BOLETIM DATALUTA
BANCO DE DADOS DAS LUTAS POR ESPAÇOS E TERRITÓRIOS

QUALIFICANDO A PESQUISA DO BANCO DE DADOS DATALUTA: AUTOMATIZAÇÃO DE BUSCA ATRAVÉS DE SCRIPT PARA VINCULAÇÃO AUTOMÁTICA DE PDF'S

Maria Eduarda Grecco Bejarano Suenaga[1]
Bernardo Mançano Fernandes[2]

RESUMO

Esta nota conceitual apresenta um *script* para *Google Apps Script* desenvolvido para automatizar a vinculação de arquivos PDF a registros em planilhas, aplicado ao banco de dados DATALUTA. A ferramenta resolve uma inconveniência metodológica: a busca manual de documentos associados à base de dados. Partindo de uma abordagem prática e acessível, o código percorre subpastas no *Google Drive*, identifica PDFs por seus códigos únicos e insere os *links* diretamente na planilha correspondente. Detalhamos os requisitos necessários para aplicá-lo e suas etapas de implementação, mostrando como a automação reduz o tempo gasto em processamento manual. O objetivo é permitir que o tempo economizado seja empregado em análises mais profundas e na produção de novos textos e mapas, tornando essa etapa mais ágil para quem organiza os dados. Destacamos que esta é uma programação feita por geógrafas(os) para geógrafas(os) e demais cientistas e militantes que utilizam o DATALUTA. Por isso, apresentamos o *script* pensando a metodologia geográfica e facilitando a implementação para quem não tem familiaridade com linguagens de programação.

Palavras-chave: Banco de dados DATALUTA. *Google Apps Script*. Automação de Planilha. Pesquisa Quali-quantitativa. Geografia Digital.

QUALIFYING THE RESEARCH OF THE DATALUTA DATABASE: SEARCH AUTOMATION THROUGH SCRIPT FOR AUTOMATIC PDF LINKING

ABSTRACT

This conceptual note presents a *script* for *Google Apps Script* developed to automate the linking of PDF files to records in *spreadsheets*, applied to the DATALUTA database. The tool addresses a methodological inconvenience: the manual search for documents associated with the database. Taking a practical and

[1] Mestranda em Geografia pela Universidade Estadual Paulista (UNESP). E-mail: grecco.unesp@gmail.com.
[2] Professor Titular (2024) e Professor Livre-Docente da Universidade Estadual Paulista (UNESP). E-mail: mancano.fernandes@unesp.br.

accessible approach, the code navigates through subfolders in *Google Drive*, identifies PDFs by their unique codes, and inserts the *links* directly into the corresponding *spreadsheet*. We detail the necessary requirements for its application and the implementation steps, demonstrating how automation reduces the time spent on manual processing. The goal is to enable the time saved to be used for deeper analyses and the production of new texts and maps, making this stage more efficient for those handling the data. We emphasize that this is a program developed by geographers for geographers, as well as other scientists and activists who use DATALUTA. Therefore, we present the *script* with geographic methodology in mind, facilitating its implementation for those unfamiliar with programming languages.

Keywords: DATALUTA database. *Google Apps Script*. *Spreadsheet* Automation. Quali-quantitative Research. Digital Geography.

CALIFICANDO LA INVESTIGACIÓN DE LA BASE DE DATOS DATALUTA: AUTOMATIZACIÓN DE BÚSQUEDA A TRAVÉS DE SCRIPT PARA LA VINCULACIÓN AUTOMÁTICA DE PDF'S

RESUMEN

Esta nota conceptual presenta un *script* para *Google Apps Script* desarrollado para automatizar la vinculación de archivos PDF a registros en hojas de cálculo, aplicado a la base de datos DATALUTA. La herramienta resuelve un inconveniente metodológico: la búsqueda manual de documentos asociados a la base de datos. A partir de un enfoque práctico y accesible, el código recorre subcarpetas en *Google Drive*, identifica los archivos PDF por sus códigos únicos e inserta los enlaces directamente en la hoja de cálculo correspondiente. Detallamos los requisitos necesarios para su aplicación y las etapas de implementación, mostrando cómo la automatización reduce el tiempo dedicado al procesamiento manual. El objetivo es permitir que el tiempo ahorrado se emplee en análisis más profundos y en la producción de nuevos textos y mapas, agilizando esta etapa para quienes manejan los datos. Destacamos que esta es una programación realizada por geógrafas(os) para geógrafas(os), así como para otros científicos y militantes que utilizan DATALUTA. Por ello, presentamos el *script* considerando la metodología geográfica y facilitando su implementación para quienes no están familiarizados con los lenguajes de programación.

