IN-MEXICO PROGRAM BACKEND DEVELOPER CERTIFICATION

Sprint 1: Server and Database Commands Google Scholar API Technical Report

Luis Enrique Ramírez Sabino
NAO ID: 3317

Monday 29th September, 2025

Endpoints: URLs Used to Access Different API Functions

Primary Endpoint

Base URL: https://serpapi.com/search
Engine Parameter: ?engine=google_scholar
Complete Endpoint Format:

```
https://serpapi.com/search?engine=google_scholar&[parameters]
```

Specific API Functions

Function	Endpoint	Purpose
General Search	/search?engine=google_scholar\&q=[query]	Search for articles, authors, and publications
Author Profile	/search?engine=google_scholar_author	Retrieve specific author information and publications
Citation Search	/search?engine=google_scholar\&cites=[↪ article_id]	Find articles citing a specific publication
Cluster Search	/search?engine=google_scholar\&cluster=[↪ cluster_id]	Find all versions of a specific article
Case Law Search	/search?engine=google_scholar\&as_sdt=4	Search legal documents and court cases

Interactive Demo

Live Testing Environment: https://serpapi.com/playground
Provides real-time API testing capabilities with parameter customization.

Authentication Methods: How to Obtain and Use Access Keys

API Key Generation

1. Account Registration: Create a free SerpApi account at https://serpapi.com
2. API Key Location: Navigate to Dashboard → API Key section
3. Key Format: Unique alphanumeric string (e.g., `abc123def456ghi789`)

Authentication Implementation

Method 1: Query Parameter (Recommended)

```
GET https://serpapi.com/search?engine=google_scholar&q=artificial+intelligence&api_key=
  ↪ YOUR_API_KEY
```

Method 2: Header Authentication

```
GET https://serpapi.com/search?engine=google_scholar&q=artificial+intelligence
Authorization: Bearer YOUR_API_KEY
```

Security Considerations

- Key Protection: Store API keys in environment variables
- Rate Limiting: Keys are subject to usage limits based on subscription plan
- Key Rotation: SerpApi allows key regeneration for security purposes

Query Parameters: Options to Filter and Customize Searches

Essential Search Parameters

Parameter	Type	Description	Example
`q`	Required*	Search query string	`q=machine+learning`
`engine`	Required	API engine selector	`engine=google\_scholar`
`api\_key`	Required	Authentication key	`api\_key=YOUR\_KEY`

Note: `q` becomes optional when using `cites` parameter.

Advanced Search Parameters

Citation and Clustering

Parameter	Purpose	Example	Usage
cites	Find citing articles	`cites ↪ =1275980731835430123`	Bibliometric analysis
cluster	Find article versions	`cluster ↪ =1275980731835430123`	Version comparison

Date Filtering

Parameter	Purpose	Example	Usage
as_ylo	Start year filter	as_ylo=2020	Recent research focus
as_yhi	End year filter	as_yhi=2023	Historical analysis
scisbd	Sort by date	scisbd=1 (abstracts), scisbd=2 (all)	Chronological results

Localization

Parameter	Purpose	Example	Usage
hl	Interface language	hl=en	English interface
lr	Content language	lr=lang_en\\|lang_es	Multilingual search

Pagination

Parameter	Purpose	Example	Usage
start	Result offset	start=10	Second page (10–19)
num	Results per page	num=20	Maximum results

Content Filtering

Parameter	Purpose	Values	Usage
as_sdt	Content type	0 (exclude patents), 7 ↪ (include patents), 4 (case law)	Content filtering
safe	Adult content	active, off	Content safety
filter	Result filtering	1 (enable), 0 (disable)	Similar/omitted results
as_vis	Citation inclusion	1 (exclude), 0 (include)	Citation filtering
as_rr	Review articles	1 (only reviews), 0 (all articles)	Content type focus

SerpApi Specific Parameters

Parameter	Purpose	Values	Usage
no_cache	Force fresh results	true, false	Real-time data
async	Asynchronous processing	true, false	Background processing
output	Response format	json, html	Data format control
zero_trace	Privacy mode	true, false	Enterprise security

