

br.atsit.in

Crie uma API CRUD usando a API do Planilhas Google - BR Atsit

12-16 minutos

Como o nome sugere, a API do Planilhas Google permite conectar uma planilha existente, analisar seus dados disponíveis e passá-los para seu aplicativo da web. A mais recente é a versão 4.0, que fornece ao usuário controle sobre muitas outras propriedades-como acesso completo à formatação de células, incluindo a configuração de cores, estilos de texto e muito mais. Imagine ter um banco de dados totalmente gratuito com uma interface de usuário intuitiva que ajuda você a visualizar seus dados e organizá-los de acordo com suas necessidades enquanto atua como um CMS. Não é legal?

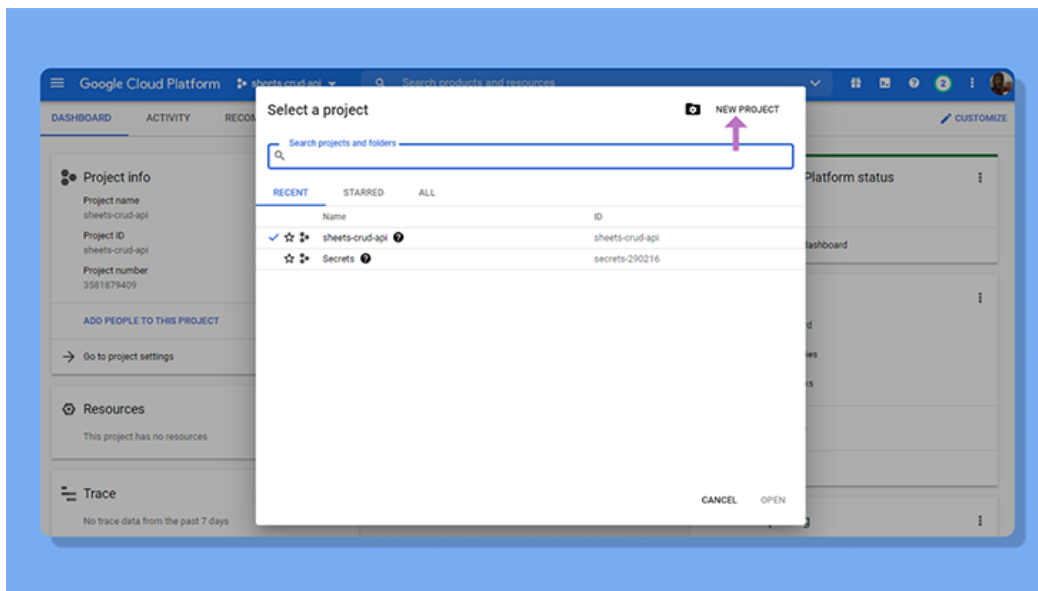
Sem mais delongas, vamos começar a criar sua própria API CRUD usando a API do Planilhas Google.

Aqui está o que vamos cobrir neste artigo:

[Configurando um novo projeto no Google Cloud Console](#)
[Vinculando a Planilha Google](#) [Configurando a API CRUD](#)
[Fazendo sua primeira solicitação](#) [Implementando operações CRUD](#)

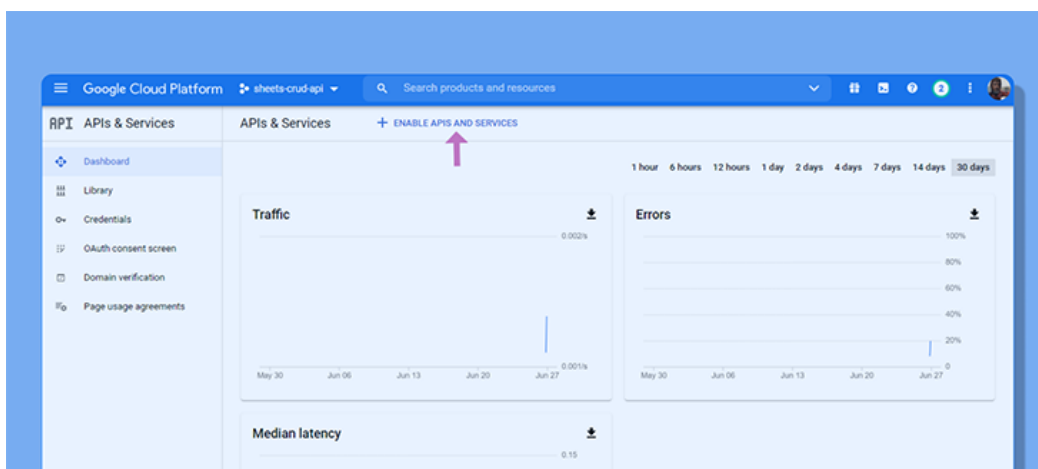
Configurando um novo projeto no Google Cloud Console

