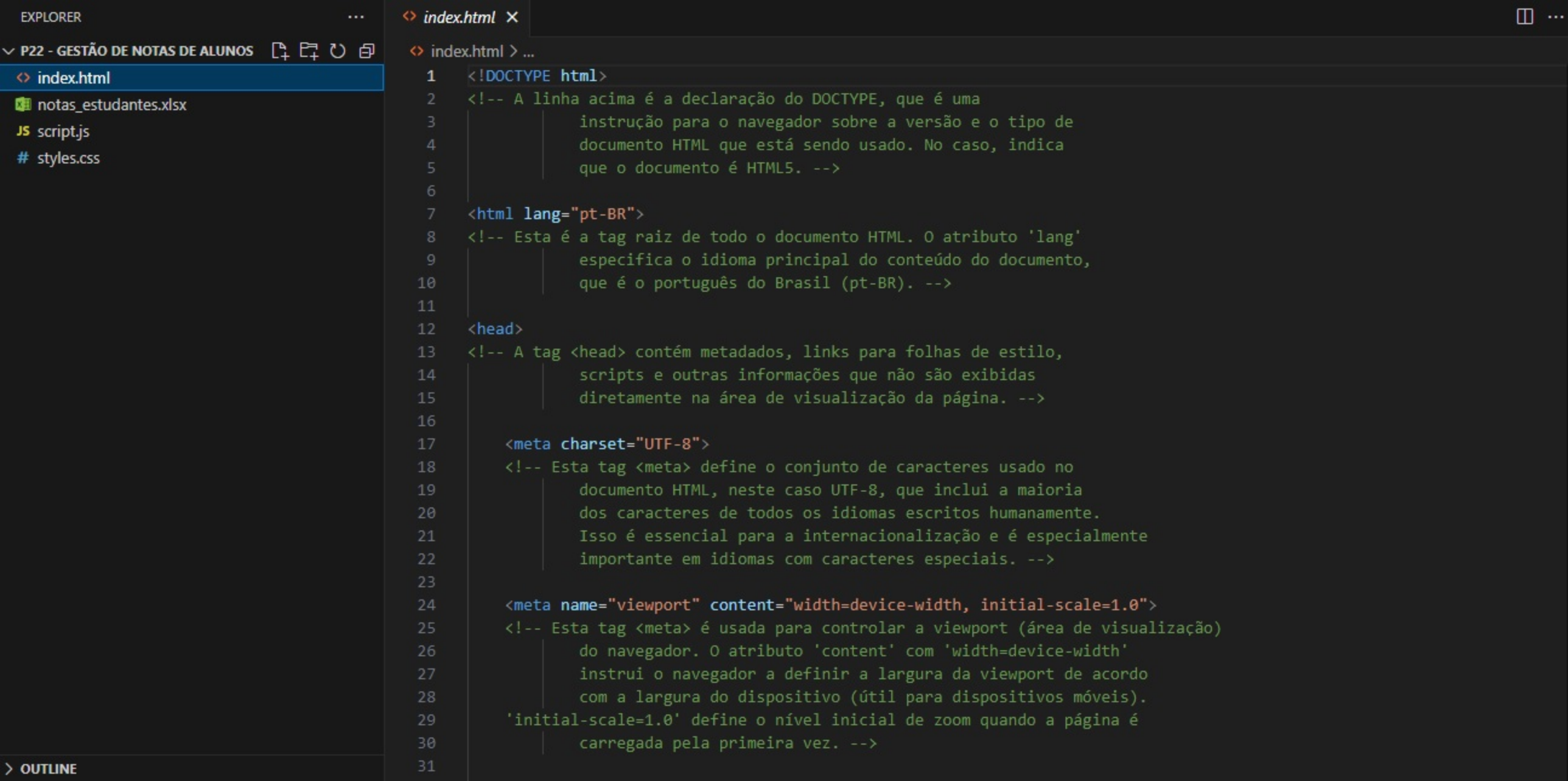
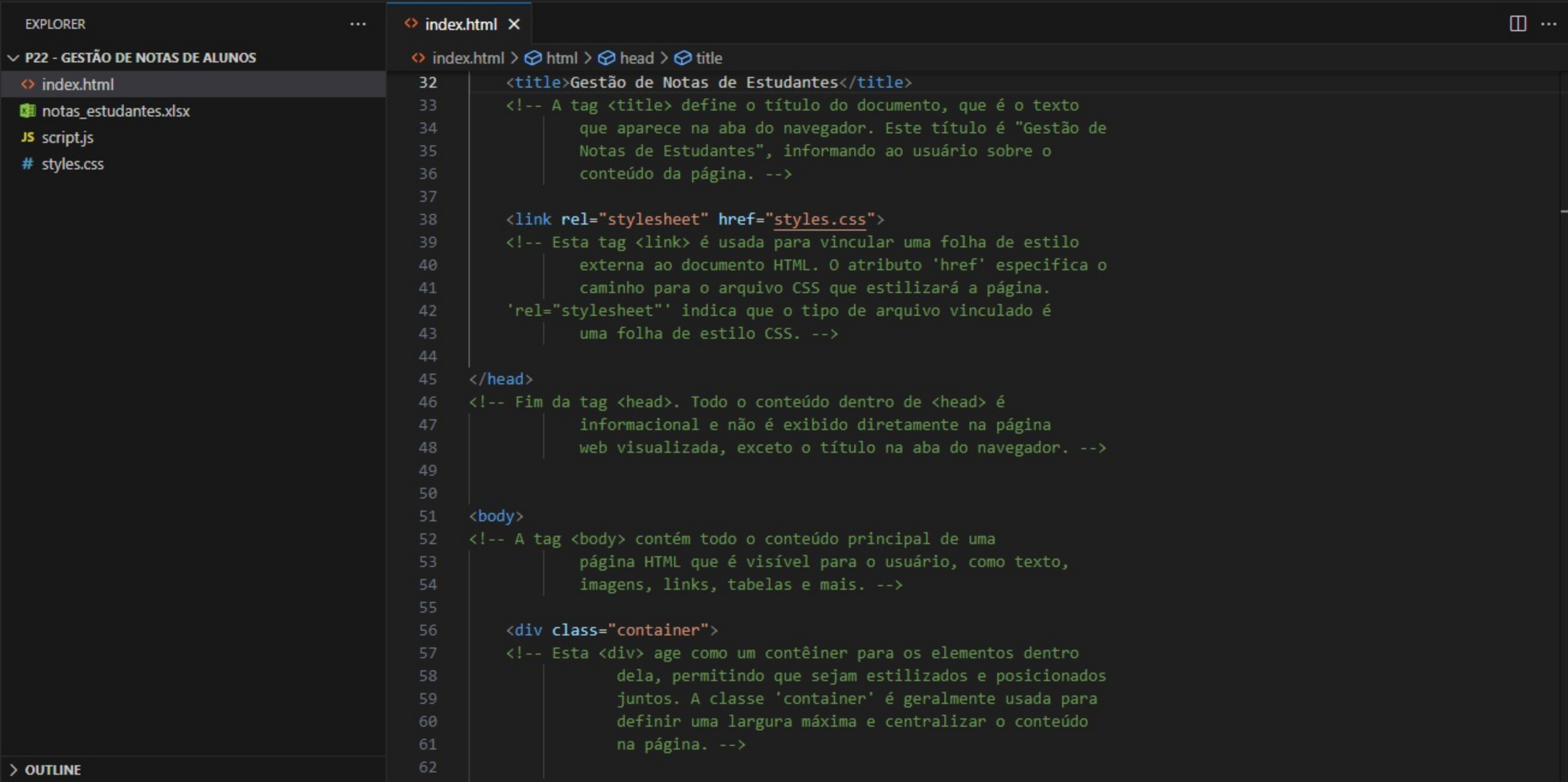
UC: Projetos - FrontEnd

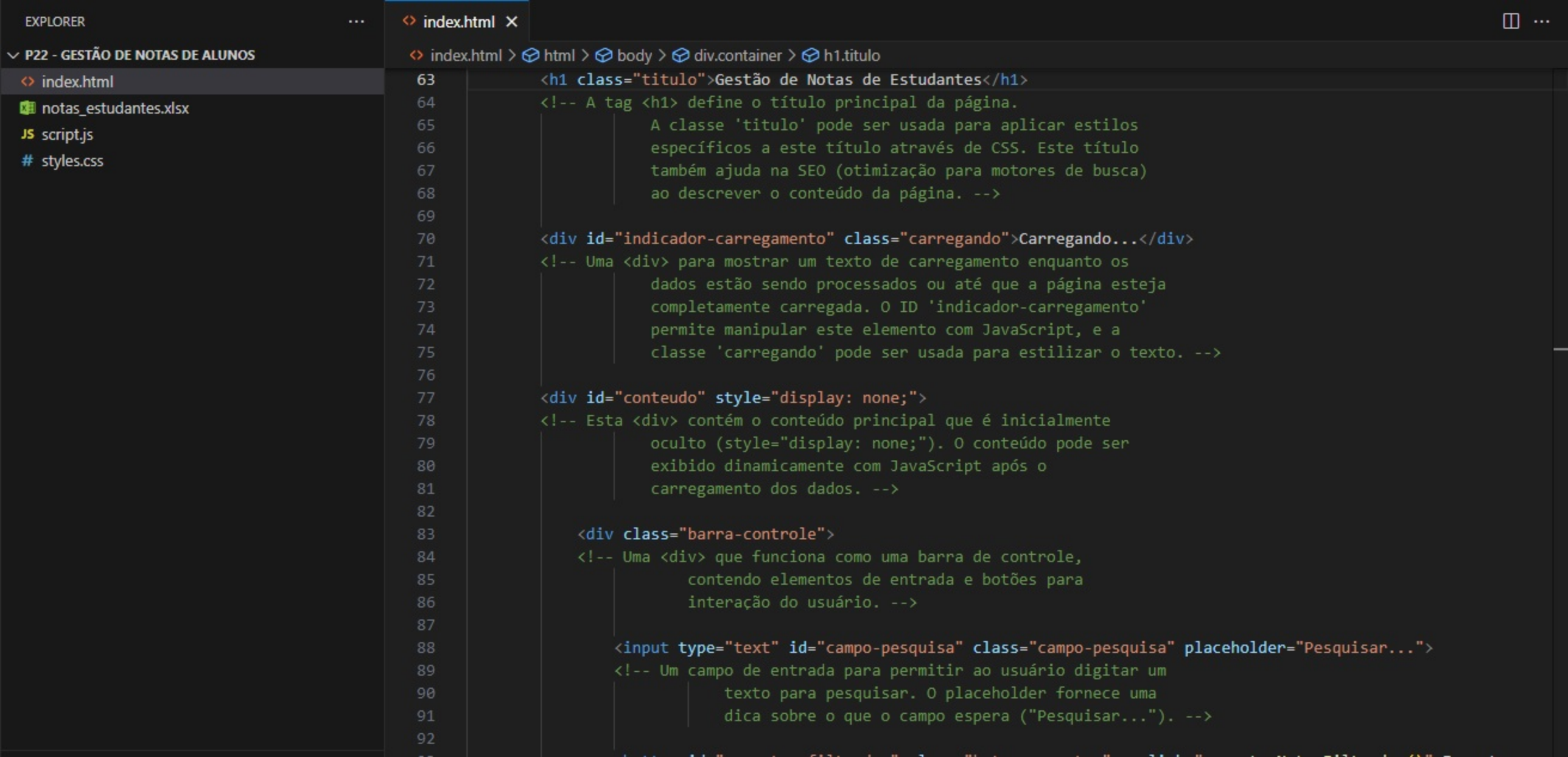
DIAS: 17 e 18/10/2024 – Parte 2

Assunto:

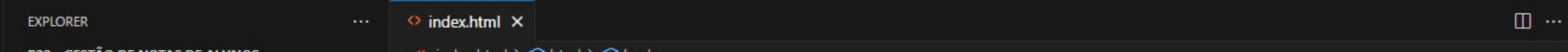
Projeto 22 – Gestão de Notas de Alunos

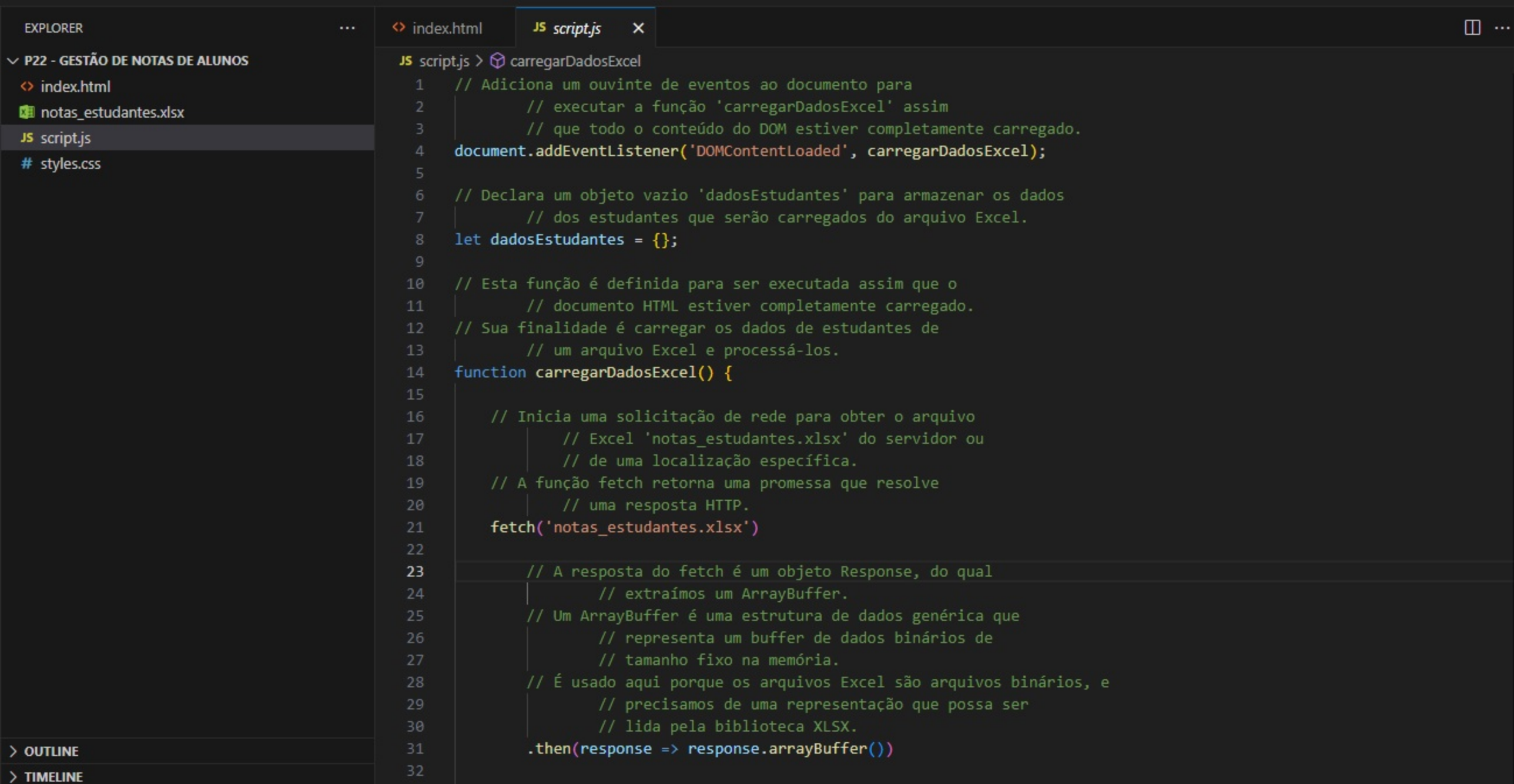

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
OUTLINE
index.html
index.html > ...
1 <!DOCTYPE html>
2 <!-- A linha acima é a declaração do DOCTYPE, que é uma
3 instrução para o navegador sobre a versão e o tipo de
4 documento HTML que está sendo usado. No caso, indica
5 que o documento é HTML5. -->
6
7 <html lang="pt-BR">
8 <!-- Esta é a tag raiz de todo o documento HTML. O atributo 'lang'
9 especifica o idioma principal do conteúdo do documento,
10 que é o português do Brasil (pt-BR). -->
11
12 <head>
13 <!-- A tag <head> contém metadados, links para folhas de estilo,
14 scripts e outras informações que não são exibidas
15 diretamente na área de visualização da página. -->
16
17 <meta charset="UTF-8">
18 <!-- Esta tag <meta> define o conjunto de caracteres usado no
19 documento HTML, neste caso UTF-8, que inclui a maioria
20 dos caracteres de todos os idiomas escritos humanamente.
21 Isso é essencial para a internacionalização e é especialmente
22 importante em idiomas com caracteres especiais. -->
23
24 <meta name="viewport" content="width=device-width, initial-scale=1.0">
25 <!-- Esta tag <meta> é usada para controlar a viewport (área de visualização)
26 do navegador. O atributo 'content' com 'width=device-width'
27 instrui o navegador a definir a largura da viewport de acordo
28 com a largura do dispositivo (útil para dispositivos móveis).
29 'initial-scale=1.0' define o nível inicial de zoom quando a página é
30 carregada pela primeira vez. -->
31