Palabras clave: Base de datos DATALUTA. *Google Apps Script*. Automatización de Hojas de Cálculo. Investigación Cuali-cuantitativa. Geografía Digital.

INTRODUÇÃO

Sair do recorte de jornal e passar a exportar em PDF, através do comando de impressão, os jornais, blogs e sites, ou seja, páginas da internet, acompanha o movimento da realidade e das tecnologias, do avanço técnico-científico-informacional e das novas formas de imprensa, em que o modelo tradicional (impresso, televisivo e radiofônico) migrou para o espaço digital. Imagine se ainda dependêssemos do recorte de jornais para importar os dados, com certeza não teríamos a

agilidade e escala de pesquisa que temos hoje. Propomos aqui uma ferramenta de auxílio para os trabalhos coletivos que produzem o Banco de Dados das Lutas por Espaços e Territórios (DATALUTA).

Estamos avançando na construção de nosso método, nossos procedimentos metodológicos para processar os dados tabulados, assim como fizemos quando saímos das páginas de jornais, passamos pelo Excel e agora utilizamos o JotForms. Observamos a qualificação técnica em processos cada vez mais avançados de organizar dados. Somamos em coro, e também em proposta, àquilo que já foi muito propagado por colegas no XVII e XVIII Encontros da Rede Brasileira de Pesquisa das Lutas por Espaços e Territórios, ambos na Escola Nacional Florestan Fernandes, Guararema - SP, quando se dizia algo como: "devemos dedicar nosso tempo analisando os dados e interpretando o que eles nos evidenciam da realidade. Ficar processando tabelas despende um tempo que pode ser empregado em outros trabalhos."

Há a possibilidade da sistematização dos dados em *Not Only SQL* (NoSQL) e processamento com a linguagem *Generative Pre-trained Transformer* (GPT) através da recente parceria com outros pesquisadores da área de Ciências da Computação, o que poderá mudar substancialmente a representação dos dados. A expectativa por uma solução futura melhor não deve impedir a adoção de medidas imediatas. Embora as alternativas disponíveis no presente possam não ser tão surpreendentes quanto a solução projetada, elas são fundamentais para garantir avanços graduais que permitam um uso otimizado. Devemos ser agentes ativos de um processo que é tão fundamental para nós.

Essa postura ativa reflete a própria natureza da Rede Brasileira de Pesquisa das Lutas por Espaços e Territórios (REDE DATALUTA), um coletivo nacional com vínculos internacionais, associada ao Conselho Latino-americano de Ciências Sociais (CLACSO) e integrante do *Latin American Geographies - United Kingdom* (LAG-UK) que desde 2004 dedica-se ao registro das ações dos movimentos socioterritoriais a partir de dados de ocupação e manifestação no campo. Em 2020, essa atuação expandiu-se para o registro de ações e conflitualidades de movimentos socioespaciais e outras instituições em contextos diversos: nos espaços agrário e urbano, nas florestas e nas águas, além do monitoramento da estrangeirização de terras, dados da questão agrária e das Jornadas Universitárias em Defesa da Reforma Agrária (JURA). É precisamente essa diversidade de frentes e o volume de dados acumulados que sustentam a potência analítica da REDE, materializando-se em pesquisas que convertem o registro técnico em análise territorial e/ou espacial e resistência política.

Boletim DATALUTA | Brasília/DF, Brasil | v. 19, n. 196 | p. 1-13 | mar. 2026 | ISSN 2177-4463
https://periodicos.unb.br/index.php/BD

Observamos isso, por exemplo, na publicação *Tempo e espaço na leitura das ações dos movimentos socioterritoriais e socioespaciais*, em que Almeida, Santos e Baratelli (2023) demonstram que a indissociabilidade entre o município e a data da ação permite ler a coexistência de diferentes temporalidades e conflitos de modelos produtivos no mesmo espaço. Essa prática é detalhada por Lima et al. (2025) no trabalho *Critérios de seleção das notícias para a base de dados do DATALUTA*, evidenciando como o monitoramento de algoritmos e notícias gera dados sólidos para diagnósticos territoriais em contextos de crise, conforme observado nas análises de Cavalcante e Moura (2022) sobre o Nordeste brasileiro e de Buscioli (2023) sobre a diversidade de resistências na Argentina.

