In-Mexico Program
Backend Developer Certification

Server and Database Commands

Technical Report

Name: Jared Alexander Trujillo Ortiz
NAO ID: 3347
Date: 30 September 2025

Table of contents

Introduction

The Google Scholar API via SerpApi allows you to scrape and extract academic research data from Google Scholar without manually browsing the website.

Authorization

Authentication is done with a SerpApi API key. Add it as a query parameter in order to use the API properly.

The key is given when creating a SerpApi account.

Example:

https://serpapi.com/search?engine=google_scholar&q=AI&**api_key=YOUR_KEY**

Endpoints

Endpoint	Description
GET https://serpapi.com/search?engine=google_scholar	**Scholar Search:** Articles, cases, citations
GET https://serpapi.com/search?engine=google_scholar_author	**Scholar Author:** Profile, articles, citations, co-authors
GET https://serpapi.com/search?engine=google_scholar_author	**Account API:** Check usage and limits

Query parameters

engine

Requirement level: Required

Proper name: SerpApi engine

Description:

Selects the SerpApi engine. Use engine=google_scholar for article searches or engine=google_scholar_author for author profiles.

Example:

```
https://serpapi.com/search?engine=google_scholar&q=AI&api_key=YOUR_KEY
```

api_key

Requirement level: Required

Proper name: API key (authentication)

Description:

Your SerpApi private key. Required for all requests; keep secret and use environment variables.

Example:

```
https://serpapi.com/search?engine=google_scholar&q=AI&api_key=YOUR_KEY
```

q

Requirement level: Required (unless using cites or cluster alone)

Proper name: Search query

Description:

Free-text search string. Use + or %20 for spaces and helpers like author: or source: to narrow results.

Example:

```
https://serpapi.com/search?engine=google_scholar&q=deep+learning&api_key=YOUR
_KEY
```

cites

Requirement level: Optional

Proper name: Cited-by ID

Description:

Return documents that cite a given article. Useful for citation tracking and impact analysis. Makes q optional; with q searches within citing docs.

Example:

```
https://serpapi.com/search?engine=google_scholar&cites=1275980731835430123&ap
i_key=YOUR_KEY
```

cluster

Requirement level: Optional

Proper name: All-versions ID (cluster)

Description:

Return all versions of a paper. Use alone (do not combine with q or cites). Useful to find PDFs and preprints.

Example:

```
https://serpapi.com/search?engine=google_scholar&cluster=1275980731835430123&
api_key=YOUR_KEY
```

as_ylo / as_yhi

Requirement level: Optional

Proper name: Year range (from/to)

Description:

Filter results by publication year range. Combine as_ylo (from) and as_yhi (to) to limit results to a date window.

Example:

```
https://serpapi.com/search?engine=google_scholar&q=reinforcement+learning&as_
ylo=2019&as_yhi=2024&api_key=YOUR_KEY
```

start / num

Requirement level: Optional

Proper name: Pagination (offset and page size)

Description:

Use start to offset results and num to set results per page. For google_scholar num is 1–20. Use together for paging through results.

Example:

```
https://serpapi.com/search?engine=google_scholar&q=AI&start=20&num=10&api_key
=YOUR_KEY
```

hl

Requirement level: Optional

Proper name: Interface language (hl)

Description:

Set UI language / localization (two-letter code). Can affect labels and localized result ordering.

Example:

```
https://serpapi.com/search?engine=google_scholar&q=machine+learning&hl=en&api
_key=YOUR_KEY
```

no_cache

Requirement level: Optional

Proper name: No-cache (fresh fetch)

Description:

Set no_cache=true to bypass SerpApi cache and fetch fresh results. Cached identical searches are free for ~1 hour.

Example:

```
https://serpapi.com/search?engine=google_scholar&q=AI&no_cache=true&api_key=Y
OUR_KEY
```

output

Requirement level: Optional

Proper name: Output format

Description:

Set output=json (default) for structured results or output=html for raw HTML capture.

Example:

```
https://serpapi.com/search?engine=google_scholar&q=AI&output=json&api_key=YOU
R_KEY
```

json_restrictor

Requirement level: Optional

Proper name: JSON restrictor (field selector)

Description:

Return only selected fields to reduce response size and speed parsing. Useful in production to save bandwidth.