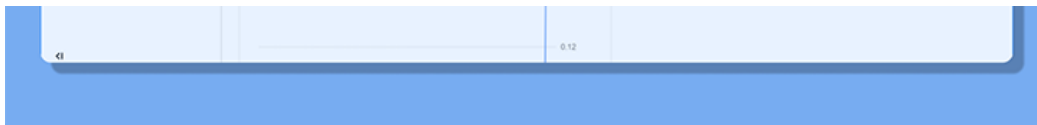
Assim como acontece com o acesso a qualquer serviço de API, devemos primeiro configurar a autenticação e autorização. Acesse [Google Cloud](https://cloud.google.com/) e inscreva-se se ainda não tiver criado uma conta. Em seguida, siga as etapas abaixo para fazer um novo projeto.



Clique em **Novo projeto** , dê a ele um nome adequado e clique em **Criar** .

Em seguida, clique no menu de navegação e vá para **APIs e Serviços** .

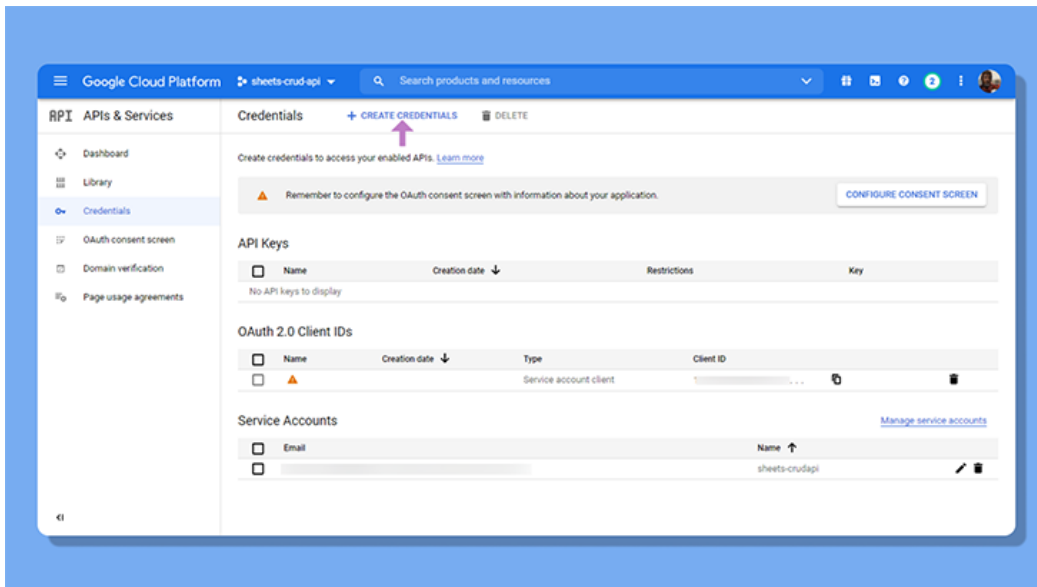




Você será redirecionado para a biblioteca API.

Pesquise a API do Planilhas Google e ative-a.

Volte para o painel de APIs e serviços. Vá para Credenciais e crie uma nova credencial.



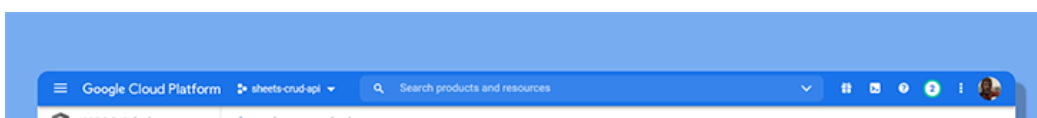
Clique em Conta de serviço . Dê um nome a ela, defina o resto como está e clique em Concluído .