Essa aplicabilidade alcança a defesa direta de territórios e a retificação de narrativas oficiais. Na *Nota Técnica do DATALUTA Floresta contra o marco temporal*, Costa, Sobreiro Filho e Lopes (2023) utilizam a sistematização de centenas de ações para comprovar o protagonismo indígena e combater a invisibilização parlamentar, dinâmica reforçada por Costa (2022) em *Movimentos socioterritoriais indígenas das florestas e processos de resistência*. A capacidade de projetar esses dados em escala global é evidenciada por Jorge et al. (2022) na nota conceitual *Aplicabilidade dos ODS na pesquisa*, que prova cientificamente serem os movimentos sociais os reais agentes cumpridores da Agenda 2030. Assim, a base qualifica ações de autonomia, como discutido por Jorge et al. (2024) em *Ações dos movimentos socioterritoriais e a produção de territorialidades emancipadoras no Brasil*, visibilizando práticas agroecológicas que reproduzem a vida para além do capital.

Embora a densidade dessas análises evidencie a maturidade da Rede, a sustentação desse volume produtivo impõe desafios operacionais constantes. A agilidade metodológica, mencionada anteriormente, esbarra muitas vezes na etapa de manuseio e recuperação documental, processo que, se não automatizado, consome o tempo que poderia ser destinado à reflexão crítica. Aqui, buscamos apresentar uma ferramenta metodológica, um processo, que ajudará, dentro da metodologia do DATALUTA, a organizar os diversos dados de acordo com as ações e demandas dos movimentos e suas relações com os Objetivos do Desenvolvimento Sustentável. Certamente, alguns colegas que já realizaram alguma pesquisa coletiva ou utilizaram os dados para fins acadêmicos ou para divulgação nas mídias populares, com um recorte que fez necessária a filtragem dos dados, ou seja, separar dados específicos da totalidade, já tiveram que pesquisar PDF por PDF e entendem o quão trabalhoso e contraproducente isso é.

Analisar cada PDF que está sendo estudado é atividade necessária, todavia demanda muito tempo quando não se operacionaliza a busca pelos PDFs, isso significa que se for necessário reanalisar este mesmo PDF, é preciso refazer todo o caminho da busca novamente. O que apresentamos neste trabalho é, concretamente, um script de programação que gera o link de

armazenamento da pasta, relacionando-o com o código do PDF, fazendo com que um simples link já redirecione para o PDF. Mas para isso, são necessários alguns cuidados, que também serão apresentados aqui.

COMPREENSÕES FUNDAMENTAIS

O *Google Apps Script* é uma plataforma de desenvolvimento baseada em nuvem, impulsionada pelo Google Drive, que utiliza a linguagem *JavaScript* para integrar e automatizar tarefas no ecossistema do *Google Workspace*. A execução das soluções programáticas ocorre diretamente nos servidores da Google, o que elimina a necessidade de configurar ambientes de desenvolvimento locais e otimiza significativamente os recursos de infraestrutura (Google, s. d.). No escopo deste projeto, contudo, a execução exige o emprego do *Google Sheets* exclusivamente em formato nativo, garantindo a plena compatibilidade com os métodos internos de manipulação de dados da plataforma.

Em termos práticos, essa ferramenta funciona como um assistente digital que reside dentro do próprio navegador. Ao invés de precisarmos instalar programas complexos ou possuir um computador de alto desempenho, ele utiliza a própria estrutura da Google para rodar o que foi estipulado. Essa integração direta significa que o código "conversa" nativamente com as planilhas e pastas, desde que os arquivos tenham sido criados dentro do sistema Google, garantindo que a comunicação entre os dados e a automação seja imediata e sem erros de leitura.

No que diz respeito à mecânica de controle, as automações são colocadas em movimento por meio de uma arquitetura orientada a eventos. Segundo o Google (s. d.), os scripts podem ser acionados de maneira flexível: via menus personalizados, botões, ações diretas do usuário ou gatilhos (*triggers*) baseados em cronogramas de tempo. Essa versatilidade permite que o processamento sistemático de dados em planilhas e a gestão de arquivos na nuvem ocorram de forma ininterrupta e escalável.

