UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Marcos David dos Santos Lopes

GERAÇÃO DE CÓDIGO EM GOOGLE APPS SCRIPT PARA DESENVOLVIMENTO RÁPIDO DE APIS WEB A PARTIR DE DADOS EM NUVEM

Santa Maria, RS
2023

Marcos David dos Santos Lopes

GERAÇÃO DE CÓDIGO EM GOOGLE APPS SCRIPT PARA DESENVOLVIMENTO RÁPIDO DE APIS WEB A PARTIR DE DADOS EM NUVEM

Trabalho Final de Graduação apresentado ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Sistemas de Informação**.

ORIENTADORA: Profa. Andrea Schwertner Charão

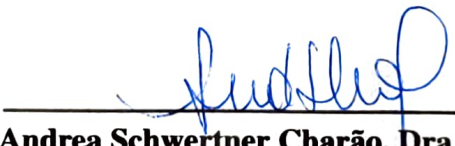
Santa Maria, RS
2023

Marcos David dos Santos Lopes

GERAÇÃO DE CÓDIGO EM GOOGLE APPS SCRIPT PARA DESENVOLVIMENTO RÁPIDO DE APIS WEB A PARTIR DE DADOS EM NUVEM

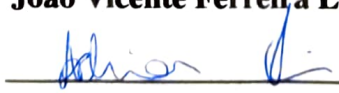
Trabalho Final de Graduação apresentado ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Sistemas de Informação**.

Aprovado em 18 de julho de 2023:



Andrea Schwertner Charão, Dra. (UFSM)
(Presidente/Orientadora)

João Vicente Ferreira Lima, Dr. (UFSM)



Adriano Pereira, Dr. (UFSM)

Santa Maria, RS
2023

GERAÇÃO DE CÓDIGO EM GOOGLE APPS SCRIPT PARA DESENVOLVIMENTO RÁPIDO DE APIS WEB A PARTIR DE DADOS EM NUVEM

CODE GENERATION IN GOOGLE APPS SCRIPT FOR RAPID DEVELOPMENT OF WEB APIS FROM CLOUD-BASED DATA

Marcos David dos Santos Lopes[1], **Andrea Schwertner Charão**[2]

RESUMO

Grande parte da demanda atual de software tem a forma de aplicações web, que se estruturam em torno de APIs (*Application Programming Interfaces*) para acesso a dados. Nesta área, também é tendência o consumo de dados e serviços em nuvem, passando por plataformas voltadas para o usuário final, como por exemplo o Google Workspace. Nesta plataforma, o Google Apps Script (GAS) se apresenta como uma solução para desenvolvimento rápido de aplicações inseridas no ambiente em nuvem da Google, possibitando a construção de APIs que respondam a requisições de acesso a dados em nuvem. Mesmo com facilidades, o desenvolvimento em GAS tem etapas repetitivas que são passíveis de automatização. Neste trabalho, explora-se estas tendências com o desenvolvimento de uma ferramenta para geração automatizada de código de APIs web em GAS, para acesso a dados armazenados em planilhas no serviço Google Sheets. Para avaliação da ferramenta, foram conduzidos testes com desenvolvedores, que corroboraram a proposta deste trabalho.

Palavras-chave: Geração de código. APIs. Aplicações Web.

ABSTRACT

Much of the current demand for software takes the form of web applications, which are structured around APIs (*Application Programming Interfaces*) for accessing data. In this domain, the consumption of data and cloud services is also a trend, through platforms aimed at the end user, such as Google Workspace. On this platform, Google Apps Script (GAS) presents itself as a solution for the rapid development of applications inserted in Google's cloud environment, enabling the construction of APIs that respond to requests for access to data in the cloud. Even with such facilities, development on this platform has repetitive steps that are subject to automation. In this work, we explore these trends by developing a tool for automated generation of web APIs code in GAS, enabling access to data stored in spreadsheets in the Google Sheets service. To evaluate the tool, tests were conducted with developers, which corroborated the proposal of this work.

Keywords: Code generation. APIs. Web applications.

[1] Autor: Graduando em Bacharelado em Sistemas de Informação pela Universidade Federal de Santa Maria.

[2] Orientadora: Professora do Departamento de Linguagens e Sistemas de Computação da Universidade Federal de Santa Maria.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento de software tem um impacto significativo em diversos aspectos da economia mundial (ZACKIEWICZ, 2015). No mundo globalizado, cada vez mais movido por tecnologias, a importância do desenvolvimento de software transparece em diferentes áreas e cenários da atividade humana (PRESSMAN, 2010). A computação possibilita descobertas e impulsiona o crescimento em praticamente todo tipo de indústria (BSA, 2016). A indústria de software, que compreende aplicativos, computação em nuvem e afins, é um grande impulsionador da economia dos EUA, agregando mais de US$ 1,6 trilhão no PIB e impactando em cerca de 14 milhões de empregos bem remunerados(BSA, 2020).

Neste contexto, a demanda por software inovador e cada vez mais eficiente cresce de forma exponencial, exigindo tempo e recursos humanos para novas implementações e manutenções. Com aplicações mais complexas, repetição e redundância de códigos podem ocorrer com mais frequência, tomando tempo de desenvolvimento que poderia ser empregado em outros pontos.

No campo do desenvolvimento de software para a web, uma tendência é o emprego de APIs (*Application Programming Interfaces*) que viabilizam a comunicação entre cliente e servidor, com requisições e respostas que trafegam pela rede, seguindo o protocolo HTTP. As APIs contribuem para acelerar o desenvolvimento de software, permitindo organizar o acesso a dados e funcionalidades de forma reusável. Com a crescente disponibilização de serviços em forma da APIs, elas se tornaram uma peça importante no mercado de desenvolvimento de software, com muitos negócios viabilizados por este meio (TAN et al., 2016; VUKOVIC et al., 2016).

Outra tendência no desenvolvimento de software para a web é o acesso a dados e serviços em nuvem, inclusive interligando plataformas voltadas para o usuário final, como por exemplo o Google Workspace[3]. Uma das ferramentas integrantes do ecossistema de aplicações do Google Workspace é o Google Apps Script (GAS), uma plataforma em nuvem para desenvolvimento rápido de aplicações inseridas no ambiente em nuvem da Google. Usando GAS, é possível adicionar funcionalidades para planilhas, documentos, sites e outros serviços da plataforma (FERREIRA, 2014). Além disso, é possível criar serviços Web que respondem a requisições, construindo assim APIs que acessam dados em nuvem. Como exemplo, podemos citar a criação de *endpoints* para realizarmos ações de alteração, criação, deleção e leitura de dados em determinada planilha em nuvem.

Seguindo estas tendências, há empresas que oferecem serviços de criação rápida de APIs a partir de dados em nuvem, ou ainda que auxiliam no processo de desenvolvimento de aplicações utilizando o GAS (Google Apps Script). A ideia geral é abreviar o desenvolvimento de aplicações web de pequeno porte, que se beneficiam de uma infraestrutura poderosa e gratuita oferecida pela Google.

Neste trabalho, busca-se explorar estas tendências, com a geração automatizada de código em GAS para acesso a dados tabulares armazenados em planilhas no serviço Google Spreadsheets. Como contribuição nessa linha, este trabalho apresenta o desenvolvimento de uma ferramenta voltada para usuários desenvolvedores, sejam profissionais ou acadêmicos

[3] <https://workspace.google.com>

da área. Esta ferramenta, acessível pela web, permite que o usuário acesse seus arquivos na plataforma Google e selecione uma planilha para usar como fonte de dados para geração automática de código. A ferramenta gera o código em Google Apps Script para atender requisições HTTP GET e POST que manipulam dados (leitura, escrita, alteração e remoção) da planilha selecionada. O código gerado é exibido ao usuário desenvolvedor, que pode estudá-lo ou modificá-lo, para depois realizar as demais etapas para implantação (*deploy*) da API na nuvem Google, que atualmente não são passíveis de automatização. Com esta ferramenta, espera-se oferecer uma solução que permita criar APIs de forma rápida, com o intuito de simplificar o conhecimento necessário para construir uma API web destinada a aplicações de pequeno porte.

2 FUNDAMENTAÇÃO E TRABALHOS RELACIONADOS

Nesta seção, são apresentados trabalhos relacionados, ferramentas consideradas para utilização no projeto e assuntos relevantes ao tema, considerando a contribuição dos mesmos para a concretização deste projeto.

2.1 APIs Web

A definição da sigla API se dá por *Application Programming Interface* (Interface de Programação de Aplicação). API é uma interface que define o contrato das comunicações entre aplicações, sem a interação do usuário (DE, 2017), estabelecendo a forma com que as solicitações e respostas são feitas. Com a padronização de comunicação impostas por uma API, diferentes aplicações de diferentes contextos podem se comunicar. Isso facilita a criação de soluções personalizadas, atendendo as demandas especificas de usuários e organizações.

