Biblioteca Virtual

[ctcbibliotecavirtual@gmail.com](mailto:ctcbibliotecavirtual@gmail.com)  
0409@Cetec

TOKEN\_SECRETO = "Cmm41170297140108";

O projeto consiste em um Acervo Virtual onde serão disponibilizados arquivos em PDF para os alunos do CETEC consultar e estudar.

**Método de acesso**

Todo curso na plataforma CGD terá uma lição com o nome biblioteca. Dentro desta lição será carregada uma pagina do acervo Biblioteca Virtual.

Nela será possível fazer filtros e busca de Títulos.

**Arquitetura**

**FrontEnd**

Exibir uma página com campo de busca e um filtro por área.

Os itens/arquivos devem ser responsivos e exibidos em forma de ícones

Criar paginação

Criar comportamento dos ícones

O ícone deve ter a capa do pdf, Título, autor, Editora, Ano e categoria, botão para descrição flutuante.  
ao clicar no item deve ir para a pagina de descrição onde vai aparecer a capa do pdf e resumo

**BackEnd**

Deve existir uma conta no drive para armazenar os arquivos.

Uma planilha que deve conter os campos:  
Título Autor Editora Ano Link Categoria Descrição Download

Link da planilha

<https://docs.google.com/spreadsheets/d/14jAz1NfDyb8KRloA5L5it3POwG6rSHqoIrc262NQrGs/edit?usp=sharing>

quero começar um projeto com firebase. terão dois tipos de acesso, o publico e acesso adm.

o projeto é uma biblioteca virtual. onde serão mostrados icones com as capas de livros, título e autor e um botao de descrição. esse botão de descrição irá direcionar para a pagina de descrição. onde terão todas as informações do livro. e um botão "Ler". a pagina de leitura deve ser o mais clean possível.

a pagina de adm, terá o mesmo comportamento, mas deve ter uma pagina onde mostra a lista dos livros cadastrados e um botão para incluir. ao clicar em um item da lista deve ser direcionado à pagina de descrição desse livro. nela , alem de conter a pagina normal deve haver botoes para editar e excluir. ao clicar em excluir deve aparecer um alert na tela.

os arquivos devem ser guardado no servidor de modo não acessivel ao publico diretamente. somente o site pode fazer requisições.

**✅ VISÃO GERAL DO FUNCIONAMENTO**

* 🔒 Planilha 1: Armazena os **dados visíveis** da obra (título, autor, categoria etc.) e um **token aleatório**.
* 🔐 Planilha 2: Armazena os **links reais dos PDFs**, **associados ao token**.
* 🧠 A lógica do front-end usa o token para buscar os dados e o link (via Google Apps Script ou fetch() de CSV).
* 📄 O link real nunca fica visível diretamente no HTML da lista.

**🪜 PASSO A PASSO COMPLETO**

**🔧 1. Criar as duas planilhas no Google Sheets**

**📄 Planilha 1: biblioteca\_dados**

Colunas:

* Título
* Autor
* Ano
* Categoria
* Descrição
* Token ✅ (coluna que liga com a outra planilha)

**📄 Planilha 2: biblioteca\_links**

Colunas:

* Token
* LinkDoPDF (pode ser Drive, GitHub, S3, etc.)

**✳️ 2. Gerar os tokens aleatórios**

Você pode gerar os tokens de duas formas:

**✔️ Manualmente:**

Use esta fórmula no Google Sheets:

excel

CopiarEditar

=JOIN("", ARRAYFORMULA(CHAR(RANDBETWEEN(65, 90) + RANDBETWEEN(0, 1)\*32)))

**✔️ Com Google Apps Script:**

javascript

CopiarEditar

function gerarTokens() {

const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("biblioteca\_dados");

const dados = sheet.getDataRange().getValues();

for (let i = 1; i < dados.length; i++) {

if (!dados[i][5]) { // Coluna 6 = Token

const token = [...Array(10)].map(() => Math.random().toString(36)[2]).join('');

sheet.getRange(i + 1, 6).setValue(token);

}

}

}

**🌐 3. Tornar as planilhas públicas para leitura**

* Acesse Arquivo > Compartilhar > Publicar na web
* Copie os **links de publicação em CSV**

Exemplo:

plaintext

CopiarEditar

https://docs.google.com/spreadsheets/d/e/.../pub?output=csv

**🧠 4. (Opcional) Criar script do Google Apps Script para consulta segura**

Isso permite que, a partir de um token, você recupere **dados + link** num JSON.

**Exemplo:**

javascript

CopiarEditar

function doGet(e) {

const token = e.parameter.token;

const sheet1 = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("biblioteca\_dados");

const sheet2 = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("biblioteca\_links");

const dados = sheet1.getDataRange().getValues();

const links = sheet2.getDataRange().getValues();

const livro = dados.find(l => l[5] === token); // Coluna Token

const link = links.find(l => l[0] === token); // Token

if (!livro || !link) return ContentService.createTextOutput(JSON.stringify({ erro: "Token inválido" }));

const resultado = {

titulo: livro[0],

autor: livro[1],

ano: livro[2],

categoria: livro[3],

descricao: livro[4],

link: link[1]

};

return ContentService

.createTextOutput(JSON.stringify(resultado))

.setMimeType(ContentService.MimeType.JSON);

}

**🖼️ 5. No front-end (HTML/JS)**

**Página da lista:**

* Busca os dados da planilha 1 (sem links!)
* Exibe os cards com botão/link:

js

CopiarEditar

window.location.href = `descricao.html?token=${token}`

**Página descricao.html:**

* Ao carregar, lê o token da URL
* Faz uma requisição para o Apps Script ou para a planilha de links
* Mostra o livro + botão de leitura

**🔒 6. Benefícios do modelo**

* 🔐 Link do PDF real **não fica exposto na lista**
* 🔁 Você pode **revogar ou atualizar links** apenas trocando a planilha 2
* 🧼 Organização limpa e modular
* 🚀 Fácil de escalar e manter

### 📌 ****Resumo final do impacto****

| **Arquivo** | **Muda?** | **O que muda** |
| --- | --- | --- |
| index.html | ❌ | Nada — estrutura e UI mantidas |
| dados.js | ✅ | Traz Token ao invés de Link |
| card.js | ✅ | Botão Ver mais redireciona com ?token=... |
| filtro.js | ❌ | Continua igual |
| paginacao.js | ❌ | Continua igual |
| visualizar.js | ✅ | Nova lógica: buscar dados + link pelo token |
| descricao.html | ✅ | Nova estrutura com base no token |
| utils.js | ❌ | Continua igual |