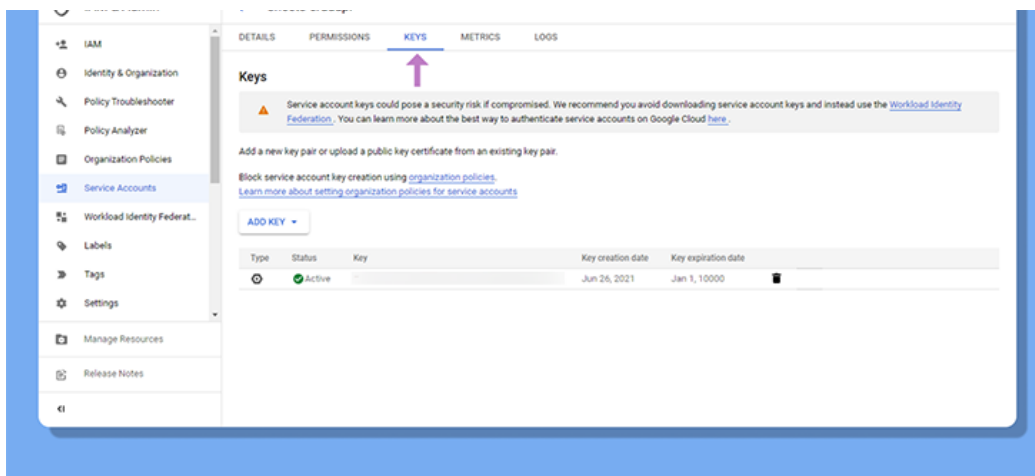
Você acabou de criar uma conta de bot para a planilha que tem permissão para ler e gravar operações no planilha.

Copie o e-mail de serviço que criamos. Isso será usado quando conectarmos a planilha ao projeto do Google Cloud.

Clique no e-mail da conta de serviço e vá para Chaves

▪





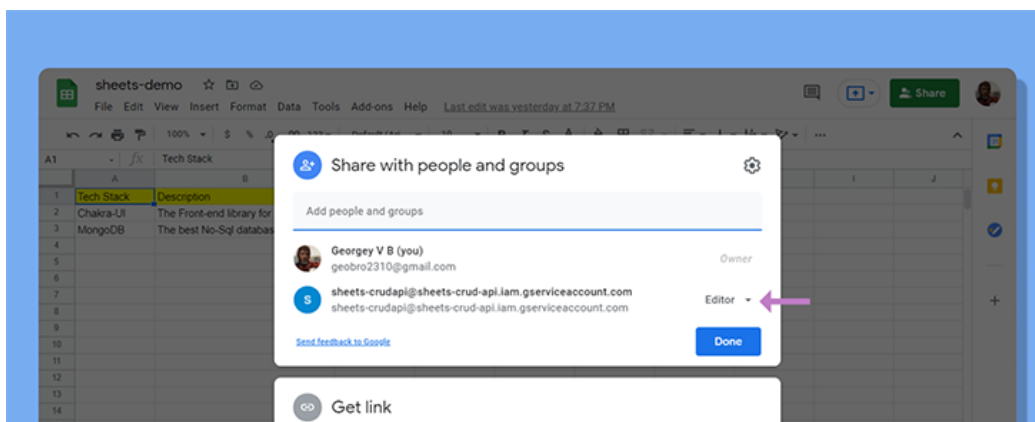
Vá em frente e crie uma nova chave, definindo o tipo de arquivo como JSON. Um arquivo será baixado em breve e, se possível, você deve movê-lo para a pasta onde espera configurar os arquivos iniciais.

Vinculando a planilha do Google

Agora, conectaremos nossa planilha para o projeto Google Cloud. Vá para o [Google Docs](#) e faça uma nova planilha. Nomeie a planilha.

Insira alguns dados fictícios para que tenhamos algo para buscar durante o teste da API.

Agora, vamos adicionar a conta de serviço e atribuir a ela a função de Editor, que concede permissões para ler, gravar, atualizar e excluir dados.





Clique em Compartilhar e adicione o e-mail de serviço que copiamos recentemente, certifique-se de fornecer o acesso de editor e desmarque Notificar pessoas.

Isso é tudo que você precisa fazer! Agora vamos ao editor de código e configurar os arquivos iniciais para a API.