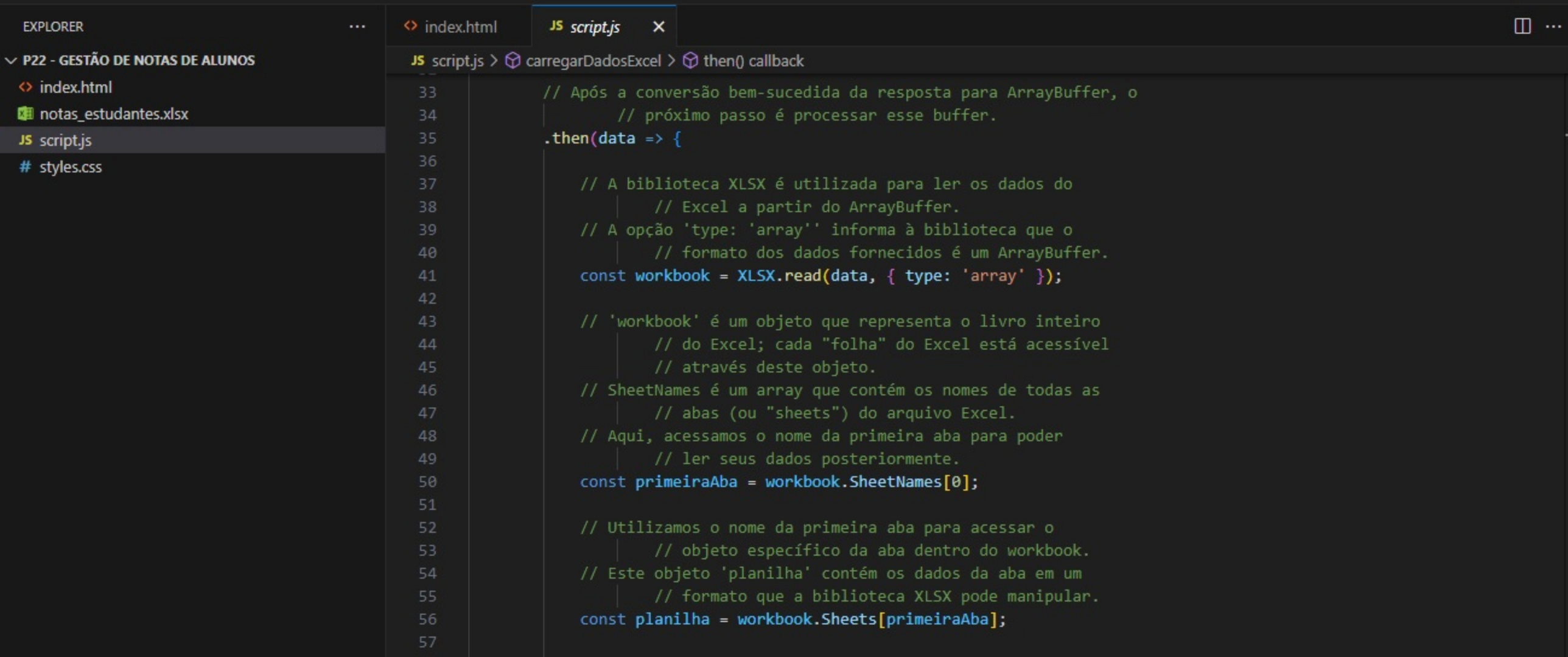

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
OUTLINE
index.html
index.html > html > head > title
32 <title>Gestão de Notas de Estudantes</title>
33 <!-- A tag <title> define o título do documento, que é o texto
34 que aparece na aba do navegador. Este título é "Gestão de
35 Notas de Estudantes", informando ao usuário sobre o
36 conteúdo da página. -->
37
38 <link rel="stylesheet" href="styles.css">
39 <!-- Esta tag <link> é usada para vincular uma folha de estilo
40 externa ao documento HTML. O atributo 'href' especifica o
41 caminho para o arquivo CSS que estilizará a página.
42 'rel="stylesheet"' indica que o tipo de arquivo vinculado é
43 uma folha de estilo CSS. -->
44
45 </head>
46 <!-- Fim da tag <head>. Todo o conteúdo dentro de <head> é
47 informacional e não é exibido diretamente na página
48 web visualizada, exceto o título na aba do navegador. -->
49
50
51 <body>
52 <!-- A tag <body> contém todo o conteúdo principal de uma
53 página HTML que é visível para o usuário, como texto,
54 imagens, links, tabelas e mais. -->
55
56
57 <!-- Esta age como um contêiner para os elementos dentro
58 dela, permitindo que sejam estilizados e posicionados
59 juntos. A classe 'container' é geralmente usada para
60 definir uma largura máxima e centralizar o conteúdo
61 na página. -->
62

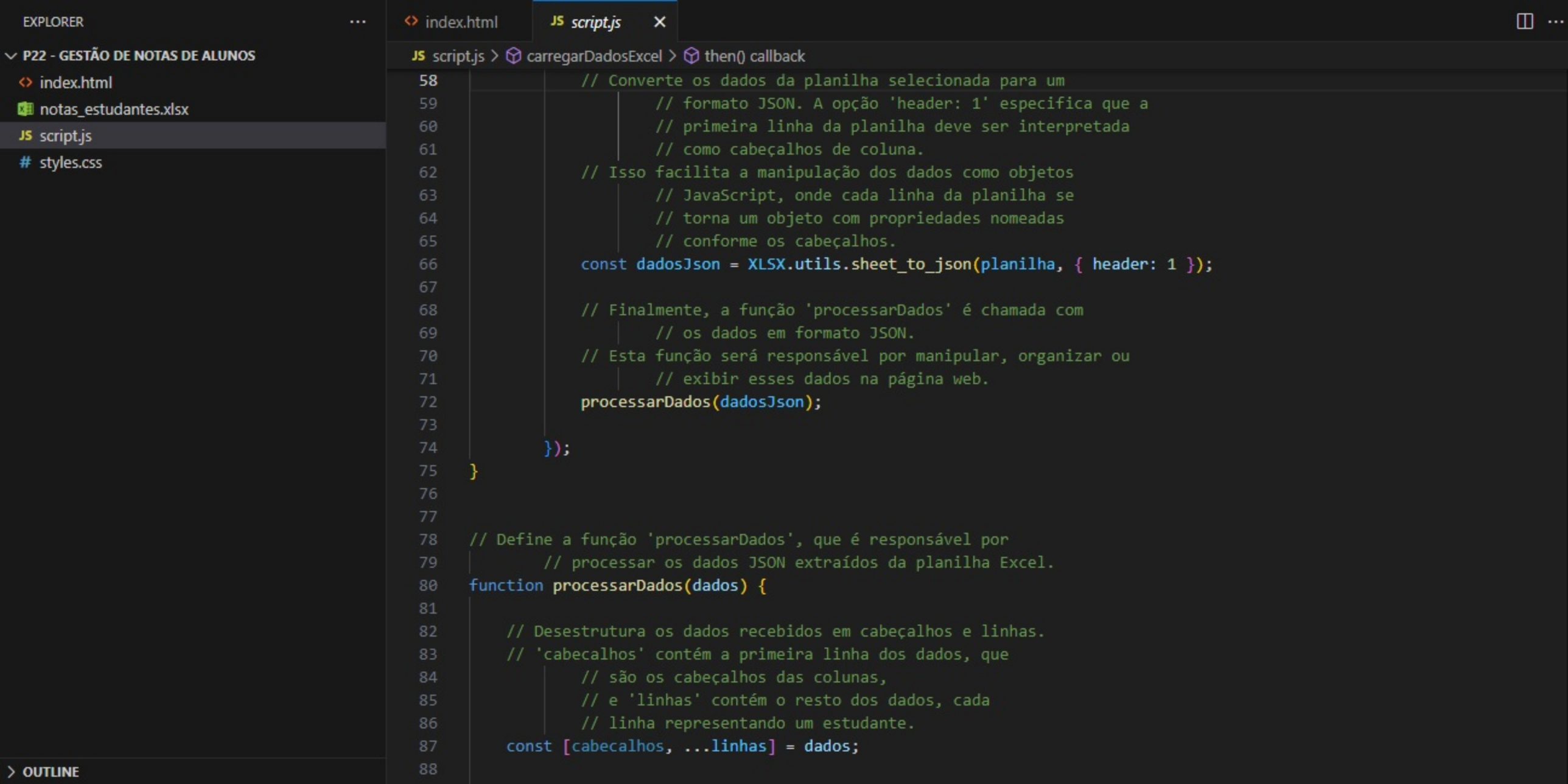

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
index.html
index.html > html > body > div.container > h1.titulo
63 <h1 class="titulo">Gestão de Notas de Estudantes</h1>
64 <!-- A tag <h1> define o título principal da página.
65 A classe 'titulo' pode ser usada para aplicar estilos
66 específicos a este título através de CSS. Este título
67 também ajuda na SEO (otimização para motores de busca)
68 ao descrever o conteúdo da página. -->
69
70 Carregando...
71 <!-- Uma para mostrar um texto de carregamento enquanto os
72 dados estão sendo processados ou até que a página esteja
73 completamente carregada. O ID 'indicador-carregamento'
74 permite manipular este elemento com JavaScript, e a
75 classe 'carregando' pode ser usada para estilizar o texto. -->
76
77
78 <!-- Esta contém o conteúdo principal que é inicialmente
79 oculto (style="display: none;"). O conteúdo pode ser
80 exibido dinamicamente com JavaScript após o
81 carregamento dos dados. -->
82
83
84 <!-- Uma que funciona como uma barra de controle,
85 contendo elementos de entrada e botões para
86 interação do usuário. -->
87
88 <input type="text" id="campo-pesquisa" class="campo-pesquisa" placeholder="Pesquisar...">
89 <!-- Um campo de entrada para permitir ao usuário digitar um
90 texto para pesquisar. O placeholder fornece uma
91 dica sobre o que o campo espera ("Pesquisar..."). -->
92

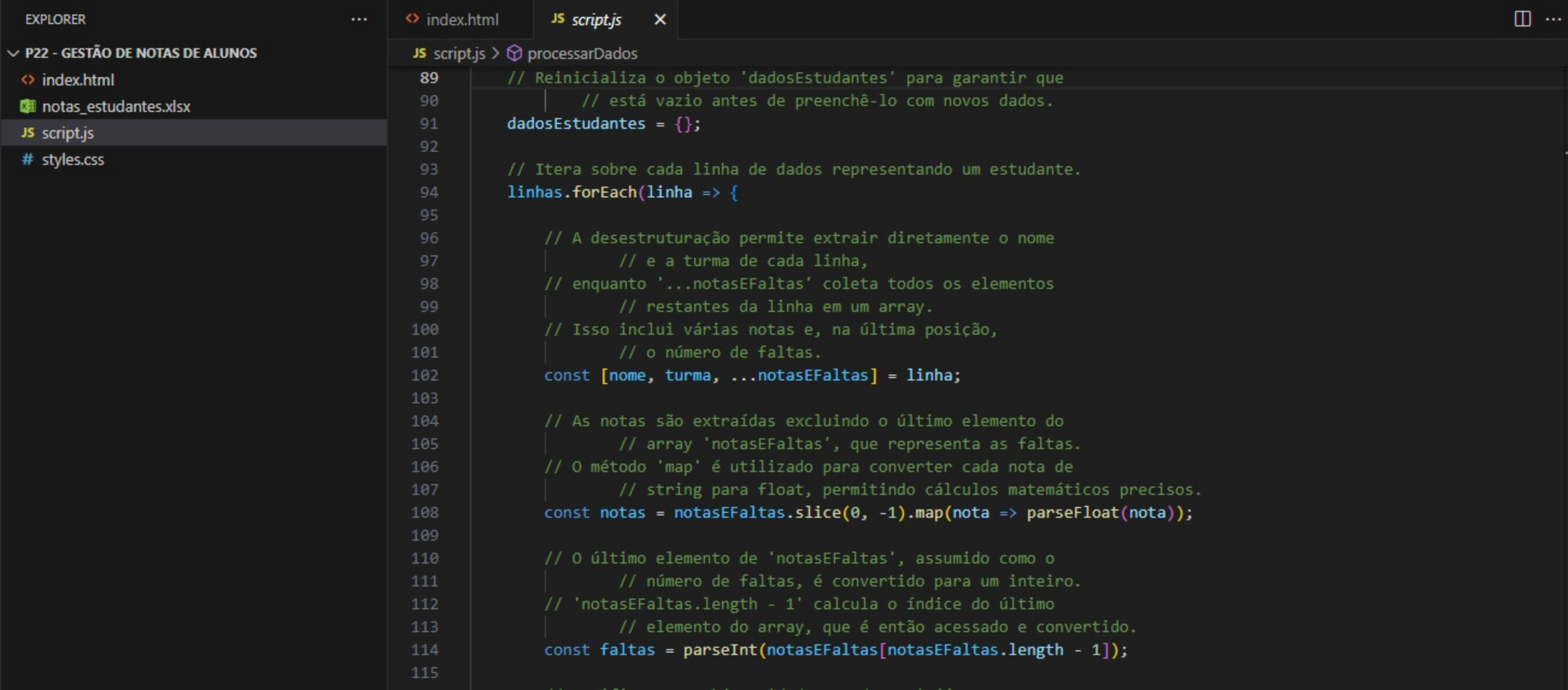
```
93              <button id="exportar-filtrados" class="botao-exportar" onclick="exportarNotasFiltradas()">Exportar
                Notas Filtradas</button>
94              <!-- Um botão que permite ao usuário exportar dados filtrados.
95                       O evento 'onclick' chama a função 'exportarNotasFiltradas()'
96                       definida no JavaScript. -->
97
98          </div>
99
100         <div id="container-abas" class="abas-container">
101         <!-- Uma <div> que serve como um contêiner para as
102                  abas de navegação. -->
103
104             <ul class="abas" id="abasTurmas"></ul>
105             <!-- Uma lista não ordenada que será preenchida com abas
106                      para cada turma dinamicamente via JavaScript. -->
107
108             <div class="conteudo-abas" id="conteudoAbas"></div>
109             <!-- Uma <div> que conterá o conteúdo associado a cada aba
110                      selecionada. O conteúdo é gerado e controlado
111                      dinamicamente por JavaScript. -->
112
113         </div>
114     </div>
115 </div>
116
117 <script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.16.9/xlsx.full.min.js"></script>
118 <!-- Um script externo que é usado para lidar com arquivos Excel (.xlsx).
119          Este script facilita a leitura e escrita de arquivos
120          Excel com JavaScript. -->
121
```