Um dos padrões para criação de API mais comum no momento é a API REST (*Representational State Transfer*). REST é um conjunto coordenado de restrições de arquitetura que tenta minimizar a dependência e a escalabilidade das implementações dos componentes (FIELDING, 2000). O REST foi criado com o objetivo de manter o protocolo HTTP para realizar transações na internet sem o auxílio de outro protocolo, recebendo ajuda apenas de seus métodos internos (FERREIRA, 2021). Os métodos (verbos) HTTP mais comumente utilizados em uma API REST (MDN, 2023) são:

- *GET*: O método GET solicita a representação de um recurso específico. Requisições utilizando o método GET devem retornar apenas dados.
- *POST*: O método POST é utilizado para submeter uma entidade a um recurso específico, frequentemente causando uma mudança no estado do recurso ou efeitos colaterais no servidor.
- *PUT*: O método PUT substitui todas as atuais representações do recurso de destino pela carga de dados da requisição.
- *PATCH*: O método PATCH é utilizado para aplicar modificações parciais em um recurso.
- *DELETE*: O método DELETE remove um recurso específico.

2.2 Google Apps Script

O Google Apps Script[4] é uma solução de *cloud computing* que permite integrar e automatizar tarefas nos produtos do ecossistema do Google Workspace. A plataforma tem o intuito de facilitar o desenvolvimento de soluções empresariais de forma rápida e integrada, automatizando e ampliando o leque de possibilidades dos recursos já disponíveis. Para o desenvolvimento, pode ser utilizado o próprio navegador já que os scripts são executados nos servidores do Google. A linguagem utilizada para o desenvolvimento é o JavaScript, com acesso a bibliotecas integradas a diversos recursos de aplicações do ecossistema.

Atualmente, o GAS (Google Apps Script) é constituído por dois tipos de cotas: a cota simples, que abrange contas padrão da Google e não possui custos financeiros, e a cota das contas do Google Workspace (GOOGLE, 2023b). Cada tipo de cota apresenta limitações distintas no número de ocorrências de cada ação. Por exemplo, a cota simples possui um limite de 90 minutos por dia para a execução total de acionadores, enquanto a cota do Google Workspace permite um limite diário de 6 horas. Em relação ao número de requisições, a cota simples permite até 20.000 requisições por dia a partir de um script, enquanto a cota do Google Workspace permite até 100.000 requisições por dia(GOOGLE, 2023a). Essas requisições são o número de vezes que o Google Apps Script faz chamadas externas para outros serviços ou APIs da web usando a função *UrlFetchApp.fetch()*.

2.3 Automação de código

A geração de código não é um tema discutido apenas recentemente, visto que na década de 50 a compilação dos códigos já era considerada um alto nível de automatização (RICH; WATERS, 1988). A automação de código consiste em o usuário definir o que espera do programa, permitindo que este gere, de forma automática e sem a assistência do usuário, o código correspondente às tarefas definidas (SYRIANI; LUHUNU; SAHRAOUI, 2018).

Assim, a automação de código tem o intuito de reduzir tarefas recorrentes e repetitivas, reduzindo o tempo utilizado para escrever código, permitindo com que o desenvolvedor ou estudante direcione seu tempo a outras tarefas, como realizar pequenos ajustes no código e entender o funcionamento e objetivo tanto do código gerado quanto do código existente.

A elaboração automática de código tem sido tópico de diversas discussões na atualidade. Através da utilização de algoritmos de aprendizagem de máquina, a inteligência artificial tem sido capaz de aprender com exemplos já existentes para criar código de forma automática. No entanto, existem limitações para o funcionamento desta abordagem, pois os exemplos gerados nem sempre são precisos ou adequados para a necessidade do usuário, aumentando esse impacto quando se tratam de problemas mais abstratos e subjetivos.

2.4 Soluções correlatas

O desenvolvimento de software em nuvem tem se tornado cada vez mais comum no cenário atual. Nesse contexto, é notável o aumento da quantidade de ferramentas que visam suprir essa demanda. Entre as opções disponíveis, existem muitas soluções

[4] <https://developers.google.com/apps-script>

que se integram ao Google Workspace. Tratam-se de ferramentas que são executadas no navegador web, sem a necessidade da instalação de software localmente, tornando o processo de uso e desenvolvimento dinâmico e produtivo.

A possibilidade de trabalhar com a Google API se torna atrativa para o desenvolvimento, visto que o usuário não precisa se preocupar com servidores dedicados para a execução dos scripts e hospedagem dos dados. Além disso, essas ferramentas permitem uma conexão direta com a conta do usuário no Google, permitindo que a aplicação tenha acesso aos dados e informações necessárias para a geração de código de acordo com a necessidade do usuário, fazendo com que a criação de soluções ocorra de forma mais ágil e flexível.

A seguir serão apresentadas algumas opções existentes no mercado.

2.4.1 API SpreadSheets

API SpreadSheets[5] se apresenta como uma ferramenta para desenvolvedores que desejem prover acesso a planilhas em nuvem, via uma API gerada especificamente para cada fonte de dados. Com isso, o usuário desenvolvedor não precisa se preocupar com a configuração de um servidor ou mesmo com as linhas de código para implementar a API, podendo concentrar-se nas aplicações que irão utilizá-la.

O processo de uso é simples, via interface web, e exige poucos cliques para obter *endpoints* para requisições clássicas de criação, leitura, alteração, remoção (CRUD) de dados da planilha. Ao final, a ferramenta apresenta exemplos de códigos em diferentes linguagens para envio de requisições com a API gerada. Um destes exemplos é apresentado no Código 1, que faz uma requisição em Python usando uma API gerada pela ferramenta API SpreadSheets.

```
# pip install requests

import requests

r = requests.post("https://api.apispreadsheets.com/data/
    jTz4RMONLqI690XD/", headers={}, json={"data": {"#":1,"Data":"13/09"
    }})

if r.status_code == 201:
  # SUCCESS
  pass
else:
  # ERROR
  pass
```

Bloco de código 1 – Exemplo de requisição feita em Python em uma API gerada pelo API SpreadSheets.

Esta ferramenta, no entanto, só disponibiliza para os usuários desenvolvedores as URLs para as requisições web. O usuário não tem acesso ao código em Google Apps Script, o que impossibilita o entendimento de como o código foi gerado e dificulta customizar as implementações conforme a necessidade do usuário. Isso reduz o potencial das aplicações,

[5] <https://www.apispreadsheets.com/>

pois o desenvolvedor fica restrito às funcionalidades e recursos disponibilizados pela ferramenta, o que limita a inovação e criatividade no desenvolvimento de software.

2.4.2 Sheet DB

Sheet DB[6] é outra ferramenta que permite a criação de uma API utilizando planilhas armazenadas no Google Sheets. Seu funcionamento é similar ao da solução anterior, com a diferença que os tutoriais e exemplos de uso são mais genéricos. Os códigos de requisição mostrados pela ferramenta (ver exemplo no Código 2) são exemplos estáticos de código, não são gerados dinamicamente para a API criada. A ferramenta também "oculta" do usuário o código gerado no ecossistema da Google.

```
// Sort results by id in descending order, take two
// and return the age as an integer.

fetch('https://sheetdb.io/api/v1/58f61be4dda40?sort_by=id&sort_order=
    desc&limit=2&cast_numbers=age')
  .then((response) => response.json())
  .then((data) => console.log(data));
```

Bloco de código 2 – Exemplo de requisição é apresentado de forma mais genérica no Sheet DB.

3 DEFINIÇÃO DOS REQUISITOS

A especificação de requisitos é uma atividade fundamental em engenharia de software (SOMMERVILLE, 2018). Os requisitos ditos "funcionais" são as funcionalidades e comportamentos esperados do software que atendem às necessidades e expectativas dos usuários e das partes interessadas.

3.1 Requisitos Funcionais

Para o desenvolvimento deste trabalho, especificou-se os seguintes requisitos funcionais para a ferramenta:

- **RF01** – Autenticação: A ferramenta deve permitir que o usuário faça login usando suas credenciais na plataforma Google.
- **RF02** – Seleção de planilha: A ferramenta deve permitir que o usuário visualize a lista de suas planilhas e selecione a que será usada como fonte de dados.
- **RF03** – Seleção de aba e colunas: A ferramenta deve permitir que o usuário selecione uma aba/página da planilha e as colunas obrigatórias nas requisições HTTP.
- **RF04** – Geração de código: A ferramenta deve gerar o código em Google Apps Script para atender requisições HTTP GET e POST usando como fonte de dados a planilha selecionada. O código resultante deve ser retornado ao usuário, que deve realizar as demais etapas para implantação (*deploy*) em <https://script.google.com>.

[6] <https://sheetdb.io/>

3.2 Diagrama de casos de uso

A partir dos requisitos especificados, produziu-se um diagrama de casos de uso. Este tipo de diagrama é uma representação visual que descreve as interações entre os atores (usuários) e o sistema, identificando as interações previstas pelos requisitos do sistema (SOMMERVILLE, 2018).