Na rotina de pesquisa, isso significa que o usuário tem total controle sobre quando e como o trabalho será feito. A ferramenta não exige que se digitem códigos complexos toda vez que for utilizada; ela pode ser ativada por um simples clique em um botão criado na barra de tarefas ou programada para trabalhar sozinha em horários específicos. Essa flexibilidade transforma tarefas manuais repetitivas em processos automáticos que funcionam em segundo plano, permitindo que a equipe se dedique exclusivamente à análise dos resultados.

Para estruturar essa lógica, a plataforma adota o *JavaScript* que, segundo Sebesta (2022), consolidou-se como uma linguagem imperativa de alta flexibilidade. O *Guia JavaScript* da *Mozilla Developer Network* (s. d.) reforça que a linguagem funciona como o pilar da programação em nuvem, suportando estruturas de dados complexas e iterações. Sob a ótica da ciência da computação, um script é um conjunto de instruções e dados que residem na memória e são executados pelo processador para realizar tarefas específicas (Sebesta, 2022). O paradigma de orientação a objetos da linguagem (Mozilla, 2025) organiza o código em entidades com propriedades e métodos específicos, permitindo que o algoritmo interaja com "objetos" que representam a interface da planilha ou o diretório de arquivos, substituindo a busca manual por algoritmos de identificação de URLs (*Uniform Resource Locators*).

Traduzindo para a realidade da organização de dados, a linguagem trata cada elemento do trabalho como uma peça individual com funções próprias. A planilha é vista como um objeto que "sabe" ler informações, enquanto a pasta de arquivos é um objeto que "sabe" organizar documentos. O script, portanto, atua como um coordenador que dá ordens a essas peças, pedindo que a planilha forneça um nome de arquivo e que a pasta localize o endereço digital (o link) correspondente a esse nome, eliminando a necessidade de o pesquisador abrir e fechar pastas manualmente.

A operacionalização desse fluxo utiliza uma hierarquia de serviços, objetos e métodos. Um "Serviço" (como o *SpreadsheetApp* ou o *DriveApp*) atua como o objeto global que fornece o ponto de entrada para a API do aplicativo correspondente. Um "Método" é uma função específica contida nesses objetos, responsável por executar ações exatas. O fluxo de trabalho utiliza o serviço *SpreadsheetApp* para isolar intervalos de células via método *getRange()*, enquanto o *getValue()* extrai o dado alfanumérico que servirá de parâmetro para que o *DriveApp* localize o arquivo correspondente no repositório digital através de seu Identificador Único (ID).

Para o usuário, essa hierarquia funciona como uma linha de comando em uma biblioteca. O "Serviço" é a própria biblioteca; o "Objeto" é a estante específica; e o "Método" é o ato de retirar o livro da prateleira. Quando o script é executado, ele vai até a "estante" das planilhas, identifica a posição exata da informação (como uma coordenada de linha e coluna) e "lê" o conteúdo ali presente. Com essa informação em mãos, ele se dirige à "estante" de arquivos no Drive para encontrar o documento exato, utilizando um código de identificação que funciona como um RG único para cada PDF.

A navegação em estruturas de diretórios complexas exige o uso de iteradores, mecanismos projetados para percorrer coleções de dados de forma ordenada (Sebesta, 2016). Como o Google

Drive organiza dados em estruturas de árvore (nós e folhas), o script utiliza os métodos *hasNext()* e *next()* para processar os arquivos. O método *hasNext()* realiza um teste lógico para verificar a existência de itens subsequentes, enquanto o *next()* recupera o objeto e avança a posição do iterador. Durante a execução, o objeto *Logger*, por meio do método *log()*, atua como a interface de depuração (*debugging*), registrando o histórico de eventos e permitindo o monitoramento do fluxo de controle e do estado das variáveis em tempo real (Google, s. d.).

Essa etapa final é o que garante a segurança e a precisão do processo. Os "iteradores" agem como um vistoriador que percorre cada corredor de um arquivo físico, conferindo pasta por pasta até encontrar o que procura, sem pular nenhuma etapa. Já o *Logger* funciona como um diário de bordo automático: ele anota cada sucesso ou dificuldade encontrada pelo script. Se um arquivo não for localizado, o sistema registra exatamente qual código apresentou problema, permitindo que saibamos exatamente onde o banco de dados precisa de correção, sem precisar revisar todo o trabalho novamente.

