

src/assets/styles/main.css

```
1 /* Variáveis Globais (opcional, mas recomendado para fácil manutenção) */
2 :root {
3   --sidebar-width: 300px;
4   --sidebar-width-mobile: 80vw;
5   --sidebar-bg-color: #f8f9fa;
6   --primary-accent-color: #007bff; /* Azul usado no item selecionado */
7   --secondary-accent-color: #4CAF50;
8   --button-bg-color: #f0f0f0;
9   --button-hover-bg-color: #e0e0e0;
10  --border-color: #ccc;
11  --marker-red: red;
12  --marker-blue: blue;
13  --path-color: #6FA1EC; /* Cor do caminho */
14  --text-color: #333;
15  --error-color: red;
16  --error-bg-color: #ffebee;
17  --transition-speed: 0.3s;
18 }
19 /* Reset Básico e Estilos Globais */
20 body {
21   margin: 0;
22   padding: 0;
23   font-family: Arial, sans-serif;
24   overflow-x: hidden; /* Evita barra de rolagem horizontal geral */
25   color: var(--text-color);
26 }
27 /* Contêiner Principal da Aplicação */
28 .app-container {
29   position: relative; /* Necessário para posicionamento absoluto da sidebar e botão */
30   width: 100vw;
31   height: 100vh;
32   display: flex; /* Mantém flexibilidade, mas a sidebar será absoluta */
33   overflow: hidden; /* Impede que o conteúdo interno cause barras de rolagem */
34 }
```

```
35 /* Barra de Navegação Superior */
36 .top-nav {
37   position: absolute;
38   top: 15px;
39   right: 15px;
40   z-index: 1001;
41   display: flex;
42   gap: 10px;
43 }
44
45 .nav-link {
46   display: inline-block;
47   padding: 8px 12px;
48   background-color: var(--button-bg-color);
49   color: var(--text-color);
50   text-decoration: none;
51   border-radius: 4px;
52   font-weight: 500;
53   box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
54   transition: background-color 0.2s ease;
55 }
56
57 .nav-link:hover {
58   background-color: var(--button-hover-bg-color);
59 }
60
61 /* Botão para Abrir/Fehar Sidebar */
62 .menu-toggle-button {
63   position: absolute; /* Ou 'fixed' se quiser que fique visível ao rolar o mapa */
64   top: 15px;
65   left: 15px;
66   z-index: 1001; /* Acima de tudo */
67   background-color: var(--button-bg-color);
68   color: var(--text-color);
69   border: 1px solid var(--border-color);
70   border-radius: 4px;
71   padding: 8px 12px;
```

```
72  font-size: 1.5em; /* Tamanho do ícone/texto */
73  cursor: pointer;
74  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
75  transition: background-color 0.2s ease;
76 }
77 .menu-toggle-button:hover {
78  background-color: var(--button-hover-bg-color);
79 }
80 /* Sidebar (Menu Lateral) */
81 .sidebar {
82  position: absolute; /* Sobrepõe o mapa */
83  top: 0;
84  left: 0;
85  height: 100vh;
86  width: var(--sidebar-width);
87  background-color: var(--sidebar-bg-color);
88  padding: 20px;
89  box-shadow: 2px 0 5px rgba(0, 0, 0, 0.1);
90  transform: translateX(-100%); /* Escondido fora da tela à esquerda */
91  transition: transform var(--transition-speed) ease-in-out;
92  z-index: 1000; /* Abaixo do botão toggle, acima do mapa */
93  display: flex;
94  flex-direction: column;
95  overflow-y: auto; /* Permite rolagem interna se necessário */
96  box-sizing: border-box; /* Garante que padding não aumente a largura total */
97 }
98 .sidebar.is-open {
99  transform: translateX(0); /* Move para a posição visível */
100 }
101 /* Conteúdo da Sidebar */
102 .sidebar h1 {
103  font-size: 1.5rem;
104  margin-top: 40px; /* Espaço para o botão de toggle não sobrepor o título */
105  margin-bottom: 20px;
106 }
107 .sidebar .floor-selector {
108  margin-bottom: 15px;
```

```
109 }
110 .sidebar .floor-selector select {
111   width: 100%;
112   padding: 8px;
113   border-radius: 5px;
114   border: 1px solid var(--border-color);
115   box-sizing: border-box;
116 }
117 .sidebar .search-box {
118   width: 100%; /* Ocupa toda a largura disponível no padding */
119   padding: 10px;
120   margin-bottom: 20px;
121   border: 1px solid var(--border-color);
122   border-radius: 5px;
123   box-sizing: border-box;
124 }
125 .sidebar .local-item {
126   padding: 10px;
127   margin-bottom: 10px;
128   background-color: white
129   border: 1px solid #eee; /* Borda sutil */
130   border-radius: 5px;
131   cursor: pointer;
132   transition: background-color 0.2s ease, color 0.2s ease;
133   box-shadow: 0 1px 3px rgba(0,0,0,0.05); /* Sombra mais suave */
134 }
135 .sidebar .local-item:hover {
136   background-color: #e9ecf;
137 }
138 .sidebar .local-item.selected {
139   background-color: var(--primary-accent-color);
140   color: white
141   font-weight: bold;
142   border-color: var(--primary-accent-color);
143 }
144 /* Contêiner do Mapa */
145 .map-container {
```

```
146 flex-grow: 1; /* Ocupa o espaço restante */
147 position: relative; /* Para posicionar elementos internos */
148 width: 100%; /* Garante que ocupe toda a largura disponível */
149 height: 100vh; /* Ocupa toda a altura */
150 overflow: hidden; /* Essencial para o pan/zoom funcionar corretamente */
151 background-color: □ #e0e0e0; /* Cor de fundo enquanto a imagem carrega ou se não cobrir tudo */
152 }
153 /* Imagem do Mapa */
154 .map-image {
155 width: 100%;
156 height: 100%;
157 background-size: contain; /* Ou 'cover' dependendo do efeito desejado */
158 background-repeat: no-repeat;
159 background-position: center center;
160 position: absolute; /* Permite que o transform funcione corretamente */
161 top: 0;
162 left: 0;
163 transform-origin: center center; /* Ponto de referência para zoom/pan */
164 cursor: grab; /* Indica que pode ser arrastado */
165 /* A propriedade 'transform' (scale, translate) será aplicada via JS */
166 }
167 .map-image:active {
168 cursor: grabbing; /* Indica que está sendo arrastado */
169 }
170 /* Marcadores no Mapa */
171 .location-marker,
172 .user-marker {
173 position: absolute;
174 width: 15px; /* Tamanho unificado */
175 height: 15px;
176 border-radius: 50%;
177 border: 2px solid white;
178 transform: translate(-50%, -50%); /* Centraliza no ponto exato */
179 box-shadow: 0 0 5px rgba(0, 0, 0, 0.5);
180 z-index: 20; /* Acima do caminho, abaixo do popup */
181 }
182 .location-marker {
```

```
183 background-color: var(--marker-red);
184 cursor: pointer;
185 }
186 .user-marker {
187 background-color: var(--marker-blue);
188 z-index: 21; /* ligeiramente acima do marcador de local, se necessário */
189 }
190 /* Linha do Caminho/Rota */
191 .path-line {
192 position: absolute;
193 height: 4px; /* Espessura da linha */
194 background-color: var(--path-color);
195 transform-origin: 0 50%; /* Origem para rotação/comprimento */
196 z-index: 10; /* Abaixo dos marcadores */
197 pointer-events: none; /* Linha não deve ser clicável */
198 }
199 /* Opcional: Indicador de direção (se desejar) */
200
201 .path-line::after {
202 content: '';
203 position: absolute;
204 width: 0;
205 height: 0;
206 right: 0;
207 top: 50%;
208 transform: translate(0, -50%);
209 border-top: 4px solid transparent;
210 border-bottom: 4px solid transparent;
211 border-left: 6px solid var(--path-color);
212 }
213
214 /* Popup de Informação do Marcador */
215 .location-popup {
216 position: absolute;
217 background-color: rgba(0, 0, 0, 0.75); /* Fundo semi-transparente */
218 color: white
219 padding: 5px 10px;
```

```
220 border-radius: 4px;  
221 font-size: 0.9em;  
222 white-space: nowrap; /* Impede quebra de linha */  
223 transform: translate(-50%, -130%); /* Posiciona um pouco acima do marcador */  
224 z-index: 30; /* Acima dos marcadores */  
225 pointer-events: none; /* Popup não interfere com cliques no mapa/marcador */  
226 box-shadow: 0 2px 4px rgba(0,0,0,0.2);  
227 }  
228 /* Controles do Mapa (Zoom, etc.) */  
229 .controls {  
230 position: absolute;  
231 bottom: 20px;  
232 right: 20px;  
233 z-index: 50; /* Acima do mapa e elementos, mas abaixo da sidebar/botão */  
234 display: flex;  
235 flex-direction: column;  
236 gap: 5px; /* Espaçamento entre botões */  
237 background: rgba(255, 255, 255, 0.8); /* Fundo levemente transparente */  
238 padding: 10px;  
239 border-radius: 5px;  
240 box-shadow: 0 2px 4px rgba(0,0,0,0.1);  
241 }  
242 .controls button {  
243 padding: 8px 12px;  
244 background-color: var(--button-bg-color);  
245 border: 1px solid var(--border-color);  
246 border-radius: 4px;  
247 cursor: pointer;  
248 transition: background-color 0.2s ease;  
249 }  
250 .controls button:hover {  
251 background-color: var(--button-hover-bg-color);  
252 }  
253 /* Indicadores de Estado (Carregamento, Erro) */  
254 .loading,  
255 .error {  
position: absolute; /* Para centralizar sobre o mapa ou sidebar */
```

```
257 left: 50%;  
258 top: 50%;  
259 transform: translate(-50%, -50%);  
260 padding: 15px 20px;  
261 border-radius: 5px;  
262 text-align: center;  
263 font-weight: bold;  
264 z-index: 500; /* Acima da maioria dos elementos */  
265 box-shadow: 0 2px 10px rgba(0,0,0,0.1);  
266 }  
267 .loading {  
268 background: rgba(255, 255, 255, 0.9);  
269 color: var(--text-color);  
270 }  
271 .error {  
272 background-color: var(--error-bg-color);  
273 color: var(--error-color);  
274 border: 1px solid var(--error-color);  
275 }  
276  
277 /* Prompt para modo manual de localização */  
278 .manual-location-prompt {  
279 position: absolute;  
280 bottom: 20px;  
281 left: 50%;  
282 transform: translateX(-50%);  
283 background: rgba(0, 0, 0, 0.75);  
284 color: white  
285 padding: 10px 15px;  
286 border-radius: 4px;  
287 box-shadow: 0 2px 4px rgba(0, 0, 0, 0.3);  
288 font-weight: bold;  
289 z-index: 50;  
290 }  
291  
292 /* Conteúdo Principal */  
293 .main-content {
```

```
294 margin-left: 0;
295 transition: margin-left var(--transition-speed) ease-in-out;
296 width: 100%;
297 height: 100%;
298 position: relative;
299 }
300
301 /* Waypoint marker para debugging */
302 .waypoint-marker {
303 position: absolute;
304 width: 8px;
305 height: 8px;
306 background-color: #44bb44;
307 border-radius: 50%;
308 transform: translate(-50%, -50%);
309 z-index: 15;
310 }
311
312 /* Media queries para responsividade */
313 @media screen and (max-width: 768px) {
314 .sidebar {
315 width: var(--sidebar-width-mobile);
316 padding: 15px;
317 }
318
319 .sidebar h1 {
320 font-size: 1.3rem;
321 margin-top: 55px;
322 }
323
324 .menu-toggle-button {
325 top: 10px;
326 left: 10px;
327 padding: 7px 10px;
328 font-size: 1.2em;
329 }
330
```

```
331 .controls {
332   bottom: 15px;
333   right: 15px;
334   padding: 8px;
335 }
336
337 .controls button {
338   padding: 6px 10px;
339 }
340
341 /* Stack botões em dispositivos muito pequenos */
342 .top-nav {
343   flex-direction: column;
344   top: 10px;
345   right: 10px;
346   gap: 5px;
347 }
348
349 .nav-link {
350   padding: 6px 10px;
351   font-size: 0.9em;
352 }
353
354 .location-marker,
355 .user-marker {
356   width: 12px;
357   height: 12px;
358 }
359
360 /* Reduz tamanho do popup em telas pequenas */
361 .location-popup {
362   font-size: 0.8em;
363   padding: 4px 8px;
364 }
365
366 /* Ajusta loading/error para telas pequenas */
367 .loading,
```

```
368 .error {
369     padding: 10px 15px;
370     font-size: 0.9em;
371     max-width: 80vw;
372 }
373
374 .manual-location-prompt {
375     padding: 8px 12px;
376     font-size: 0.9em;
377     max-width: 80vw;
378 }
379 }
380
381 /* Para telas muito pequenas */
382 @media screen and (max-width: 480px) {
383     .sidebar {
384         width: 90vw;
385         padding: 10px;
386     }
387
388     .sidebar h1 {
389         font-size: 1.2rem;
390         margin-top: 50px;
391     }
392
393     .menu-toggle-button {
394         top: 8px;
395         left: 8px;
396     }
397
398     .controls {
399         bottom: 10px;
400         right: 10px;
401     }
402 }
```