Um caso de uso identifica os atores envolvidos em uma interação e nomeia o tipo de interação. Depois, são adicionadas informações descrevendo a interação com o sistema, que pode ser uma descrição textual ou um ou mais modelos gráficos (SOMMERVILLE, 2018).

Os atores em um processo, que podem ser humanos ou outros sistemas, são representados como bonecos palitos. Cada classe de interação é representada como uma elipse nomeada, onde linhas fazem a ligação entre atores do sistema e a interação.

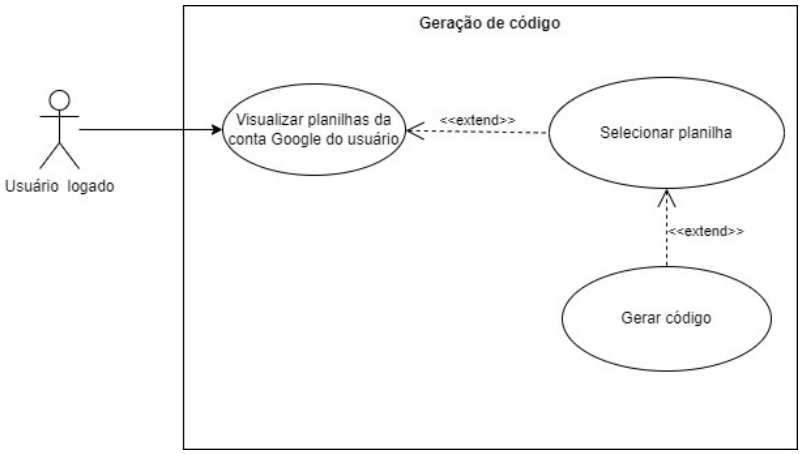
A Tabela 1 distingue atores e suas interações com a ferramenta proposta, ressaltando que é imprescindível a autenticação do usuário na plataforma Google para acesso às funcionalidades da ferramenta.

Ator	**Descrição**
Não autenticado	Usuário que não fez login na plataforma Google. Não possui acesso a nenhum recurso da ferramenta.
Autenticado	Usuário autenticado na plataforma Google. Possui acesso às funcionalidades de seleção de tabela/página/colunas e geração de código.

Tabela 1 – Atores do sistema

O caso de uso descrito na Figura 1 representa a principal interação dos usuários com a ferramenta, permitindo a visualização das planilhas às quais o usuário tem acesso com sua conta da Google (RQ02), a seleção de abas/colunas (RQ03) e a geração de código (RQ04) a partir da planilha selecionada.

Figura 1 – Diagrama de casos de uso da geração de código



Fonte: Imagem do autor

3.3 Geração de código: requisições HTTP

Para o processo de geração de código ser possível, a ferramenta precisa de permissões de acesso ao Google Drive do usuário. Com as devidas permissões, a ferramenta deve possibilitar escolher a planilha que será a fonte de dados para as requisições GET e POST.

Cada planilha pode ter uma ou mais abas/páginas, acessíveis separadamente por diferentes requisições POST de inserção, remoção ou atualização de dados, ou ainda requisições GET de consulta aos dados. Cada requisição pode manipular uma ou mais linhas da planilha, possivelmente usando uma das colunas como chave, imitando um banco de dados relacional.

4 DESENVOLVIMENTO DA FERRAMENTA

A ferramenta desenvolvida é resultado da interação de 3 grandes partes *front end*, *back end* e base do código. Podemos visualizar o funcionamento da integração das mesmas na Figura 2. Como podemos observar, o processo de autenticação feito através do Google API tem um papel crucial no funcionamento da ferramenta. O fluxograma apresenta as etapas necessárias para conseguirmos acesso às planilhas do usuário, que ficam armazenadas em seu Google Drive. Portanto, é necessário solicitar permissões para o utilizador.

Após o usuário permitir o acesso da ferramenta às planilhas, recebemos um "Token Access". Com isso, é possível fazer requisições para obter dados que possibilitam listar as planilhas no *front end* da ferramenta. Isso possibilita que o usuário selecione uma planilha, a partir da qual será feita a identificação das colunas para a geração do código.

Nas próximas subseções, serão apresentadas as responsabilidades da autenticação, *front end*, *back end* e a base do código.

4.1 Autenticação

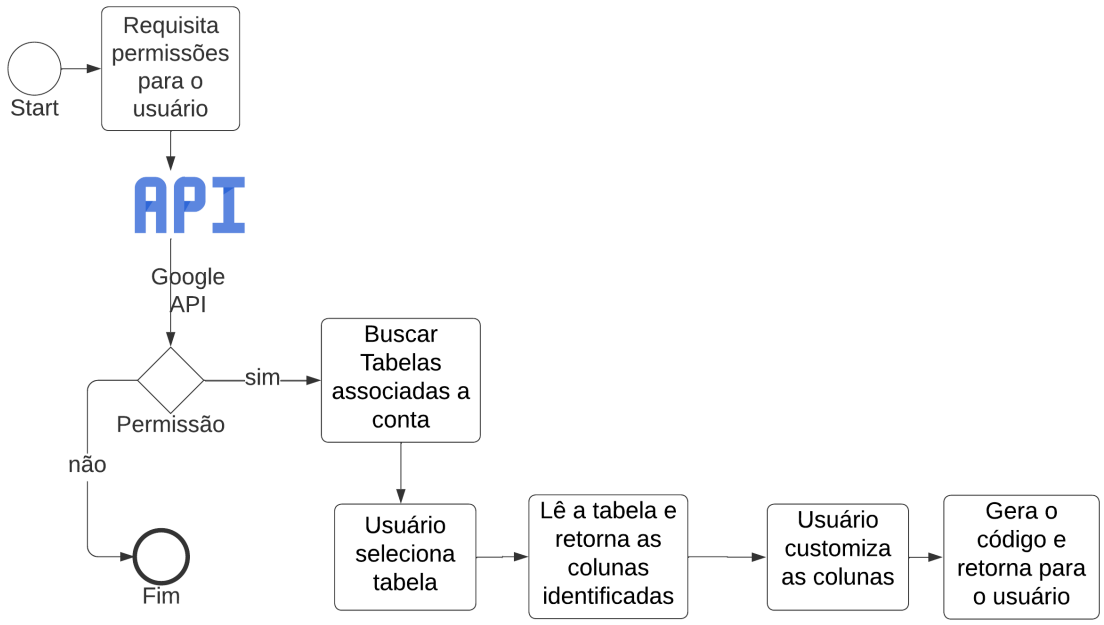
Todo o processo de autenticação ocorre pelas APIs do Google, utilizando o protocolo OAuth 2.0[7]. Esse processo é vital neste trabalho, pois precisamos das informações pertencentes ao usuário para podermos acessar os dados das planilhas, como nome, id e conteúdo, a fim de implementar os requisitos da ferramenta. Isso é feito em dois passos, conforme descrito a seguir.

O primeiro passo está representado na Figura 3, onde se tem o diagrama de sequência respectivo ao login. Como se pode observar, o sistema de autenticação da Google é responsável pelo processo de reconhecer o usuário.

O segundo passo é necessário para obter a permissão de acesso aos dados armazenados no Google Drive do utilizador. Isso está retratado na Figura 4, onde temos o diagrama de sequência referente à obtenção dos nomes das planilhas relacionadas à conta do usuário. É importante ressaltar que, para as requisições feitas para os servidores da Google no intuito de buscar os dados, se faz necessário enviar juntamente o "permissionToken" do usuário em questão.

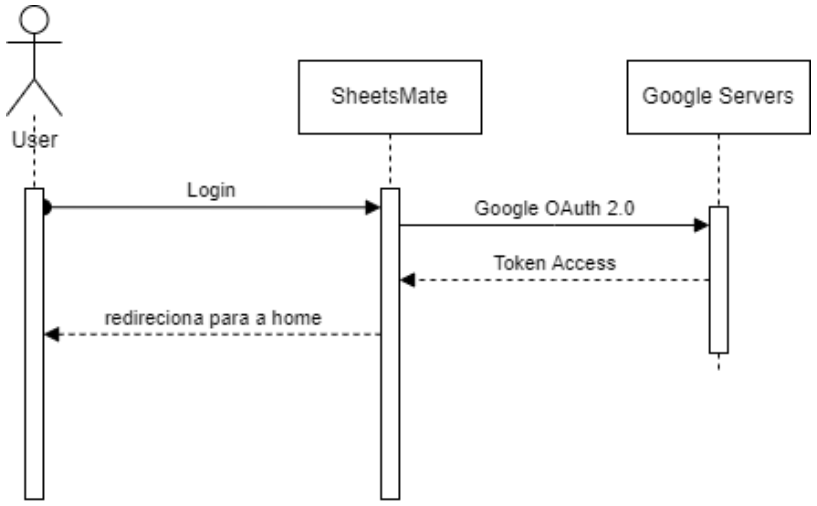
[7] <https://datatracker.ietf.org/doc/html/rfc6749>

Figura 2 – Fluxograma da aplicação.



