

## src/components/LocationAdmin.vue

```
1 <template>
2   <div class="admin-container">
3     <router-link to="/">Voltar para o Mapa</router-link> <h1>Admin de Locais</h1>
4
5     <div v-if="localLoading" class="loading">Carregando...</div>
6     <div v-if="localError" class="error">{{ localError }}</div>
7     <div v-if="localSuccessMessage" class="success">{{ localSuccessMessage }}</div>
8
9     <button @click="showAddForm" :disabled="localLoading || showForm" class="btn btn-primary">Adicionar Novo Local</button>
10
11   <div v-if="showForm" class="form-section">
12     <h2>{{ isEditing ? 'Editar Local' : 'Adicionar Local' }}</h2>
13     <form @submit.prevent="handleSubmit">
14       <div class="form-group" v-if="isEditing">
15         <label for="id">ID:</label>
16         <input type="text" id="id" :value="formData.id" disabled>
17       </div>
18       <div class="form-group">
19         <label for="nome">Nome:</label>
20         <input type="text" id="nome" v-model="formData.nome" required>
21       </div>
22       <div class="form-group">
23         <label for="endereco">Endereço/Descrição (para geocodificação no backend, se houver):</label>
24         <input type="text" id="endereco" v-model="formData.endereco" required>
25       </div>
26       <div class="form-group">
27         <label for="andar">Andar (ex: terreo, primeiro):</label>
28         <input type="text" id="andar" v-model="formData.andar" required>
29       </div>
30       <div class="form-actions">
31         <button type="submit" :disabled="localLoading" class="btn btn-success">{{ isEditing ? 'Salvar Alterações' :
32 'Criar Local' }}</button>
33         <button type="button" @click="cancelForm" :disabled="localLoading" class="btn btn-secondary">Cancelar</button>
34       </div>
35     </form>
36   </div>
37 </div>
38 </template>
```

```

35 </div>
36
37 <h2>Locais Existentes</h2>
38 <div class="table-container">
39   <table class="locations-table">
40     <thead>
41       <tr>
42         <th>ID</th>
43         <th>Nome</th>
44         <th>Andar</th>
45         <th>Endereço</th>
46         <th>X%</th>
47         <th>Y%</th>
48         <th>Ações</th>
49       </tr>
50     </thead>
51     <tbody>
52       <tr v-if="locations.length === 0 && !localLoading && !localError">
53         <td colspan="7" style="text-align: center;">Nenhum local cadastrado.</td>
54       </tr>
55       <tr v-if="locations.length === 0 && localError">
56         <td colspan="7" style="text-align: center; color: red;">Falha ao carregar locais.</td>
57       </tr>
58       <tr v-for="loc in locations" :key="loc.id">
59         <td>{{ loc.id }}</td>
60         <td>{{ loc.nome }}</td>
61         <td>{{ loc.andar }}</td>
62         <td>{{ loc.endereco || 'N/A' }}</td>
63         <td>{{ loc.x?.toFixed ? loc.x.toFixed(2) : loc.x }}</td>
64         <td>{{ loc.y?.toFixed ? loc.y.toFixed(2) : loc.y }}</td>
65         <td>
66           <button @click="showEditForm(loc)" :disabled="localLoading || showForm" class="btn btn-warning btn-
sm">Editar</button>
67           <button @click="confirmDelete(loc.id)" :disabled="localLoading || showForm" class="btn btn-danger btn-
sm">Excluir</button>
68         </td>
69       </tr>

```

```

70     </tbody>
71 </table>
72 </div>
73 </div>
74 </template>
75
76 <script setup>
77 import { ref, onMounted } from 'vue';
78 import { useRouter } from 'vue-router';
79 // Importa as funções CRUD do serviço centralizado
80 import {
81     createLocation,
82     updateLocation,
83     deleteLocation,
84     fetchAdminLocations // Usa a função específica para buscar locais para o admin
85 } from '@services/LocationService.js';
86
87 const router = useRouter();
88 const locations = ref([]);
89 // Estados locais para feedback dentro do componente admin
90 const localLoading = ref(false);
91 const localError = ref(null);
92 const localSuccessMessage = ref(null);
93
94 const showForm = ref(false);
95 const isEditing = ref(false);
96 const formData = ref({
97     id: null,
98     nome: '',
99     endereco: '',
100     andar: '',
101     // x: null, // Descomente se adicionar inputs X/Y
102     // y: null, // Descomente se adicionar inputs X/Y
103 });
104
105 // Limpa mensagens locais após um tempo
106 function clearLocalMessages() {

```

```
107     setTimeout(() => {
108         localError.value = null;
109         localSuccessMessage.value = null;
110     }, 5000); // Limpa após 5 segundos
111 }
112
113 // --- Funções de Interação com o Serviço ---
114
115 async function loadLocations() {
116     localLoading.value = true;
117     localError.value = null; // Limpa erro anterior
118     locations.value = []; // Limpa locais antes de buscar
119     try {
120         const fetchedLocais = await fetchAdminLocations(); // Usa a função do serviço
121         if (fetchedLocais) {
122             locations.value = fetchedLocais;
123         } else {
124             // Se fetchAdminLocations retornar null, o erro já foi setado no serviço,
125             // mas podemos colocar uma msg genérica aqui se quisermos.