Example:

```
https://serpapi.com/search?engine=google_scholar&q=AI&json_restrictor=organic
_results.title,organic_results.link&api_key=YOUR_KEY
```

author_id

Requirement level: Required when using engine=google_scholar_author

Proper name: Author profile ID

Description:

Use with engine=google_scholar_author to fetch an author's profile, articles, and citations.

Example:

```
https://serpapi.com/search?engine=google_scholar_author&author_id=LSsXyncAAAA
J&api_key=YOUR_KEY
```

Common composite examples

1) Topic search with year filter + pagination (useful for literature review pages):

```
https://serpapi.com/search?engine=google_scholar&q=reinforcement
+learning&as_ylo=2018&as_yhi=2024&start=0&num=20&api_key=YOUR_KE
Y
```

2) Cited-by search (find all papers that cite a high-impact paper):

```
https://serpapi.com/search?engine=google_scholar&cites=127598073
1835430123&num=20&api_key=YOUR_KEY
```

3) Author profile (get up to 100 articles sorted by pubdate):

```
https://serpapi.com/search?engine=google_scholar_author&author_i
d=LSsXyncAAAAJ&num=100&sort=pubdate&api_key=YOUR_KEY
```

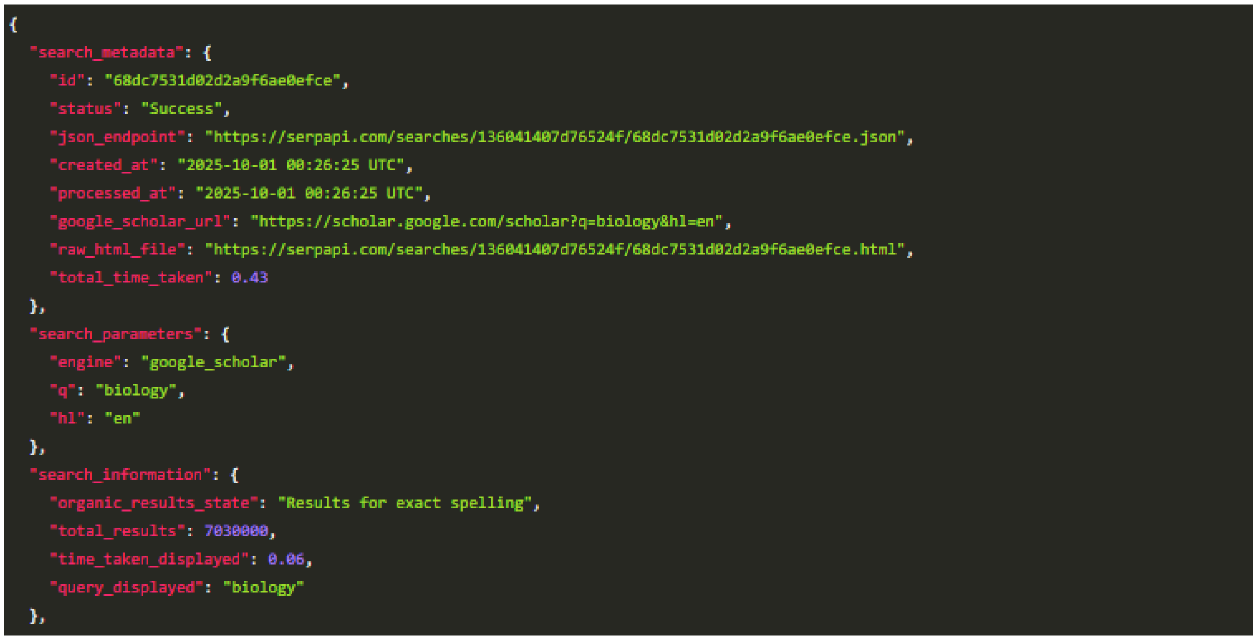
In case for more information about query parameters go to the official documentation: https://serpapi.com/google-scholar-api