Configurando a API CRUD

Estaremos usando alguns pacotes para a API: [Express](#), [dotenv](#) e [googleapis](#). Antes de fazer o download, vamos inicializar o npm usando o seguinte comando:

npm init-y

Agora instale os pacotes:

npm install express dotenv googleapis

Adicione o nodemon como uma dependência dev (isso irá certifique-se de que o servidor de desenvolvimento seja reiniciado sempre que fizermos qualquer alteração no código):

npm install nodemon--save-dev

Com tudo isso feito, crie um novo arquivo chamado index.js.

Iniciar exigindo dotenv e inicializando express.

require ('dotenv'). config (); const express=require

```
('express'); const app=express (); app.listen (3000 ||  
process.env.PORT, ()=> {console.log ('Instalado e  
funcionando !!');});
```

Crie um novo script no arquivo package.json:

```
"dev": "nodemon index.js"
```

E se tudo funcionar bem, o nodemon reiniciará o servidor toda vez que salvarmos o arquivo.

```
npm run dev
```

Fazendo sua primeira solicitação

Com tudo isso feito, vamos ver se nossa planilha está realmente vinculada ao projeto do Google Cloud.

Importe o seguinte do pacote googleapis:

```
const {google}=require ('googleapis');
```

Crie uma rota GET:

```
app.get ('/', async (req, res)=> {res.send ("Hello  
Google!");});
```

Crie um token de autenticação em seguida, consistindo em um keyFile que aponta para o arquivo credentials.json que baixamos e escopos que fornecem acesso completo para realizar operações de leitura e gravação.

```
const auth=new google.auth.GoogleAuth (  
{keyFile:'credentials.json',  
escopos:'https://www.googleapis.com  
/auth/spreadsheets'});
```

Scope	Meaning
https://www.googleapis.com/auth/spreadsheets.readonly	Allows read-only access to the user's sheets and their properties.
https://www.googleapis.com/auth/spreadsheets	Allows read/write access to the user's sheets and their properties.
https://www.googleapis.com/auth/drive.readonly	Allows read-only access to the user's file metadata and file content.
https://www.googleapis.com/auth/drive.file	Per-file access to files created or opened by the app.
https://www.googleapis.com/auth/drive	Full, permissive scope to access all of a user's files. Request this scope only when it is strictly necessary.

Fonte: [Planilhas Google para desenvolvedores](#) .

Você sempre pode consultar a [documentação oficial do Google Developers](#) para obter ajuda adicional com isso.

Em seguida, defina o cliente, a versão mais recente da API e o spreadsheetId.

```
const client=await auth.getClient (); const  
googleSheet=google.sheets ({version:'v4', auth:  
client}); const spreadsheetId=your_spreadsheetid
```

Obtenha o ID da planilha no URL da planilha do Google, da seguinte forma:

```
https://docs.google.com/spreadsheets  
/d/{_your_database_id_}/edit#gid=0
```

No exemplo acima, gid é o ID da planilha.

Você provavelmente deve armazenar essas informações confidenciais em um arquivo de ambiente. Crie um arquivo.env e armazene o ID da

planilha conforme mostrado:

SPREADSHEET_ID=your_spreadsheet_id

E, finalmente, indique a variável de ambiente:

const spreadsheetId=process.env.SPREADSHEET_ID

Com feito isso, vamos finalmente fazer uma solicitação!