Essa integração sistêmica assegura que a correspondência entre o código da planilha e o nome do arquivo no Drive seja validada logicamente antes da atribuição do link final, garantindo a integridade do banco de dados. Para que essa precisão seja alcançada, o script utiliza métodos de manipulação de *strings* (cadeias de caracteres), como o *trim()* para remover espaços em branco e o *replace()* para desconsiderar extensões de arquivo, assegurando que o identificador alfanumérico lido seja identicamente comparado ao nome do documento armazenado.

Ou seja, esse mecanismo funciona como um "filtro de segurança" que evita erros humanos de digitação ou de nomenclatura. Antes de o sistema escrever qualquer informação na planilha, ele faz uma conferência: ele "limpa" o nome do arquivo, retira sobras de texto desnecessárias e só valida o link se houver uma combinação perfeita entre o que está na tabela e o que está na pasta. Isso garante que a informação seja confiável, impedindo que um código aponte para o arquivo errado.

Devido à sintaxe do script e aos protocolos de segurança da plataforma, os seguintes procedimentos são indispensáveis para a operacionalização da ferramenta:

1. **Gerenciamento de Permissões e Integridade:** O acesso via *Apps Script* exige a concessão de permissões de leitura e escrita no Google Drive. Para garantir a integridade da base de dados original da REDE, o processamento deve ser executado obrigatoriamente em uma cópia da pasta de diretórios, isolando o ambiente de execução para evitar colisões de dados ou exclusões acidentais por falhas lógicas.

   a. *Procedimento Prático:* A pasta que contém as notícias não pode ser a pasta original do projeto onde a equipe armazena os dados em conjunto. É preciso que uma cópia da pasta seja feita no seu próprio Drive. Isso é fundamental porque o script solicita permissões que poderiam, tecnicamente, excluir arquivos caso houvesse um código para isso. Por segurança, trabalhe sempre com uma cópia.

2. **Estrutura de Varredura por Iteração:** A arquitetura do algoritmo utiliza métodos de busca, projetados para percorrer coleções de dados organizadas de forma hierárquica (em árvore). O sistema percorre sistematicamente os nós das subpastas para indexar os arquivos antes de realizar a correspondência com a planilha.
   a. *Procedimento Prático:* O armazenamento dos PDFs deve estar obrigatoriamente em subpastas. Como este já é o padrão do banco de dados do DATALUTA (arquivamento por meses do ano), a estrutura atual já satisfaz essa exigência. O script foi programado para "mergulhar" nessas divisões para encontrar as notícias.
3. **Padronização dos Vetores de Entrada e Tratamento de Strings:** O algoritmo define a Coluna A como o vetor de entrada (*getRange*) para os parâmetros de busca. O método de comparação exige uma identidade exata entre o nome do arquivo e o conteúdo da célula, utilizando funções de limpeza para remover extensões e espaços residuais.
   a. *Procedimento Prático:* A coluna de códigos do PDF deve estar na Coluna A e os links serão inseridos automaticamente na Coluna B. Se os códigos estiverem em qualquer outra coluna, ocorrerá um erro. Além disso, os códigos na Coluna A não podem conter o termo ".pdf"; caso existam nomes com essa extensão, eles devem ser retirados antes de rodar o script.

COMO USAR O SCRIPT: UM BREVE MANUAL

Queremos apresentar este breve manual, de forma que essa nota conceitual possa auxiliar no passo a passo para a implementação do script. Esperamos que ao final desses passos, quem esteja executando consiga aplicar a funcionalidade proposta.

1. Se você precisa encontrar PDFs específicos no banco de dados, e não checar todos os PDFs, primeiro é necessário selecionar quais dados serão usados, isso é, separar os dados que serão usados na planilha bruta. Para isso, através da filtragem no Excel ou *Google Sheets*, a pesquisadora ou pesquisador deve, seja por uma tipologia de ação, uma escala, um nome de movimento, um ODS ou qualquer que seja a variável (ou combinação de

Boletim DATALUTA| Brasília/DF, Brasil | v. 19, n. 196 | p. 1-13| mar. 2026 | ISSN 2177-4463
https://periodicos.unb.br/index.php/BD

variáveis), separar apenas os dados de interesse. Assim, você terá os dados e seus respectivos PDFs. Salve essa planilha no Excel ou *Google Sheets*.