Fonte: Imagem do autor

Figura 3 – Diagrama de sequência – Login



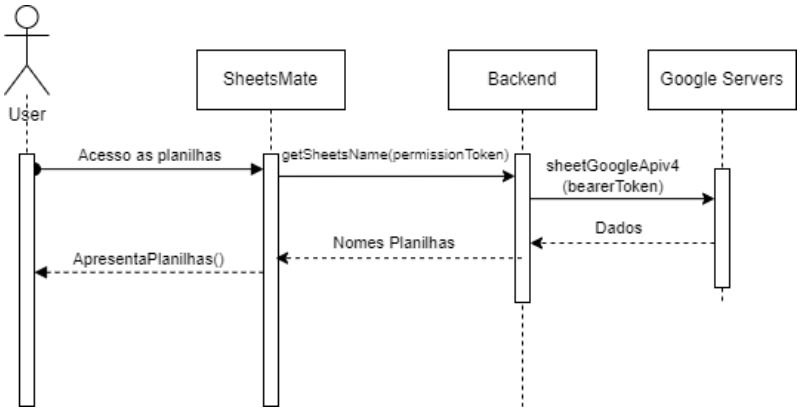
Fonte: Imagem do autor

4.2 Front-end

O *front end* é a parte visual, que permite o usuário interagir com as funcionalidades propostas pela ferramenta. Para o desenvolvimento do *front end*, foi utilizado Angular[8], um framework de código aberto que utiliza TypeScript como linguagem principal. O Angular possui uma arquitetura modular e orientada a componentes, adotando o conceito de uma "árvore de componentes", onde cada componente encapsula sua própria lógica e apresentação.

[8] <https://angular.io/>

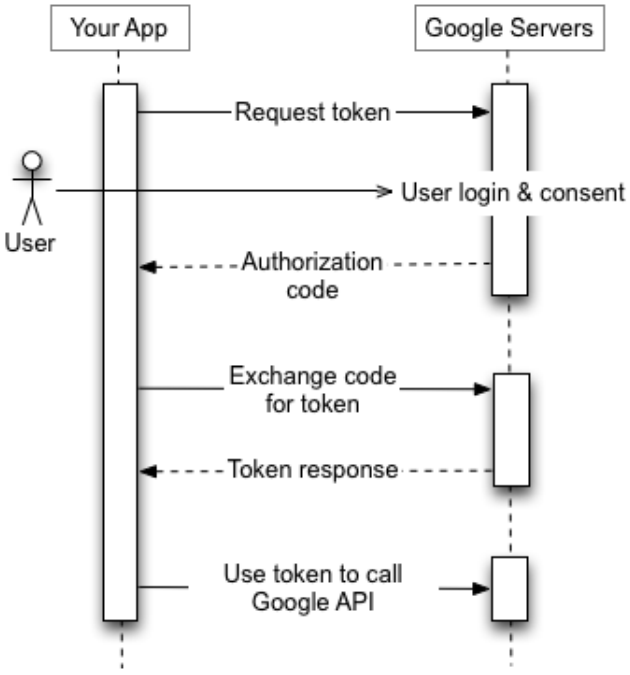
Figura 4 – Diagrama de sequência – Comunicação entre componentes



Fonte: Imagem do autor

O fluxograma da interação do *front end*, Google API e *back end*, ilustrado na Figura 2 apresentada anteriormente, está implementado e disponível em um repositório de código aberto[9]. A interação entre estas partes só é possível após o login do usuário. O processo de login é realizado através do *front end*, utilizando a autenticação da Google[10], representada na Figura 5.

Figura 5 – Fluxo para recebimento do token de acesso da Google.



Fonte: Página sobre OAUTH 2.0 da Google

O front realiza a comunicação com o *back end* passando o "permissionToken" obtido através da autorização, como representado na Figura 6. Assim, o *back end* consegue

[9] Acessível em <https://github.com/Davidlopes22/sheets-mate.github.io/>
[10] <https://developers.google.com/identity/protocols/oauth2?hl=pt-br>

realizar a comunicação para requisitar as planilhas e dados atreladas à conta do usuário.

Figura 6 – Tela de consentimento para liberação do acesso aos dados

Fonte: Imagem do autor

4.3 *Back end*

O *back end*[11] é responsável por orquestrar as informações com o *front end* e Google API, tratando os dados e passando tais informações para o *front end* de acordo com a necessidade. É também responsável por identificar a composição e posição das colunas em uma tabela, e por utilizar o código base descrito na seção 4.4 para a geração de código que será enviada e apresentada pelo *front end* como apresentado na Figura 7.

[11] Código disponível em <https://github.com/Davidlopes22/sheets-mate-backend>

No processo de desenvolvimento do *back end*, foi utilizado o próprio Google Apps Script[12]. A escolha se deu pela integração nativa com os produtos da Google, como o Google Sheets, facilitando a obtenção e manipulação dos dados necessários. Um outro ponto positivo é que, sendo o Google Apps Script uma plataforma de desenvolvimento baseado na nuvem, não há necessidade de dedicar um servidor para hospedar a aplicação, já que a mesma fica hospedada e disponível nos servidores da Google. É importante ressaltar que código do *back end* é apartado do código que é gerado para o usuário, pois tem diferentes funcionalidades e responsabilidades. Ou seja, o *back end* e o código gerado para o usuário ficam em projetos Google Apps Script distintos.

Figura 7 – Exemplo do modal de um código gerado

Fonte: Imagem do autor

4.4 Base do código

A base de código foi desenvolvida com o objetivo de ser o mais genérica possível, se atentando a algumas particularidades do funcionamento do GAS. A mesma foi desenvolvida para receber as requisições, tratar os dados e manipular as planilhas. Os ajustes da base ocorreram durante o desenvolvimento, sendo necessário testar a base em diversas tabelas, tentando encontrar cenários em que pudessem ocorrer falhas ou um mau funcionamento.

4.4.1 Acesso à planilha

Para ser possível acessarmos os dados da planilha selecionada ou fazer qualquer tipo de alteração, primeiro são necessárias algumas chamadas em Google Apps Script (ver Código 3).

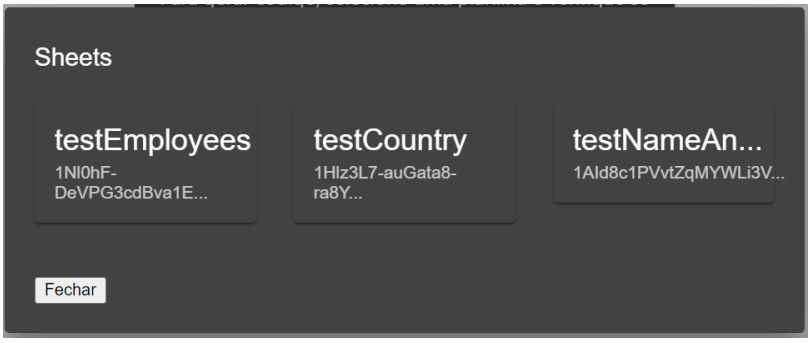
[12] <https://script.google.com/home>

```
const spreadsheetUrl = 'https://docs.google.com/spreadsheets/d/18
   p01Amy4VhgjZBZ2ABO-zmtPzAeZBlEAPj7X8P3gxg0/';
const file = SpreadsheetApp.openByUrl(spreadsheetUrl);
const sheetExample = file.getSheetByName('sheetExample');
```

Bloco de código 3 – Processo para acessar a planilha via GAS

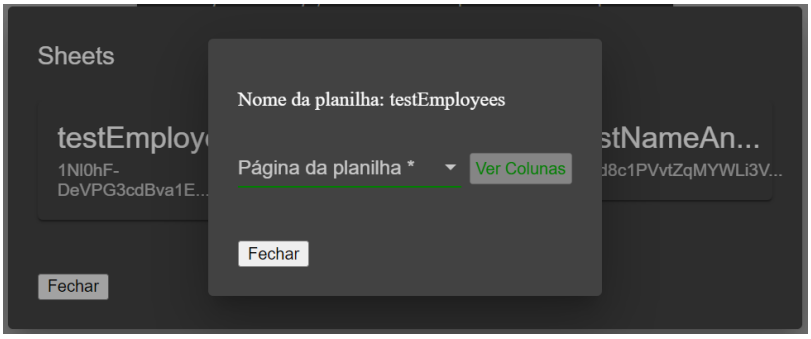
Para conseguirmos o acesso, precisamos da URL e o nome da página da planilha, visto que uma planilha pode ter várias páginas. No Código 3, essas informações estão respectivamente nas constantes `spreadsheetUrl` e `sheetExample`. Os valores para estas constantes são fornecidos pelo usuário através do *front end*, como apresentado nas Figuras 8 e 9.

Figura 8 – Exemplo do modal para seleção da planilha



Fonte: Imagem do autor

Figura 9 – Exemplo do modal para seleção da página de uma planilha



Fonte: Imagem do autor

Esse passo é essencial, pois é a partir da constante `sheetExample` que será possível acessar os métodos para manipular e acessar os dados da planilha.