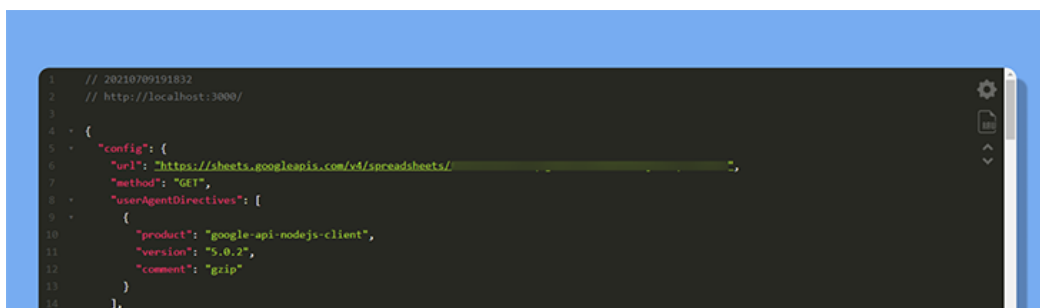
const getMetaData=await

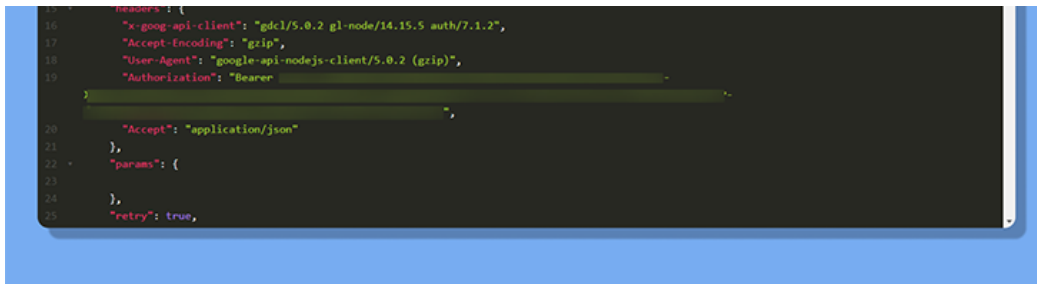
googleSheet.spreadsheets.get ({auth, spreadsheetId, range:'Sheet1! A: B'}); res.send (getMetaData);

Certifique-se de nomear as variáveis como fizemos acima, porque também é a abreviação para escrever auth: auth.

Cada chamada de API leva em dois parâmetros, que são auth e spreadsheetId. O intervalo define o intervalo de células a serem editadas. Se você não tiver certeza dos valores, pode sempre usar a interface da planilha. Usaremos quando se trata de ler os valores das células na próxima seção.

Por enquanto, vá em frente e faça uma solicitação GET para a URL raiz em localhost: 3000. Se você seguiu todas as etapas, receberá uma resposta longa da API.





Implementando operação CRUD

1. Leia os valores das células

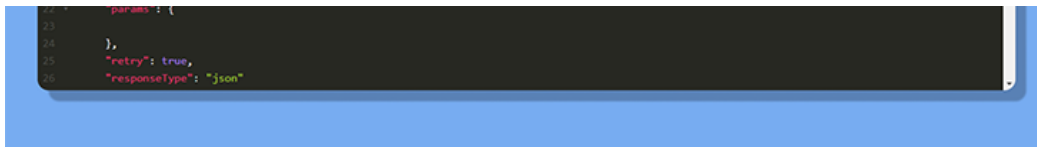
Por enquanto, comente a solicitação anterior e vamos realmente ler os valores das células que inserimos. Para ler os valores das células, usaremos o método `spreadsheets.values.get`.

```
const getSheetData=await  
googleSheet.spreadsheets.values.get ({auth,  
spreadsheetId, range:'Sheet1! A: B'}); res.send  
(getSheetData);
```

Como eu disse antes, o método sempre leva em `auth` e `spreadsheetId`. O parâmetro de intervalo define a área da célula para ler e escrever. Nesse caso, faremos alterações apenas nas duas primeiras colunas, A e B.

Vá em frente e faça uma solicitação GET.