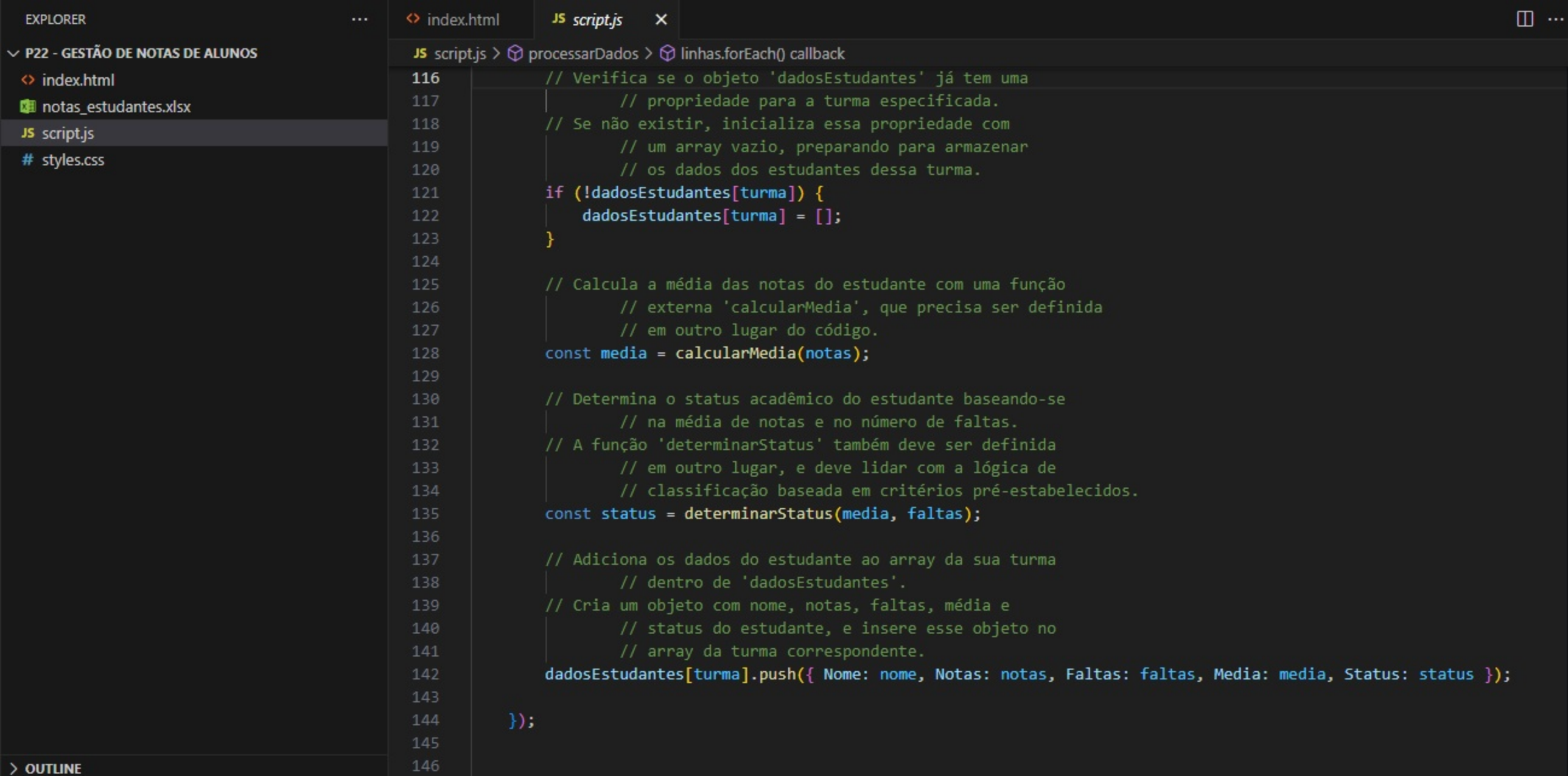

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
index.html
index.html > html > body
<script src="script.js"></script>
<!-- A inclusão do arquivo JavaScript local 'script.js' que
contém a lógica específica para esta aplicação, como
carregar dados, gerar abas e manipular eventos. -->
</body>
</html>
<!-- Fim da tag <body> e do documento HTML. -->

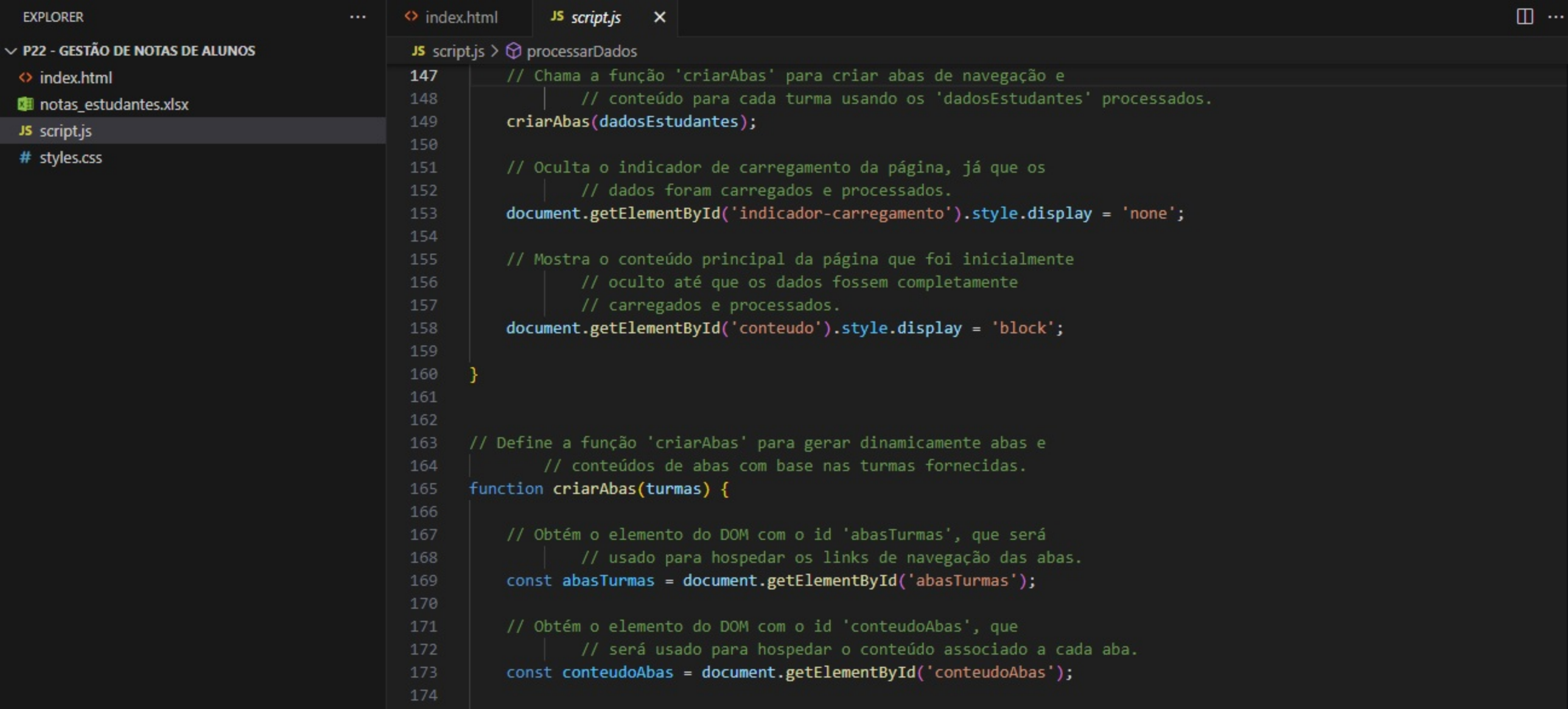

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
OUTLINE
TIMELINE
index.html
script.js
script.js > carregarDadosExcel
// Adiciona um ouvinte de eventos ao documento para
// executar a função 'carregarDadosExcel' assim
// que todo o conteúdo do DOM estiver completamente carregado.
document.addEventListener('DOMContentLoaded', carregarDadosExcel);
// Declara um objeto vazio 'dadosEstudantes' para armazenar os dados
// dos estudantes que serão carregados do arquivo Excel.
let dadosEstudantes = {};
// Esta função é definida para ser executada assim que o
// documento HTML estiver completamente carregado.
// Sua finalidade é carregar os dados de estudantes de
// um arquivo Excel e processá-los.
function carregarDadosExcel() {
// Inicia uma solicitação de rede para obter o arquivo
// Excel 'notas_estudantes.xlsx' do servidor ou
// de uma localização específica.
// A função fetch retorna uma promessa que resolve
// uma resposta HTTP.
fetch('notas_estudantes.xlsx')
// A resposta do fetch é um objeto Response, do qual
// extraímos um ArrayBuffer.
// Um ArrayBuffer é uma estrutura de dados genérica que
// representa um buffer de dados binários de
// tamanho fixo na memória.
// É usado aqui porque os arquivos Excel são arquivos binários, e
// precisamos de uma representação que possa ser
// lida pela biblioteca XLSX.
.then(response => response.arrayBuffer())

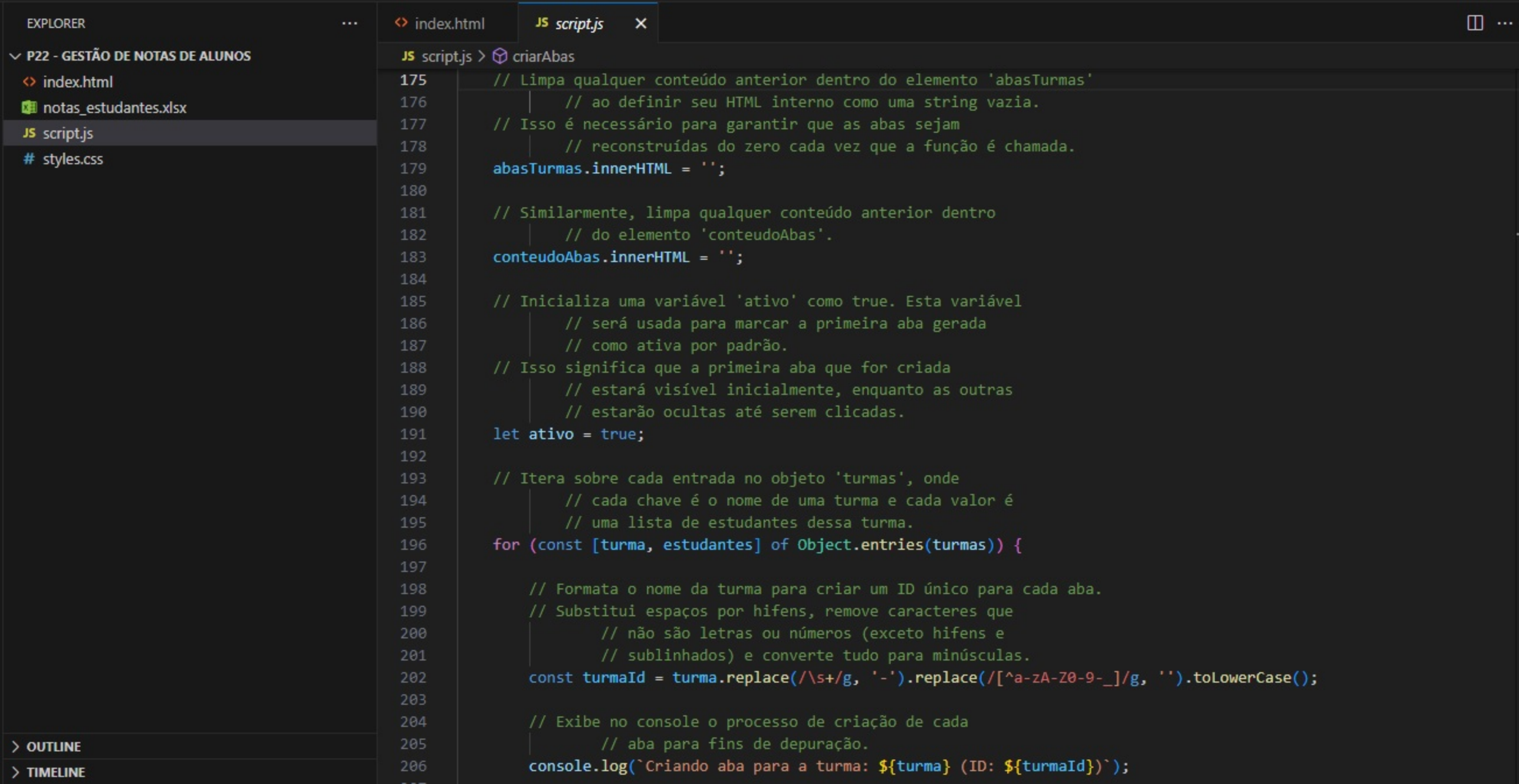

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
index.html
script.js
script.js > carregarDadosExcel > then() callback
// Após a conversão bem-sucedida da resposta para ArrayBuffer, o
// próximo passo é processar esse buffer.
.then(data => {
// A biblioteca XLSX é utilizada para ler os dados do
// Excel a partir do ArrayBuffer.
// A opção 'type: 'array'' informa à biblioteca que o
// formato dos dados fornecidos é um ArrayBuffer.
const workbook = XLSX.read(data, { type: 'array' });
// 'workbook' é um objeto que representa o livro inteiro
// do Excel; cada "folha" do Excel está acessível
// através deste objeto.
// SheetNames é um array que contém os nomes de todas as
// abas (ou "sheets") do arquivo Excel.
// Aqui, acessamos o nome da primeira aba para poder
// ler seus dados posteriormente.
const primeiraAba = workbook.SheetNames[0];
// Utilizamos o nome da primeira aba para acessar o
// objeto específico da aba dentro do workbook.
// Este objeto 'planilha' contém os dados da aba em um
// formato que a biblioteca XLSX pode manipular.
const planilha = workbook.Sheets[primeiraAba];

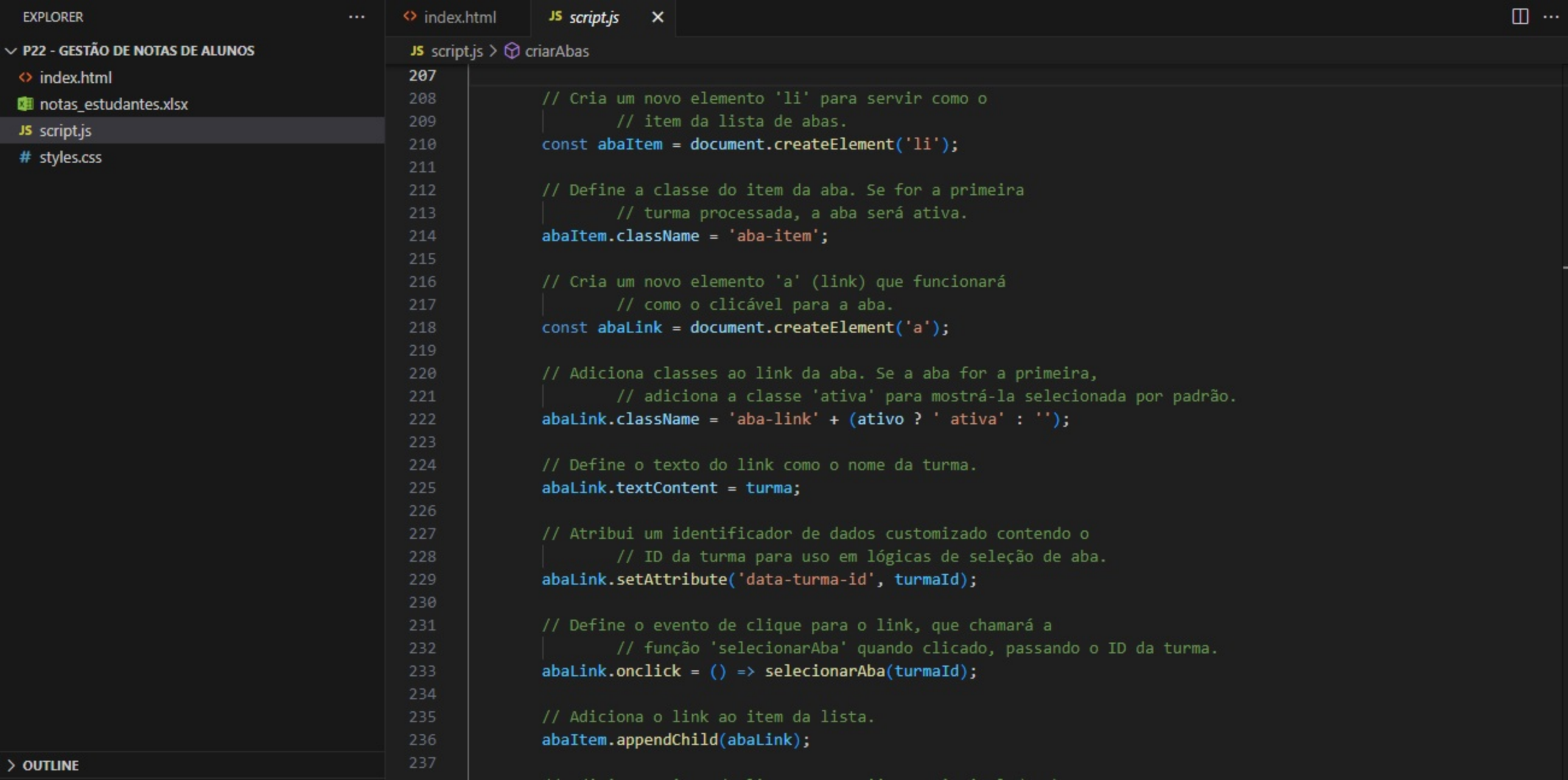

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
OUTLINE
index.html
script.js
script.js > carregarDadosExcel > then() callback
// Converte os dados da planilha selecionada para um
// formato JSON. A opção 'header: 1' especifica que a
// primeira linha da planilha deve ser interpretada
// como cabeçalhos de coluna.
// Isso facilita a manipulação dos dados como objetos
// JavaScript, onde cada linha da planilha se
// torna um objeto com propriedades nomeadas
// conforme os cabeçalhos.
const dadosJson = XLSX.utils.sheet_to_json(planilha, { header: 1 });
// Finalmente, a função 'processarDados' é chamada com
// os dados em formato JSON.
// Esta função será responsável por manipular, organizar ou
// exibir esses dados na página web.
processarDados(dadosJson);
});
}
// Define a função 'processarDados', que é responsável por
// processar os dados JSON extraídos da planilha Excel.
function processarDados(dados) {
// Desestrutura os dados recebidos em cabeçalhos e linhas.
// 'cabecalhos' contém a primeira linha dos dados, que
// são os cabeçalhos das colunas,
// e 'linhas' contém o resto dos dados, cada
// linha representando um estudante.
const [cabecalhos, ...linhas] = dados;