126             localError.value = "Não foi possível carregar os locais.";
127         }
128     } catch (err) {
129         console.error("Erro não capturado ao carregar locais:", err);
130         localError.value = `Falha ao carregar dados: ${err.message}`;
131     } finally {
132         localLoading.value = false;
133         if(localError.value) clearLocalMessages(); // Limpa msg de erro tbm
134     }
135 }
136
137 async function handleAddLocation() {
138     localLoading.value = true;
139     localError.value = null;
140     localSuccessMessage.value = null;
141     try {
142         // Prepara os dados do formData
143         const dataToSend = {
```

```
144     nome: formData.value.nome,
145     endereco: formData.value.endereco,
146     andar: formData.value.andar,
147     // x: formData.value.x, // Envia se for editável
148     // y: formData.value.y, // Envia se for editável
149 };
150 const success = await createLocation(dataToSend); // Usa a função do serviço
151 if (success) {
152     localSuccessMessage.value = "Local adicionado com sucesso!"; // Mensagem local
153     await loadLocations(); // Recarrega a lista
154     cancelForm(); // Esconde o formulário
155     clearLocalMessages();
156 } else {
157     // Se retornou false, o erro provavelmente está no estado global do serviço
158     // Poderíamos buscá-lo ou mostrar uma msg genérica
159     localError.value = "Falha ao adicionar local.";
160     clearLocalMessages();
161 }
162 } catch (err) {
163     console.error("Erro ao adicionar local:", err);
164     localError.value = `Erro ao criar: ${err.message}`;
165     clearLocalMessages();
166 } finally {
167     localLoading.value = false;
168 }
169 }
170
171 async function handleUpdateLocation() {
172     localLoading.value = true;
173     localError.value = null;
174     localSuccessMessage.value = null;
175     try {
176         const dataToSend = {
177             id: formData.value.id,
178             nome: formData.value.nome,
179             endereco: formData.value.endereco,
180             andar: formData.value.andar,
```

```
181     // x: formData.value.x, // Envia se for editável
182     // y: formData.value.y, // Envia se for editável
183 };
184 const success = await updateLocation(dataToSend); // Usa a função do serviço
185 if (success) {
186     localSuccessMessage.value = "Local atualizado com sucesso!"; // Mensagem local
187     await loadLocations(); // Recarrega a lista
188     cancelForm(); // Esconde o formulário
189     clearLocalMessages();
190 } else {
191     localError.value = "Falha ao atualizar local.";
192     clearLocalMessages();
193 }
194 } catch (err) {
195     console.error("Erro ao atualizar local:", err);
196     localError.value = `Erro ao atualizar: ${err.message}`;
197     clearLocalMessages();
198 } finally {
199     localLoading.value = false;
200 }
201 }
202
203 async function handleDeleteLocation(id) {
204     localLoading.value = true;
205     localError.value = null;
206     localSuccessMessage.value = null;
207     try {
208         const success = await deleteLocation(id); // Usa a função do serviço
209         if (success) {
210             localSuccessMessage.value = "Local excluído com sucesso!"; // Mensagem local
211             await loadLocations(); // Recarrega a lista
212             clearLocalMessages();
213         } else {
214             localError.value = "Falha ao excluir local.";
215             clearLocalMessages();
216         }
217     } catch (err) {
```

```
218     console.error("Erro ao excluir local:", err);
219     localError.value = `Erro ao excluir: ${err.message}`;
220     clearLocalMessages();
221   } finally {
222     localLoading.value = false;
223   }
224 }
225
226
227 // --- Funções de UI ---
228
229 function resetForm() {
230   formData.value = { id: null, nome: '', endereco: '', andar: '' /*, x: null, y: null */ };
231 }
232
233 function showAddForm() {
234   resetForm();
235   isEditing.value = false;
236   showForm.value = true;
237   localError.value = null;
238   localSuccessMessage.value = null;
239 }
240
241 function showEditForm(location) {
242   isEditing.value = true;
243   // Preenche o form com os dados do local. Certifique-se que a API retorna todos necessários.