4.4.2 Requisição

O Google Apps Script não oferece suporte nativo para a criação de *endpoints*. No entanto, é possível contornar essa limitação utilizando os métodos `doGet` e `doPost`, que aceitam solicitações do tipo GET e POST, respectivamente. Assim, podemos implementar em `doPost` os métodos que precisam receber dados, como POST e UPDATE, e em

`doGet` os métodos do tipo GET e DELETE, que não precisam receber dados no corpo da requisição.

```
function doPost(e) {
  if (isMissingProperties(e, ["actionRequest"])) {
    return errorMessage("Missing request actionRequest");
  }
  const requestAction = e.parameter.actionRequest;
  var jsonData = e.postData?.contents;

  if (!jsonData) {
    return errorMessage("post data cannot be empty");
  }
  jsonData = JSON.parse(jsonData);
  switch(requestAction) {
    case "postSite": return postSite(e, jsonData, columnsSiteSheet);
    case "postProject": return postProject(e,jsonData,
        columnsProjectSheet);
    case "deleteByKeys": return deleteRows(jsonData,
        projectSheet, keys);
    case "deleteWithoutKeys": return deleteRows(jsonData,
        projectSheet,columnsProjectSheet);
    case "updateUsingKeys": return updateByKey(jsonData, projectSheet,
        columnsProjectSheet);
    case "updateByquery": return updateByQuery(e, jsonData,
        projectSheet, columnsProjectSheet);
    default: return errorMessage("Invalid request action");
  }
}
```

Bloco de código 4 – Exemplo de implementação do método doPost em Google Apps Script

No código 4, é apresentado um exemplo de implementação do `doPost`. O parâmetro `actionRequest` é utilizado dentro de uma estrutura `switch` para realizar o encaminhamento para diferentes métodos.

4.4.3 Manipulação de dados da planilha

Para acessarmos e editarmos os dados de uma planilha, iremos utilizar a constante `sheetExample` apresentada na subseção 4.4.1, que guarda uma referência à página da planilha que queremos manipular.

```
function postProject(data, columns) {

  var lastRow = sheetExample.getLastRow() + 1;
  var range = sheetExample.getRange(lastRow, columns[0][0], 1,
      columns.length);
  range.setValues(data);
  return successMessage("Project added!");

}
```

Bloco de código 5 – Exemplo de implementação de um método para inserir dados em uma planilha em Google Apps Script

No bloco de código 5, temos um exemplo de inserção de dados em uma linha. O método recebe os dados que irão ser inseridos e as colunas da tabela. As colunas nada mais são que uma estrutura auxiliar que foi criada para apoiar os métodos. A estrutura é composta por uma matriz aninhada, onde cada matriz é composta por três dados: posição da coluna, nome da coluna e um `boolean` que indica se esse dado é obrigatório em momentos de inserção.

Tabela 2 – Exemplo de tabela: Projetos

ID	Title	Dev	Client
1	Projeto A	Dev1	Cliente1
2	Projeto B	Dev2	Cliente2
3	Projeto C	Dev3	Cliente3

No método do bloco de código 5, primeiro é encontrada a última linha com dados na tabela. Tomando como exemplo a Tabela 2, esta operação irá retornar a linha 4. Após isso, obtém-se o "`range`", que é o intervalo de células que foi encontrado através dos parâmetros passados no método. Esses parâmetros são: (1) a linha em que queremos fazer a inserção (a primeira linha livre após os dados da planilha, ou seja linha 5), (2) a posição da primeira coluna utilizando da estrutura auxiliar para encontrar o dado (primeira coluna na tabela 2, seria a coluna 1), (3) o número de linhas que devem ser incluídas no intervalo e (4) o parâmetro que representa quantas colunas devem ser incluídas (4 colunas, de acordo com a planilha de exemplo).

5 AVALIAÇÃO DE FERRAMENTA

Para avaliar as funcionalidades e objetivos da ferramenta desenvolvida, foram elaborados dois instrumentos de coleta de dados, que foram aplicados em sequência junto a uma amostra do público-alvo. Dado o tempo disponível para este Trabalho de Conclusão de Curso, os instrumentos não tinham a pretensão de obter resultados com significância estatística, mas sim de realizar uma avaliação qualitativa, coletando opiniões de potenciais usuários da ferramenta.

Ambos instrumentos foram concebidos para serem respondidos assincronamente por estudantes e profissionais da área de Computação. Os instrumentos se complementaram, à medida que o primeiro foi de preenchimento mais rápido e preparou possíveis voluntários para o segundo, que exigiu mais tempo dos respondentes. As seções seguintes detalham a composição, aplicação e resultados obtidos por meio desses 2 instrumentos.

5.1 Primeiro instrumento

O primeiro instrumento foi um questionário com 3 questões objetivas sobre experiência dos respondentes com APIs REST e Google Apps Script, e um campo aberto para que possíveis voluntários pudessem deixar seus contatos para testes com a ferramenta.

Este primeiro instrumento, criado no Google Forms e transcrito no Apêndice 1, foi enviado para os grupos institucionais de e-mail de acadêmicos matriculados do segundo semestre de 2023, nos cursos de Sistemas de Informação e Ciência da Computação da Universidade Federal de Santa Maria. A comunicação foi realizada por esse canal com o

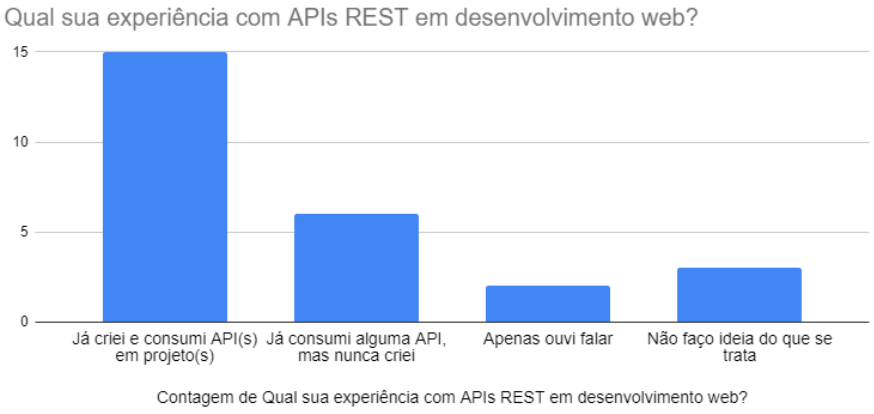
intuito de alcançar o público que faz parte da comunidade estudantil desses cursos que, segundo o portal UFSM em Números[13], totaliza 313 acadêmicos. O questionário obteve um total de 26 respondentes, sendo que 6 deixaram dados para contato, se dispondo a realizar testes com a ferramenta.

5.1.1 Resultados

A partir das 26 respostas obtidas no formulário, foi possível identificar o conhecimento e experiência prévia dos respondentes em assuntos fundamentais para o entendimento da ferramenta. Os resultados sumarizados a seguir auxiliaram na tomada de decisões para os próximos passos da avaliação.

Na Figura 10, pode-se observar que a grande maioria dos respondentes (92.3%) relatou algum tipo de conhecimento relacionado a APIs REST.

Figura 10 – Conhecimento dos respondentes referente a APIs REST



Fonte: Imagem do autor

Ainda assim, como se pode notar no gráfico da Figura 11, apenas 26.9% dos respondentes tiveram algum tipo de contato com Google Apps Script, logo percebeu-se que o desconhecimento dos demais poderia ser um entrave ao teste e utilização da ferramenta. Isso motivou a criação de um material auxiliar, disponibilizado junto com a ferramenta, para sanar eventuais dúvidas e facilitar a compreensão da tecnologia e da sua utilização.

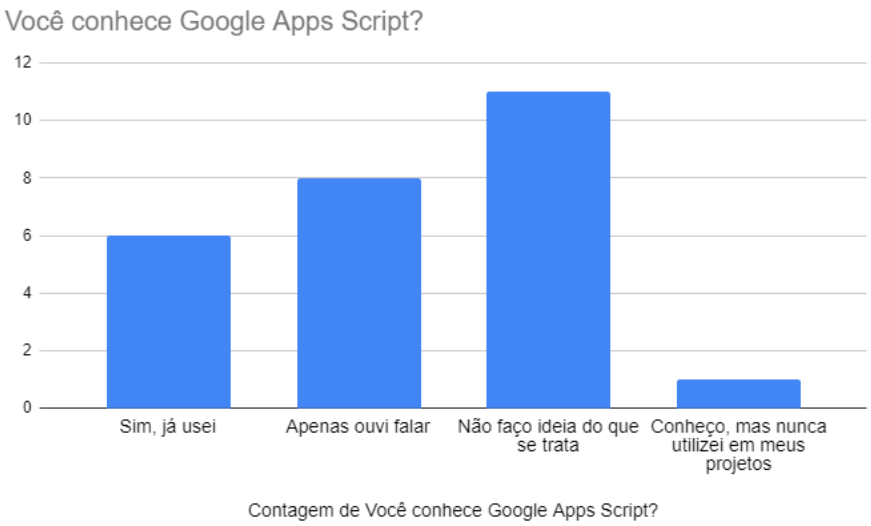
Na Figura 12, observa-se 50% dos respondentes não sabiam da possibilidade de criar APIs web para acessar dados em planilhas através do GAS. Esse resultado reforçou a necessidade de um material explicativo adicional. Além disso, esse dado sugere que a ferramenta pode representar uma oportunidade de aprendizado a quem desconhece as potencialidades das tecnologias envolvidas.