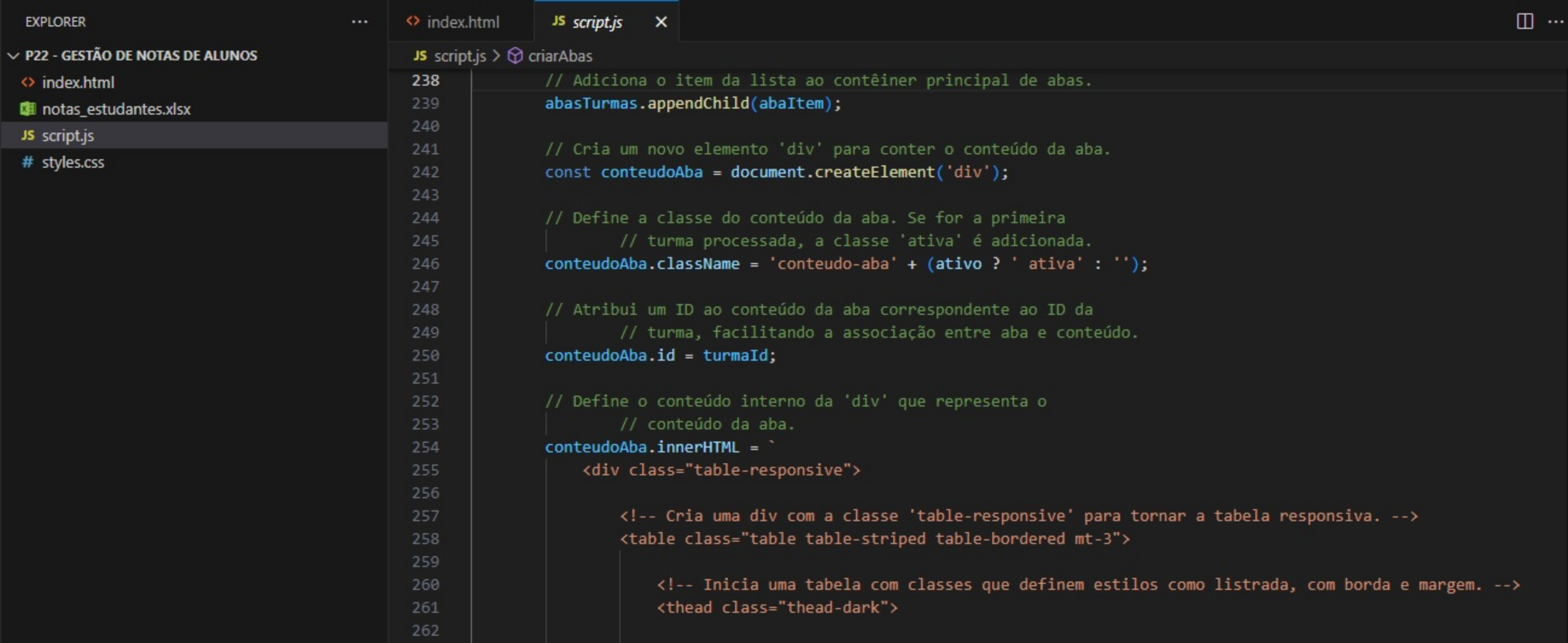

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
index.html
script.js
script.js > processarDados
// Reinicializa o objeto 'dadosEstudantes' para garantir que
// está vazio antes de preenchê-lo com novos dados.
dadosEstudantes = {};
// Itera sobre cada linha de dados representando um estudante.
linhas.forEach(linha => {
// A desestruturação permite extrair diretamente o nome
// e a turma de cada linha,
// enquanto '...notasEFaltas' coleta todos os elementos
// restantes da linha em um array.
// Isso inclui várias notas e, na última posição,
// o número de faltas.
const [nome, turma, ...notasEFaltas] = linha;
// As notas são extraídas excluindo o último elemento do
// array 'notasEFaltas', que representa as faltas.
// O método 'map' é utilizado para converter cada nota de
// string para float, permitindo cálculos matemáticos precisos.
const notas = notasEFaltas.slice(0, -1).map(nota => parseFloat(nota));
// O último elemento de 'notasEFaltas', assumido como o
// número de faltas, é convertido para um inteiro.
// 'notasEFaltas.length - 1' calcula o índice do último
// elemento do array, que é então acessado e convertido.
const faltas = parseInt(notasEFaltas[notasEFaltas.length - 1]);

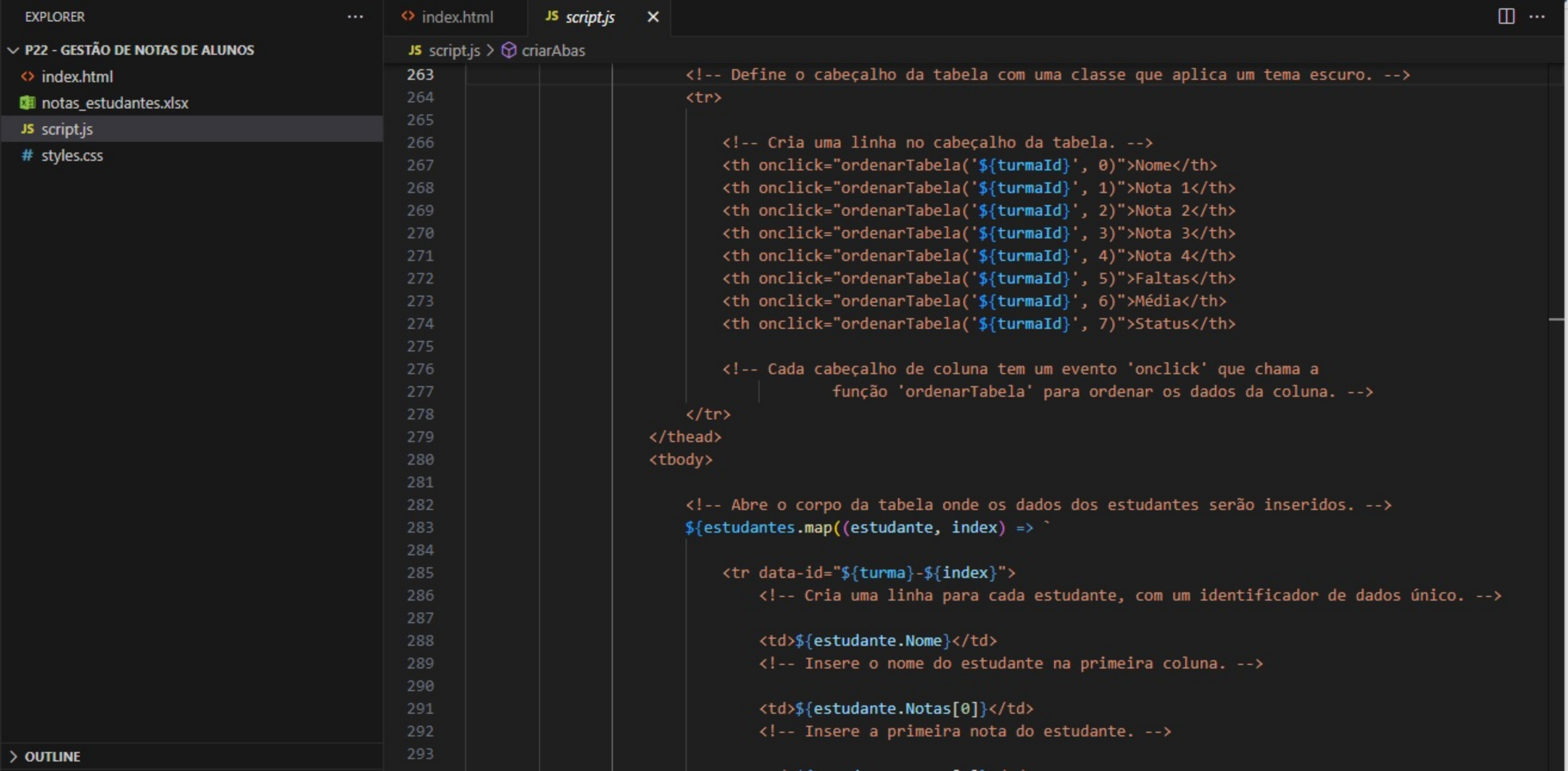

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
OUTLINE
index.html
script.js
script.js > processarDados > linhas.forEach() callback
116 // Verifica se o objeto 'dadosEstudantes' já tem uma
117 // propriedade para a turma especificada.
118 // Se não existir, inicializa essa propriedade com
119 // um array vazio, preparando para armazenar
120 // os dados dos estudantes dessa turma.
121 if (!dadosEstudantes[turma]) {
122 dadosEstudantes[turma] = [];
123 }
124
125 // Calcula a média das notas do estudante com uma função
126 // externa 'calcularMedia', que precisa ser definida
127 // em outro lugar do código.
128 const media = calcularMedia(notas);
129
130 // Determina o status acadêmico do estudante baseando-se
131 // na média de notas e no número de faltas.
132 // A função 'determinarStatus' também deve ser definida
133 // em outro lugar, e deve lidar com a lógica de
134 // classificação baseada em critérios pré-estabelecidos.
135 const status = determinarStatus(media, faltas);
136
137 // Adiciona os dados do estudante ao array da sua turma
138 // dentro de 'dadosEstudantes'.
139 // Cria um objeto com nome, notas, faltas, média e
140 // status do estudante, e insere esse objeto no
141 // array da turma correspondente.
142 dadosEstudantes[turma].push({ Nome: nome, Notas: notas, Faltas: faltas, Media: media, Status: status });
143
144 });
145
146

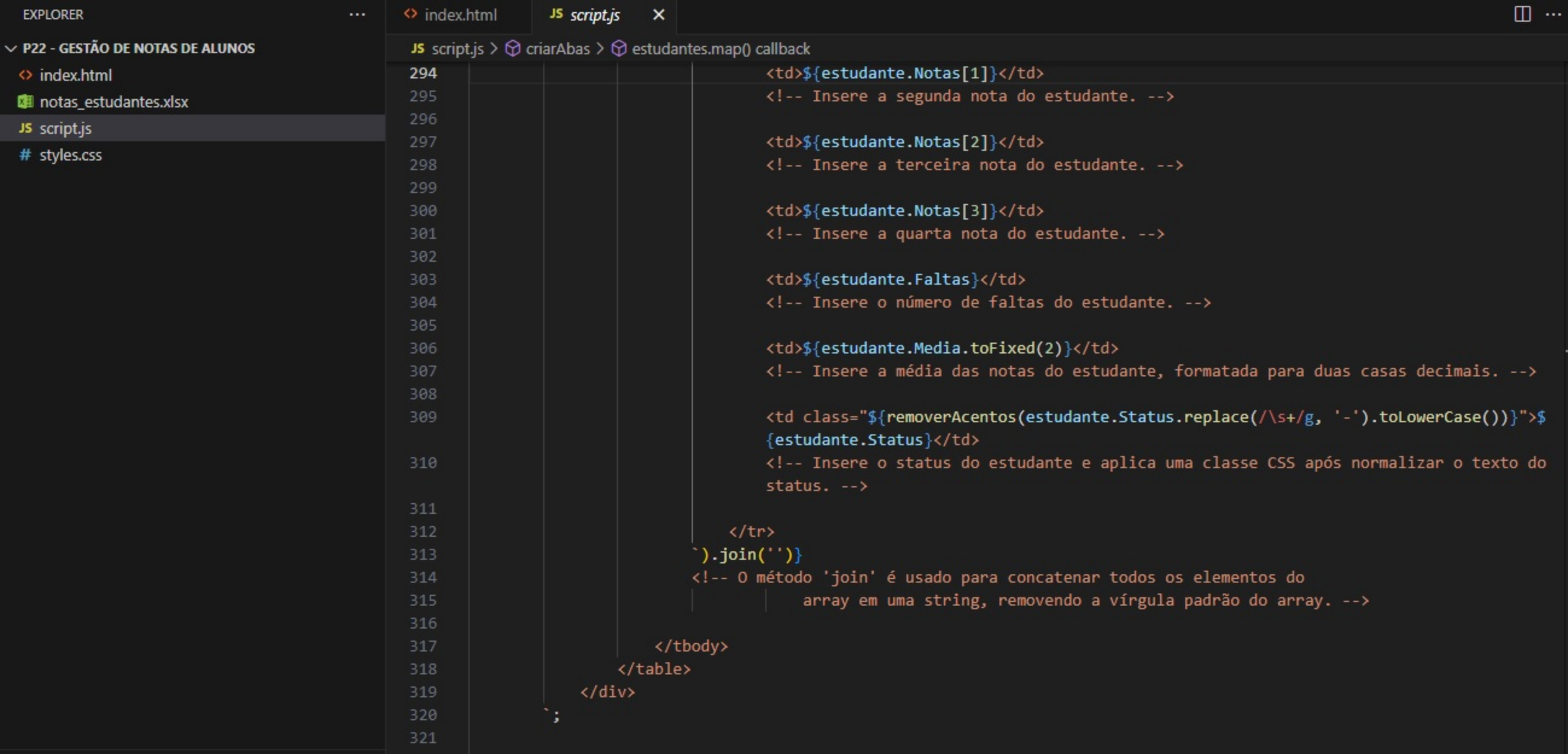

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
index.html
script.js
script.js > processarDados
147 // Chama a função 'criarAbas' para criar abas de navegação e
148 // conteúdo para cada turma usando os 'dadosEstudantes' processados.
149 criarAbas(dadosEstudantes);
150
151 // Oculta o indicador de carregamento da página, já que os
152 // dados foram carregados e processados.
153 document.getElementById('indicador-carregamento').style.display = 'none';
154
155 // Mostra o conteúdo principal da página que foi inicialmente
156 // oculto até que os dados fossem completamente
157 // carregados e processados.
158 document.getElementById('conteudo').style.display = 'block';
159
160 }
161
162
163 // Define a função 'criarAbas' para gerar dinamicamente abas e
164 // conteúdos de abas com base nas turmas fornecidas.
165 function criarAbas(turmas) {
166
167 // Obtém o elemento do DOM com o id 'abasTurmas', que será
168 // usado para hospedar os links de navegação das abas.
169 const abasTurmas = document.getElementById('abasTurmas');
170
171 // Obtém o elemento do DOM com o id 'conteudoAbas', que
172 // será usado para hospedar o conteúdo associado a cada aba.
173 const conteudoAbas = document.getElementById('conteudoAbas');
174

