September 30th 2025

IN-MEXICO PROGRAM BACKEND DEVELOPER CERTIFICATION

Server and Database Commands

Made By:

Emilio Flores Licea

NAO ID: 3307

GIT Repository link:

https://github.com/FloresEmilioIg/Challenge3

Google Scholar API

Endpoints:

- **Google Scholar Search Endpoint:** This endpoint is used to perform general searches for academic content. Example of code:

  GET /search?engine=google_scholar&q={search_query}

- **Google Scholar Author Search Endpoint:** This endpoint allows users to retrieve information about authors. It uses an author name or ID as a parameter. Example of code:

  GET /search?engine=google_scholar_author&q={author_name}

- **Google Scholar Cited By Search Endpoint:** This endpoint enables users to find articles that cite a specific article. Example of code:

  GET /search?engine=google_scholar&cites={article_id}

- **Google Scholar Article Versions/Cluster Search Endpoint:** This endpoint is used to find different versions or related articles within a cluster based on a unique article ID (Cluster). Example of code:

  GET /search?engine=google_scholar&cluster={article_id}

Authentication methods:

Normally, Google Scholar does not offer a public, direct API for general use. So, the next best thing that can be used are third-party scraping services. We'll use SerpApi as our option on this case.

As to how we will use it to gain access to access keys or tokens it will be done in a few steps:

1. **Register and Obtain an API Key:**
   - Sign up for an account on SerpApi.
   - Navigate to the API Key Management or Developer section within your account dashboard.

- Generate a unique API key or token, often by clicking a "Create" or "Generate" button.
- Copy the generated API key.

2. **Integrate the API Key into Your Code:**
    - Consult the documentation provided by the third-party service for details on how to use their API.
    - Typically, you will include the API key as a parameter in your API requests, often in the Auth field or as a key parameter in the request URL.
    - Configure other necessary parameters for your query, such as the search query (q), engine type (e.g., google_scholar), and any filters (e.g., year range, language).
    - Execute your code to send the API request and receive the structured data (usually in JSON format).

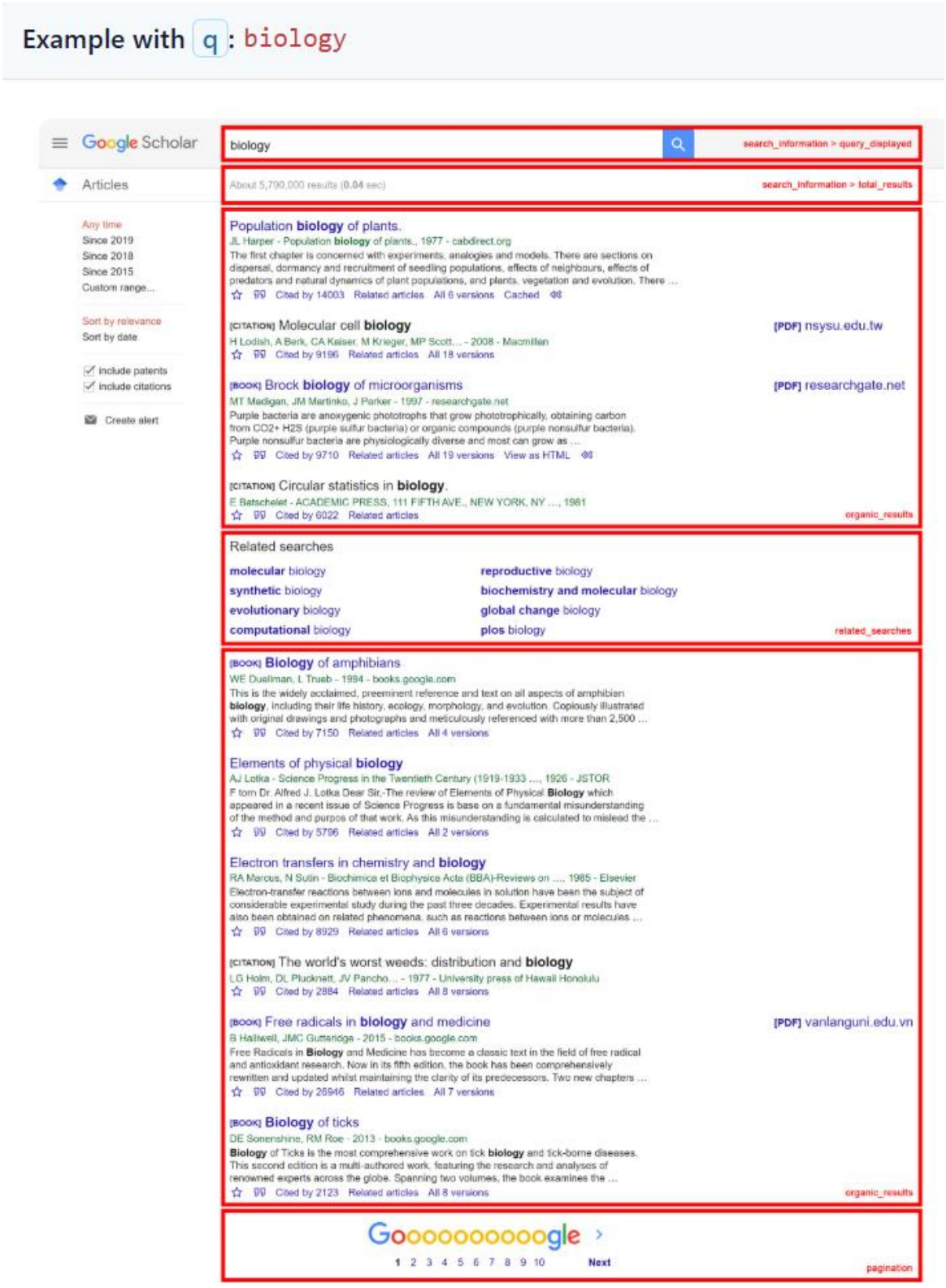
Query parameters:

Here are some parameters that can be used to customize the search in the API:

- **q:** The main search query (example, "Physics"). This is typically a required parameter.
- **cites:** A unique ID for an article to find articles that cite it.
- **as_ylo:** Filters results to include only those published from a specified starting year.
- **as_yhi:** Filters results to include only those published up to a specified ending year.
- **scisbd:** Filters results based on whether to include only abstract results (set to 1) or all results (set to 0).
- **as_vis:** Controls whether to include or exclude citations (e.g., 1 to exclude, 0 to include).
- **safe:** Filters adult content from results (e.g., "active" to enable SafeSearch, "off" to disable).
- **cluster:** A unique ID for an article to find all available versions of it.
- **author_id:** Used with the Google Scholar Author API to retrieve information about a specific author.

Response formats:

The return format is structured the following way:



Usage limits:

The limits of SerpApi are 250 searches for the free plan that is currently being used.

Code examples:

Here is a demonstration of how to use Google Scholar API in a few programming languages:

Code to integrate — Java

```java
Map<String, String> parameter = new HashMap<>();

parameter.put("engine", "google_scholar");
parameter.put("q", "biology");
parameter.put("api_key", "22d39513fa0109a11b2d9d0a09c553203ac80715fee6

GoogleSearch search = new GoogleSearch(parameter);

try {
  JsonObject results = search.getJson();
  var organic_results = results.get("organic_results");
} catch (SerpApiSearchException ex) {
  System.out.println("Exception:");
  System.out.println(ex.toString());
}
```

Code to integrate — JavaScript

```javascript
const { getJson } = require("serpapi");

getJson({
  engine: "google_scholar",
  q: "biology",
  api_key: "22d39513fa0109a11b2d9d0a09c553203ac80715fee63e3f4b6c9a82fe
}, (json) => {
  console.log(json["organic_results"]);
});
```

Code to integrate | Python

```python
from serpapi import GoogleSearch

params = {
  "engine": "google_scholar",
  "q": "biology",
  "api_key": "22d39513fa0109a11b2d9d0a09c553203ac80715fee63e3f4b6c9a82
}

search = GoogleSearch(params)
results = search.get_dict()
organic_results = results["organic_results"]
```

Code to integrate | PHP

```php
require 'path/to/google-search-results.php';
require 'path/to/restclient.php';

$query = [
 "engine" => "google_scholar",
 "q" => "biology",
];

$search = new GoogleSearch('22d39513fa0109a11b2d9d0a09c553203ac80715fe
$result = $search->get_json($query);
$organic_results = $result->organic_results;
```