244   formData.value = {
245     id: location.id,
246     nome: location.nome || '',
247     endereco: location.endereco || '', // Necessita que a API retorne 'endereco'
248     andar: location.andar || '',
249     // x: location.x, // Preenche se for editável
250     // y: location.y, // Preenche se for editável
251   };
252   showForm.value = true;
253   localError.value = null;
254   localSuccessMessage.value = null;
```

```
255 }
256
257 function cancelForm() {
258   showForm.value = false;
259   isEditing.value = false;
260   resetForm();
261 }
262
263 function handleSubmit() {
264   // Validação básica
265   if (!formData.value.nome || !formData.value.endereco || !formData.value.andar) {
266     localError.value = "Por favor, preencha nome, endereço e andar.";
267     clearLocalMessages();
268     return;
269   }
270   // Adicione validação para X/Y se forem editáveis
271
272   if (isEditing.value) {
273     if (!formData.value.id) {
274       localError.value = "ID do local ausente para edição.";
275       clearLocalMessages();
276       return;
277     }
278     handleUpdateLocation();
279   } else {
280     handleAddLocation();
281   }
282 }
283
284 function confirmDelete(id) {
285   if (window.confirm(`Tem certeza que deseja excluir o local com ID "${id}"? Esta ação não pode ser desfeita.`)) {
286     handleDeleteLocation(id);
287   }
288 }
289
290 // Carrega locais iniciais ao montar o componente
291 onMounted(() => {
```



```
292     loadLocations();
293 });
294 </script>
295
296 <style scoped>
297 /* Estilos específicos para o container geral da página admin */
298 .admin-container {
299     padding: 25px;
300     max-width: 900px; /* Limita a largura para melhor leitura */
301     margin: 20px auto; /* Centraliza */
302     font-family: sans-serif;
303     background-color: #fff;
304     border-radius: 8px;
305     box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
306 }
307
308 /* Estilos para links e botões */
309 a {
310     color: #007bff;
311     text-decoration: none;
312     margin-bottom: 15px;
313     display: inline-block;
314 }
315 a:hover {
316     text-decoration: underline;
317 }
318
319 h1, h2 {
320     color: #333;
321     margin-bottom: 15px;
322 }
323
324 .loading, .error, .success {
325     padding: 12px 15px;
326     margin-bottom: 20px;
327     border-radius: 5px;
328     text-align: center;
```

```
329     font-size: 0.95em;
330 }
331 .loading { background-color: #eef; color: #333; }
332 .error { background-color: #f8d7da; color: #721c24; border: 1px solid #f5c6cb; }
333 .success { background-color: #d4edda; color: #155724; border: 1px solid #c3e6cb; }
334
335 .btn {
336     padding: 10px 18px;
337     border: none;
338     border-radius: 5px;
339     cursor: pointer;
340     margin: 5px;
341     font-size: 1em;
342     transition: background-color 0.2s ease, box-shadow 0.2s ease;
343     box-shadow: 0 1px 3px rgba(0,0,0,0.1);
344 }
345 .btn:hover:not(:disabled) {
346     opacity: 0.9;
347     box-shadow: 0 2px 5px rgba(0,0,0,0.15);
348 }
349 .btn-primary { background-color: #007bff; color: white; }
350 .btn-success { background-color: #28a745; color: white; }
351 .btn-warning { background-color: #ffc107; color: #212529; border: 1px solid #d39e00;}
352 .btn-danger { background-color: #dc3545; color: white; }
353 .btn-secondary { background-color: #6c757d; color: white; }
354 .btn-sm { padding: 6px 12px; font-size: 0.85em; }
355 .btn:disabled { background-color: #e9ecef; color: #6c757d; cursor: not-allowed; box-shadow: none; }
356
357
358 /* Estilos para o formulário */
359 .form-section {
360     margin-top: 25px;
361     margin-bottom: 30px;
362     padding: 20px;
363     border: 1px solid #ddd;
364     border-radius: 8px;
365     background-color: #fdfdfd;
```

```
366 }
367 .form-group {
368     margin-bottom: 15px;
369 }
370 .form-group label {
371     display: block;
372     margin-bottom: 6px;
373     font-weight: bold;
374     color: #555;
375 }
376 .form-group input[type="text"],
377 .form-group input[type="number"], /* Estilo para inputs numéricos (X/Y) */
378 .form-group select {
379     width: 100%; /* Full width */
380     padding: 10px;
381     border: 1px solid #ccc;
382     border-radius: 4px;
383     box-sizing: border-box; /* Include padding in width */
384     font-size: 1em;
385 }
386 .form-group input[disabled] {
387     background-color: #e9ecef;
388     cursor: not-allowed;
389 }
390 .form-actions {
391     margin-top: 20px;
392     text-align: right; /* Alinha botões à direita */
393 }
394
395 /* Estilos para a tabela */
396 .table-container {
397     overflow-x: auto; /* Permite rolagem horizontal em telas pequenas */
398 }
399 .locations-table {
400     width: 100%;
401     margin-top: 25px;
402     border-collapse: collapse;
```

```
403     font-size: 0.9em;
404 }
405 .locations-table th,
406 .locations-table td {
407     border: 1px solid #dee2e6;
408     padding: 10px 12px;
409     text-align: left;
410     vertical-align: middle;
411 }
412 .locations-table th {
413     background-color: #f8f9fa;
414     font-weight: bold;
415     color: #495057;
416 }
417 .locations-table tr:nth-child(even) {
418     background-color: #f8f9fa;
419 }
420 .locations-table tr:hover {
421     background-color: #e9ecef;
422 }
423 .locations-table td button {
424     margin-right: 5px; /* Espaço entre botões de ação */
425 }
426 </style>
```