Response formats

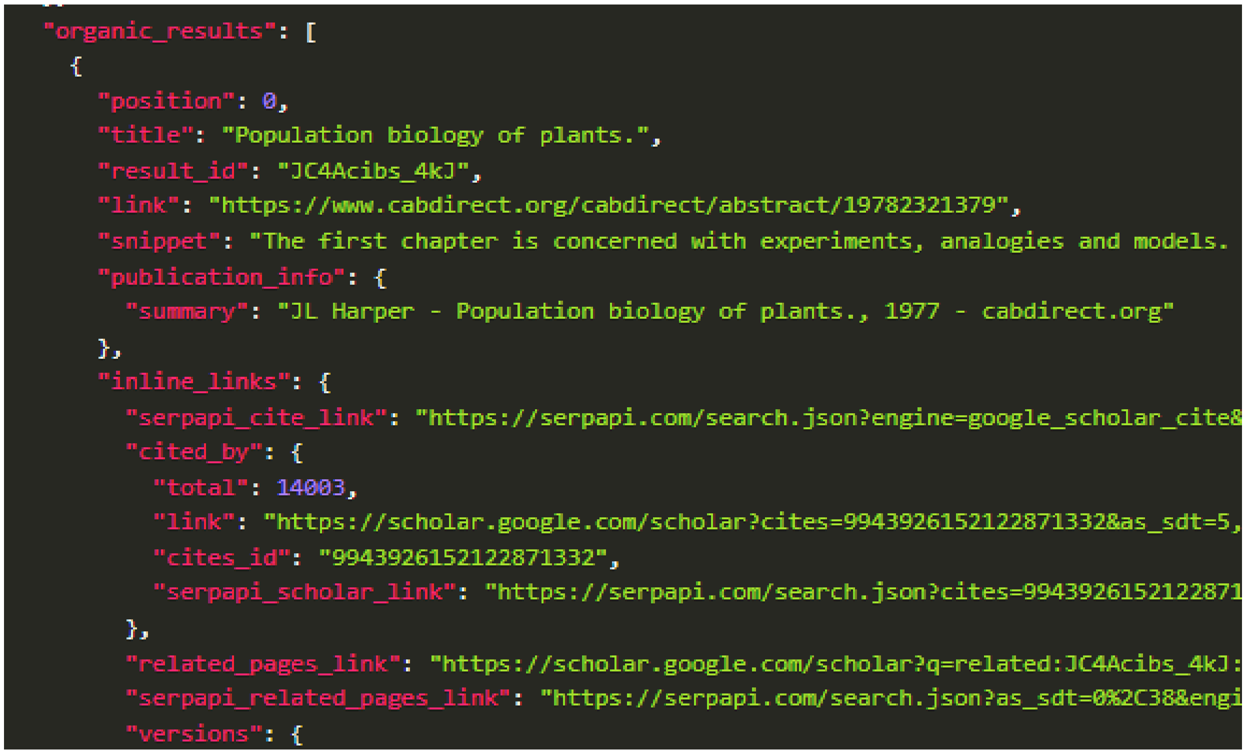
Responses are returned in JSON by default. Example fields:
- search_metadata: Info about request.
- organic_results: Search results (Scholar Search).
- author, articles, cited_by: Author data.