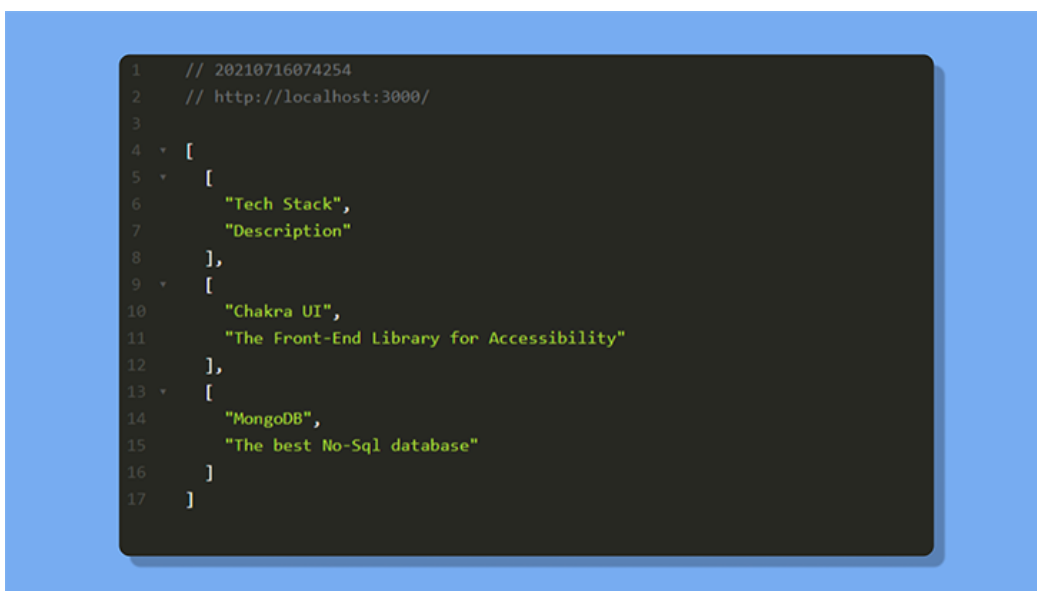




A resposta contém um monte de informações, incluindo os valores das células, a cor das células, a geolocalização e o fuso horário. Vamos direcionar os valores da célula aqui.

`res.send (getSheetData.data.values);`

A resposta parece muito mais concisa agora.



Observe que também estamos obtendo os cabeçalhos das colunas reais nesses resultados. Você pode omitir esses e enviar de volta apenas os valores das células abaixo da primeira linha.

Veja como podemos alterar o intervalo. Selecione a área que deseja incluir em sua resposta. A área selecionada é denotada por um intervalo. Em nosso exemplo, é da coluna A à coluna B.

Visto que precisamos incluir os valores das células

sob os títulos das colunas na linha um, podemos começar a selecionar a partir da linha dois. Portanto, o novo intervalo agora é Folha1! A2: B.

A resposta parece muito melhor agora!

```
1 // 20210716075503
2 // http://localhost:3000/
3
4 [
5   [
6     "Chakra UI",
7     "The Front-End Library for Accessibility"
8   ],
9   [
10    "MongoDB",
11    "The best No-Sql database"
12  ]
13 ]
```

2. Crie e poste dados

Feito isso, vamos prosseguir para postar dados na planilha.

Configure uma rota POST:

```
app.post('/post', assíncrono (req, res)=> {res.send("Dados enviados!");});
```

Siga o mesmo procedimento acima, configurando o token de autenticação e definindo o ID da planilha.

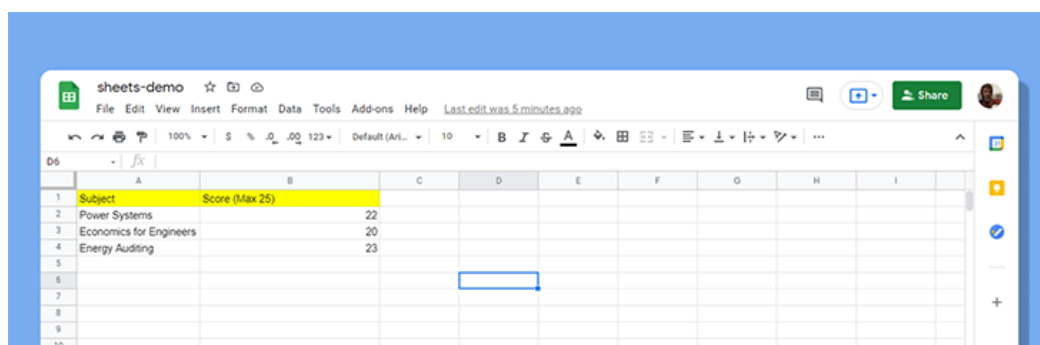
Para postar dados, usaremos o método `spreadsheets.values.append`. A API do Google anexará valores à planilha dependendo do número de valores passados na solicitação.

O método permanece o mesmo. Vamos passar `auth`, `spreadsheetId` e um intervalo. Junto com isso, passamos agora mais duas propriedades: `valueInputOption` e `resource`.

```
const response=await
googleSheet.spreadsheets.values.append ({auth,
spreadsheetId, range:'Sheet1! A2: B',
valueInputOption:'USER_ENTERED', recurso:
{valores: [['NextJS','A estrutura para produção']]]});
res.send (resposta)
```