Response Formats: How the Returned Data is Structured

JSON Response Structure (Default)

Main Response Schema

```
{
  "search_metadata": {
    "id": "search_id_string",
    "status": "Processing|Success|Error",
    "json_endpoint": "https://serpapi.com/searches/[id].json",
    "created_at": "2025-09-26 12:00:00 UTC",
    "processed_at": "2025-09-26 12:00:01 UTC",
    "google_scholar_url": "https://scholar.google.com/scholar?q=...",
    "raw_html_file": "https://serpapi.com/searches/[id]/show_html",
    "total_time_taken": 1.23
  },
  "search_parameters": {
    "engine": "google_scholar",
    "q": "search query",
    "hl": "en"
  },
  "organic_results": [
  {
    "position": 1,
    "title": "Article Title",
    "result_id": "unique_result_id",
    "link": "https://example.com/article",
    "snippet": "Article description...",
    "publication_info": {
      "summary": "Author Name - Journal, Year",
      "authors": [
      {
        "name": "Author Name",
        "link": "https://scholar.google.com/citations?user=...",
        "serpapi_scholar_link": "https://serpapi.com/search?author_id=...",
        "author_id": "author_unique_id"
      }
      ]
    },
    "resources": [
    {
      "title": "Resource Title",
      "file_format": "PDF",
      "link": "https://example.com/paper.pdf"
    }
    ],
    "inline_links": {
      "serpapi_cite_link": "https://serpapi.com/search?engine=google_scholar_cite&q=...",
      "cited_by": {
        "total": 150,
        "link": "https://scholar.google.com/scholar?cites=...",
        "serpapi_scholar_link": "https://serpapi.com/search?engine=google_scholar&cites=..."
      },
      "versions": {
        "total": 5,
        "link": "https://scholar.google.com/scholar?cluster=...",
        "serpapi_scholar_link": "https://serpapi.com/search?engine=google_scholar&cluster=..."
      }
    }
  }
  ],
  "pagination": {
    "current": 1,
    "next": "https://serpapi.com/search?engine=google_scholar&q=...&start=10",
    "other_pages": {
      "2": "https://serpapi.com/search?engine=google_scholar&q=...&start=10",
      "3": "https://serpapi.com/search?engine=google_scholar&q=...&start=20"
    }
  },
  "serpapi_pagination": {
    "current": 1,
    "next_link": "https://serpapi.com/search?engine=google_scholar&q=...&start=10",
    "next": "https://serpapi.com/search?engine=google_scholar&q=...&start=10"
  }
}
```

Error Response Format

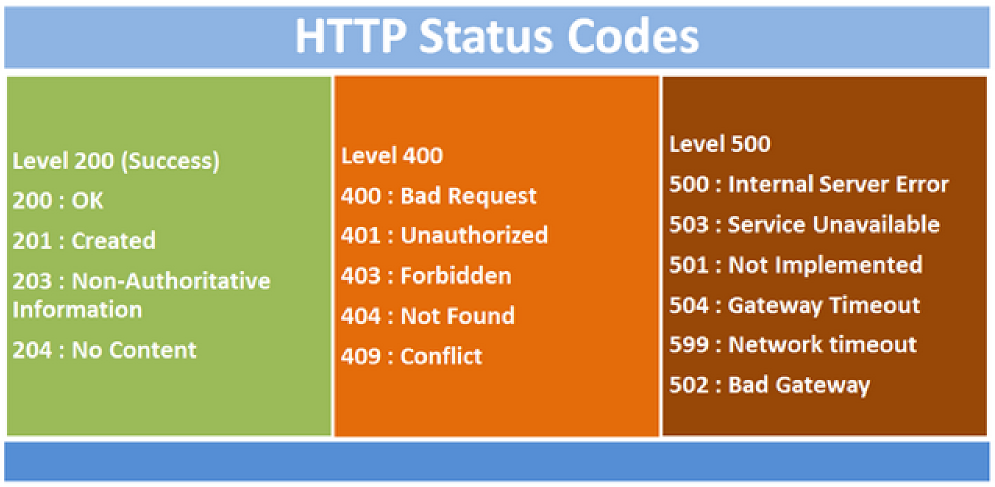
```
{
  "error": "Error message describing the issue",
  "search_metadata": {
    "id": "search_id_string",
    "status": "Error",
    "created_at": "2025-09-26 12:00:00 UTC"
  }
}
```

