

**Projeto Engenharia de Software POP**  
**Desenvolvimento de um Script Java para Geração de Páginas Markdown**

Erick Augusto Warmling

Ibirama, 2025

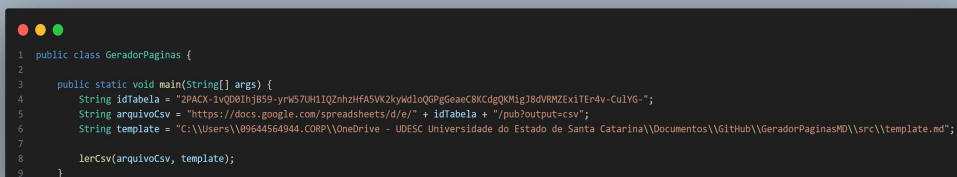
## GERADOR DE PÁGINAS MARKDOWN

O Gerador de Páginas Markdown foi desenvolvido em Java na bolsa de extensão Engenharia de Software POP, com o objetivo de otimizar a criação das páginas de apresentação dos Trabalhos de Conclusão de Curso (TCCs) no site do Departamento de Engenharia de Software da Universidade do Estado de Santa Catarina (UDESC). O programa lê um arquivo CSV hospedado no Google Sheets e, a partir de um template pré-definido, gera automaticamente os arquivos Markdown (.md), preenchendo com as informações completas de cada TCC, como aluno, título, orientador, membros da banca examinadora, área, semestre, data, hora, local da defesa e o link do Repositório Institucional da UDESC.

## MÉTODOS

### 1. main (String[] args)

Figura 1 - Método main



```
1 public class GeradorPaginas {
2
3     public static void main(String[] args) {
4         String idTabela = "2PACX-1vQD0lhj859-yrW57Uhl1Q2nhzHFASVK2kyWd1oG6Pg6eacC8KcdgQKHig38dVRMZE1Er4v-Cu1YG-";
5         String arquivoCsv = "https://docs.google.com/spreadsheets/d/e/" + idTabela + "/pub?output=csv";
6         String template = "C:\\Users\\b9644564944.CORP\\OneDrive - UDESC Universidade do Estado de Santa Catarina\\Documentos\\GitHub\\GeradorPaginasMD\\src\\template.md";
7
8         lerCsv(arquivoCsv, template);
9     }
10 }
```

Fonte: O autor, 2025.

## Descrição

- É o método principal que irá executar o fluxo do programa, ele chama o método lerCsv para realizar a leitura dos dados da planilha e gerar os arquivos Markdown (.md).

## Fluxo

- Define o idTabela e o arquivoCsv para montar a URL da planilha;
- Define o caminho do template Markdown.
- Chama o método lerCsv, passando como parâmetro o arquivoCsv e o template.

## 2. lerCsv (String arquivoCsv, String template)

Figura 2- Método lerCsv

```
1 private static void lerCsv (String arquivoCsv, String template) {
2     try (BufferedReader br = new BufferedReader(new InputStreamReader(new URL(arquivoCsv).openStream()))){
3         br.readLine();
4         String linhaTcc;
5
6         while ((linhaTcc = br.readLine()) != null) {
7             String[] dadosPlanilha = linhaTcc.split(",(?=[^\"']*\"|\"'\"]*(\"'\"]|\"'\"]*))", -1);
8
9             for (int i = 0; i < dadosPlanilha.length; i++) {
10                 dadosPlanilha[i] = dadosPlanilha[i].replaceAll("\"\"\"", "\"").trim();
11             }
12
13             String nomeArquivo = gerarNomeArquivo(dadosPlanilha[5], dadosPlanilha[0]);
14             String templateArquivo = gerarArquivoTemplate(template, dadosPlanilha);
15
16             salvarPaginaMd(nomeArquivo, templateArquivo);
17         }
18     } catch (FileNotFoundException e) {
19         System.err.println("Arquivo CSV não foi encontrado: " + arquivoCsv);
20         e.printStackTrace();
21     } catch (IOException e) {
22         System.err.println("Erro ao ler o arquivo CSV: " + e.getMessage());
23         e.printStackTrace();
24     }
25 }
26 }
```

Fonte: O Autor, 2025.