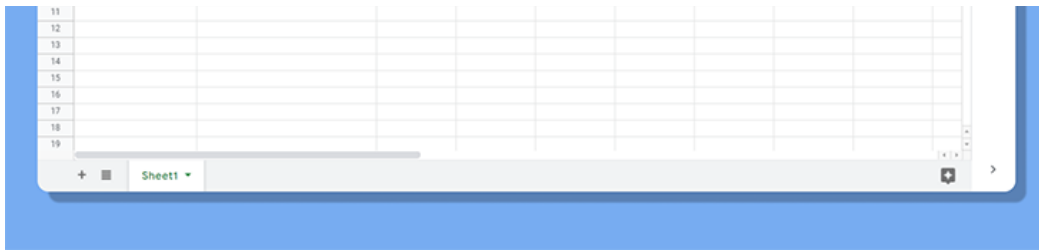
`valueInputOption` pode ter duas opções, `"RAW"` ou `"USER_ENTERED"`. Se for `"RAW"`, o que quer que o usuário tenha inserido será armazenado como está. Se você usar `"USER_ENTERED"`, a entrada do usuário sempre será analisada quando passada-se o usuário inserir um número, ele será analisado como um número.

Isso é realmente útil em certos casos de uso-para Por exemplo, digamos que você esteja construindo um formulário React que envia os dados enviados para uma planilha. Vou usar o exemplo de uma planilha simples com uma pontuação correspondente a cada assunto.

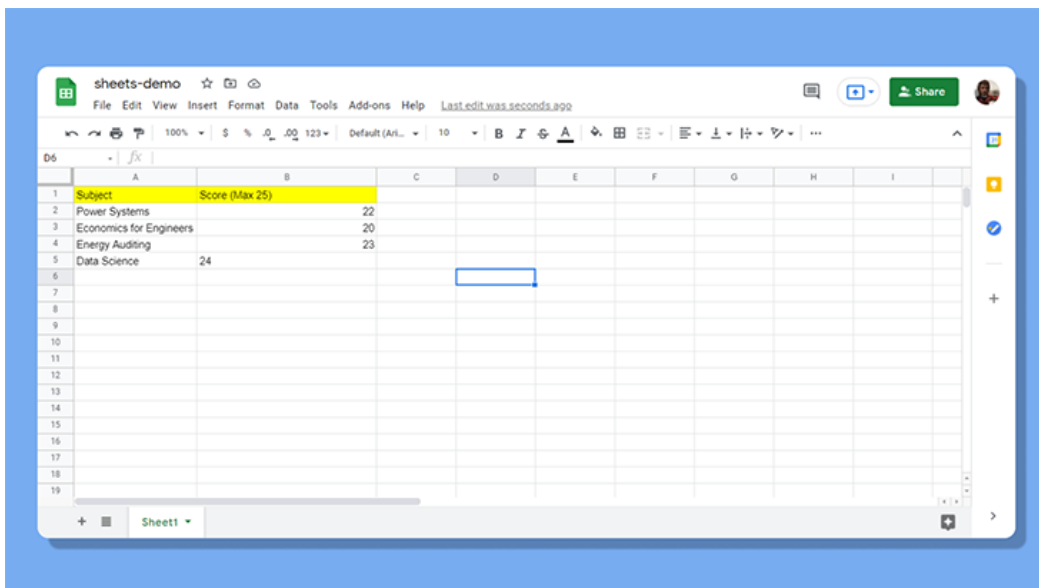


The screenshot shows a Google Sheet interface with a table containing the following data:

	A	B	C	D	E	F	G	H	I
1	Subject	Score (Max 25)							
2	Power Systems	22							
3	Economics for Engineers	20							
4	Energy Auditing	23							
5									
6									
7									
8									
9									
10									



Se `valueInputOption` for definido como `"USER_ENTERED"`, os dados serão postados e reconhecidos como um número. Mas se eu definir o parâmetro como `"RAW"` e passar a pontuação como uma string, os dados são postados, mas o Planilhas Google não parece tratar a pontuação como um número.



O recurso obtém os valores das células a serem adicionados à planilha. Você também pode inserir várias entradas adicionando outro conjunto de matrizes.

`resource: {values: [['NextJS', 'The framework for Production'], ['Jest', 'The testing framework for React']] }`

Vá em frente e faça uma solicitação POST. Você pode

usar qualquer testador de API como [Postman](#) para obter ajuda com isso.

3. Atualizar os valores das células

Para atualizar os valores das células, usaremos o método `spreadsheets.values.update`.