Here to see more response examples: https://serpapi.com/google-scholar-api

Example of JSON response:



```
{
  "search_metadata": {
    "id": "68dc7531d02d2a9f6ae0efce",
    "status": "Success",
    "json_endpoint": "https://serpapi.com/searches/136041407d76524f/68dc7531d02d2a9f6ae0efce.json",
    "created_at": "2025-10-01 00:26:25 UTC",
    "processed_at": "2025-10-01 00:26:25 UTC",
    "google_scholar_url": "https://scholar.google.com/scholar?q=biology&hl=en",
    "raw_html_file": "https://serpapi.com/searches/136041407d76524f/68dc7531d02d2a9f6ae0efce.html",
    "total_time_taken": 0.43
  },
  "search_parameters": {
    "engine": "google_scholar",
    "q": "biology",
    "hl": "en"
  },
  "search_information": {
    "organic_results_state": "Results for exact spelling",
    "total_results": 7030000,
    "time_taken_displayed": 0.06,
    "query_displayed": "biology"
  },
```



```
  "organic_results": [
    {
      "position": 0,
      "title": "Population biology of plants.",
      "result_id": "JC4Acibs_4kJ",
      "link": "https://www.cabdirect.org/cabdirect/abstract/19782321379",
      "snippet": "The first chapter is concerned with experiments, analogies and models.
      "publication_info": {
        "summary": "JL Harper - Population biology of plants., 1977 - cabdirect.org"
      },
      "inline_links": {
        "serpapi_cite_link": "https://serpapi.com/search.json?engine=google_scholar_cite&
        "cited_by": {
          "total": 14003,
          "link": "https://scholar.google.com/scholar?cites=9943926152122871332&as_sdt=5,
          "cites_id": "9943926152122871332",
          "serpapi_scholar_link": "https://serpapi.com/search.json?cites=994392615212287l
        },
        "related_pages_link": "https://scholar.google.com/scholar?q=related:JC4Acibs_4kJ:
        "serpapi_related_pages_link": "https://serpapi.com/search.json?as_sdt=0%2C38&engi
        "versions": {
```

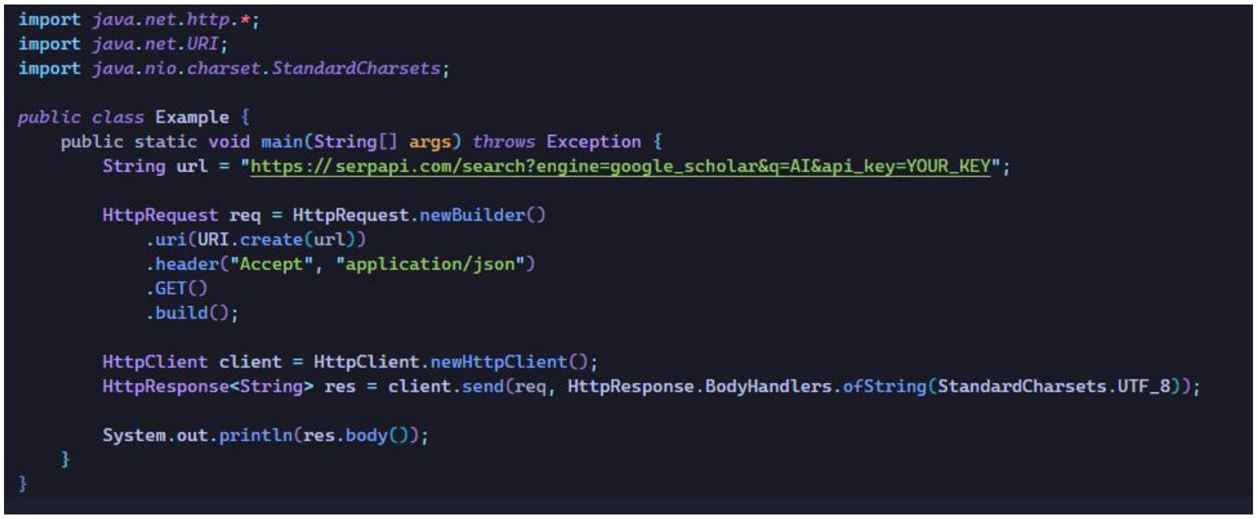
Usage limits

Plan	Price (monthly)	Monthly Searches	Notes
Free	$0	250	Free tier
Developer	$75	5,000	Paid monthly
Production	$150	15,000	Paid monthly
Big Data	$275	30,000	Paid monthly
Enterprise	Custom	Custom	