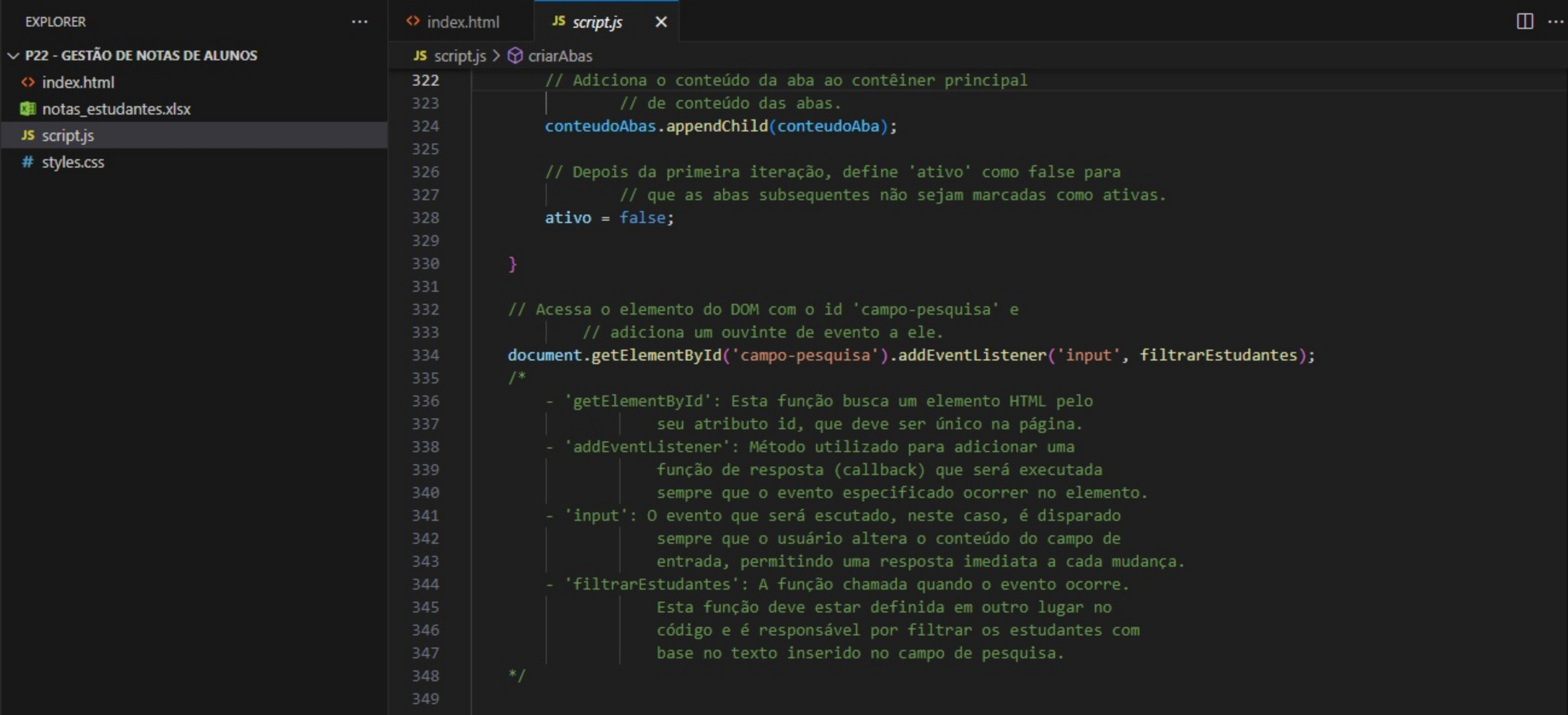

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
OUTLINE
TIMELINE
index.html
script.js
script.js > criarAbas
175 // Limpa qualquer conteúdo anterior dentro do elemento 'abasTurmas'
176 // ao definir seu HTML interno como uma string vazia.
177 // Isso é necessário para garantir que as abas sejam
178 // reconstruídas do zero cada vez que a função é chamada.
179 abasTurmas.innerHTML = '';
180
181 // Similarmente, limpa qualquer conteúdo anterior dentro
182 // do elemento 'conteudoAbas'.
183 conteudoAbas.innerHTML = '';
184
185 // Inicializa uma variável 'ativo' como true. Esta variável
186 // será usada para marcar a primeira aba gerada
187 // como ativa por padrão.
188 // Isso significa que a primeira aba que for criada
189 // estará visível inicialmente, enquanto as outras
190 // estarão ocultas até serem clicadas.
191 let ativo = true;
192
193 // Itera sobre cada entrada no objeto 'turmas', onde
194 // cada chave é o nome de uma turma e cada valor é
195 // uma lista de estudantes dessa turma.
196 for (const [turma, estudantes] of Object.entries(turmas)) {
197
198 // Formata o nome da turma para criar um ID único para cada aba.
199 // Substitui espaços por hifens, remove caracteres que
200 // não são letras ou números (exceto hifens e
201 // sublinhados) e converte tudo para minúsculas.
202 const turmaId = turma.replace(/\s+/g, '-').replace(/[^a-zA-Z0-9-_]/g, '').toLowerCase();
203
204 // Exibe no console o processo de criação de cada
205 // aba para fins de depuração.
206 console.log(`Criando aba para a turma: ${turma} (ID: ${turmaId})`);

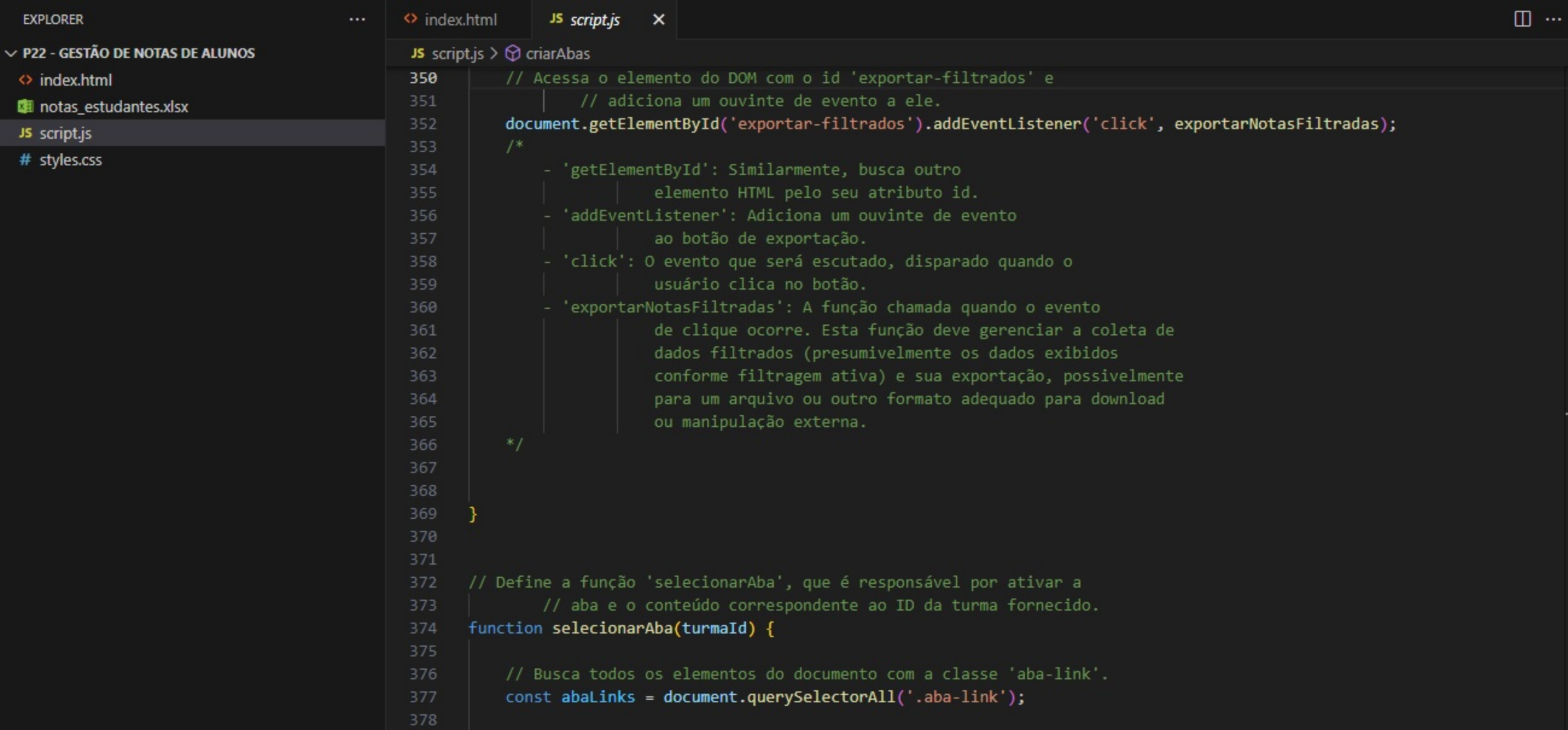

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
OUTLINE
index.html
script.js
script.js > criarAbas
207
208 // Cria um novo elemento 'li' para servir como o
209 // item da lista de abas.
210 const abaItem = document.createElement('li');
211
212 // Define a classe do item da aba. Se for a primeira
213 // turma processada, a aba será ativa.
214 abaItem.className = 'aba-item';
215
216 // Cria um novo elemento 'a' (link) que funcionará
217 // como o clicável para a aba.
218 const abaLink = document.createElement('a');
219
220 // Adiciona classes ao link da aba. Se a aba for a primeira,
221 // adiciona a classe 'ativa' para mostrá-la selecionada por padrão.
222 abaLink.className = 'aba-link' + (ativo ? ' ativa' : '');
223
224 // Define o texto do link como o nome da turma.
225 abaLink.textContent = turma;
226
227 // Atribui um identificador de dados customizado contendo o
228 // ID da turma para uso em lógicas de seleção de aba.
229 abaLink.setAttribute('data-turma-id', turmaId);
230
231 // Define o evento de clique para o link, que chamará a
232 // função 'selecionarAba' quando clicado, passando o ID da turma.
233 abaLink.onclick = () => selecionarAba(turmaId);
234
235 // Adiciona o link ao item da lista.
236 abaItem.appendChild(abaLink);
237

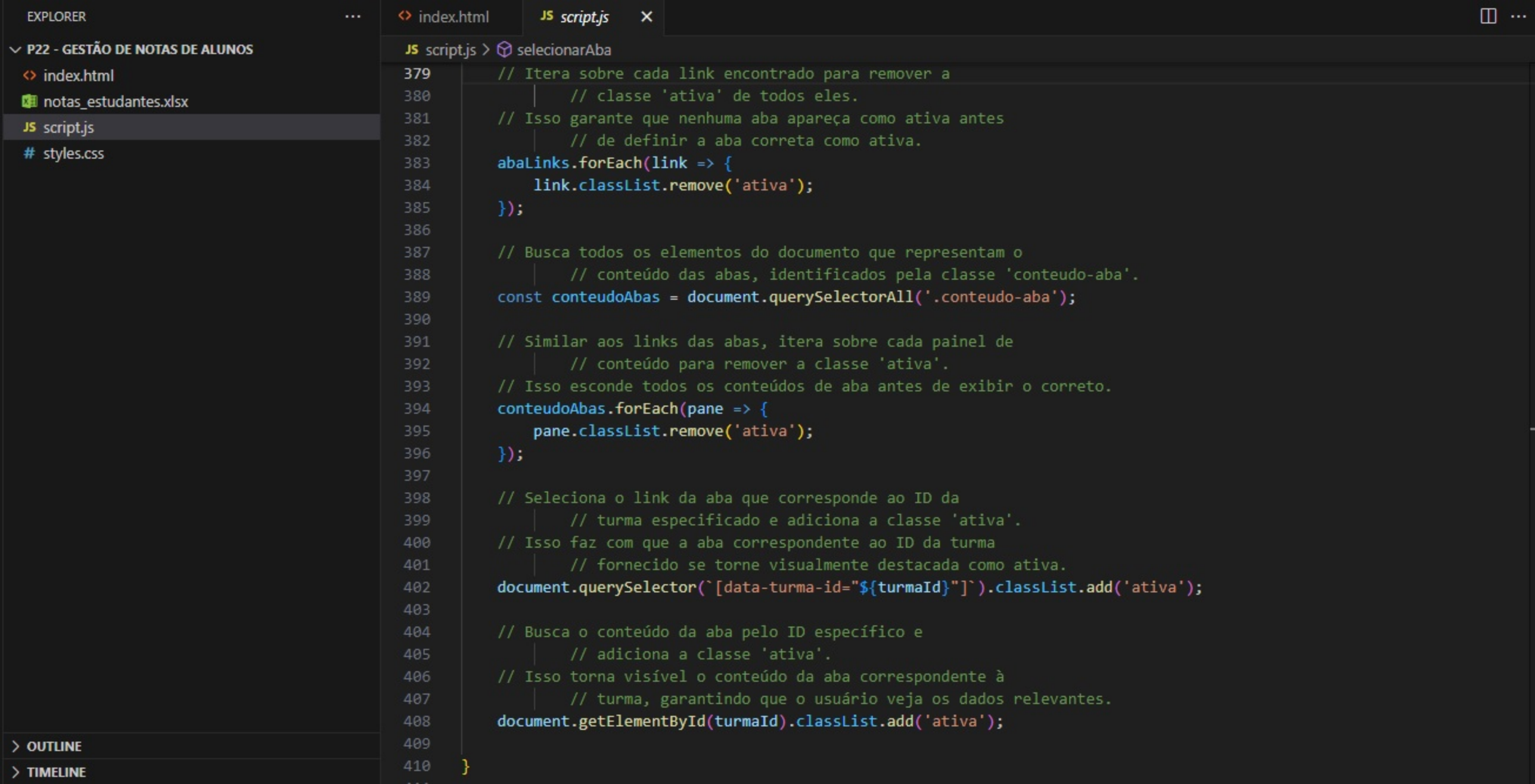

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
index.html
script.js
script.js > criarAbas
238 // Adiciona o item da lista ao contêiner principal de abas.
239 abasTurmas.appendChild(abaItem);
240
241 // Cria um novo elemento 'div' para conter o conteúdo da aba.
242 const conteudoAba = document.createElement('div');
243
244 // Define a classe do conteúdo da aba. Se for a primeira
245 // turma processada, a classe 'ativa' é adicionada.
246 conteudoAba.className = 'conteudo-aba' + (ativo ? ' ativa' : '');
247
248 // Atribui um ID ao conteúdo da aba correspondente ao ID da
249 // turma, facilitando a associação entre aba e conteúdo.
250 conteudoAba.id = turmaId;
251
252 // Define o conteúdo interno da 'div' que representa o
253 // conteúdo da aba.
254 conteudoAba.innerHTML = `
255
256
257 <!-- Cria uma div com a classe 'table-responsive' para tornar a tabela responsiva. -->
258 <table class="table table-striped table-bordered mt-3">
259
260 <!-- Inicia uma tabela com classes que definem estilos como listrada, com borda e margem. -->
261 <thead class="thead-dark">
262