Vá em frente e faça uma rota PUT.

```
app.put('/update', async (req, res)=> {res.send  
("Célula atualizada!");});
```

O método leva `auth` e `spreadsheetId` normalmente. Certifique-se de que o intervalo aponte para uma única linha, a menos que você esteja atualizando várias linhas.

Aqui, irei especificar o intervalo: "Folha1! A2: B2", que é apenas a segunda linha. O resto permanece o mesmo. Você pode definir `valueInputOption` como "RAW" ou "USER_ENTERED". E, por fim, insira os valores da célula que deseja substituir por meio do recurso.

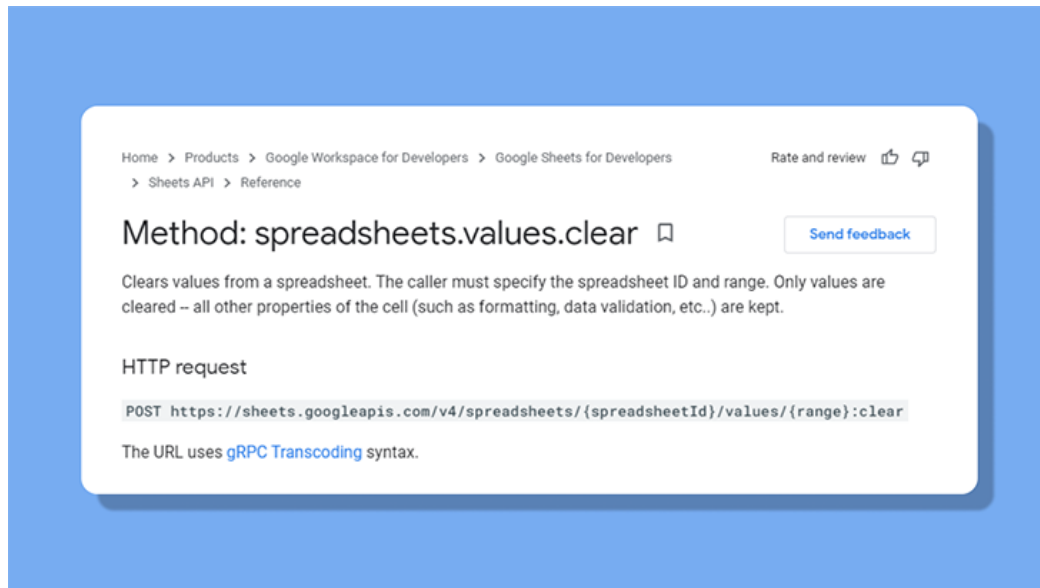
```
const response=await  
googleSheet.spreadsheets.values.update ({auth,  
spreadsheetId, range:'Sheet1! A2: B2',  
valueInputOption:'USER_ENTERED', recurso:  
{valores: [['Jamstack','Futuro da Web']]]}); res.send  
(resposta)
```

Vá em frente e faça uma solicitação PUT no testador de API. Os valores das células devem ser atualizados

agora.

4. Excluir valores de células

A API do Planilhas recomenda o uso de uma solicitação POST para usar o método `spreadsheets.values.clear`.



Fonte: [Documentação da API do Planilhas Google](#) .

Então, faremos uma nova rota POST.

```
app.post ('/delete', async (req, res)=> {res.send  
("Deleted Cell com sucesso!");});
```

Este método é bastante direto. Tudo o que você precisa fazer é especificar a linha e a coluna da planilha por meio da propriedade `range`.

```
const response=await  
googleSheet.spreadsheets.values.clear ({auth,  
spreadsheetId, range:"Sheet1! A5: B5"});
```

Faça uma nova solicitação à rota `/delete` para ver as

alterações.

Bem, parabéns! Isso é algo novo! Implementamos operações CRUD usando o Planilhas Google. Se você tiver alguma dúvida, pode dar uma olhada em [este repositório em meu GitHub](#) .

Conclusão

Parece que descobrimos um banco de dados totalmente novo que é gratuito e tem uma interface limpa para gerenciamento de dados. Você pode usá-lo com uma variedade de linguagens, incluindo Python, Go, Ruby e muito mais.

Embora haja um limite no número de solicitações que você pode fazer-100 solicitações por 100 segundos-se você veja o lado bom, a API do Google Sheets oferece uma ótima maneira de começar a aprender APIs e pode ser usada para integrar projetos de pequena escala.