2. O próximo passo é criar uma planilha no *Google Sheets* a partir da sua conta Google. Nessa planilha nova, você pode optar por copiar apenas os códigos específicos dos PDFs que selecionou ou transferir todos os dados (com outras colunas que não sejam a do código) de uma vez. É essencial que os códigos fiquem na coluna A, enquanto a coluna B deve permanecer vazia para receber os links posteriormente. Reforçamos que não funcionará importar uma planilha pronta (como um arquivo .xlsx enviado do computador); ela precisa ser criada nativamente no *Google Sheets* para que o serviço *SpreadsheetApp* consiga ler e escrever nas células corretamente.
3. Depois de preparar a planilha, vá até o menu "Extensões" e selecione "*Apps Script*". Isso abrirá o editor de scripts em uma nova janela. Você verá um arquivo chamado Código.gs com um texto padrão. Apague tudo o que estiver lá e cole o script que apresentamos no final deste texto. Esse ambiente é onde o "cérebro" da automação vai morar, permitindo que o código interaja com seus arquivos.
4. Você precisará copiar para o seu Google Drive pessoal a pasta que contém as subpastas com os arquivos necessários. Por exemplo, se estiver trabalhando com dados de 2020 a 2023, faça o download dessas pastas e as coloque diretamente em "Meu Drive", sem qualquer subpasta intermediária. Isso é fundamental para a integridade dos dados, uma vez que o *Apps Script* solicita permissões que podem, inclusive, excluir arquivos caso um código seja feito para isso. Depois de ter esses arquivos no seu Drive, abra a pasta principal e observe a URL no navegador: o *ID* da pasta é a sequência de caracteres que aparece após "*folders*/" no link. Copie esse *ID* e, no script que você colou, substitua o texto "COLOCAR AQUI ID DA PASTA" por esse código, mantendo as aspas. Esse ID é o parâmetro que o método *getFolderById()* utiliza para localizar o diretório exato na nuvem.
5. Com tudo configurado, clique no botão de salvar (representado por um disquete) e depois em "Executar" (ícone de play) no editor do *Apps Script*. O Google solicitará permissões para acessar seu Drive e *Sheets*, conceda todas. O script processará os dados, buscando os arquivos nas subpastas e inserindo os links correspondentes na coluna B da sua planilha. Caso algum código não for encontrado, o script registrará um *log* para que você possa verificar manualmente.

6. Se o script não funcionar na primeira tentativa, revise os erros mais frequentes: códigos com a extensão ".pdf" na sua planilha (embora o script tente tratar isso, confira se há inconsistências), a ordem das colunas incorreta (mantenha sempre os códigos na Coluna A) ou arquivos não encontrados (verifique se o código na planilha corresponde exatamente ao nome do arquivo no Drive). Se o script não funcionar na primeira tentativa, revise essas possibilidades e execute novamente.

METODOLOGIA

Estes procedimentos metodológicos, que apresentamos a seguir, representam um novo espaço dentro das experiências da REDE DATALUTA, de modo que procuramos ser didáticos, embora estejamos trabalhando com uma linguagem pouco conhecida, mas que tem relação direta com todas as atividades que realizamos em nossas pesquisas.

Conhecendo a possibilidade de integração entre o *Apps Script* e os serviços Google, recorremos ao *JavaScript Guide* (Mozilla, s.d.), como fonte para o script, e utilizamos o documento oficial do *Google Apps Script* no Workspace (Google, s.d), que contém todas as informações necessárias sobre a plataforma, elaborada pela própria desenvolvedora.

Em programação, uma função é um bloco de código que executa uma tarefa específica. No *Google Apps Script*, as funções são definidas com um nome e contêm as instruções a serem executadas. Assim, quando criamos uma função chamada "preencherCelula" para preencher uma célula de uma planilha com um valor específico (em nosso caso o link do PDF), podemos utilizá-la em diversas partes do código, facilitando a organização e evitando a repetição de blocos de código que já foram feitos.

Para o *Apps Script*, o que é nomeado como objetos são representações de serviços do Google, como o "*SpreadsheetApp*", que permite acessar e modificar planilhas do *Google Sheets*, e também o "*DriveApp*", que oferece recursos para interagir com arquivos e pastas no *Google Drive*. Cada objeto possui diversos métodos que permitem realizar ações sobre os dados ou funcionalidades desses serviços. Sendo assim, o método "*getRange()*" do "*SpreadsheetApp*" é utilizado para obter um intervalo de células, enquanto o método "*getValue()*" pode ser usado para obter o valor de uma célula específica.