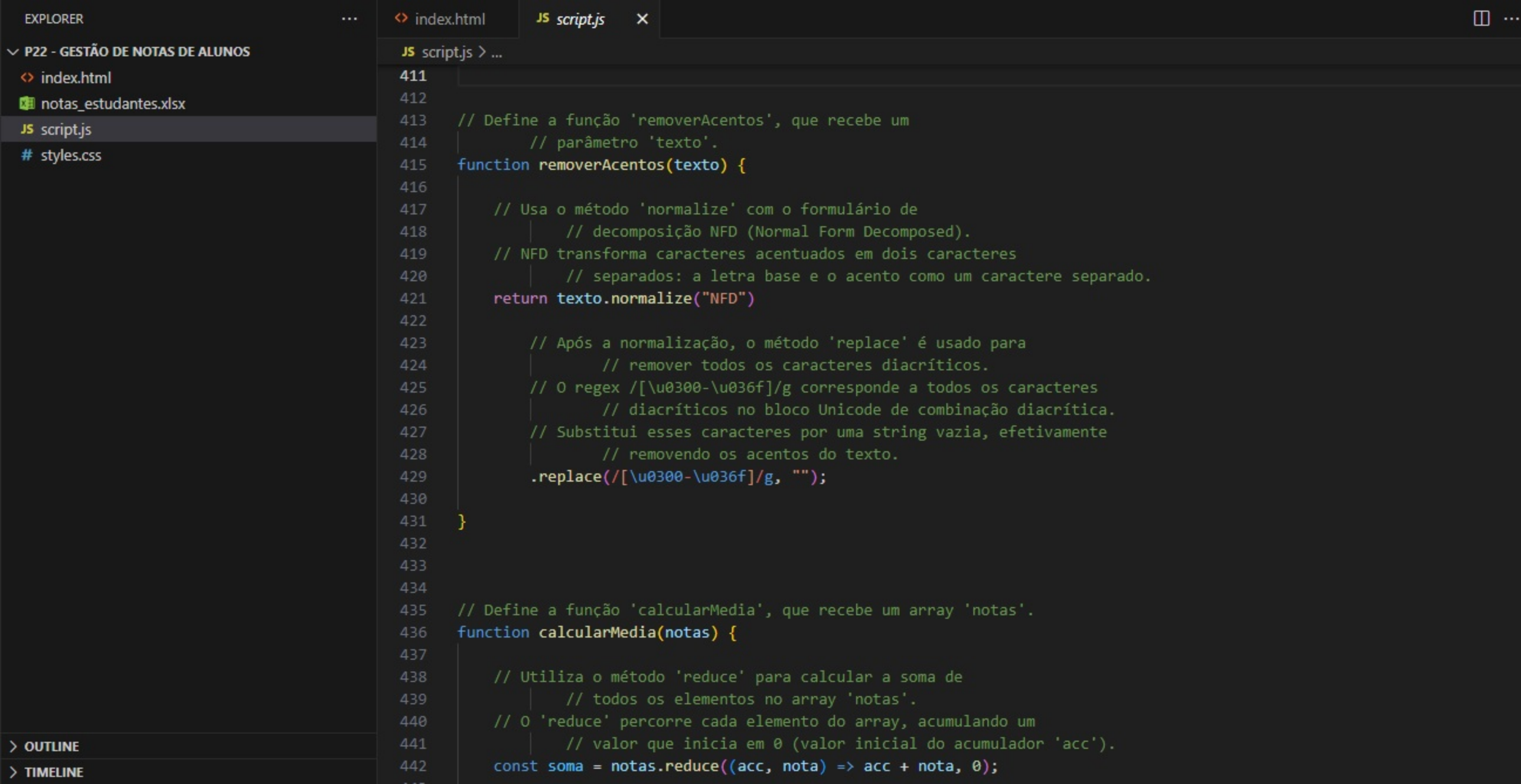

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
OUTLINE
index.html
script.js
script.js > criarAbas
263 <!-- Define o cabeçalho da tabela com uma classe que aplica um tema escuro. -->
264 <tr>
265
266 <!-- Cria uma linha no cabeçalho da tabela. -->
267 <th onclick="ordenarTabela('${turmaId}', 0)">Nome</th>
268 <th onclick="ordenarTabela('${turmaId}', 1)">Nota 1</th>
269 <th onclick="ordenarTabela('${turmaId}', 2)">Nota 2</th>
270 <th onclick="ordenarTabela('${turmaId}', 3)">Nota 3</th>
271 <th onclick="ordenarTabela('${turmaId}', 4)">Nota 4</th>
272 <th onclick="ordenarTabela('${turmaId}', 5)">Faltas</th>
273 <th onclick="ordenarTabela('${turmaId}', 6)">Média</th>
274 <th onclick="ordenarTabela('${turmaId}', 7)">Status</th>
275
276 <!-- Cada cabeçalho de coluna tem um evento 'onclick' que chama a
277 função 'ordenarTabela' para ordenar os dados da coluna. -->
278 </tr>
279 </thead>
280 <tbody>
281
282 <!-- Abre o corpo da tabela onde os dados dos estudantes serão inseridos. -->
283 ${estudantes.map((estudante, index) => `
284
285 <tr data-id="${turma}-${index}">
286 <!-- Cria uma linha para cada estudante, com um identificador de dados único. -->
287
288 <td>${estudante.Nome}</td>
289 <!-- Insere o nome do estudante na primeira coluna. -->
290
291 <td>${estudante.Notas[0]}</td>
292 <!-- Insere a primeira nota do estudante. -->
293


EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
index.html
script.js
script.js > criarAbas > estudantes.map() callback
294 <td>${estudante.Notas[1]}</td>
295 <!-- Insere a segunda nota do estudante. -->
296
297 <td>${estudante.Notas[2]}</td>
298 <!-- Insere a terceira nota do estudante. -->
299
300 <td>${estudante.Notas[3]}</td>
301 <!-- Insere a quarta nota do estudante. -->
302
303 <td>${estudante.Faltas}</td>
304 <!-- Insere o número de faltas do estudante. -->
305
306 <td>${estudante.Media.toFixed(2)}</td>
307 <!-- Insere a média das notas do estudante, formatada para duas casas decimais. -->
308
309 <td class="${removerAcentos(estudante.Status.replace(/\s+/g, '-').toLowerCase())}">${estudante.Status}</td>
310 <!-- Insere o status do estudante e aplica uma classe CSS após normalizar o texto do status. -->
311
312 </tr>
313 `).join('')}
314 <!-- O método 'join' é usado para concatenar todos os elementos do
315 array em uma string, removendo a vírgula padrão do array. -->
316
317 </tbody>
318 </table>
319
320 `;
321


EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
index.html
script.js
script.js > criarAbas
// Adiciona o conteúdo da aba ao contêiner principal
// de conteúdo das abas.
conteudoAbas.appendChild(conteudoAba);
// Depois da primeira iteração, define 'ativo' como false para
// que as abas subsequentes não sejam marcadas como ativas.
ativo = false;
}
// Acessa o elemento do DOM com o id 'campo-pesquisa' e
// adiciona um ouvinte de evento a ele.
document.getElementById('campo-pesquisa').addEventListener('input', filtrarEstudantes);
/*
- 'getElementById': Esta função busca um elemento HTML pelo
seu atributo id, que deve ser único na página.
- 'addEventListener': Método utilizado para adicionar uma
função de resposta (callback) que será executada
sempre que o evento especificado ocorrer no elemento.
- 'input': O evento que será escutado, neste caso, é disparado
sempre que o usuário altera o conteúdo do campo de
entrada, permitindo uma resposta imediata a cada mudança.
- 'filtrarEstudantes': A função chamada quando o evento ocorre.
Esta função deve estar definida em outro lugar no
código e é responsável por filtrar os estudantes com
base no texto inserido no campo de pesquisa.
*/

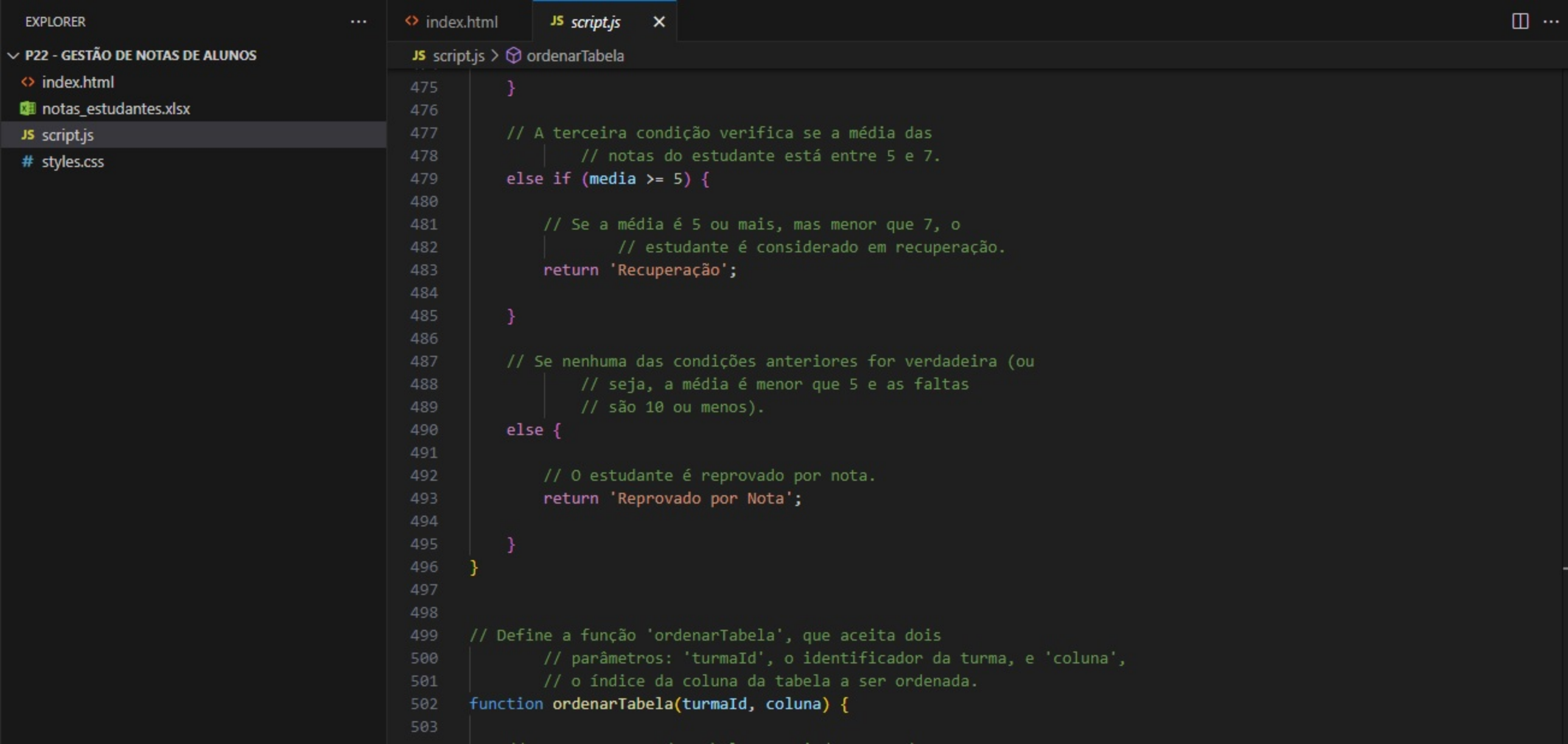

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
index.html
script.js
script.js > criarAbas
// Acessa o elemento do DOM com o id 'exportar-filtrados' e
// adiciona um ouvinte de evento a ele.
document.getElementById('exportar-filtrados').addEventListener('click', exportarNotasFiltradas);
/*
- 'getElementById': Similarmente, busca outro
elemento HTML pelo seu atributo id.
- 'addEventListener': Adiciona um ouvinte de evento
ao botão de exportação.
- 'click': O evento que será escutado, disparado quando o
usuário clica no botão.
- 'exportarNotasFiltradas': A função chamada quando o evento
de clique ocorre. Esta função deve gerenciar a coleta de
dados filtrados (presumivelmente os dados exibidos
conforme filtragem ativa) e sua exportação, possivelmente
para um arquivo ou outro formato adequado para download
ou manipulação externa.
*/
}
// Define a função 'selecionarAba', que é responsável por ativar a
// aba e o conteúdo correspondente ao ID da turma fornecido.
function selecionarAba(turmaId) {
// Busca todos os elementos do documento com a classe 'aba-link'.
const abaLinks = document.querySelectorAll('.aba-link');

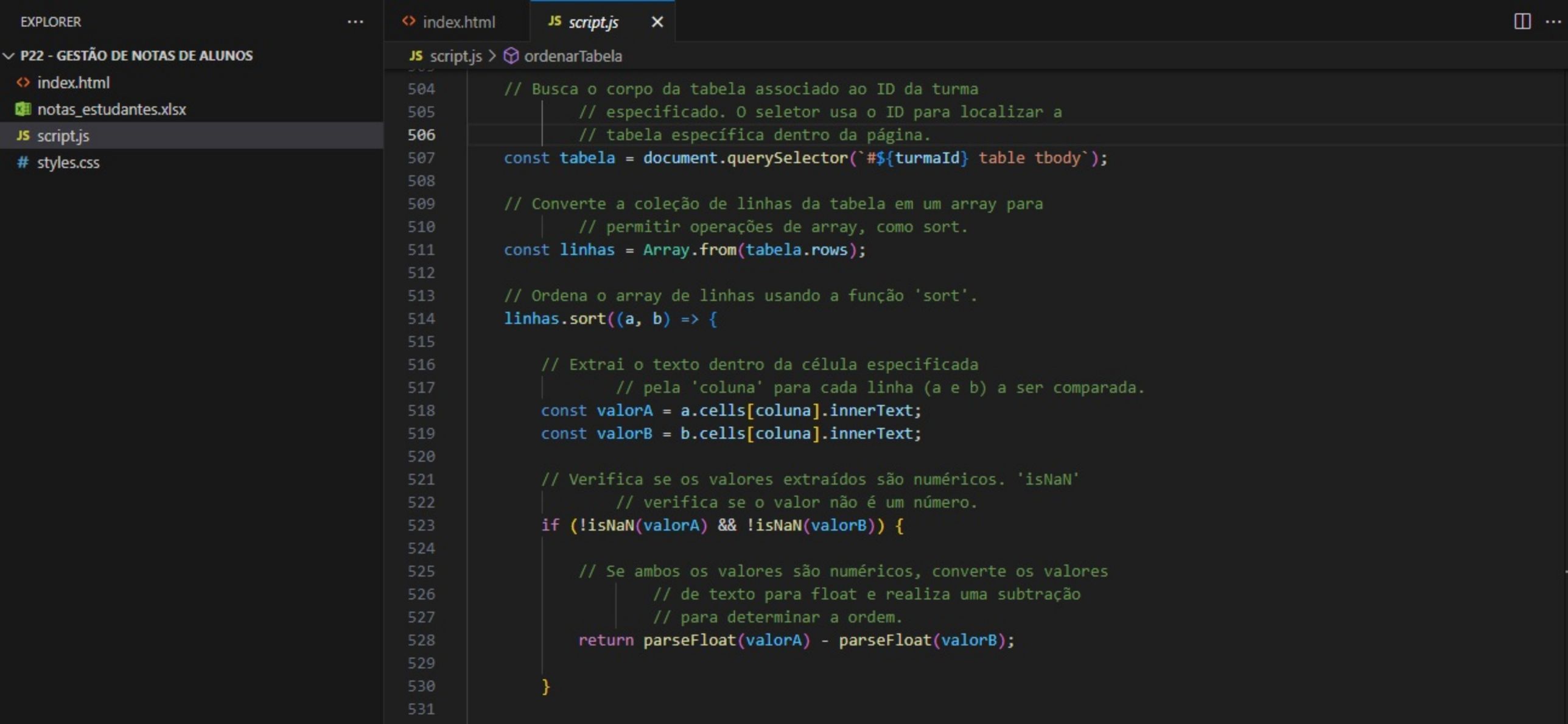

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
OUTLINE
TIMELINE
index.html
script.js
script.js > selecionarAba
379 // Itera sobre cada link encontrado para remover a
380 // classe 'ativa' de todos eles.
381 // Isso garante que nenhuma aba apareça como ativa antes
382 // de definir a aba correta como ativa.
383 abaLinks.forEach(link => {
384 link.classList.remove('ativa');
385 });
386
387 // Busca todos os elementos do documento que representam o
388 // conteúdo das abas, identificados pela classe 'conteudo-aba'.
389 const conteudoAbas = document.querySelectorAll('.conteudo-aba');
390
391 // Similar aos links das abas, itera sobre cada painel de
392 // conteúdo para remover a classe 'ativa'.
393 // Isso esconde todos os conteúdos de aba antes de exibir o correto.
394 conteudoAbas.forEach(pane => {
395 pane.classList.remove('ativa');
396 });
397
398 // Seleciona o link da aba que corresponde ao ID da
399 // turma especificado e adiciona a classe 'ativa'.
400 // Isso faz com que a aba correspondente ao ID da turma
401 // fornecido se torne visualmente destacada como ativa.
402 document.querySelector(`[data-turma-id="${turmaId}"]`).classList.add('ativa');
403
404 // Busca o conteúdo da aba pelo ID específico e
405 // adiciona a classe 'ativa'.
406 // Isso torna visível o conteúdo da aba correspondente à
407 // turma, garantindo que o usuário veja os dados relevantes.
408 document.getElementById(turmaId).classList.add('ativa');
409
410 }