.env

```
1 | VITE_GOOGLE_APPS_SCRIPT_URL=https://script.google.com/macros/s/AKfycbwB1PV-SS6VafK03N_ohVBi-PI0_eoVSBTbtLdpg10V3-  
| Dxu1PYSClITd1b-h0l0qJNk/exec
```

index.html

```
1 <!DOCTYPE html>
2 <html lang="pt-BR">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <title>Mapa Campus Russas</title>
7   </head>
8   <body>
9     <div id="app"></div>
10    <script type="module" src="/src/main.js"></script>
11  </body>
12 </html>
```

src/components/AlertMessage.vue

```
1 <template>
2   <transition name="fade">
3     <div
4       v-if="show"
5       :class="['alert', `alert-${type}`, isClosable ? 'closable' : '' ]"
6       role="alert"
7     >
8       <div class="alert-content">
9         <strong v-if="title" class="alert-title">{{ title }}</strong>
10        <p class="alert-message">{{ message }}</p>
11      </div>
12      <button
13        v-if="isClosable"
14        class="alert-close"
15        @click="close"
16        aria-label="Fechar"
17      >
18        &times;
19      </button>
20    </div>
21  </transition>
22</template>
23
24 <script setup>
25 import { defineProps, defineEmits, ref, watch, onMounted } from 'vue';
26
27 const props = defineProps({
28   message: {
29     type: String,
30     required: true
31   },
32   title: {
33     type: String,
34     default: ''
35   }
36 }
```

```
35 },
36 type: {
37   type: String,
38   default: 'info',
39   validator: (value) => ['info', 'success', 'warning', 'error'].includes(value)
40 },
41 isClosable: {
42   type: Boolean,
43   default: true
44 },
45 autoClose: {
46   type: Boolean,
47   default: false
48 },
49 duration: {
50   type: Number,
51   default: 5000
52 }
53 );
54
55 const emit = defineEmits(['close']);
56 const show = ref(true);
57
58 const close = () => {
59   show.value = false;
60   emit('close');
61 };
62
63 // Fecha automaticamente após a duração especificada
64 let timeoutId = null;
65
66 const startAutoCloseTimer = () => {
67   if (props.autoClose && props.duration > 0) {
68     timeoutId = setTimeout(() => {
69       close();
70     }, props.duration);
71 }
```

```
72 };
73
74 // Limpa o temporizador quando o componente é desmontado
75 onMounted(()) => {
76   startAutoCloseTimer();
77 });
78
79 // Reseta o temporizador quando as props mudam
80 watch(()) => [props.message, props.type], () => {
81   if (timeoutId) {
82     clearTimeout(timeoutId);
83   }
84   show.value = true;
85   startAutoCloseTimer();
86 });
87 </script>
88
89 <style scoped>
90 .alert {
91   display: flex;
92   align-items: flex-start;
93   justify-content: space-between;
94   padding: 12px 16px;
95   margin-bottom: 16px;
96   border-radius: 4px;
97   font-size: 0.95rem;
98   box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
99 }
100
101 .alert-content {
102   flex: 1;
103 }
104
105 .alert-title {
106   display: block;
107   margin-bottom: 4px;
108   font-weight: 600;
```

```
109 }
110
111 .alert-message {
112   margin: 0;
113   line-height: 1.4;
114 }
115
116 .alert-close {
117   background: transparent;
118   border: none;
119   font-size: 1.5rem;
120   padding: 0 8px;
121   margin-left: 8px;
122   cursor: pointer;
123   opacity: 0.7;
124   transition: opacity 0.2s;
125 }
126
127 .alert-close:hover {
128   opacity: 1;
129 }
130
131 /* Tipos de alerta */
132 .alert-info {
133   background-color: #e3f2fd;
134   border-left: 4px solid #2196f3;
135   color: #0d47a1;
136 }
137
138 .alert-success {
139   background-color: #e8f5e9;
140   border-left: 4px solid #4caf50;
141   color: #1b5e20;
142 }
143
144 .alert-warning {
145   background-color: #fff3e0;
```

```
146 border-left: 4px solid #ff9800;
147 color: #e65100;
148 }
149
150 .alert-error {
151 background-color: #ffebee;
152 border-left: 4px solid #f44336;
153 color: #b71c1c;
154 }
155
156 /* Animações */
157 .fade-enter-active,
158 .fade-leave-active {
159 transition: opacity 0.3s ease;
160 }
161
162 .fade-enter-from,
163 .fade-leave-to {
164 opacity: 0;
165 }
166
167 /* Media Query para telas menores */
168 @media (max-width: 768px) {
169 .alert {
170 padding: 10px 12px;
171 font-size: 0.9rem;
172 }
173 }
174 </style>
```