### Descrição

- O método lerCsv é responsável por ler o arquivo CSV hospedado na URL fornecida na variável arquivoCsv. Ele processa cada linha da planilha e para cada uma delas, gera um arquivo Markdown (.md) de acordo com o template pré-definido. Durante o processo, os dados da planilha são extraídos e formatados, criando uma página de apresentação do TCC, com as informações cadastradas na planilha.

### Fluxo

- Recebe como parâmetro o caminho do arquivoCsv e do template;
- Utiliza o BufferedReader para a leitura do arquivo CSV, ignorando a primeira linha da planilha (cabeçalho);
- Para cada linha da planilha, processa os dados, realiza a formatação e preenche no template;
- O arquivo será salvo com o nome adequado;
- Em casos de erro, uma mensagem será exibida no console.

### 3. gerarNomeArquivo (String semestre, String aluno)

Figura 3- Método gerarNomeArquivo

```
1 private static String gerarNomeArquivo (String semestre, String aluno) {  
2     String semestreFormatado = semestre.replace("/", "-").replace(" ", "-");  
3     String nomeFormatado = Normalizer.normalize(aluno.toLowerCase(), Normalizer.Form.NFD)  
4         .replaceAll("[^a-z0-9\\s-]", "")  
5         .trim()  
6         .replace(" ", "-");  
7  
8     return semestreFormatado + "-" + nomeFormatado + ".md";  
9 }
```

Fonte: O Autor, 2025.

#### Descrição

- O método gerarNomeArquivo é responsável por formatar o nome do arquivo Markdown (.md) a ser gerado, utilizando como parâmetros o **semestre** e **aluno**, de forma a criar um título padronizado.

#### Fluxo

- O semestre é formatado para substituir barras por hífens;
- O nome do aluno é transformado em minúsculo, removendo aspas e caracteres especiais, colocando hífens no lugar dos espaços.
- O arquivo será a combinação do semestre e do nome do aluno, com a extensão Markdown (.md) no final.

#### Exemplo

Com base nos valores fornecidos abaixo:

- semestre = "2024/2";
- aluno = "Fernando dos Santos";

O método **gerarNomeArquivo()** retornará:

*2024-2-fernando-dos-santos.md*

#### 4. gerarArquivoTemplate (String template, String [ ] dadosPlanilha)

Figura 4 - Método gerarArquivoTemplate

```
1 private static String gerarArquivoTemplate (String template, String[] dadosPlanilha) {
2     StringBuilder sb = new StringBuilder();
3
4     try (BufferedReader bufferedReader = new BufferedReader(new FileReader(template))) {
5         String linhaTemplate;
6
7         while ((linhaTemplate = bufferedReader.readLine()) != null) {
8             linhaTemplate = linhaTemplate.replace("$titulo$", dadosPlanilha[1].trim())
9                                     .replace("$nome$", dadosPlanilha[0].trim())
10                                    .replace("$area$", dadosPlanilha[4].trim())
11                                    .replace("$data$", dadosPlanilha[6].trim())
12                                    .replace("$hora$", dadosPlanilha[7].trim())
13                                    .replace("$local$", dadosPlanilha[8].trim())
14                                    .replace("$semestre$", dadosPlanilha[5].trim())
15                                    .replace("$orientador$", dadosPlanilha[2].trim())
16                                    .replace("$membros$", dadosPlanilha[3].trim())
17                                    .replace("$linkTcc$", dadosPlanilha[9].trim());
18
19             sb.append(linhaTemplate).append("\n");
20         }
21     } catch (FileNotFoundException e) {
22         System.err.println("O template não foi encontrado: " + template);
23         e.printStackTrace();
24     } catch (IOException e) {
25         System.err.println("Erro ao ler o template: " + e.getMessage());
26         e.printStackTrace();
27     }
28
29     return sb.toString();
30 }
```

Fonte: O Autor, 2025.