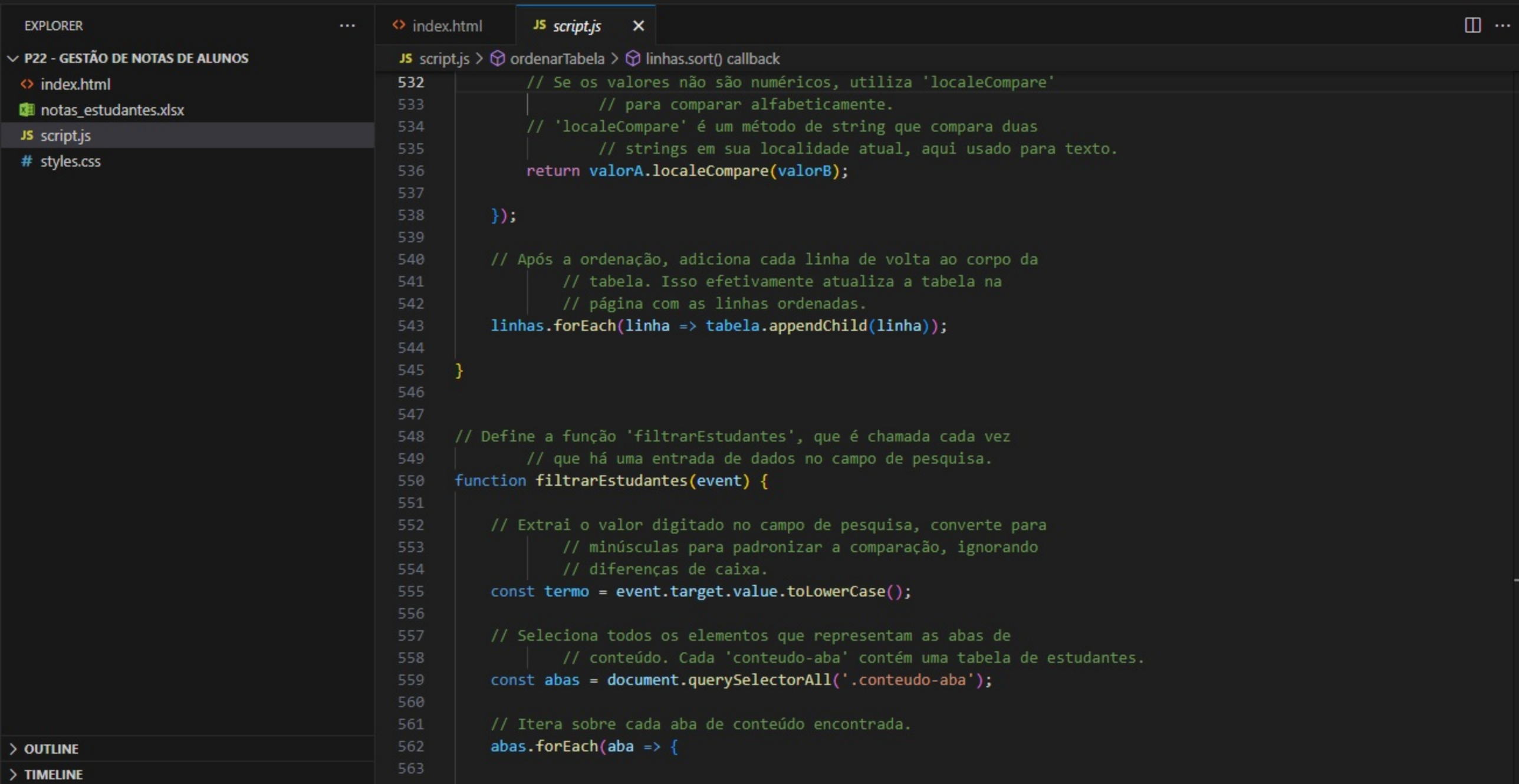

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
OUTLINE
TIMELINE
index.html
script.js
script.js > ...
411
412
413 // Define a função 'removerAcentos', que recebe um
414 // parâmetro 'texto'.
415 function removerAcentos(texto) {
416
417 // Usa o método 'normalize' com o formulário de
418 // decomposição NFD (Normal Form Decomposed).
419 // NFD transforma caracteres acentuados em dois caracteres
420 // separados: a letra base e o acento como um caractere separado.
421 return texto.normalize("NFD")
422
423 // Após a normalização, o método 'replace' é usado para
424 // remover todos os caracteres diacríticos.
425 // O regex /[\u0300-\u036f]/g corresponde a todos os caracteres
426 // diacríticos no bloco Unicode de combinação diacrítica.
427 // Substitui esses caracteres por uma string vazia, efetivamente
428 // removendo os acentos do texto.
429 .replace(/[\u0300-\u036f]/g, "");
430
431 }
432
433
434
435 // Define a função 'calcularMedia', que recebe um array 'notas'.
436 function calcularMedia(notas) {
437
438 // Utiliza o método 'reduce' para calcular a soma de
439 // todos os elementos no array 'notas'.
440 // O 'reduce' percorre cada elemento do array, acumulando um
441 // valor que inicia em 0 (valor inicial do acumulador 'acc').
442 const soma = notas.reduce((acc, nota) => acc + nota, 0);

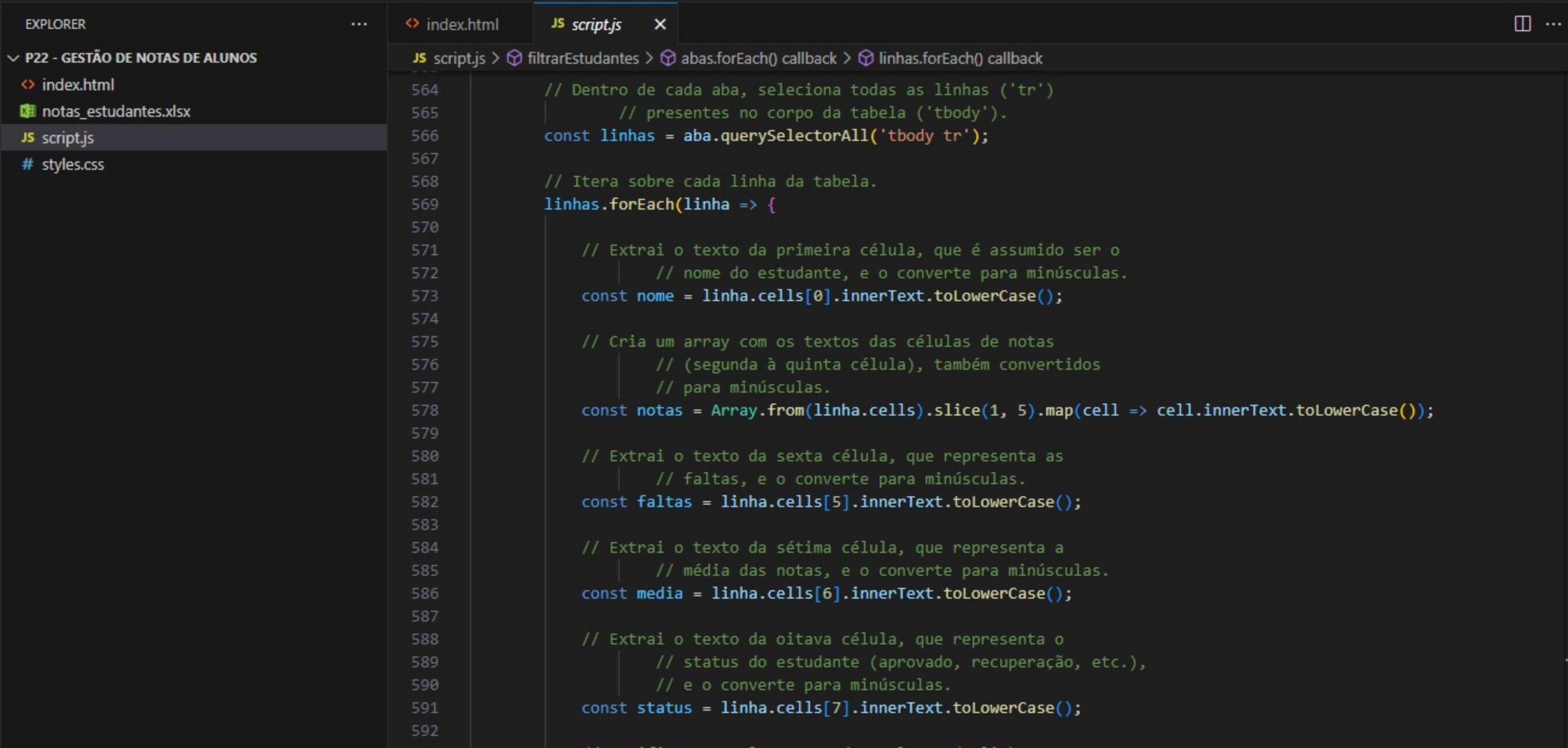
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
OUTLINE
TIMELINE

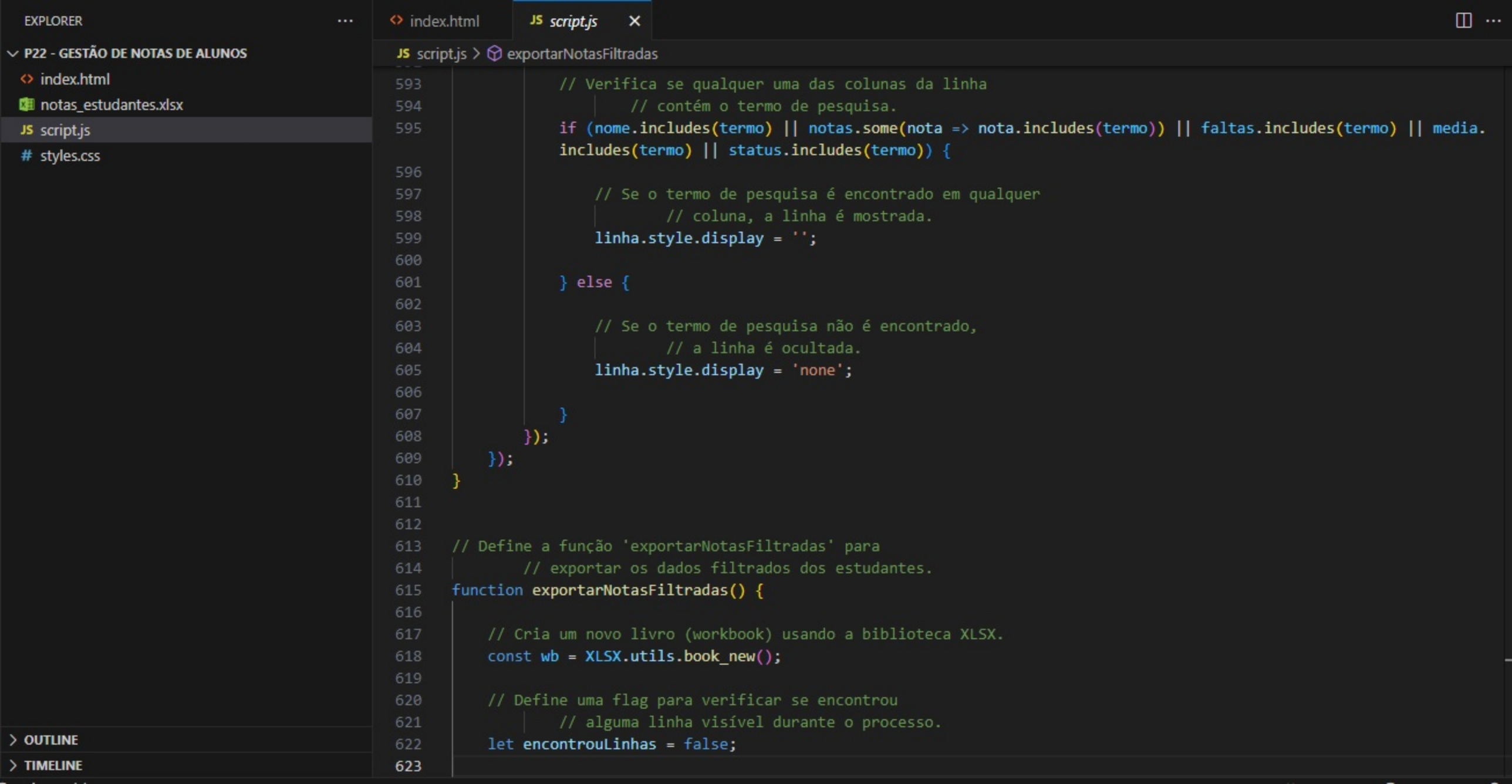
```
443
444     // Divide a soma total pelo número de elementos no
445             // array 'notas' para calcular a média.
446     // 'notas.length' fornece o total de elementos no array.
447     return soma / notas.length;
448
449 }
450
451
452 // Define a função 'determinarStatus', que recebe dois
453         // parâmetros: 'media', a média das notas do estudante,
454         // e 'faltas', o número total de faltas do estudante.
455 function determinarStatus(media, faltas) {
456
457     // A primeira condição verifica se o número de faltas
458             // do estudante é maior que 10.
459     if (faltas > 10) {
460
461         // Se o estudante tem mais de 10 faltas, independente da
462                 // média de notas, ele é automaticamente reprovado por faltas.
463         return 'Reprovado por faltas';
464
465     }
466
467     // A segunda condição verifica se a média das notas do
468             // estudante é igual ou superior a 7.
469     else if (media >= 7) {
470
471         // Se a média é 7 ou mais, o estudante é
472                 // considerado aprovado.
473         return 'Aprovado';
474
```

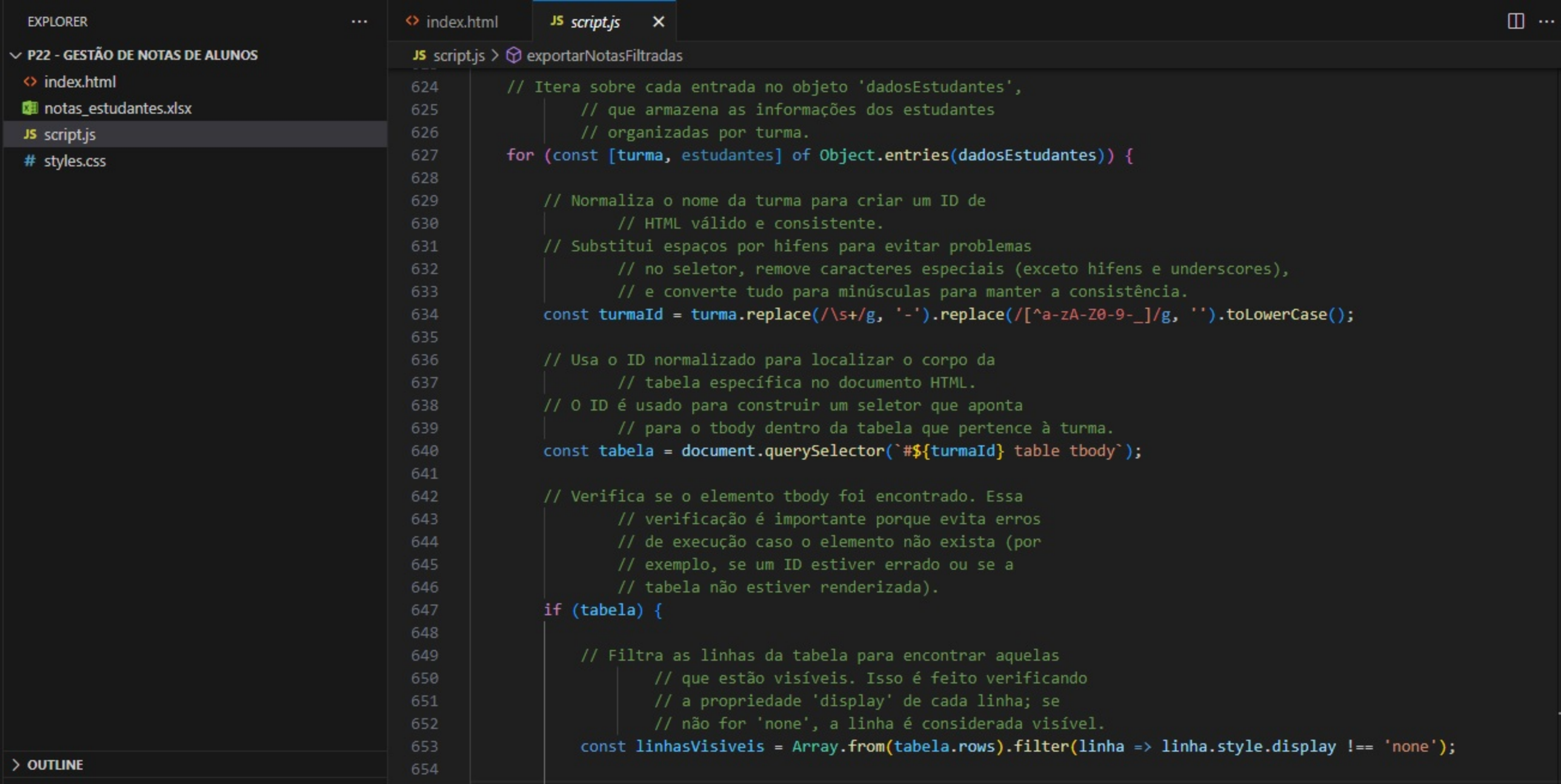

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
index.html
script.js
script.js > ordenarTabela
475 }
476
477 // A terceira condição verifica se a média das
478 // notas do estudante está entre 5 e 7.
479 else if (media >= 5) {
480
481 // Se a média é 5 ou mais, mas menor que 7, o
482 // estudante é considerado em recuperação.
483 return 'Recuperação';
484
485 }
486
487 // Se nenhuma das condições anteriores for verdadeira (ou
488 // seja, a média é menor que 5 e as faltas
489 // são 10 ou menos).
490 else {
491
492 // O estudante é reprovado por nota.
493 return 'Reprovado por Nota';
494
495 }
496 }
497
498
499 // Define a função 'ordenarTabela', que aceita dois
500 // parâmetros: 'turmaId', o identificador da turma, e 'coluna',
501 // o índice da coluna da tabela a ser ordenada.
502 function ordenarTabela(turmaId, coluna) {
503