HTML Response Format

- **Purpose:** Debugging and feature development
- **Content:** Raw HTML from Google Scholar
- **Access:** Set `output=html` parameter
- **Use Cases:** Custom parsing, feature not yet supported by JSON

Response Status Flow

1. **Processing:** Search is being executed
2. **Success:** Results available and complete
3. **Error:** Search failed with error message



HTTP Status Codes

Level 200 (Success)	Level 400	Level 500
200 : OK	400 : Bad Request	500 : Internal Server Error
201 : Created	401 : Unauthorized	503 : Service Unavailable
203 : Non-Authoritative Information	403 : Forbidden	501 : Not Implemented
204 : No Content	404 : Not Found	504 : Gateway Timeout
	409 : Conflict	599 : Network timeout
		502 : Bad Gateway

Figure 1: HTTP Status code.

Usage Limits: Restrictions on the Number of Requests

Free Plan Limitations

- **Monthly Searches:** 100 free searches per month
- **Search Counting:** Only successful searches count toward monthly limit
- **Excluded from Count:** Cached searches, errored searches, failed searches
- **Result Volume Impact:** Number of results per response doesn't affect credit usage

Paid Plan Options

Based on SerpApi pricing structure:

Plan Level	**Monthly Searches**	**Approximate Cost**	**Target Usage**
Free	100	\$0	Development/Testing
Starter	5,000+	\$50+	Small Projects
Professional	15,000+	\$150+	Production Use
Enterprise	100,000+	Custom	Large Scale

Usage Management Features

- Automatic Early Renewal: Available for preventing service interruption
- Manual Renewal: On-demand plan renewal option
- Usage Monitoring: Real-time search counter tracking
- Upgrade/Downgrade: Flexible plan changes without restrictions

Cache System Benefits

- Cache Duration: 1 hour for identical queries
- Free Cache Access: Cached results don't count toward monthly limit
- Performance Improvement: Faster response times for repeated queries
- Cost Optimization: Reduces API consumption for duplicate requests

Rate Limiting Considerations

- Concurrent Requests: Based on subscription plan
- Geographic Distribution: Global server availability
- API Uptime: 99.997% uptime guarantee

Code Examples: Demonstrations of API Usage

Java Implementation

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;

public class GoogleScholarAPI {
  private static final String BASE_URL = "https://serpapi.com/search";
  private static final String API_KEY = "YOUR_API_KEY"; // Store in environment variable

  public static String searchArticles(String query) throws Exception {
    // Encode query parameters
    String encodedQuery = URLEncoder.encode(query, "UTF-8");
    String urlString = BASE_URL + "?engine=google_scholar&q=" + encodedQuery + "&api_key=" +
  ↪ API_KEY;

    // Create HTTP connection
    URL url = new URL(urlString);
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    connection.setRequestMethod("GET");
    connection.setRequestProperty("Accept", "application/json");

    // Read response
    BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));
    StringBuilder response = new StringBuilder();
    String line;
    while ((line = reader.readLine()) != null) {
      response.append(line);
```

```java
        }
        reader.close();

        return response.toString();
    }

    // Example usage
    public static void main(String[] args) {
        try {
            String results = searchArticles("artificial intelligence");
            JsonObject jsonResponse = JsonParser.parseString(results).getAsJsonObject();
            System.out.println("Search Status: " +
            jsonResponse.getAsJsonObject("search_metadata").get("status").getAsString());
        } catch (Exception e) {
            System.err.println("Error: " + e.getMessage());
        }
    }
}
```

Advanced Java with Apache HttpClient

```java
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

public class AdvancedGoogleScholarAPI {
    private CloseableHttpClient httpClient;
    private final String apiKey;

    public AdvancedGoogleScholarAPI(String apiKey) {
        this.apiKey = apiKey;
        this.httpClient = HttpClients.createDefault();
    }

    public String searchWithPagination(String query, int startIndex, int numResults) throws Exception {
        String url = String.format(
        "https://serpapi.com/search?engine=google_scholar&q=%s&start=%d&num=%d&api_key=%s",
        URLEncoder.encode(query, "UTF-8"), startIndex, numResults, apiKey
        );

