Google Scholar API

Server and Database Commands

Luis David Martínez Gutiérrez

Sprint 1

29 September 2025

Table of Contents

1. Overview

This document provides a comprehensive technical guide for developers on using the Google Scholar API, accessed via the SerpApi service. The purpose of this report is to outline the necessary procedures for connecting to the API, making requests, and interpreting the responses. This information will form the technical foundation for the Java application designed to automate the integration of researcher data for the university.

2. API Endpoints

SerpApi simplifies API interaction by using a single, unified endpoint for all search queries. The specific data source and search type are controlled by a mandatory engine parameter within the request.

- **Primary Endpoint URL:** https://serpapi.com/search

To query Google Scholar, the engine parameter must be set to one of the following values, depending on the required data:

- **google_scholar**: For general searches of articles, profiles, and case law.
- **google_scholar_profiles**: To search specifically for author profiles by name or affiliation.
- **google_scholar_author**: To retrieve the complete publication history for a specific author using their unique author_id. This will be the primary engine for this project.

- **google_scholar_cite**: To retrieve the citation data and formats for a specific article.

3. Authentication methods

All requests to the SerpApi endpoint must be authenticated. Authentication is performed by including your private API key in every request.

- **Method:** API Key in URL Parameter
- **Parameter Name:** api_key
- **Details:** The API key is a unique string assigned to your account. It must be kept confidential and should be included as a query parameter in the request URL. Calls made without a valid api_key will fail.

4. Query Parameters

To customize and refine API requests, a series of query parameters must be appended to the endpoint URL. Each parameter defines a specific aspect of the search. For every request, two parameters are mandatory: `api_key`, which must contain your private authentication key, and `engine`, which specifies the search engine to use (e.g., `google_scholar_author`). For the core functionality of this project, the most critical parameter is `author_id`, which accepts the unique identifier of a researcher's profile to

retrieve their complete list of publications. For broader searches, the `q` parameter can be used for general article queries by keyword, while `mauthors` is used to find author profiles by name. Finally, the output can be controlled through several utility parameters. The `hl` parameter sets the language of the results (e.g., `'en'` for English), while pagination is handled using `start` to define the result offset and `num` to specify the number of results to return per page.

5. Response Format

The API returns data structured in **JSON** (JavaScript Object Notation) format, which is lightweight and easily parsable by most programming languages, including Java.

A typical response from the google_scholar_author engine is organized as follows:

- search_metadata: Contains metadata about the API request itself, such as its status, the time it took to complete, and the URL used.
- author: A JSON object containing detailed information about the researcher's profile, including their name, affiliation, email, and list of academic interests.
- articles: A JSON array where each element is an object representing a single publication. Each article object contains fields like title, authors, publication, link, and cited_by.

- cited_by: A summary object containing citation metrics like total citations and h-index.
- pagination: An object containing URLs to the next and previous pages of results, essential for retrieving all articles.

6. Usage Limits

The SerpApi free plan is subject to specific usage limits that must be considered during development and testing.

- **Monthly Limit:** The free plan allows for **100 successful searches per month**.
- **Tracking:** Each successful API call (HTTP 200 OK) decrements this counter. It is critical to use the API efficiently to stay within this limit during the development phase.
- **Exceeding Limits:** Once the limit is reached, subsequent API calls will fail until the monthly cycle resets. Usage can be monitored from the SerpApi account dashboard.

7. Code Example (Java)

The following Java code provides a practical example of how to construct a URL, make a GET request using the built-in HttpClient, and handle the response.

```
import java.io.IOException;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class ScholarApiExample {

    public static void main(String[] args) {
        final String API_KEY = "YOUR_SECRET_API_KEY";
        final String AUTHOR_ID = "z_vX_bsAAAAJ";

        // Build the dynamic URL for the request
        String url = String.format(
            "https://serpapi.com/search.json?engine=google_scholar_author&author_id=%s&api_key=%s&num=20",
            AUTHOR_ID, API_KEY
        );

        // Create an HTTP client and build the request
        HttpClient client = HttpClient.newHttpClient();
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create(url))
                .header("Accept", "application/json")
                .build();

        // Send the request and process the response
        try {
            HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());

            if (response.statusCode() == 200) {
                System.out.println("API Request Successful!");
                System.out.println("--- RESPONSE BODY ---");
                System.out.println(response.body());
            } else {
                System.err.println("Error: Received response code " + response.statusCode());
                System.err.println("--- ERROR BODY ---");
                System.err.println(response.body());
            }

        } catch (IOException | InterruptedException e) {
            System.err.println("An exception occurred while making the API request.");
            e.printStackTrace();
        }
    }
}
```

8. Conclusion

The SerpApi service provides a reliable and straightforward RESTful interface for accessing Google Scholar data. By using the google_scholar_author engine, developers can effectively retrieve comprehensive publication data for specific researchers. Successful implementation requires careful management of the private api_key, adherence to monthly usage limits, and proper parsing of the structured JSON response. The provided Java example serves as a solid starting point for the development of the data integration module.

9. References

- SerpApi. (n.d.). *Google Scholar API*. SerpApi. Recuperado el 29 de septiembre de 2025, de https://serpapi.com/google-scholar-api
- SerpApi. (n.d.). *Dashboard*. SerpApi. Recuperado el 29 de septiembre de 2025, de https://serpapi.com/dashboard