src/components/AppSidebar.vue

```
1 <script setup>
2 import { ref, computed, watch } from 'vue';
3 import LocationItem from './LocationItem.vue';
4
5 const props = defineProps({
6   isOpen: Boolean,
7   allLocais: Array,
8   floors: Array,
9   currentFloorId: String,
10  selectedLocalId: [Number, String, null],
11  loading: Boolean,
12  error: String
13 });
14
15 const emit = defineEmits(['update:currentFloorId', 'select-local', 'close']);
16
17 const searchTerm = ref('');
18
19 // Filtra locais baseado na busca E opcionalmente no andar atual
20 // Decisão: Mostrar sempre locais do andar atual + resultados da busca em outros andares
21 const filteredLocais = computed(() => {
22   const term = searchTerm.value.toLowerCase().trim();
23   if (!term) {
24     // Sem busca, mostra apenas os do andar atual
25     return props.allLocais.filter(local => local.andar === props.currentFloorId);
26   } else {
27     // Com busca, mostra os que batem com a busca (em qualquer andar)
28     return props.allLocais.filter(local =>
29       local.nome.toLowerCase().includes(term)
30     );
31   }
32 });
33
34
```

```
35 const internalCurrentFloor = ref(props.currentFloorId);
36
37 // Observa mudança externa no currentFloorId
38 watch(() => props.currentFloorId, (newVal) => {
39     internalCurrentFloor.value = newVal;
40 });
41
42
43 const changeFloor = () => {
44     //searchTerm.value = '' // Limpa busca ao trocar de andar? Opcional.
45     emit('update:currentFloorId', internalCurrentFloor.value);
46 };
47
48 const handleSelectLocal = (local) => {
49     emit('select-local', local);
50     // Opcional: Fechar sidebar ao selecionar?
51     // emit('close');
52 };
53
54 </script>
55
56 <template>
57     <div class="sidebar" :class="{ 'is-open': isOpen }">
58         <h1>Locais</h1>
59
60         <div class="floor-selector">
61             <select v-model="internalCurrentFloor" @change="changeFloor">
62                 <option v-for="floor in floors" :key="floor.id" :value="floor.id">
63                     {{ floor.nome }}
64                 </option>
65             </select>
66         </div>
67
68         <div class="search-container">
69             <input v-model="searchTerm" class="search-box" placeholder="Buscar local..." />
70         </div>
71     </div>
```

```
72      <div v-if="error" class="error">
73          {{ error }}
74      </div>
75
76      <div v-if="loading && !allLocais.length" class="status-message">Carregando locais...</div>
77
78      <div v-if="!loading && !filteredLocais.length && searchTerm" class="status-message">
79          Nenhum local encontrado para "{{ searchTerm }}".
80      </div>
81      <div v-if="!loading && !filteredLocais.length && !searchTerm" class="status-message">
82          Nenhum local neste andar.
83      </div>
84
85      <div class="locations-list">
86          <div
87              v-for="local in filteredLocais"
88              :key="local.id"
89              class="location-item"
90              :class="{ 'selected': selectedLocalId === local.id }"
91              @click="handleSelectLocal(local)"
92          >
93              {{ local.nome }}
94          </div>
95      </div>
96
97      </div>
98  </template>
99
100 <style scoped>
101 .sidebar {
102     width: 300px;
103     height: 100vh;
104     background: white;
105     position: fixed;
106     left: 0;
107     top: 0;
108     transform: translateX(-100%);
```

```
109     transition: transform 0.3s ease;
110     z-index: 1000;
111     box-shadow: 2px 0 5px rgba(0, 0, 0, 0.1);
112     padding: 20px;
113 }
114
115 .sidebar.is-open {
116     transform: translateX(0);
117 }
118
119 .floor-selector select {
120     width: 100%;
121     padding: 8px;
122     margin-bottom: 15px;
123     border: 1px solid #ddd;
124     border-radius: 4px;
125     background: white;
126 }
127
128 .search-container {
129     margin-bottom: 15px;
130 }
131
132 .search-box {
133     width: 100%;
134     padding: 8px;
135     border: 1px solid #ddd;
136     border-radius: 4px;
137 }
138
139 .locations-list {
140     max-height: calc(100vh - 200px);
141     overflow-y: auto;
142 }
143
144 .location-item {
145     padding: 10px;
```

```
146     margin: 5px 0;
147     background: #f8f9fa;
148     border: 1px solid #ddd;
149     border-radius: 4px;
150     cursor: pointer;
151     transition: all 0.2s ease;
152 }
153
154 .location-item:hover {
155     background: #e9ecf;
156 }
157
158 .location-item.selected {
159     background: #007bff;
160     color: white;
161     border-color: #0056b3;
162 }
163
164 .error {
165     color: #dc3545;
166     padding: 10px;
167     margin-bottom: 15px;
168     background: #f8d7da;
169     border-radius: 4px;
170 }
171
172 .status-message {
173     padding: 10px;
174     color: #6c757d;
175     text-align: center;
176 }
177 </style>
```

src/components/LocationAdmin.vue

```
1 <template>
2   <div class="admin-container">
3     <router-link to="/">Voltar para o Mapa</router-link>
4     <h1>Admin de Locais</h1>
5
6     <div v-if="localLoading" class="loading">Carregando...</div>
7     <div v-if="localError" class="error">{{ localError }}</div>
8     <div v-if="localSuccessMessage" class="success">{{ localSuccessMessage }}</div>
9
10    <button @click="showAddForm" :disabled="localLoading || showForm" class="btn btn-primary">Adicionar Novo Local</button>
11
12    <div v-if="showForm" class="form-section">
13      <h2>{{ isEditing ? 'Editar Local' : 'Adicionar Local' }}</h2>
14      <form @submit.prevent="handleSubmit">
15        <div class="form-group" v-if="isEditing">
16          <label for="id">ID:</label>
17          <input type="text" id="id" :value="formData.id" disabled>
18        </div>
19        <div class="form-group">
20          <label for="tipo">Tipo:</label>
21          <select id="tipo" v-model="formData.tipo" required>
22            <option v-for="tipo in tiposDisponiveis" :key="tipo" :value="tipo">
23              {{ tipo.charAt(0).toUpperCase() + tipo.slice(1) }}
24            </option>
25          </select>
26        </div>
27        <div class="form-group">
28          <label for="nome">Nome:</label>
29          <input type="text" id="nome" v-model="formData.nome" required>
30        </div>
31        <div class="form-group">
32          <label for="andar">Andar:</label>
33          <select id="andar" v-model="formData.andar" required>
34            <option v-for="andar in andaresDisponiveis" :key="andar" :value="andar">
```

```

35      {{ andar.charAt(0).toUpperCase() + andar.slice(1) }}
36  </option>
37  </select>
38 </div>
39 <div class="form-group">
40   <label for="x">Coordenada X (0-100):</label>
41   <input type="number" id="x" v-model="formData.x" required step="0.01" min="0" max="100">
42 </div>
43 <div class="form-group">
44   <label for="y">Coordenada Y (0-100):</label>
45   <input type="number" id="y" v-model="formData.y" required step="0.01" min="0" max="100">
46 </div>
47 <div class="form-actions">
48   <button type="submit" :disabled="localLoading" class="btn btn-success">{{ isEditing ? 'Salvar Alterações' :
49 'Criar Local' }}</button>
50   <button type="button" @click="cancelForm" :disabled="localLoading" class="btn btn-secondary">Cancelar</button>
51 </div>
52 </form>
53 </div>
54 <h2>Locais Existentes</h2>
55 <div class="table-container">
56   <table class="locations-table">
57     <thead>
58       <tr>
59         <th>ID</th>
60         <th>Tipo</th>
61         <th>Nome</th>
62         <th>X</th>
63         <th>Y</th>
64         <th>Andar</th>
65         <th>Ações</th>
66       </tr>
67     </thead>
68     <tbody>
69       <tr v-if="locations.length === 0 && !localLoading && !localError">
70         <td colspan="7" style="text-align: center;">Nenhum local cadastrado.</td>

