['title']}")
        print(f"Authors: {', '.join(result['authors'])}")
```

- Java (using Spring boot and HttpClient)

```java
import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

@RestController
@RequestMapping("/apisaludos")
public class ScholarController {

    @Value("${serpapi.key}")
    private String API_KEY;

    private static final String BASE_URL = "https://serpapi.com/search?engine=google_scholar";

    @GetMapping("/buscar")
    public String searchArticles(@RequestParam String q) throws Exception {
        String url = BASE_URL + "&q=" + q.replace(" ", "+") + "&api_key=" + API_KEY;

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

        HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());
        return response.body(); // Returns JSON
    }
}
```

- JavaScript (Node.js, using node-fetch)

```javascript
const fetch = require('node-fetch');

const params = new URLSearchParams({
    engine: 'google_scholar',
    q: 'python programming',
    api_key: 'YOUR_API_KEY'
});

fetch(`https://serpapi.com/search?${params}`)
    .then(response => response.json())
    .then(data => {
        if (data.search_metadata.status === 'Success') {
            data.organic_results.forEach(result => {
                console.log(`Title: ${result.title}`);
                console.log(`Authors: ${result.authors.join(', ')}`);
            });
        }
    });
```