For more information: https://serpapi.com/pricing

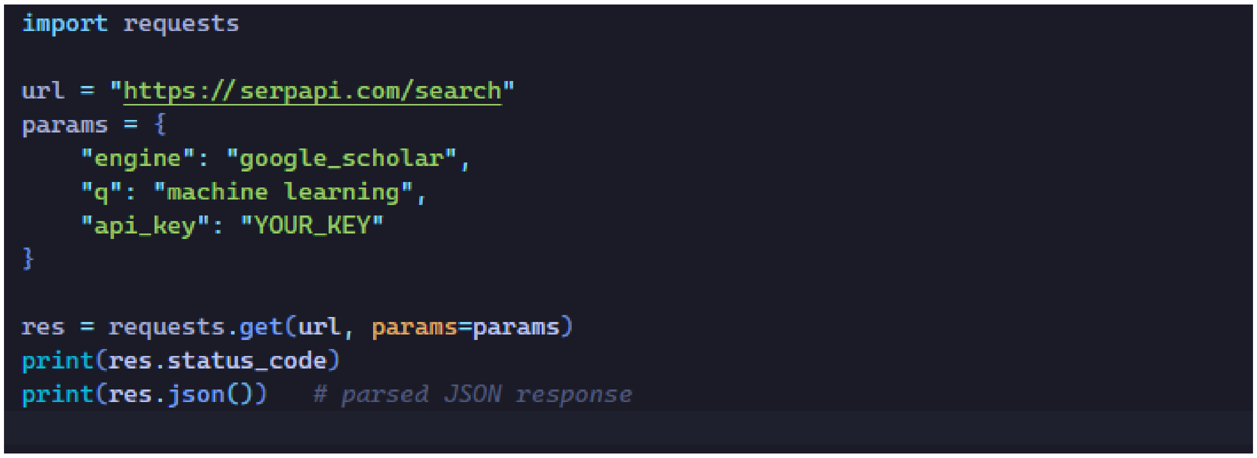
Code examples

Java



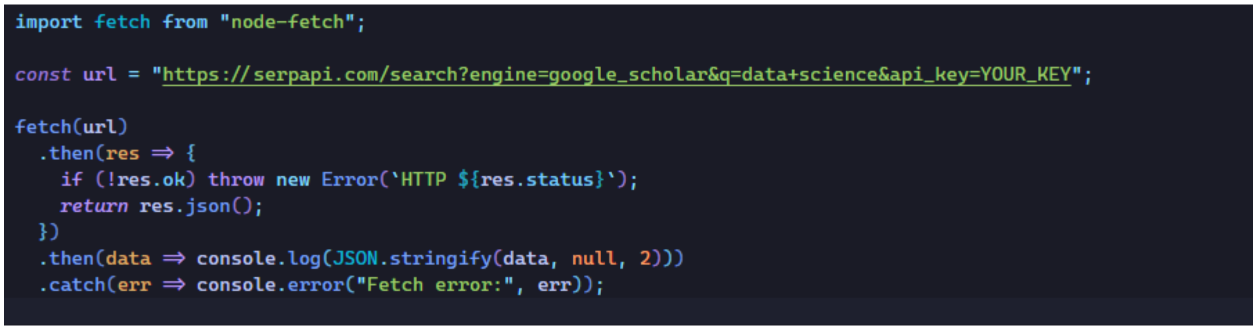
```java
import java.net.http.*;
import java.net.URI;
import java.nio.charset.StandardCharsets;

public class Example {
    public static void main(String[] args) throws Exception {
        String url = "https://serpapi.com/search?engine=google_scholar&q=AI&api_key=YOUR_KEY";

        HttpRequest req = HttpRequest.newBuilder()
            .uri(URI.create(url))
            .header("Accept", "application/json")
            .GET()
            .build();

        HttpClient client = HttpClient.newHttpClient();
        HttpResponse<String> res = client.send(req, HttpResponse.BodyHandlers.ofString(StandardCharsets.UTF_8));

        System.out.println(res.body());
    }
}
```

Python



```python
import requests

url = "https://serpapi.com/search"
params = {
    "engine": "google_scholar",
    "q": "machine learning",
    "api_key": "YOUR_KEY"
}

res = requests.get(url, params=params)
print(res.status_code)
print(res.json())    # parsed JSON response
```

Javascript



```javascript
import fetch from "node-fetch";

const url = "https://serpapi.com/search?engine=google_scholar&q=data+science&api_key=YOUR_KEY";

fetch(url)
  .then(res => {
    if (!res.ok) throw new Error(`HTTP ${res.status}`);
    return res.json();
  })
  .then(data => console.log(JSON.stringify(data, null, 2)))
  .catch(err => console.error("Fetch error:", err));
```