```

```
71      </tr>
72      <tr v-if="locations.length === 0 && localError">
73          <td colspan="7" style="text-align: center; color: red;">Falha ao carregar locais.</td>
74      </tr>
75      <tr v-for="loc in locations" :key="loc.id">
76          <td>{{ loc.id }}</td>
77          <td>{{ loc.tipo || 'N/A' }}</td>
78          <td>{{ loc.nome }}</td>
79          <td>{{ loc.x?.toFixed ? loc.x.toFixed(2) : loc.x }}</td>
80          <td>{{ loc.y?.toFixed ? loc.y.toFixed(2) : loc.y }}</td>
81          <td>{{ loc.andar }}</td>
82          <td>
83              <button @click="showEditForm(loc)" :disabled="localLoading || showForm" class="btn btn-warning btn-sm">Editar</button>
84              <button @click="confirmDelete(loc.id)" :disabled="localLoading || showForm" class="btn btn-danger btn-sm">Excluir</button>
85          </td>
86      </tr>
87  </tbody>
88 </table>
89 </div>
90 </div>
91 </template>
92
93 <script setup>
94 import { ref, onMounted, watch } from 'vue';
95 import { useRouter } from 'vue-router';
96 import locationService from '@/services/LocationService.js'; // Importe a instância
97
98 const router = useRouter();
99 const locations = ref([]);
100 const localLoading = ref(false);
101 const localError = ref(null);
102 const localSuccessMessage = ref(null);
103
104 const showForm = ref(false);
105 const isEditing = ref(false);
```

```
106 const tiposDisponiveis = ['location', 'Waypoints'];
107 const andaresDisponiveis = ['terreo', 'primeiro'];
108
109 const saveFormToStorage = () => {
110   localStorage.setItem('locationFormData', JSON.stringify({
111     ...formData.value,
112     showForm: showForm.value,
113     isEditing: isEditing.value,
114   }));
115 };
116
117 const loadFormFromStorage = () => {
118   const savedData = localStorage.getItem('locationFormData');
119   if (savedData) {
120     const parsed = JSON.parse(savedData);
121     showForm.value = parsed.showForm;
122     isEditing.value = parsed.isEditing;
123     delete parsed.showForm;
124     delete parsed.isEditing;
125     formData.value = parsed;
126   }
127 };
128
129 const validateCoordinates = (value) => {
130   if (value === null || value === undefined) return 0;
131   const num = parseFloat(value);
132   return Math.max(0, Math.min(100, num));
133 };
134
135 const formData = ref({
136   id: null,
137   tipo: 'location',
138   nome: '',
139   andar: 'terreo',
140   x: null,
141   y: null,
142 });
```

```
143
144 const resetForm = () => {
145   formData.value = {
146     id: null,
147     tipo: 'location',
148     nome: '',
149     andar: 'terreo',
150     x: null,
151     y: null,
152   };
153   localStorage.removeItem('locationFormData');
154 };
155
156 onMounted(() => {
157   loadLocations();
158   loadFormFromStorage();
159 });
160
161 watch([formData, showForm, isEditing], () => {
162   saveFormToStorage();
163 }, { deep: true });
164
165 function clearLocalMessages() {
166   setTimeout(() => {
167     localError.value = null;
168     localSuccessMessage.value = null;
169   }, 5000);
170 }
171
172 async function loadLocations() {
173   localLoading.value = true;
174   localError.value = null;
175   locations.value = [];
176   try {
177     localLoading.value = true;
178     const result = await locationService.getAllLocations(); // Use a instância
179     locations.value = result;
```

```
180 } catch (error) {
181     localError.value = `Erro ao carregar locais: ${error.message}`;
182     locations.value = [];
183 } finally {
184     localLoading.value = false;
185     if (localError.value) clearLocalMessages();
186 }
187 }
188
189 async function handleAddLocation() {
190     localLoading.value = true;
191     localError.value = null;
192     localSuccessMessage.value = null;
193     try {
194         const dataToSend = {
195             nome: formData.value.nome,
196             andar: formData.value.andar,
197             x: formData.value.x,
198             y: formData.value.y,
199         };
200         const success = await locationService.createLocation(dataToSend); // Use a instância
201         if (success) {
202             localSuccessMessage.value = 'Local adicionado com sucesso!';
203             await loadLocations();
204             cancelForm();
205             clearLocalMessages();
206         } else {
207             localError.value = 'Falha ao adicionar local.';
208             clearLocalMessages();
209         }
210     } catch (err) {
211         console.error('Erro ao adicionar local:', err);
212         localError.value = `Erro ao criar: ${err.message}`;
213         clearLocalMessages();
214     } finally {
215         localLoading.value = false;
216     }
}
```

```
217 }
218
219 async function handleUpdateLocation() {
220   localLoading.value = true;
221   localError.value = null;
222   localSuccessMessage.value = null;
223   try {
224     const dataToSend = {
225       id: formData.value.id,
226       nome: formData.value.nome,
227       andar: formData.value.andar,
228       x: formData.value.x,
229       y: formData.value.y,
230     };
231     const success = await locationService.updateLocation(dataToSend.id, dataToSend); // Use a instância
232     if (success) {
233       localSuccessMessage.value = 'Local atualizado com sucesso!';
234       await loadLocations();
235       cancelForm();
236       clearLocalMessages();
237     } else {
238       localError.value = 'Falha ao atualizar local.';
239       clearLocalMessages();
240     }
241   } catch (err) {
242     console.error('Erro ao atualizar local:', err);
243     localError.value = `Erro ao atualizar: ${err.message}`;
244     clearLocalMessages();
245   } finally {
246     localLoading.value = false;
247   }
248 }
249
250 async function handleDeleteLocation(id) {
251   localLoading.value = true;
252   localError.value = null;
253   localSuccessMessage.value = null;
```

```
254 try {
255   const success = await locationService.deleteLocation(id); // Use a instância
256   if (success) {
257     localSuccessMessage.value = 'Local excluído com sucesso!';
258     await loadLocations();
259     clearLocalMessages();
260   } else {
261     localError.value = 'Falha ao excluir local.';
262     clearLocalMessages();
263   }
264 } catch (err) {
265   console.error('Erro ao excluir local:', err);
266   localError.value = `Erro ao excluir: ${err.message}`;
267   clearLocalMessages();
268 } finally {
269   localLoading.value = false;
270 }
271 }
272
273 const handleSubmit = async () => {
274   try {
275     if (localLoading.value) return;
276     formData.value.x = validateCoordinates(formData.value.x);
277     formData.value.y = validateCoordinates(formData.value.y);
278     localLoading.value = true;
279     localError.value = '';
280     if (isEditing.value) {
281       await handleUpdateLocation();
282     } else {
283       await handleAddLocation();
284     }
285     resetForm();
286     showForm.value = false;
287     isEditing.value = false;
288     localStorage.removeItem('locationFormData');
289     await loadLocations();
290   } catch (error) {
```

```
291     console.error('Erro ao salvar local:', error);
292     localError.value = 'Erro ao salvar local. Por favor, tente novamente.';
293 } finally {
294     localLoading.value = false;
295 }
296 };
297
298 function showAddForm() {
299     resetForm();
300     isEditing.value = false;
301     showForm.value = true;
302     localError.value = null;
303     localSuccessMessage.value = null;
304 }
305
306 function showEditForm(location) {
307     isEditing.value = true;
308     formData.value = {
309         id: location.id,
310         nome: location.nome || '',
311         andar: location.andar || '',
312         x: location.x,
313         y: location.y,
314     };
315     showForm.value = true;
316     localError.value = null;
317     localSuccessMessage.value = null;
318 }
319
320 function cancelForm() {
321     resetForm();
322     showForm.value = false;
323     isEditing.value = false;
324     localStorage.removeItem('locationFormData');
325 }
326
327 function confirmDelete(id) {
```

```
328 if (window.confirm(`Tem certeza que deseja excluir o local com ID "${id}"? Esta ação não pode ser desfeita.`)) {  
329     handleDeleteLocation(id);  
330 }  
331 }  
332 </script>  
333  
334 <style scoped>  
335 /* ... (seus estilos CSS) ... */  
336 </style>  
337  
338 <style scoped>  
339 .admin-container {  
340     padding: 20px;  
341     height: 100vh;  
342     overflow-y: auto;  
343     max-width: 1200px;  
344     margin: 0 auto;  
345     font-family: system-ui, -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, sans-serif;  
346     background-color: #f8f9fa;  
347 }  
348  
349 .admin-container h1 {  
350     color: #2c3e50;  
351     margin-bottom: 1.5rem;  
352     font-size: 2rem;  
353     font-weight: 600;  
354 }  
355  
356 .admin-container h2 {  
357     color: #2c3e50;  
358     margin: 1.5rem 0;  
359     font-size: 1.5rem;  
360     font-weight: 500;  
361 }  
362  
363 a {  
364     color: #3498db;
```

```
365 text-decoration: none;
366 margin-right: 10px;
367 padding: 8px 12px;
368 border-radius: 4px;
369 background-color: #fff;
370 box-shadow: 0 1px 3px rgba(0,0,0,0.1);
371 transition: all 0.2s ease;
372 }
373
374 a:hover {
375   background-color: #3498db;
376   color: #fff;
377   text-decoration: none;
378 }
379
380 .form-section {
381   margin: 20px 0;
382   padding: 25px;
383   background-color: #fff;
384   border-radius: 8px;
385   box-shadow: 0 2px 4px rgba(0,0,0,0.1);
386 }
387
388 .form-group {
389   margin-bottom: 1.2rem;
390 }
391
392 label {
393   display: block;
394   margin-bottom: 0.5rem;
395   color: #2c3e50;
396   font-weight: 500;
397   font-size: 0.95rem;
398 }
399
400 input[type="text"],
401 input[type="number"],
```

```
402 select {
403   width: 100%;
404   padding: 10px 12px;
405   border: 2px solid #e2e8f0;
406   border-radius: 6px;
407   font-size: 1rem;
408   background-color: #fff;
409   transition: all 0.2s ease;
410 }
411
412 input[type="text"]:hover,
413 input[type="number"]:hover {
414   border-color: #cbd5e0;
415 }
416
417 input[type="text"]:focus,
418 input[type="number"]:focus {
419   outline: none;
420   border-color: #3498db;
421   box-shadow: 0 0 0 3px rgba(52, 152, 219, 0.2);
422 }
423
424 .form-actions {
425   margin-top: 1.5rem;
426   display: flex;
427   gap: 12px;
428 }
429
430 /* Estilos para a tabela */
431 .table-container {
432   margin-top: 20px;
433   max-height: calc(100vh - 500px);
434   overflow-y: auto;
435   background-color: #fff;
436   border-radius: 8px;
437   box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
438   scrollbar-width: thin;
```

```
439 scrollbar-color: #94a3b8 #f1f5f9;
440 }
441
442 .table-container::-webkit-scrollbar {
443   width: 8px;
444 }
445
446 .table-container::-webkit-scrollbar-track {
447   background: #f1f5f9;
448   border-radius: 8px;
449 }
450
451 .table-container::-webkit-scrollbar-thumb {
452   background-color: #94a3b8;
453   border-radius: 8px;
454   border: 2px solid #f1f5f9;
455 }
456
457 .locations-table {
458   width: 100%;
459   border-collapse: separate;
460   border-spacing: 0;
461 }
462
463 .locations-table thead {
464   position: sticky;
465   top: 0;
466   z-index: 1;
467 }
468
469 .locations-table th {
470   background-color: #f8fafc;
471   color: #1e293b;
472   font-weight: 600;
473   padding: 16px;
474   text-align: left;
475   border-bottom: 2px solid #e2e8f0;
```

```
476 white-space: nowrap;  
477 }  
478  
479 .locations-table td {  
480   padding: 14px 16px;  
481   border-bottom: 1px solid #e2e8f0;  
482   color: #475569;  
483 }  
484  
485 .locations-table tr:last-child td {  
486   border-bottom: none;  
487 }  
488  
489 .locations-table tbody tr {  
490   transition: all 0.2s ease;  
491 }  
492  
493 .locations-table tbody tr:hover {  
494   background-color: #f1f5f9;  
495 }  
496  
497 .locations-table td button {  
498   margin-right: 8px;  
499   padding: 6px 12px;  
500   border-radius: 4px;  
501   font-size: 0.875rem;  
502   transition: all 0.2s ease;  
503 }  
504  
505 /* Estilos para os botões */  
506 .btn {  
507   padding: 8px 16px;  
508   border: none;  
509   border-radius: 6px;  
510   font-weight: 500;  
511   cursor: pointer;  
512   transition: all 0.2s ease;
```

```
513 }
514
515 .btn:disabled {
516   opacity: 0.6;
517   cursor: not-allowed;
518 }
519
520 .btn-primary {
521   background-color: #3498db;
522   color: white;
523 }
524
525 .btn-primary:hover:not(:disabled) {
526   background-color: #2980b9;
527 }
528
529 .btn-success {
530   background-color: #2ecc71;
531   color: white;
532 }
533
534 .btn-success:hover:not(:disabled) {
535   background-color: #27ae60;
536 }
537
538 .btn-warning {
539   background-color: #f1c40f;
540   color: #2c3e50;
541 }
542
543 .btn-warning:hover:not(:disabled) {
544   background-color: #f39c12;
545 }
546
547 .btn-danger {
548   background-color: #e74c3c;
549   color: white;
```

```
550 }
551
552 .btn-danger:hover:not(:disabled) {
553   background-color: #c0392b;
554 }
555
556 .btn-secondary {
557   background-color: #95a5a6;
558   color: white;
559 }
560
561 .btn-secondary:hover:not(:disabled) {
562   background-color: #7f8c8d;
563 }
564 </style>
```