Os loops, como "for" e "while", permitem percorrer coleções de dados, como linhas de uma planilha ou arquivos dentro de uma pasta (em nosso caso, o armazenamento em meses do ano), realizando operações automatizadas de forma frequente. Outro componente são as condicionais,

como o "*if*", que são usadas para tomar decisões dentro do código com base em condições específicas, neste caso, verificar se o arquivo foi encontrado na pasta e, caso afirmativo, realizar a ação de inserir o link na planilha.

Para a depuração dos dados, recorremos ao *Logger*, uma ferramenta necessária para entender o que está acontecendo durante a execução do script, especialmente em casos de erro ou comportamento inesperado. O "*Logger.log()*" permite registrar mensagens e variáveis no console de execução, facilitando a identificação de problemas e proporcionando uma visão mais clara do fluxo do script. Isso possibilita que erros e inconsistências sejam encontrados de forma facilitada.

Em resumo, nosso código para o *Google Apps Script* combina conceitos de programação, como funções, objetos, loops e condicionais, com a capacidade de interagir diretamente com os serviços do Google, neste caso o *Google Sheets*. Como mencionamos, o exposto na metodologia, almeja apresentar a compreensão desses componentes, que são fundamentais para o desenvolvimento de scripts eficientes e automatizados.

O SCRIPT

Nesta parte da nota estamos apresentando o script que deve ser copiado e colado, ou escrito, no *Google Apps Script*. Esperamos que, com o que foi apresentado até então, esta sintaxe não pareça indecifrável. Para facilitar a compreensão, tudo que está depois das duas barras (//), na cor verde, é uma descrição do processo executado pelo script.

```
function  buscarLinksEmSubpastas() {
var  pastaPrincipalId = "COLOCAR AQUI ID DA PASTA"; // ID da pasta principal: Identificador único (ID) para o serviço DriveApp
var  pastaPrincipal = DriveApp.getFolderById(pastaPrincipalId); // Objeto que localiza a pasta raiz no Google Drive
var  subpastas = pastaPrincipal.getFolders(); // Obtém todas as subpastas: Utiliza um iterador para percorrer os diretórios
var  planilha = SpreadsheetApp.getActiveSpreadsheet(); // Acessa o serviço de planilhas nativo do Google
var  aba = planilha.getActiveSheet(); // Define a aba ativa para a manipulação dos dados
var  dados = aba.getDataRange().getValues(); // Coleta os valores das células (Matriz de dados) para processamento
var  mapaArquivos = {}; // Objeto de mapeamento para associar nomes de arquivos aos seus URLs

// Log para verificar se a pasta principal foi carregada: Interface de depuração (Logger)
Logger.log("Pasta principal carregada: " + pastaPrincipal.getName());

// Percorre todas as subpastas dentro da pasta principal: Estrutura de repetição (Loop While)
while (subpastas.hasNext()) {
var  subpasta = subpastas.next();
Logger.log("Subpasta encontrada: " + subpasta.getName()); // Log da subpasta para monitoramento do fluxo
```


Boletim DATALUTA | Brasília/DF, Brasil | v. 19, n. 196 | p. 1-13 | mar. 2026 | ISSN 2177-4463
https://periodicos.unb.br/index.php/BD

```
var  arquivos = subpasta.getFiles();

// Percorre todos os arquivos dentro da subpasta: Iteração interna para coleta de arquivos
while (arquivos.hasNext()) {
var  arquivo = arquivos.next();
mapaArquivos[arquivo.getName().replace('.pdf', '')] = arquivo.getUrl(); // Remove a extensão .pdf
para comparação lógica
Logger.log("Arquivo encontrado: " + arquivo.getName() + " - Link: " + arquivo.getUrl()); // Log do
arquivo processado
}
}

// Percorre a planilha e insere os links: Estrutura de controle (Loop For) que cruza os dados
for (var  i = 1; i < dados.length; i++) {
var  codigo = dados[i][0].toString().trim(); // Garantir que o código seja tratado como string e sem
espaços extras
if (mapaArquivos[codigo]) { // Condicional que valida a existência do arquivo antes da atribuição
aba.getRange(i + 1, 2).setValue(mapaArquivos[codigo]); // Método que insere o link na coluna B
(Vetor de saída)
Logger.log("Link inserido para o código: " + codigo + " - Link: " + mapaArquivos[codigo]); // Log da
inserção do link
} else {
Logger.log("Código não encontrado: " + codigo); /// Log de erro para identificação de
inconsistências no banco de dados
}
}
}
```