5.2 Segundo instrumento

O segundo instrumento foi elaborado para guiar um teste com usuários desenvolvedores. Tal instrumento foi composto por um material de apoio e um questionário de

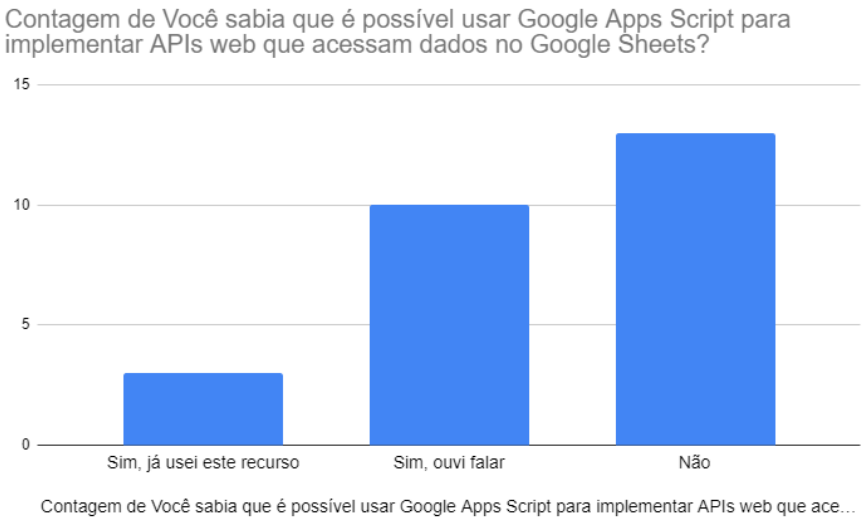
[13] <https://portal.ufsm.br/ufsm-em-numeros>

Figura 11 – Conhecimento dos respondentes sobre Google Apps Script.



Fonte: Imagem do autor

Figura 12 – Conhecimento dos respondentes sobre implementação de APIs com Google Apps Script

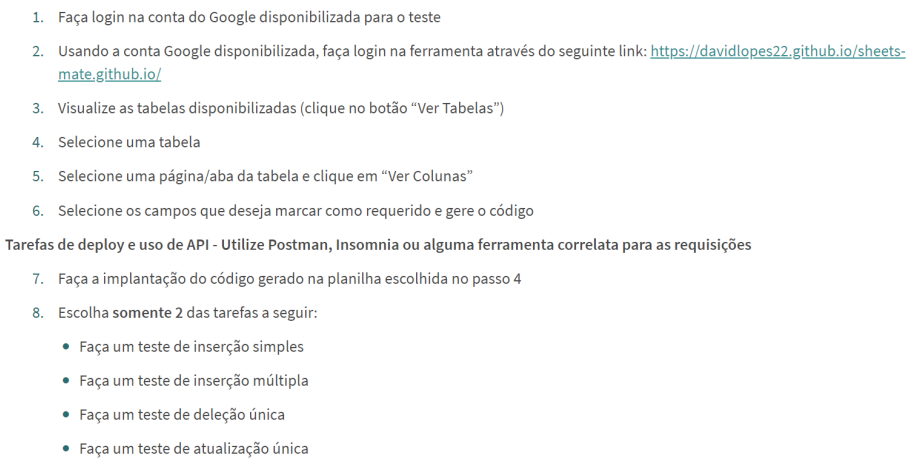


Fonte: Imagem do autor

resposta anônima, reunidos em uma único documento web[14].

O material de apoio foi desenvolvido para auxiliar os testadores durante a realização do teste, contendo 2 vídeos explicativos (um sobre construção de APIs web com Google Apps Script e outro sobre a ferramenta desenvolvida) e um roteiro com tarefas de teste (ver Figura 13). Esse material continha os passos necessários desde o acesso à ferramenta até a implantação do código e utilização dos *endpoints* da API gerada.

Figura 13 – Conhecimento dos respondentes sobre implementação de APIs com Google Apps Script



Fonte: Imagem do autor

A composição do questionário se deu por 13 perguntas, baseadas no Modelo de Aceitação de Tecnologia (em inglês, *Technology Acceptance Model* – TAM). O TAM foi escolhido por ser amplamente utilizado para estudar a aceitação de determinada tecnologia por indivíduos, considerando a influência de fatores humanos (LAI, 2017). Embora se saiba que este modelo não é especificamente voltado a usuários desenvolvedores, e também sabendo-se da carência de instrumentos de coleta sistematizados e padronizados para este público, entendeu-se que o TAM poderia revelar observações úteis para avaliação da ferramenta.

As perguntas foram estruturadas da seguinte maneira:

- 10 tinham uma escala Likert de 1 até 7, onde 1 representava "Discordo completamente" e 7 "Concordo completamente", conforme orientação do TAM;
- duas eram perguntas abertas, uma para sugestões de melhorias e outra para registrar eventuais erros/bugs;
- uma pergunta sobre a experiência com APIs REST em desenvolvimento web.

O segundo instrumento foi enviado às 6 pessoas que deixaram contato no primeiro formulário. Como nem todas continuaram disponíveis para testes, e também como forma de ampliar o perfil e número de respondentes, enviou-se o instrumento para alguns egressos

[14] Disponível em <https://liascript.github.io/course/?https://raw.githubusercontent.com/Davidlopes22/sheetsMate-docs/main/aux-docs/README.md>

recentemente graduados nos cursos de Sistemas de Informação e Ciência da Computação. O questionário, que está apresentado no Apêndice 2, obteve um total de 5 respostas.

5.2.1 Resultados

Os resultados do questionário forneceram informações importantes sobre a ferramenta, tanto sobre sua funcionalidade quanto pontos para melhoria. Todos os usuários conseguiram concluir as tarefas propostas no instrumento. Um dos usuários teve algumas dificuldades com o processo de implementação do código e utilização dos *endpoints*. As dúvidas foram sanadas após este usuário assistir o vídeo sobre a ferramenta disponibilizado e acessar a seção de utilização de *endpoints* na ferramenta.

O questionário revelou que foram reconhecidos pontos positivos na ferramenta, como a possibilidade de criar APIs de forma rápida, conforme indica o gráfico da Figura 14. O código gerado pela ferramenta, segundo os testadores, é legível e organizado.

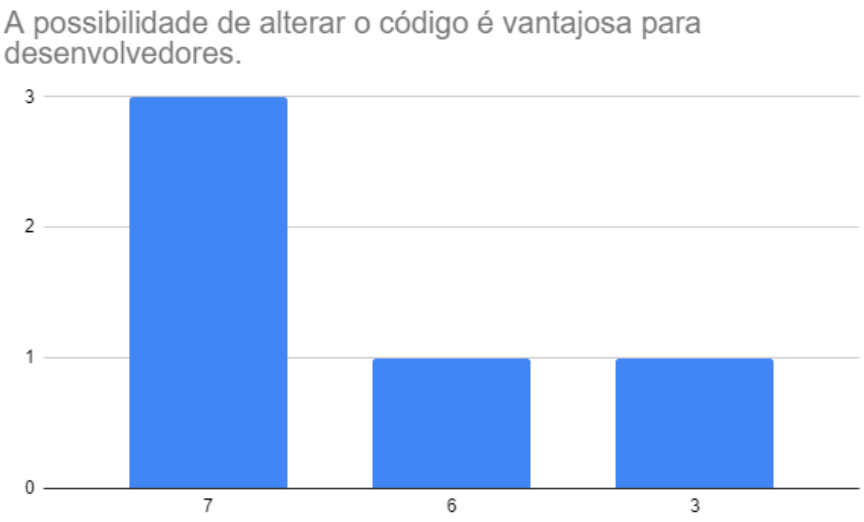
Figura 14 – Respostas sobre rapidez de desenvolvimento comparado a uma criação manual

Fonte: Imagem do autor

A possibilidade de customizar o código gerado foi outro ponto positivo, como visto na Figura 15. A ideia de disponibilizar todo o código para os usuários era justamente para oferecer uma maior flexibilidade no seu uso, em comparação com soluções correlatas. Porém, um dos usuários relatou o seguinte:

> ...Uma questão interessante caso a ferramenta receba melhorias é como fornecer funcionalidade extra sem exigir que o usuário copie n linhas de código.
>
> Por exemplo, queries podem ser mais poderosas, por exemplo, quero triplicar todos os registros cujo valor é abaixo de 10? Para dar suporte a queries mais complexas exigirá muito mais volume de código.
>
> ...Uma sugestão seria que o usuário pudesse definir quais recursos são necessários. Outra ideia pode ser criar uma solução mais dinâmica que permita acessar outros recursos do próprio Google Sheets como comparação, etc.