src/components/LocationItem.vue

```
1 <script setup>
2 import { computed } from 'vue';
3
4 const props = defineProps({
5   local: {
6     type: Object,
7     required: true
8   },
9   isSelected: {
10     type: Boolean,
11     default: false
12   }
13 });
14
15 const emit = defineEmits(['select']);
16
17 const selectThisLocal = () => {
18   emit('select', props.local);
19 };
20
21 const classes = computed(() => ({
22   'local-item': true,
23   'selected': props.isSelected
24 }));
25 </script>
26
27 <template>
28   <div :class="classes" @click="selectThisLocal">
29     <strong>{{ local.nome }}</strong>
30     <span v-if="local.andar !== 'terreo'"> ({{ local.andar === 'primeiro' ? '1º' : local.andar }} Andar)</span>
31   </div>
32 </template>
33
34 <style scoped>
```

```
35 /* Estilos específicos para local-item já estão em main.css
36 .local-item {
37 }
38 }
39 .local-item.selected {
40 }
41 }*/
42 span {
43     font-size: 0.8em;
44     color: #666;
45 }
46 .local-item.selected span {
47     color: white; /* Ajusta cor do andar quando selecionado */
48 }
49
50 </style>
```

src/components/MapControls.vue

```
1 <script setup>
2 const emit = defineEmits(['zoom-in', 'zoom-out', 'update-location', 'set-location-manual']);
3 </script>
4
5 <template>
6   <div class="controls">
7     <button @click="emit('zoom-in')" title="Aproximar">+</button>
8     <button @click="emit('zoom-out')" title="Afastar">-</button>
9     <button @click="emit('update-location')" title="Obter localização automática">GPS</button>
10    <button @click="emit('set-location-manual')" title="Definir localização manualmente">Manual</button>
11  </div>
12 </template>
13
14 <style scoped>
15 /* Estilos dos controlos já estão em main.css */
16 button {
17   font-weight: bold;
18 }
19 </style>
```

```
75 });
76 // Estilo do cursor quando arrastando
77 const isDragging = ref(false);
78 const handleMouseDown = () => {
79   isDragging.value = true;
80 };
81 const handleMouseUp = () => {
```

src/composables/useGeolocation.js

```
1 import { ref } from 'vue';
2
3 // --- Configurações ---
4 // Coordenadas aproximadas do ponto de referência no Campus UFC Russas (Ex: Centro)
5 // !!! SUBSTITUIR PELAS COORDENADAS REAIS DO SEU PONTO DE REFERÊNCIA !!!
6 const campusReferenceCoordinates = {
7   latitude: -4.944444, // Latitude do Campus UFC Russas
8   longitude: -37.955556, // Longitude do Campus UFC Russas
9 };
10
11 // Dimensões em metros (aproximadas) cobertas pelo mapa a partir do ponto de referência
12 // !!! AJUSTAR COM BASE NO SEU MAPA E PONTO DE REFERÊNCIA !!!
13 // Se o ponto de referência é (0,0), estas são as distâncias máximas X e Y visíveis no mapa.
14 // Se o ponto de referência é o centro (50,50), ajuste a lógica de conversão.
15 // Vamos assumir que (0,0) no mapa % corresponde a campusReferenceCoordinates
16 const campusDimensions = {
17   widthMeters: 500, // Largura aproximada do campus em metros
18   heightMeters: 400, // Altura aproximada do campus em metros
19 };
20 // --- Fim Configurações ---
21
22
23 export function useGeolocation() {
24   const userPosition = ref(null); // Posição no mapa { x: number, y: number }
25   const geolocationError = ref(null);
26   const isGettingLocation = ref(false);
27
28   /**
29    * Converte coordenadas geográficas (lat/lng) para coordenadas percentuais no mapa.
30    * Assume que campusReferenceCoordinates corresponde ao canto superior esquerdo (0%, 0%) do mapa.
31    */
32   const convertGeoToMapCoordinates = (lat, lng) => {
33     // Fatores de conversão (aproximados)
34     const metersPerLatDegree = 111320;
```

```
35 const metersPerLngDegree = 111320 * Math.cos((campusReferenceCoordinates.latitude * Math.PI) / 180);
36
37 // Calcula a diferença em metros da referência
38 const latDiff = lat - campusReferenceCoordinates.latitude;
39 const lngDiff = lng - campusReferenceCoordinates.longitude;
40
41 // Distância em metros (Y cresce para baixo em geo, para cima em lat)
42 const distanceY = -latDiff * metersPerLatDegree; // Invertido
43 const distanceX = lngDiff * metersPerLngDegree;
44
45 // Converte para porcentagem (0-100)
46 let x = (distanceX / campusDimensions.widthMeters) * 100;
47 let y = (distanceY / campusDimensions.heightMeters) * 100;
48
49 // Limita os valores entre 0 e 100%
50 x = Math.max(0, Math.min(100, x));
51 y = Math.max(0, Math.min(100, y));
52
53 console.log(`Geo (${lat.toFixed(5)}, ${lng.toFixed(5)}) -> Map (${x.toFixed(2)}%, ${y.toFixed(2)}%)`);
54 return { x, y };
55};
56
57 /**
58 * Tenta obter a localização geográfica do usuário e convertê-la para o mapa.
59 */
60 const getUserLocationAuto = () => {
61   if (!navigator.geolocation) {
62     geolocationError.value = "Geolocalização não é suportada por este navegador.";
63     userPosition.value = null; // Garante que não há posição antiga
64     return;
65   }
66
67   isGettingLocation.value = true;
68   geolocationError.value = "Obtendo sua localização...";
69   userPosition.value = null; // Limpa posição anterior enquanto busca
70
71   navigator.geolocation.getCurrentPosition(
```

```
72  (position) => {
73    const userLat = position.coords.latitude;
74    const userLng = position.coords.longitude;
75    console.log(`Localização geográfica recebida: ${userLat}, ${userLng}`);
76
77    userPosition.value = convertGeoToMapCoordinates(userLat, userLng);
78    geolocationError.value = null; // Limpa erro/mensagem
79    isGettingLocation.value = false;
80  },
81  (err) => {
82    console.error("Erro ao obter geolocalização:", err);
83    switch (err.code) {
84      case err.PERMISSION_DENIED:
85        geolocationError.value = "Permissão de localização negada.";
86        break;
87      case err.POSITION_UNAVAILABLE:
88        geolocationError.value = "Informação de localização indisponível.";
89        break;
90      case err.TIMEOUT:
91        geolocationError.value = "Tempo esgotado ao buscar localização.";
92        break;
93      default:
94        geolocationError.value = "Erro desconhecido ao obter localização.";
95        break;
96    }
97    // Definir posição manual pode ser uma alternativa aqui
98    // error.value += " Você pode definir sua localização manualmente clicando no mapa.";
99    userPosition.value = null; // Garante que não usa posição inválida
100   isGettingLocation.value = false;
101 },
102 {
103   enableHighAccuracy: true, // Tenta obter a localização mais precisa
104   timeout: 10000, // Tempo máximo para obter (10 segundos)
105   maximumAge: 0, // Força obter uma nova localização (não usa cache)
106 }
107 );
108 };
```

```
109
110 /**
111 * Define a posição do usuário manualmente com base em um clique no mapa.
112 * @param {number} clickXPercent - Coordenada X do clique em porcentagem (0-100).
113 * @param {number} clickYPercent - Coordenada Y do clique em porcentagem (0-100).
114 */
115 const setUserLocationManually = (clickXPercent, clickYPercent) => {
116     const finalX = Math.max(0, Math.min(100, clickXPercent));
117     const finalY = Math.max(0, Math.min(100, clickYPercent));
118     userPosition.value = { x: finalX, y: finalY };
119     geolocationError.value = null; // Limpa qualquer erro anterior
120     console.log(`Localização definida manualmente: X: ${finalX.toFixed(2)}%, Y: ${finalY.toFixed(2)}%`);
121 }
122
123
124 return {
125     userPosition,          // ref({x, y} | null)
126     geolocationError,      // ref(string | null)
127     isGettingLocation,     // ref(boolean)
128     getUserLocationAuto,   // function
129     setUserLocationManually // function (xPerc, yPerc)
130 };
131 }
```