CONSIDERAÇÕES FINAIS

Esta nota conceitual nasce de uma necessidade prática vivenciada no cotidiano da pesquisa: a urgência em otimizar etapas excessivamente operacionais que, embora fundamentais, consomem um tempo precioso que deveria ser dedicado à análise de conteúdo. Nosso trabalho envolve a operacionalização da pesquisa, mas não deve se restringir a tarefas repetitivas, como o esforço exaustivo de copiar e colar códigos entre planilhas e diretórios para localizar arquivos individualmente. Ao automatizar essa busca, o tempo que seria gasto em processos manuais é redirecionado para a análise de fato dos PDFs, portanto às ações.

A solução encontrada através do script é uma alternativa possível para simplificar a rotina acadêmica sem perder o rigor. Embora utilize a linguagem *JavaScript*, trata-se de uma sintaxe simples e acessível, que não exige um domínio profundo de programação para ser implementada. Mais do que apresentar um código, esta nota busca encorajar todos a se apropriarem dessas tecnologias, seja pedindo auxílio a colegas de outras áreas ou utilizando ferramentas de Inteligência Artificial para adaptar e criar seus próprios scripts.

Concluímos que a integração entre a Geografia e a Programação em nuvem não é apenas um suporte técnico, mas uma estratégia metodológica que potencializa o trabalho coletivo. Almejamos que o iniciado aqui, seja o rizoma de novas atitudes e propostas que estreitarão as

fronteiras entre o que executamos no DATALUTA e as técnicas computacionais de organização de dados. Queremos tornar nossa pesquisa mais facilitada, permitindo que o nosso tempo seja empregado na qualificação da análise, na elaboração e proposição de políticas e ações para os movimentos, espaços e territórios.

REFERÊNCIAS

ALMEIDA, F. F. de; SANTOS, L. L. M.; BARATELLI, A. E. S. **Tempo e espaço na leitura das ações dos movimentos socioterritoriais e socioespaciais**. Boletim DATALUTA, Presidente Prudente, n. 181, p. 1-10, jan. 2023.

BUSCIOLI, L. D. **Las acciones de los movimientos socioterritoriales agrarios en la provincia de Santiago del Estero en Argentina**. Boletim DATALUTA, Presidente Prudente, n. 185, maio 2023.

CAVALCANTE, L. V.; MOURA, J. T. V. de. **Ações dos movimentos socioterritoriais do Nordeste em tempos de pandemia, desmonte de políticas públicas e avanço do capital no campo**. Boletim DATALUTA, Presidente Prudente, n. 174, jun. 2022.

COSTA, B. G. **Movimentos socioterritoriais indígenas das florestas e processos de resistência em tempos de pós fascismo**. Boletim DATALUTA, Presidente Prudente, n. 175, jul. 2022.

COSTA, B. G.; SOBREIRO FILHO, J.; LOPES, M. L. A. **Nota técnica do DATALUTA Floresta contra o marco temporal e em defesa das lutas e resistências dos povos indígenas**. Boletim DATALUTA, Presidente Prudente, n. 186, jun. 2023.

GOOGLE. **Google Apps Script documentation**. Google Workspace. Disponível em: https://developers.google.com/apps-script. Acesso em: 13 mar. 2026.

JORGE, A. A. et al. **Nota conceitual: aplicabilidade dos ODS na pesquisa: movimentos socioterritoriais em perspectiva comparada**. Boletim DATALUTA, Presidente Prudente, n. 169, jan. 2022.

JORGE, A. A.; ORIGUELA, C. F.; LIMA, B. B.; MOURA, J. T. V. de. **Ações dos movimentos socioterritoriais e a produção de territorialidades emancipadoras no Brasil: uma análise desde o banco de dados DATALUTA**. Boletim DATALUTA, Brasília/DF, [s.d.].

LIMA, W. F. P. et al. **Critérios de seleção das notícias para a base de dados do DATALUTA movimentos socioespaciais e socioterritoriais agrários**. Boletim DATALUTA, Brasília/DF, v. 18, n. 190, p. 1-9, jun. 2025.

MOZILLA. **JavaScript Guide**. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide. Acesso em: 13 mar. 2026.

SEBESTA, Robert W. **Concepts of programming languages**. 12. ed. Boston: Pearson, 2022.