Figura 15 – Respostas sobre a vantagem de se poder alterar o código gerado



Fonte: Imagem do autor

Outro ponto que suscitou possíveis melhorias é a forma de implantação do código pelo usuário final. Atualmente, é necessário copiar todo o código e colá-lo na página de implementação. Ainda outra questão levantada por um usuário está relacionada ao *front end*:

> Acredito que uma possível melhoria seria em relação aos modais presentes na interface, em alguns o texto está cortando.

6 CONCLUSÃO

Neste trabalho, foi proposta uma ferramenta para geração de código em Google Apps Script a partir de planilhas, com o objetivo de fornecer uma solução para criação rápida de APIs, visando reduzir o conhecimento necessário para construir uma API web para aplicações de pequeno porte. A ferramenta permite que o usuário final customize a implementação gerada, facilitando a criação de soluções para manipular dados em planilhas.

Este trabalho cumpre com o objetivo proposto, sendo que as funcionalidades projetadas foram implementadas e estão disponiveis em <https://davidlopes22.github.io/sheets-mate.github.io/>. Ao usuário fazer login utilizando uma conta da Google, o mesmo consegue visualizar as tabelas relacionadas a sua conta e ver as páginas relacionadas a tabela escolhida. A partir disso, o mesmo pode visualizar as colunas identificadas pela aplicação ou indicar manualmente a linha em que o nome das colunas está alocada na planilha. Com isso, o utilizador pode prosseguir para a geração de código.

Todo o processo de implantação do código na plataforma do Google (ver Apêndice 3) é de responsabilidade do usuário, que tem a possibilidade de modificar o código, customizando-o de acordo com as funcionalidades adicionais que deseja.

Para trabalhos futuros, pode-se desenvolver funcionalidades opcionais no código gerado, criando métodos para ações específicas, como por exemplo a validação de CPF. Outro incremento possível é o tratamento de *timeouts* no código gerado, visto que o script em GAS tem um tempo máximo de execução de 6 minutos. Essa melhoria se faz necessária para lidar com este tipo de erro, uma vez que quanto mais dados em uma planilha, maior será o tempo de execução em métodos de atualização e deleção, sendo que o código percorre pelas colunas para encontrar qual linha deve ser atualizada ou deletada.

REFERÊNCIAS

BSA. *The $1 Trillion Economic Impact of Software*. [S.l.], 2016. Disponível em: <https://www.bsa.org/reports/the-1-trillion-economic-impact-of-software>. Acesso em: 25 fevereiro 2023. Citado na página 8.

BSA. *BSA Policy Recommendations for the Biden-Harris Transition Team*. [S.l.], 2020. Disponível em: <https://www.bsa.org/policy-filings/bsa-policy-recommendations-for-the-biden-harris-transition-team>. Acesso em: 11 março 2023. Citado na página 8.

DE, B. Api management. In: ____. *API Management: An Architect's Guide to Developing and Managing APIs for Your Organization*. Berkeley, CA: Apress, 2017. p. 15–28. ISBN 978-1-4842-1305-6. Disponível em: <https://doi.org/10.1007/978-1-4842-1305-6_2>. Citado na página 9.

FERREIRA, A. G. *Interface de Programação de Aplicações (API) e Web Services*. [S.l.]: Novatec Editora, 2021. Citado na página 9.

FERREIRA, J. *Google Apps Script: Web Application Development Essentials*. [S.l.]: "O'Reilly Media, Inc.", 2014. Citado na página 8.

FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. [S.l.]: University of California, Irvine, 2000. Citado na página 9.

GOOGLE. *Google Apps Script - Quotas for Google Services*. 2023. [Online; accessed 21 de julho de 2023]. Disponível em: <https://developers.google.com/apps-script/guides/services/quotas>. Citado na página 10.

GOOGLE. *Workspace Google*. 2023. [Online; acessado em 21 de julho de 2023]. Disponível em: <https://workspace.google.com/intl/pt-BR/business/>. Citado na página 10.

LAI, P. C. The literature review of technology adoption models and theories for the novelty technology. *JISTEM-Journal of Information Systems and Technology Management*, SciELO Brasil, v. 14, p. 21–38, 2017. Citado na página 24.

MDN. *Métodos de requisição HTTP | MDN*. 2023. [Online; accessed 12. May 2023]. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods>. Citado na página 9.

PRESSMAN, R. S. *Software engineering : a practitioner's approach*. [S.l.]: McGraw-Hil, 2010. v. 7th ed. Citado na página 8.

RICH, C.; WATERS, R. C. Automatic programming: Myths and prospects. *Computer*, IEEE, v. 21, n. 8, p. 40–51, 1988. Citado na página 10.

SOMMERVILLE, I. *Engenharia de Software*. 10. ed. [S.l.]: Pearson Education do Brasil, 2018. Citado 2 vezes nas páginas 12 e 13.

SYRIANI, E.; LUHUNU, L.; SAHRAOUI, H. Systematic mapping study of template-based code generation. *Computer Languages, Systems & Structures*, Elsevier, v. 52, p. 43–62, 2018. Disponível em: <https://doi.org/10.1016/j.cl.2017.11.003>. Citado na página 10.

TAN, W. et al. From the service-oriented architecture to the web api economy. *IEEE Internet Computing*, v. 20, n. 4, p. 64–68, 2016. Citado na página 8.

VUKOVIC, M. et al. Riding and thriving on the api hype cycle. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 59, n. 3, p. 35–37, feb 2016. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/2816812>. Citado na página 8.

ZACKIEWICZ, M. A economia do software e a digitalização da economia. *Revista Brasileira de Inovação*, v. 14, n. 2, p. 315, 2015. Citado na página 8.

A APÊNDICE

1 PRIMEIRO QUESTIONÁRIO

1. Qual sua experiência com APIs REST em desenvolvimento web?

   ☐ Não faço ideia do que se trata
   ☐ Apenas ouvi falar
   ☐ Já consumi alguma API, mas nunca criei
   ☐ Já criei e consumi API(s) em projeto(s)
   ☐ Outros...

2. Você conhece Google Apps Script?

   ☐ Não faço ideia do que se trata
   ☐ Apenas ouvi falar
   ☐ Sim, já usei
   ☐ Outros...

3. Você sabia que é possível usar Google Apps Script para implementar APIs web que acessam dados no Google Sheets?

   ☐ Não
   ☐ Sim, ouvi falar
   ☐ Sim, já usei este recurso
   ☐ Outros...

4. Em um TCC, estamos desenvolvendo uma ferramenta que auxilia na criação de APIs web com Google Apps Script. Caso você possa colaborar nos testes nos próximos dias, por favor deixe seu contato abaixo:

   _____

2 SEGUNDO QUESTIONÁRIO

Na escala 1 representa Discordo completamente e 7 Concordo completamente.

1. Usar esta ferramenta me permitiria criar uma API web mais rapidamente do que se eu tivesse que criá-la manualmente.

   1 2 3 4 5 6 7

2. Usar esta ferramenta aumentaria minha produtividade no desenvolvimento de uma API web.

1 2 3 4 5 6 7

3. Usar esta ferramenta tornaria mais fácil desenvolver uma API web.

   1 2 3 4 5 6 7

4. Considero esta ferramenta útil para o desenvolvimento de aplicações web de pequeno porte.

   1 2 3 4 5 6 7

5. Foi fácil operar esta ferramenta nos testes realizados.

   1 2 3 4 5 6 7

6. A interface da ferramenta é clara e compreensível.

   1 2 3 4 5 6 7

7. O código gerado pela ferramenta é legível e organizado.

   1 2 3 4 5 6 7

8. A possibilidade de alterar o código é vantajosa para desenvolvedores.

   1 2 3 4 5 6 7

9. A possibilidade de hospedar a API e os dados na plataforma Google é vantajosa para desenvolvedores.

   1 2 3 4 5 6 7

10. Eu recomendaria esta ferramenta para desenvolvedores.

    1 2 3 4 5 6 7

11. Qual sua experiência com APIs REST em desenvolvimento web?

    ☐ Não faço ideia do que se trata
    ☐ Apenas ouvi falar
    ☐ Já consumi alguma API, mas nunca criei
    ☐ Já criei e consumi API(s) em projeto(s)