src/composables/useLocalStorage.js

```
1 import { ref, watch } from 'vue';
2
3 /**
4  * Composable para gerenciar dados no localStorage
5  * @param {string} key - Chave para armazenar o valor no localStorage
6  * @param {any} defaultValue - Valor padrão se não existir item no localStorage
7  * @returns {object} Objeto contendo valor reativo e métodos para manipulação
8 */
9 export function useLocalStorage(key, defaultValue = null) {
10    // Recupera valor do localStorage ou usa o valor padrão
11    const getStoredValue = () => {
12        try {
13            const item = window.localStorage.getItem(key);
14            return item ? JSON.parse(item) : defaultValue;
15        } catch (error) {
16            console.error(`Erro ao ler "${key}" do localStorage:`, error);
17            return defaultValue;
18        }
19    };
20
21    // Cria referência reativa com o valor inicial
22    const storedValue = ref(getStoredValue());
23
24    // Função para atualizar valor no localStorage
25    const updateValue = (newValue) => {
26        try {
27            if (newValue === null || newValue === undefined) {
28                window.localStorage.removeItem(key);
29            } else {
30                window.localStorage.setItem(key, JSON.stringify(newValue));
31            }
32            storedValue.value = newValue;
33            return true;
34        } catch (error) {
```

```
35     console.error(`Erro ao salvar "${key}" no localStorage:`, error);
36     return false;
37   }
38 };
39
40 // Atualiza localStorage quando o valor reativo muda
41 watch(storedValue, (newValue) => {
42   updateValue(newValue);
43 });
44
45 // Função para remover do localStorage
46 const removeItem = () => {
47   try {
48     window.localStorage.removeItem(key);
49     storedValue.value = defaultValue;
50     return true;
51   } catch (error) {
52     console.error(`Erro ao remover "${key}" do localStorage:`, error);
53     return false;
54   }
55 };
56
57 // Retorna valor reativo e funções de manipulação
58 return {
59   value: storedValue,
60   updateValue,
61   removeItem
62 };
63 }
```

```
1 import { ref } from 'vue';
2
3 export function useMapInteraction(initialScale = 1, minScale = 0.5, maxScale = 3, zoomStep = 1.2) {
4     const mapScale = ref(initialScale);
5     const popupInfo = ref({ visible: false, x: 0, y: 0, text: "" });
6     const settingUserLocationMode = ref(false); // Novo estado para modo de definição manual
7
8     const zoomIn = () => {
9         mapScale.value = Math.min(mapScale.value * zoomStep, maxScale);
10    };
11
12    const zoomOut = () => {
13        mapScale.value = Math.max(mapScale.value / zoomStep, minScale);
14    };
15
16    const showPopup = (local) => {
17        if (local && typeof local.x === 'number' && typeof local.y === 'number') {
18            popupInfo.value = { visible: true, x: local.x, y: local.y, text: local.nome };
19        } else {
20            console.warn("Tentativa de mostrar popup para local inválido:", local);
21            popupInfo.value = { visible: false, x: 0, y: 0, text: "" };
22        }
23    };
24
25    const hidePopup = () => {
26        popupInfo.value.visible = false;
27    };
28
29 /**
30  * Processa um clique no mapa. Pode fechar menu, definir localização manual, etc.
31  * @param {MouseEvent} event - O evento de clique.
32  * @param {HTMLElement|null} mapElement - O elemento do mapa para calcular o BoundingClientRect.
33  * @param {Function} callbackSetLocation - Função para chamar ao definir localização manual.
34  * @param {Function} callbackCloseMenu - Função para chamar se precisar fechar o menu.
35  */
36    const handleMapClick = (event, mapElement, callbackSetLocation, callbackCloseMenu) => {
```

```
37 // 1. Tenta fechar o menu se clicar fora dele
38 if (callbackCloseMenu) {
39     callbackCloseMenu(); // App.vue decide se fecha ou não
40 }
41
42 // 2. Se estiver no modo de definir localização manual
43 if (settingUserLocationMode.value) {
44     // Verifica se o elemento do mapa está disponível
45     if (!mapElement) {
46         console.error("Elemento do mapa não disponível para calcular clique.");
47         settingUserLocationMode.value = false; // Sai do modo
48         return;
49     }
50
51     // Obtém o BoundingClientRect diretamente do elemento
52     const mapRect = mapElement.getBoundingClientRect();
53     if (!mapRect) {
54         console.error("Map rect não disponível para calcular clique.");
55         settingUserLocationMode.value = false; // Sai do modo
56         return;
57     }
58
59     // Calcula coordenadas do clique relativas ao elemento da imagem do mapa
60     const clickXRelative = event.clientX - mapRect.left;
61     const clickYRelative = event.clientY - mapRect.top;
62
63     // Converte coordenadas do clique para coordenadas relativas à imagem *não escalada*
64     // Precisamos considerar a origem da transformação (centro) e a escala
65     const originX = mapRect.width / 2;
66     const originY = mapRect.height / 2;
67     const scale = mapScale.value;
68
69     // Posição na imagem como se não houvesse escala (em pixels, relativa ao canto sup esq)
70     const imageX = (clickXRelative - originX) / scale + originX;
71     const imageY = (clickYRelative - originY) / scale + originY;
72
73     // Converte para porcentagem (0-100) relativa à dimensão da imagem
```

```
74  const xPercent = (imageX / mapRect.width) * 100;
75  const yPercent = (imageY / mapRect.height) * 100;
76
77  // Chama a função de callback para efetivamente definir a localização
78  if (callbackSetLocation) {
79      // Passa as coordenadas calculadas em %
80      callbackSetLocation(xPercent, yPercent);
81  }
82
83  settingUserLocationMode.value = false; // Sai do modo após definir
84 }
85 // 3. Lógica adicional de clique no mapa (ex: deselecionar local?) pode ser adicionada aqui
86 };
87
88 /** Ativa o modo de definir localização manual */
89 const enableSetUserLocationMode = () => {
90     settingUserLocationMode.value = true;
91 };
92
93 return {
94     mapScale,           // ref(number)
95     popupInfo,          // ref({ visible, x, y, text })
96     settingUserLocationMode, // ref(boolean) - Indica se está esperando clique para definir local
97
98     zoomIn,             // function
99     zoomOut,            // function
100    showPopup,           // function(local)
101    hidePopup,           // function()
102    handleMapClick,       // function(event, mapElement, callbackSetLocation, callbackCloseMenu)
103    enableSetUserLocationMode, // function()
104 };
105 }
```