        HttpGet request = new HttpGet(url);
        request.addHeader("Accept", "application/json");

        return EntityUtils.toString(httpClient.execute(request).getEntity());
    }

    public void close() throws Exception {
        httpClient.close();
    }
}
```

Python Implementation

```python
import requests
import json
import os

class GoogleScholarAPI:
def __init__(self):
self.api_key = os.getenv('SERPAPI_KEY')
self.base_url = 'https://serpapi.com/search'

def search_articles(self, query, start=0, num=10):
params = {
    'engine': 'google_scholar',
    'q': query,
    'start': start,
    'num': num,
    'api_key': self.api_key
}

response = requests.get(self.base_url, params=params)
return response.json()

def search_author(self, author_name):
params = {
    'engine': 'google_scholar',
    'q': f'author:"{author_name}"',
    'api_key': self.api_key
}

response = requests.get(self.base_url, params=params)
return response.json()

# Usage example
api = GoogleScholarAPI()
results = api.search_articles('machine learning', num=20)
print(f"Status: {results['search_metadata']['status']}")
```

JavaScript/Node.js Implementation

```javascript
const https = require('https');

class GoogleScholarAPI {
  constructor(apiKey) {
    this.apiKey = apiKey;
    this.baseUrl = 'https://serpapi.com/search';
  }

  async searchArticles(query, options = {}) {
    const params = new URLSearchParams({
      engine: 'google_scholar',
      q: query,
      api_key: this.apiKey,
      ...options
    });

    const url = `${this.baseUrl}?${params}`;

    return new Promise((resolve, reject) => {
      https.get(url, (response) => {
        let data = ’’;
        response.on(’data’, chunk => data += chunk);
        response.on(’end’, () => {
          try {
            resolve(JSON.parse(data));
          } catch (error) {
            reject(error);
          }
        });
      }).on(’error’, reject);
    });
  }
}

// Usage
const api = new GoogleScholarAPI(’YOUR_API_KEY’);
api.searchArticles(’artificial intelligence’)
.then(results => console.log(results.search_metadata.status))
.catch(console.error);
```

cURL Example

```
# Basic search
curl "https://serpapi.com/search?engine=google_scholar&q=artificial+intelligence&api_key=YOUR_API_KEY"

# Advanced search with filters
curl "https://serpapi.com/search?engine=google_scholar&q=machine+learning&as_ylo=2020&as_yhi=2023&start=10&num=20&api_key=YOUR_API_KEY"

# Author search
curl "https://serpapi.com/search?engine=google_scholar&q=author%3A%22John+Smith%22&api_key=YOUR_API_KEY"
```

Implementation Recommendations for University Project

Project-Specific Considerations

1. **Research Focus:** Target university’s top 3 researchers
2. **Data Requirements:** 2 researchers $\times$ 3 articles = 6 total articles minimum

3. **API Budget:** 100 free searches should accommodate development and testing
4. **Error Handling:** Implement robust exception management for network issues

Optimization Strategies

1. **Cache Utilization:** Leverage 1-hour cache for repeated queries during development
2. **Pagination Management:** Use `start` and `num` parameters for controlled data retrieval
3. **Query Refinement:** Use `author:` helper for targeted researcher searches
4. **Date Filtering:** Apply `as_ylo` and `as_yhi` for recent publication focus

Integration Architecture

1. **MVC Pattern:** Implement clear separation between data models, views, and controllers
2. **Database Mapping:** Design schema based on JSON response structure
3. **Error Recovery:** Implement retry mechanisms for failed requests
4. **Data Validation:** Verify response completeness before database insertion

References

1. SerpApi. (2025). Google Scholar API Documentation. https://serpapi.com/google-scholar-api
2. SerpApi. (2025). API Pricing and Plans. https://serpapi.com/pricing
3. SerpApi. (2025). Official Java wrapper. https://github.com/serpapi/serpapi-java