#### Descrição

- O método gerarArquivoTemplate é responsável por ler o arquivo template fornecida pela variável template, identificar os marcadores pré-definidos (\$titulo\$, \$nome\$, \$area\$, etc) e substituí-los pelos valores correspondentes armazenados no array **dadosPlanilha**.

#### Fluxo

- Recebe como parâmetro o caminho do arquivo template e o array dadosPlanilha[ ];
- Utiliza o BufferedReader para a leitura do arquivo de template (template.md) linha por linha;
- Para cada linha do template, o método procura o marcador e os substitui pelos valores correspondentes armazenados no array;

- As linhas do template com as informações substituídas são armazenadas em uma StringBuilder para formar o conteúdo das páginas a serem geradas;
- Em caso de erro, uma mensagem será exibida no console;
- O método retorna uma string contendo o conteúdo completo da página, com todas as substituições realizadas.

## Exemplo

Com base nos valores do array abaixo e o no template Markdown pré-definido:

```
dadosPlanilha = ["Fernando dos Santos", "DESENVOLVIMENTO DE UM SISTEMA  
BANCÁRIO", "Maria Cristina de Souza", "José da Silva e Carlos Pereira",  
"Engenharia de Software", "2024/2", "06/12/2024", "16h", "CEAVI - Sala  
208 (Bloco Imbuia)", "https://repositorio.udesc.br/handle/UDESC/11111"];
```

Template Markdown pré-definido:

```
## $titulo$

#### Aluno(a): $nome$

#### Área: $area$

#### Data: $data$

#### Hora: $hora$

#### Local: $local$

#### Semestre: $semestre$

#### Banca Examinadora:

#### Orientador(a): $orientador$

#### Membros: $membros$

#### Link Repositório Institucional UDESC: $linkTcc$
```

Após executar o método gerarArquivoTemplate(), o conteúdo do template será visualizado conforme exemplo a seguir:

**DESENVOLVIMENTO DE UM SISTEMA BANCÁRIO**

Aluno(a): Fernando dos Santos

Área: Engenharia de Software

Data: 06/12/2024

Hora: 16h

Local: CEA VI - Sala 208 (Bloco Imbuia)

Semestre: 2024/2

Banca Examinadora:

Orientador(a): Maria Cristina de Souza

Membros: José da Silva e Carlos Pereira

Link Repositório Institucional UDESC:  
<https://repositorio.udesc.br/handle/UDESC/11111>**5. salvarPaginaMd (String nomeArquivo, String conteudoTemplate)****Figura 5 - Método salvarPaginaMd**

```
1 private static void salvarPaginaMd (String nomeArquivo, String conteudoTemplate) {  
2     try (BufferedWriter bw = new BufferedWriter(new FileWriter(nomeArquivo))) {  
3         bw.write(conteudoTemplate);  
4         System.out.println("Página criada com sucesso: " + nomeArquivo);  
5     } catch (IOException e) {  
6         System.out.println("Erro ao salvar a página: " + nomeArquivo);  
7         e.printStackTrace();  
8     }  
9 }
```

Fonte: O Autor, 2025.

## Descrição

- O método `salvarPaginaMd` é responsável por salvar o conteúdo de uma página no formato Markdown (.md), bem como escrever o conteúdo da página no arquivo com o nome fornecido.

## Fluxo

- Recebe como parâmetro o `nomeArquivo` e o `conteudoTemplate`;
- O arquivo com o nome fornecido para a escrita é aberto.
- O `BufferedWriter` é utilizado para escrever o conteúdo do template;
- Em caso de sucesso, uma mensagem de confirmação é exibida no console;
- Caso contrário, uma mensagem de erro será exibida.