src/composables/useMapRouting.js

```
1 import { ref, watch } from 'vue';
2
3 export function useMapRouting(userPositionRef, selectedLocationRef, currentFloorRef, mapScaleRef, mapDimensionsRef,
4 waypointsRef) {
5   const routeSegments = ref([]);
6   const debugWaypoints = ref([]);
7   const routingError = ref(null);
8
9   function calculateRouteSameFloor(start, end, andar, allWaypoints) {
10     if (!start || !end || !andar || !allWaypoints) {
11       console.error("Valores indefinidos em calculateRouteSameFloor");
12       return [];
13     }
14     const waypoints = allWaypoints.filter(wp => wp.andar === andar);
15     const getDist = (a, b) => Math.hypot(a.x - b.x, a.y - b.y);
16
17     const nodes = [...waypoints];
18     const tempStart = { id: 'start', x: start.x, y: start.y, neighbors: [] };
19     const tempEnd = { id: 'end', x: end.x, y: end.y, neighbors: [] };
20
21     const k = 3;
22     const sortedStart = [...waypoints].sort((a, b) => getDist(a, start) - getDist(b, start));
23     const sortedEnd = [...waypoints].sort((a, b) => getDist(a, end) - getDist(b, end));
24     tempStart.neighbors = sortedStart.slice(0, k).map(n => n.id);
25     tempEnd.neighbors = sortedEnd.slice(0, k).map(n => n.id);
26
27     nodes.push(tempStart, tempEnd);
28
29     const graph = {};
30     nodes.forEach(wp => {
31       const neighbors = wp.neighbors || wp.vizinhos || [];
32       graph[wp.id] = neighbors;
33     });
34
35     const dist = {};
```

```
35  const prev = {};
36  const queue = new Set(nodes.map(n => n.id));
37
38  nodes.forEach(n => dist[n.id] = Infinity);
39  dist['start'] = 0;
40
41  while (queue.size > 0) {
42    const u = [...queue].reduce((minNode, node) => dist[node] < dist[minNode] ? node : minNode, [...queue][0]);
43    queue.delete(u);
44
45    const neighbors = graph[u] || [];
46    neighbors.forEach(v => {
47      const from = nodes.find(n => n.id === u);
48      const to = nodes.find(n => n.id === v);
49      if (!from || !to) return;
50
51      const alt = dist[u] + getDist(from, to);
52      if (alt < dist[v]) {
53        dist[v] = alt;
54        prev[v] = u;
55      }
56    });
57  }
58
59  const path = [];
60  let u = 'end';
61  while (u && u !== 'start') {
62    const node = nodes.find(n => n.id === u);
63    if (node) path.unshift({ x: node.x, y: node.y });
64    u = prev[u];
65  }
66
67  path.unshift({ x: start.x, y: start.y });
68  return path;
69}
70
71 function calculateRouteBetweenFloors(start, end, andarStart, andarEnd, allWaypoints) {
```

```
72 if (!start || !end || !andarStart || !andarEnd || !allWaypoints) {
73   console.error("Valores indefinidos em calculateRouteBetweenFloors");
74   return [];
75 }
76 if (andarStart === andarEnd) {
77   return calculateRouteSameFloor(start, end, andarStart, allWaypoints);
78 }
79
80 const escadasSaida = allWaypoints.filter(wp =>
81   wp.andar === andarStart && wp.tipo === 'escada' && wp.andarDestino === andarEnd
82 );
83
84 const escadasChegada = allWaypoints.filter(wp =>
85   wp.andar === andarEnd && wp.tipo === 'escada' &&
86   escadasSaida.some(s => s.id === wp.idLigacao)
87 );
88
89 if (!escadasSaida.length || !escadasChegada.length) {
90   routingError.value = "Sem conexão entre andares.";
91   return [];
92 }
93
94 const escadaInicio = escadasSaida[0];
95 const escadaDestino = escadasChegada.find(e => e.idLigacao === escadaInicio.id);
96
97 if (!escadaDestino) {
98   routingError.value = "Ligaçāo de escada não encontrada.";
99   return [];
100 }
101
102 const rota1 = calculateRouteSameFloor(start, escadaInicio, andarStart, allWaypoints);
103 const rota2 = [{ x: escadaDestino.x, y: escadaDestino.y }];
104 const rota3 = calculateRouteSameFloor(escadaDestino, end, andarEnd, allWaypoints);
105
106 return [...rota1, ...rota2, ...rota3];
107 }
108 }
```

```
109 watch(
110   [userPositionRef, selectedLocationRef, currentFloorRef, mapScaleRef, mapDimensionsRef, waypointsRef],
111   () => {
112     try {
113       if (!userPositionRef.value || !selectedLocationRef.value || !currentFloorRef.value || !waypointsRef.value) {
114         routeSegments.value = [];
115         return;
116       }
117
118       const start = userPositionRef.value;
119       const endLocation = selectedLocationRef.value;
120       const andarAtual = currentFloorRef.value;
121       const waypoints = waypointsRef.value || [];
122
123       routingError.value = null;
124
125       if (!start || !endLocation) {
126         routeSegments.value = [];
127         return;
128       }
129
130       const end = { x: endLocation.x, y: endLocation.y };
131       const andarDestino = endLocation.andar;
132
133       const path = calculateRouteBetweenFloors(start, end, andarAtual, andarDestino, waypoints);
134
135       debugWaypoints.value = waypoints;
136
137       routeSegments.value = [];
138
139       for (let i = 0; i < path.length - 1; i++) {
140         const p1 = path[i];
141         const p2 = path[i + 1];
142
143         const dx = p2.x - p1.x;
144         const dy = p2.y - p1.y;
145         const length = Math.sqrt(dx * dx + dy * dy);
```

```
146     const angle = Math.atan2(dy, dx) * (180 / Math.PI);
147
148     routeSegments.value.push({
149         x: pl.x,
150         y: pl.y,
151         length,
152         angle
153     });
154 }
155 } catch (error) {
156     console.error("Erro durante o calculo da rota:", error);
157     routingError.value = "Erro interno ao calcular a rota.";
158     routeSegments.value = [];
159 }
160 },
161 { immediate: true }
162 );
163
164 return {
165     routeSegments,
166     debugWaypoints,
167     routingError
168 };
169 }
```

src/layouts/MapLayout.vue

```
1 <template>
2   <div class="map-container">
3     <div v-if="loading" class="loading">Carregando mapa...</div>
4     <div v-if="error" class="error">{{ error }}</div>
5
6     <div v-if="mapScale && mapDimensions" class="map-content" :style="{ width: `${mapDimensions.width}px`, height: `${mapDimensions.height}px` }">
7       
8
9       <div v-for="loc in locations" :key="loc.id" class="location-pin"
10         :style="{ left: `${loc.x * mapScale.x}%`, top: `${loc.y * mapScale.y}%` }"
11         @click="selectLocation(loc)">
12           {{ loc.nome }}
13         </div>
14
15       <div v-for="segment in routeSegments" :key="segment.index" class="route-segment"
16         :style="{ left: `${segment.x * mapScale.x}%`, top: `${segment.y * mapScale.y}%` ,
17                   width: `${segment.length * mapScale.x}%`, transform: `rotate(${segment.angle}deg)` }">
18         </div>
19
20       <div v-if="userPosition" class="user-position"
21         :style="{ left: `${userPosition.x * mapScale.x}%`, top: `${userPosition.y * mapScale.y}%` }">
22         Você está aqui
23       </div>
24     </div>
25
26     <div class="location-details" v-if="selectedLocation">
27       <h2>{{ selectedLocation.nome }}</h2>
28       <p>{{ selectedLocation.descricao }}</p>
29       <button @click="clearSelection">Fechar</button>
30     </div>
31
32   </div>
33 </template>
```

```
35 <script setup>
36 import { ref, computed, onMounted } from 'vue';
37 import { useMapRouting } from '@/composables/useMapRouting';
38 import { useLocationStore } from '@/stores/location';
39 import { fetchInitialData } from '@/services/LocationService'; // Importe fetchInitialData diretamente
40
41 const locationStore = useLocationStore();
42 const locations = ref([]);
43 const loading = ref(true);
44 const error = ref(null);
45 const selectedLocation = ref(null);
46 const userPosition = ref({ x: 10, y: 10 }); // Posição inicial do usuário
47 const currentFloor = ref('terreo'); // Andar inicial
48 const mapScale = ref({ x: 1, y: 1 }); // Escala inicial
49 const mapDimensions = ref({ width: 800, height: 600 }); // Dimensões do mapa
50
51 onMounted(async () => {
52   try {
53     const data = await fetchInitialData(); // Chame fetchInitialData diretamente
54     if (data) {
55       locations.value = data.locais;
56       andaresDisponiveis.value = data.floors;
57     }
58   } catch (err) {
59     error.value = `Erro ao carregar dados: ${err.message}`;
60   } finally {
61     loading.value = false;
62   }
63 });
64
65 const andaresDisponiveis = ref(null);
66
67 const andarAtualInfo = computed(() => {
68   console.log('Verificando andar atual:', currentFloor.value);
69   console.log('Andares disponíveis:', andaresDisponiveis.value);
70
71   if (!andaresDisponiveis.value) {
```

```
72     return { nome: 'Carregando...', image: '' };
73 }
74
75 const andarInfo = andaresDisponiveis.value.find(andar => andar.nome.toLowerCase() === currentFloor.value);
76
77 if (!andarInfo) {
78     return { nome: 'Andar não encontrado', image: '' };
79 }
80
81 return {
82     nome: andarInfo.nome,
83     image: `/maps/${andarInfo.nome.toLowerCase()}.svg`
84 };
85 });
86
87 const { routeSegments } = useMapRouting(
88     userPosition,
89     selectedLocation,
90     currentFloor,
91     mapScale,
92     mapDimensions,
93
94 );
95
96 /***** ✨ Windsurf Command ✨ *****/
97 /**
98 * Updates the selected location to the provided location.
99 *
100 * @param {Object} location - The location object to be set as the selected location.
101 */
102
103 /***** b7aecb73-4466-4de7-b736-0ef4c0977dfc *****/
104 function selectLocation(location) {
105     selectedLocation.value = location;
106 }
107
108 function clearSelection() {
```

```
109     selectedLocation.value = null;
110 }
111 </script>
112
113 <style scoped>
114   .map-container {
115     position: relative;
116     width: 100%;
117     height: 100vh;
118   }
119
120   .map-content {
121     position: relative;
122     border: 1px solid #ccc;
123   }
124
125   .map-image {
126     width: 100%;
127     height: 100%;
128   }
129
130   .location-pin,
131   .user-position {
132     position: absolute;
133     background-color: rgba(255, 0, 0, 0.7);
134     color: white;
135     padding: 5px;
136     border-radius: 5px;
137     cursor: pointer;
138   }
139
140   .route-segment {
141     position: absolute;
142     height: 3px;
143     background-color: blue;
144     transform-origin: left;
145 }
```

```
146  
147 .location-details,  
148 .debug-waypoints {  
149   position: absolute;  
150   top: 10px;  
151   right: 10px;  
152   background-color: white;  
153   padding: 10px;  
154   border: 1px solid #ccc;  
155 }  
156 </style>
```