12. Caso tenha encontrado dificuldades ou erros nos testes, por favor registre-os aqui.

    ---

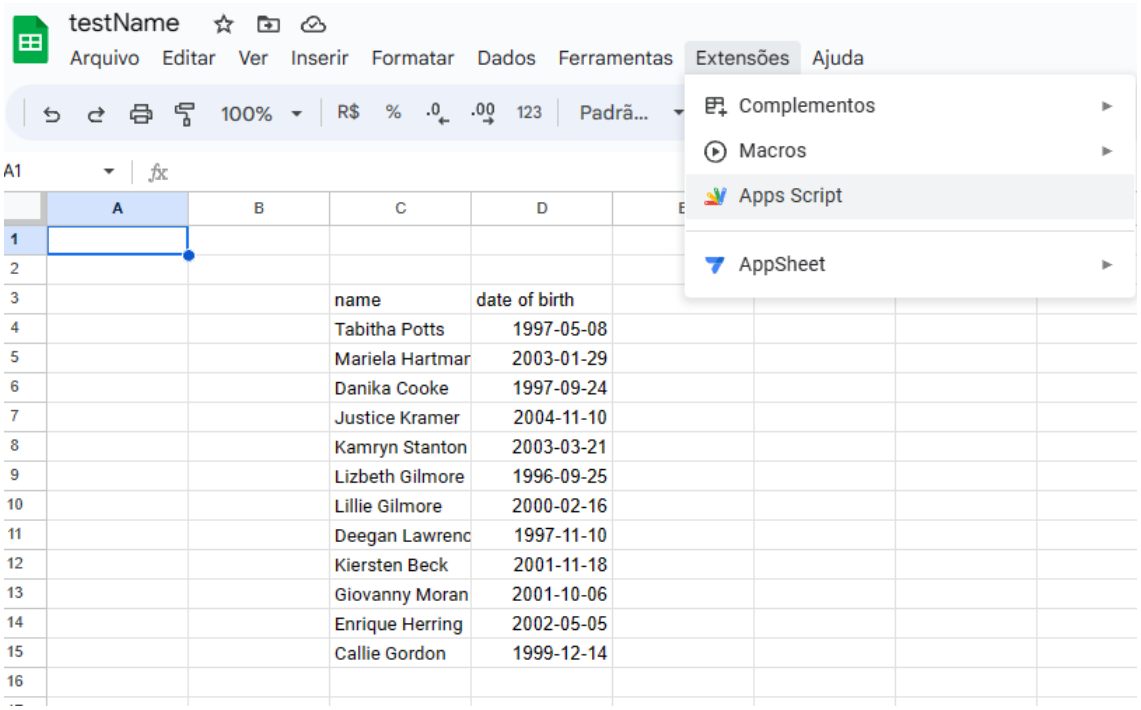
13. Você gostaria de sugerir alguma melhoria na ferramenta?

    ---

3 DEPLOY DO CÓDIGO GERADO

Esse apêndice irá apresentar o passo a passo para realizar o deploy do código gerado pela ferramenta. Após copiar o código gerado, deve ser acessado a página da planilha em questão, clicando em *Extensões* e em *Apps Script*.

Figura 16 – Seção para acessas o Apps Scripts através do Google Spreadsheets



Fonte: Imagem do autor

Em seguida, deve ser colado o código como apresentado na Figura 17 e, depois disso, clicar no botão *Implantar*.

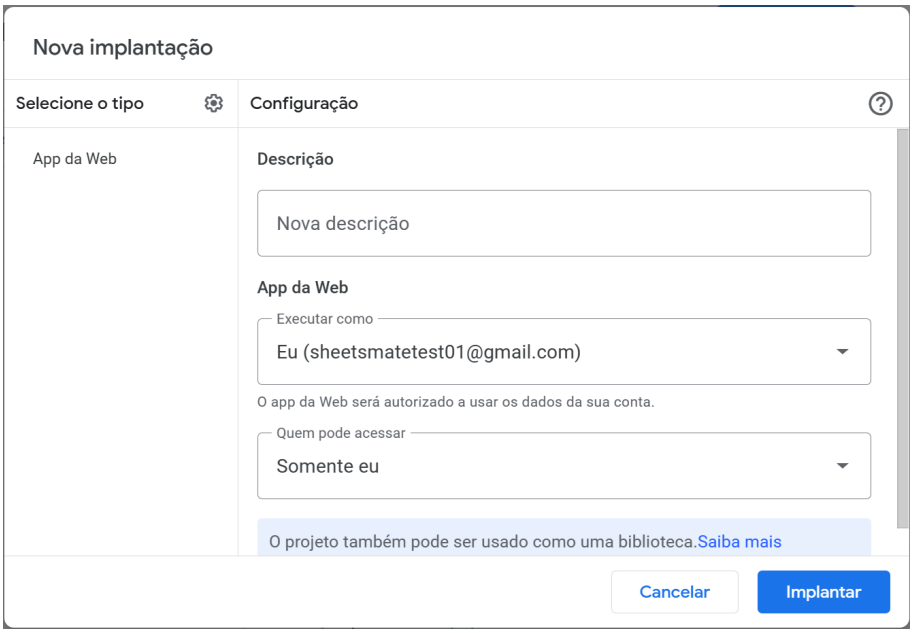
Figura 17 – Página para edição e deploy de código

Fonte: Imagem do autor

Na sequência, deve ser realizada a configuração do *deploy*. A Figura 18 é apenas um exemplo, sendo que a configuração deve ser realizada conforme o objetivo da aplica-

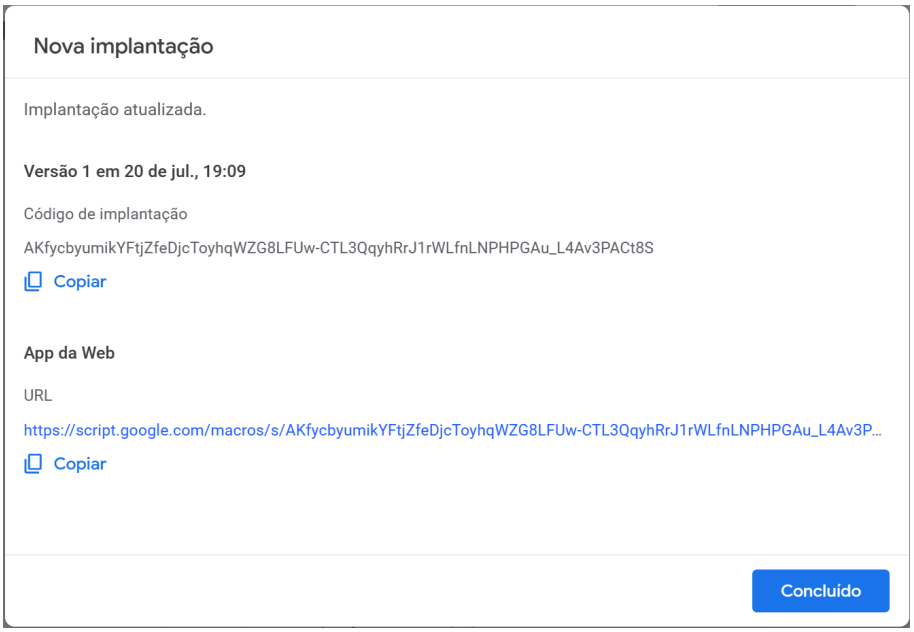
ção. Após a configuração, prosseguir com a implantação. Ocorrendo com sucesso, será apresentado um modal similar ao da Figura 19.

Figura 18 – Configuração do deploy



Fonte: Imagem do autor

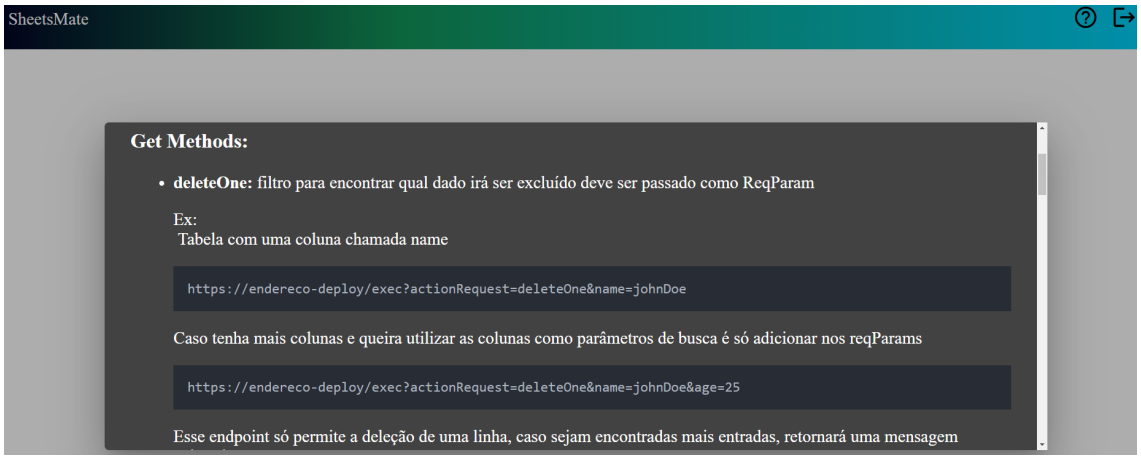
Figura 19 – Deploy realizado com sucesso



Fonte: Imagem do autor

Para auxiliar no entendimento da utilização do código, na página principal da ferramenta há uma seção para auxiliar na compreensão de como funcionam os *endpoints* gerados (Figura 20).

Figura 20 – Modal com informações de como utilizar os endpoints gerados



Fonte: Imagem do autor