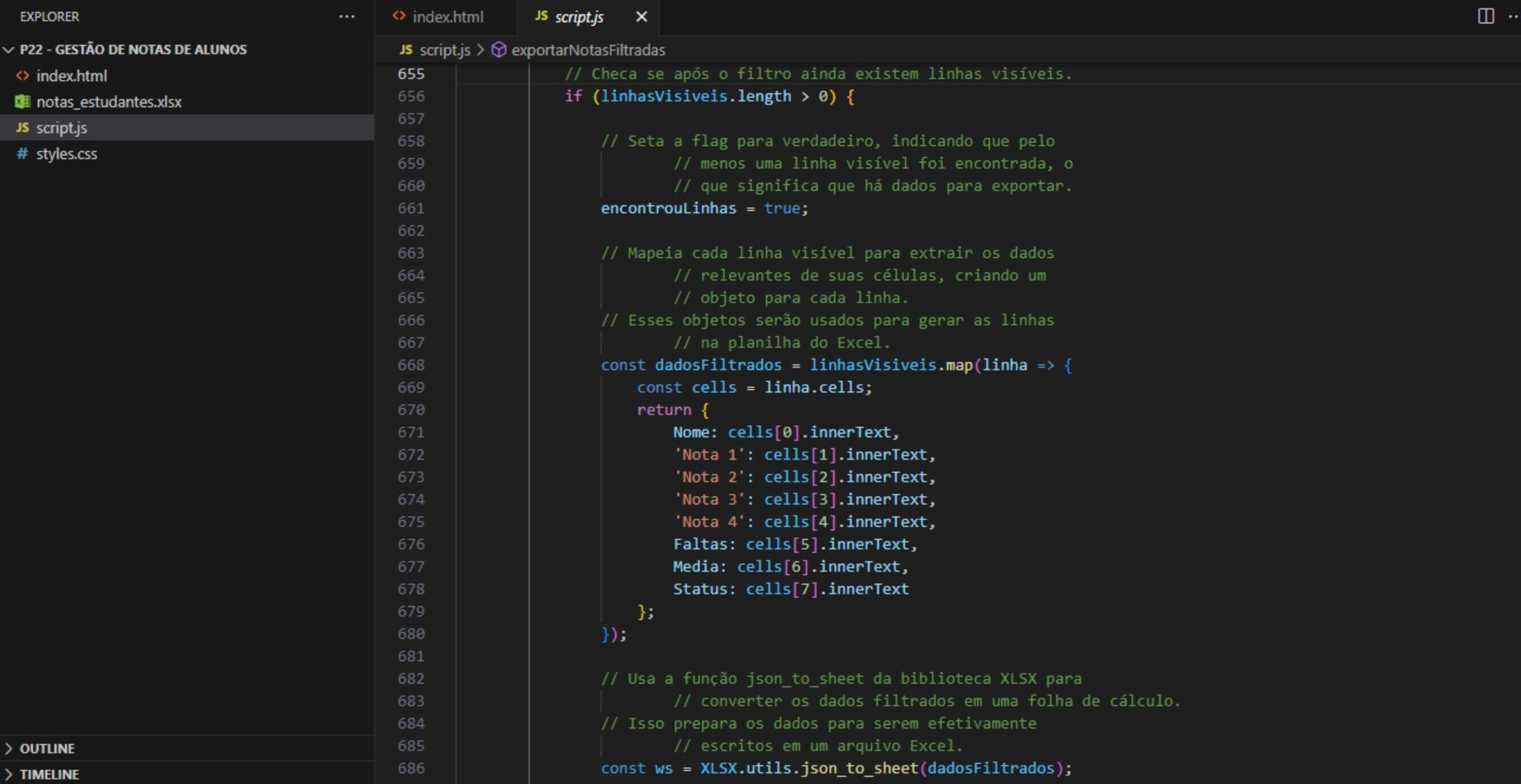

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
index.html
script.js
script.js > ordenarTabela
// Busca o corpo da tabela associado ao ID da turma
// especificado. O seletor usa o ID para localizar a
// tabela específica dentro da página.
const tabela = document.querySelector(`#${turmaId} table tbody`);
// Converte a coleção de linhas da tabela em um array para
// permitir operações de array, como sort.
const linhas = Array.from(tabela.rows);
// Ordena o array de linhas usando a função 'sort'.
linhas.sort((a, b) => {
// Extrai o texto dentro da célula especificada
// pela 'coluna' para cada linha (a e b) a ser comparada.
const valorA = a.cells[coluna].innerText;
const valorB = b.cells[coluna].innerText;
// Verifica se os valores extraídos são numéricos. 'isNaN'
// verifica se o valor não é um número.
if (!isNaN(valorA) && !isNaN(valorB)) {
// Se ambos os valores são numéricos, converte os valores
// de texto para float e realiza uma subtração
// para determinar a ordem.
return parseFloat(valorA) - parseFloat(valorB);
}

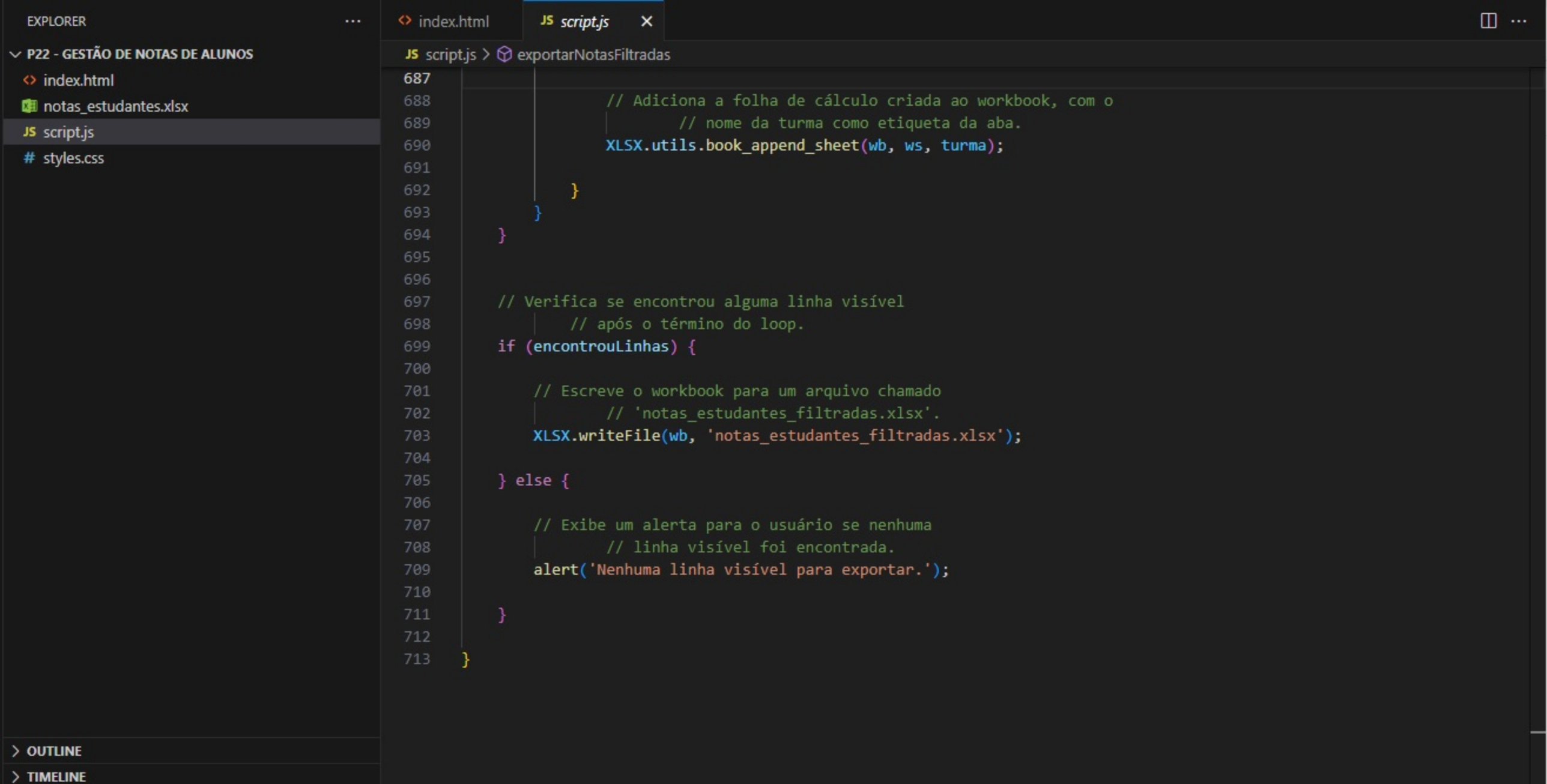

EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
OUTLINE
TIMELINE
index.html
script.js
script.js > ordenarTabela > linhas.sort() callback
// Se os valores não são numéricos, utiliza 'localeCompare'
// para comparar alfabeticamente.
// 'localeCompare' é um método de string que compara duas
// strings em sua localidade atual, aqui usado para texto.
return valorA.localeCompare(valorB);
});
// Após a ordenação, adiciona cada linha de volta ao corpo da
// tabela. Isso efetivamente atualiza a tabela na
// página com as linhas ordenadas.
linhas.forEach(linha => tabela.appendChild(linha));
}
// Define a função 'filtrarEstudantes', que é chamada cada vez
// que há uma entrada de dados no campo de pesquisa.
function filtrarEstudantes(event) {
// Extrai o valor digitado no campo de pesquisa, converte para
// minúsculas para padronizar a comparação, ignorando
// diferenças de caixa.
const termo = event.target.value.toLowerCase();
// Seleciona todos os elementos que representam as abas de
// conteúdo. Cada 'conteudo-aba' contém uma tabela de estudantes.
const abas = document.querySelectorAll('.conteudo-aba');
// Itera sobre cada aba de conteúdo encontrada.
abas.forEach(aba => {


EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
index.html
script.js
script.js > filtrarEstudantes > abas.forEach() callback > linhas.forEach() callback
564 // Dentro de cada aba, seleciona todas as linhas ('tr')
565 // presentes no corpo da tabela ('tbody').
566 const linhas = aba.querySelectorAll('tbody tr');
567
568 // Itera sobre cada linha da tabela.
569 linhas.forEach(linha => {
570
571 // Extrai o texto da primeira célula, que é assumido ser o
572 // nome do estudante, e o converte para minúsculas.
573 const nome = linha.cells[0].innerText.toLowerCase();
574
575 // Cria um array com os textos das células de notas
576 // (segunda à quinta célula), também convertidos
577 // para minúsculas.
578 const notas = Array.from(linha.cells).slice(1, 5).map(cell => cell.innerText.toLowerCase());
579
580 // Extrai o texto da sexta célula, que representa as
581 // faltas, e o converte para minúsculas.
582 const faltas = linha.cells[5].innerText.toLowerCase();
583
584 // Extrai o texto da sétima célula, que representa a
585 // média das notas, e o converte para minúsculas.
586 const media = linha.cells[6].innerText.toLowerCase();
587
588 // Extrai o texto da oitava célula, que representa o
589 // status do estudante (aprovado, recuperação, etc.),
590 // e o converte para minúsculas.
591 const status = linha.cells[7].innerText.toLowerCase();
592


EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
OUTLINE
TIMELINE
index.html
script.js
script.js > exportarNotasFiltradas
593 // Verifica se qualquer uma das colunas da linha
594 // contém o termo de pesquisa.
595 if (nome.includes(termo) || notas.some(nota => nota.includes(termo)) || faltas.includes(termo) || media.includes(termo) || status.includes(termo)) {
596
597 // Se o termo de pesquisa é encontrado em qualquer
598 // coluna, a linha é mostrada.
599 linha.style.display = '';
600
601 } else {
602
603 // Se o termo de pesquisa não é encontrado,
604 // a linha é ocultada.
605 linha.style.display = 'none';
606
607 }
608 });
609 });
610 }
611
612
613 // Define a função 'exportarNotasFiltradas' para
614 // exportar os dados filtrados dos estudantes.
615 function exportarNotasFiltradas() {
616
617 // Cria um novo livro (workbook) usando a biblioteca XLSX.
618 const wb = XLSX.utils.book_new();
619
620 // Define uma flag para verificar se encontrou
621 // alguma linha visível durante o processo.
622 let encontrouLinhas = false;
623


EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
OUTLINE
index.html
script.js
script.js > exportarNotasFiltradas
624 // Itera sobre cada entrada no objeto 'dadosEstudantes',
625 // que armazena as informações dos estudantes
626 // organizadas por turma.
627 for (const [turma, estudantes] of Object.entries(dadosEstudantes)) {
628
629 // Normaliza o nome da turma para criar um ID de
630 // HTML válido e consistente.
631 // Substitui espaços por hifens para evitar problemas
632 // no seletor, remove caracteres especiais (exceto hifens e underscores),
633 // e converte tudo para minúsculas para manter a consistência.
634 const turmaId = turma.replace(/\s+/g, '-').replace(/[^a-zA-Z0-9-_]/g, '').toLowerCase();
635
636 // Usa o ID normalizado para localizar o corpo da
637 // tabela específica no documento HTML.
638 // O ID é usado para construir um seletor que aponta
639 // para o tbody dentro da tabela que pertence à turma.
640 const tabela = document.querySelector(`#${turmaId} table tbody`);
641
642 // Verifica se o elemento tbody foi encontrado. Essa
643 // verificação é importante porque evita erros
644 // de execução caso o elemento não exista (por
645 // exemplo, se um ID estiver errado ou se a
646 // tabela não estiver renderizada).
647 if (tabela) {
648
649 // Filtra as linhas da tabela para encontrar aquelas
650 // que estão visíveis. Isso é feito verificando
651 // a propriedade 'display' de cada linha; se
652 // não for 'none', a linha é considerada visível.
653 const linhasVisiveis = Array.from(tabela.rows).filter(linha => linha.style.display !== 'none');
654


EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
OUTLINE
TIMELINE
index.html
script.js
script.js > exportarNotasFiltradas
655 // Checa se após o filtro ainda existem linhas visíveis.
656 if (linhasVisiveis.length > 0) {
657
658 // Seta a flag para verdadeiro, indicando que pelo
659 // menos uma linha visível foi encontrada, o
660 // que significa que há dados para exportar.
661 encontrouLinhas = true;
662
663 // Mapeia cada linha visível para extrair os dados
664 // relevantes de suas células, criando um
665 // objeto para cada linha.
666 // Esses objetos serão usados para gerar as linhas
667 // na planilha do Excel.
668 const dadosFiltrados = linhasVisiveis.map(linha => {
669 const cells = linha.cells;
670 return {
671 Nome: cells[0].innerText,
672 'Nota 1': cells[1].innerText,
673 'Nota 2': cells[2].innerText,
674 'Nota 3': cells[3].innerText,
675 'Nota 4': cells[4].innerText,
676 Faltas: cells[5].innerText,
677 Media: cells[6].innerText,
678 Status: cells[7].innerText
679 };
680 });
681
682 // Usa a função json_to_sheet da biblioteca XLSX para
683 // converter os dados filtrados em uma folha de cálculo.
684 // Isso prepara os dados para serem efetivamente
685 // escritos em um arquivo Excel.
686 const ws = XLSX.utils.json_to_sheet(dadosFiltrados);


EXPLORER
P22 - GESTÃO DE NOTAS DE ALUNOS
index.html
notas_estudantes.xlsx
script.js
styles.css
OUTLINE
TIMELINE
index.html
script.js
script.js > exportarNotasFiltradas
687
688 // Adiciona a folha de cálculo criada ao workbook, com o
689 // nome da turma como etiqueta da aba.
690 XLSX.utils.book_append_sheet(wb, ws, turma);
691
692 }
693 }
694 }
695
696
697 // Verifica se encontrou alguma linha visível
698 // após o término do loop.
699 if (encontrouLinhas) {
700
701 // Escreve o workbook para um arquivo chamado
702 // 'notas_estudantes_filtradas.xlsx'.
703 XLSX.writeFile(wb, 'notas_estudantes_filtradas.xlsx');
704
705 } else {
706
707 // Exibe um alerta para o usuário se nenhuma
708 // linha visível foi encontrada.
709 alert('Nenhuma linha visível para exportar.');
710
711 }
712
713 }