src/router/index.js

```
1 import { createRouter, createWebHistory } from 'vue-router';
2 import MapLayout from '@/layouts/MapLayout.vue';
3 import MapHome from '@/views/MapHome.vue';
4 import AdminPanel from '@/views/AdminPanel.vue';
5
6 const routes = [
7   {
8     path: '/',
9     component: MapLayout,
10    children: [
11      {
12        path: '',
13        name: 'MapHome',
14        component: MapHome
15      },
16      {
17        path: 'admin',
18        name: 'AdminPanel',
19        component: AdminPanel
20      }
21    ]
22  }
23];
24
25 const router = createRouter({
26   history: createWebHistory(import.meta.env.BASE_URL),
27   routes
28 });
29
30 export default router;
31
```

src/services/LocationService.js

```
1 import { ref } from 'vue';
2
3 // Dados simulados para localizações
4 const locationsData = [
5   { id: 1, nome: 'Bloco 1', descricao: 'Salas de aula e laboratórios', categoria: 'Acadêmico', andar: 1, latitude: -4.972138, longitude: -37.977455 },
6   { id: 2, nome: 'Bloco 2', descricao: 'Departamentos e coordenações', categoria: 'Administrativo', andar: 1, latitude: -4.971956, longitude: -37.977100 },
7   { id: 3, nome: 'Biblioteca', descricao: 'Acervo e salas de estudo', categoria: 'Acadêmico', andar: 1, latitude: -4.972401, longitude: -37.977012 },
8   { id: 4, nome: 'Cantina', descricao: 'Área de alimentação', categoria: 'Serviços', andar: 1, latitude: -4.972587, longitude: -37.977455 },
9   { id: 5, nome: 'Laboratório de Informática', descricao: 'Computadores para uso dos alunos', categoria: 'Acadêmico', andar: 2, latitude: -4.972138, longitude: -37.977256 },
10  { id: 6, nome: 'Auditório', descricao: 'Eventos e palestras', categoria: 'Comum', andar: 2, latitude: -4.972300, longitude: -37.977356 },
11  { id: 7, nome: 'Banheiros', descricao: 'Banheiros masculino e feminino', categoria: 'Serviços', andar: 1, latitude: -4.972456, longitude: -37.977200 },
12  { id: 8, nome: 'Elevador', descricao: 'Acesso entre andares', categoria: 'Acessibilidade', andar: 1, latitude: -4.972200, longitude: -37.977300 },
13  { id: 9, nome: 'Secretaria Acadêmica', descricao: 'Atendimento aos estudantes', categoria: 'Administrativo', andar: 1, latitude: -4.972050, longitude: -37.977250 },
14  { id: 10, nome: 'Laboratório de Química', descricao: 'Experimentos e aulas práticas', categoria: 'Acadêmico', andar: 2, latitude: -4.972350, longitude: -37.977150 }
15];
16
17 // Pontos de referência para navegação
18 const waypointsData = [
19   { id: 1, nome: 'Entrada Principal', descricao: 'Acesso principal ao campus', latitude: -4.972700, longitude: -37.977500 },
20   { id: 2, nome: 'Interseção Blocos 1-2', descricao: 'Cruzamento entre blocos', latitude: -4.972050, longitude: -37.977300 },
21   { id: 3, nome: 'Rampa de Acesso', descricao: 'Acesso para cadeirantes', latitude: -4.972400, longitude: -37.977250 },
22   { id: 4, nome: 'Escada Central', descricao: 'Escada para o segundo andar', latitude: -4.972200, longitude: -37.977200 },
23   { id: 5, nome: 'Estacionamento A', descricao: 'Área de estacionamento frontal', latitude: -4.972800, longitude: -37.977600 }
```

```
24 ];
25
26 // Informações sobre os andares
27 const floorsData = [
28   { id: 1, nome: 'Térreo', descricao: 'Andar principal com recepção' },
29   { id: 2, nome: 'Primeiro Andar', descricao: 'Salas administrativas e laboratórios especiais' }
30 ];
31
32 // Estado global do serviço de localização (para monitorar estados e erros)
33 export const locationServiceState = {
34   loading: ref(false),
35   error: ref(null),
36   lastUpdate: ref(null)
37 };
38
39 // Definindo a classe LocationService antes de usá-la
40 class LocationService {
41   /**
42    * Retorna todas as localizações
43    * @returns {Promise<Array>} Lista de todas as localizações
44    */
45   async getAllLocations() {
46     // Simulando um delay de rede
47     await new Promise(resolve => setTimeout(resolve, 300));
48     return [...locationsData];
49   }
50
51   /**
52    * Busca uma localização pelo ID
53    * @param {number} id - ID da localização
54    * @returns {Promise<Object>} A localização encontrada ou undefined
55    */
56   async getLocationById(id) {
57     await new Promise(resolve => setTimeout(resolve, 200));
58     return locationsData.find(loc => loc.id === id);
59   }
60 }
```

```
61 /**
62  * Cria uma nova localização
63  * @param {Object} locationData - Dados da nova localização
64  * @returns {Promise<Object>} A nova localização criada
65 */
66 async createLocation(locationData) {
67   await new Promise(resolve => setTimeout(resolve, 400));
68
69   // Simulando criação (na prática, isso seria feito por um backend)
70   const newId = Math.max(...locationsData.map(loc => loc.id)) + 1;
71   const newLocation = { id: newId, ...locationData };
72   locationsData.push(newLocation);
73
74   return newLocation;
75 }
76
77 /**
78  * Atualiza uma localização existente
79  * @param {number} id - ID da localização a ser atualizada
80  * @param {Object} locationData - Novos dados para a localização
81  * @returns {Promise<Object>} A localização atualizada
82 */
83 async updateLocation(id, locationData) {
84   await new Promise(resolve => setTimeout(resolve, 300));
85
86   const index = locationsData.findIndex(loc => loc.id === id);
87   if (index === -1) {
88     throw new Error('Localização não encontrada');
89   }
90
91   const updatedLocation = { ...locationsData[index], ...locationData };
92   locationsData[index] = updatedLocation;
93
94   return updatedLocation;
95 }
96
97 /**
```

```
98 * Remove uma localização
99 * @param {number} id - ID da localização a ser removida
100 * @returns {Promise<boolean>} Verdadeiro se removido com sucesso
101 */
102 async deleteLocation(id) {
103     await new Promise(resolve => setTimeout(resolve, 300));
104
105     const index = locationsData.findIndex(loc => loc.id === id);
106     if (index === -1) {
107         throw new Error('Localização não encontrada');
108     }
109
110     locationsData.splice(index, 1);
111     return true;
112 }
113
114 /**
115 * Retorna todos os pontos de referência
116 * @returns {Promise<Array>} Lista de waypoints
117 */
118 async getAllWaypoints() {
119     await new Promise(resolve => setTimeout(resolve, 200));
120     return [...waypointsData];
121 }
122
123 /**
124 * Retorna informações sobre os andares
125 * @returns {Promise<Array>} Lista de andares
126 */
127 async getFloors() {
128     await new Promise(resolve => setTimeout(resolve, 100));
129     return [...floorsData];
130 }
131
132 /**
133 * Filtra localizações por categoria
134 * @param {string} categoria - Categoria para filtrar
```

```
135 * @returns {Promise<Array>} Localizações filtradas
136 */
137 async getLocationsByCategory(categoria) {
138     await new Promise(resolve => setTimeout(resolve, 200));
139     return locationsData.filter(loc => loc.categoria === categoria);
140 }
141
142 /**
143 * Filtra localizações por andar
144 * @param {number} andar - Número do andar
145 * @returns {Promise<Array>} Localizações filtradas
146 */
147 async getLocationsByFloor(andar) {
148     await new Promise(resolve => setTimeout(resolve, 200));
149     return locationsData.filter(loc => loc.andar === andar);
150 }
151
152
153 const locationService = new LocationService();
154
155 /**
156 * Busca todos os dados iniciais necessários para o mapa
157 * @param {boolean} forceRefresh - Se verdadeiro, recarrega os dados mesmo se já estiverem em cache
158 * @returns {Promise<Object>} Objeto com locais, waypoints e andares
159 */
160 export async function fetchInitialData(forceRefresh = false) {
161     locationServiceState.loading.value = true;
162     locationServiceState.error.value = null;
163
164     try {
165         // Simulando tempo de requisição
166         await new Promise(resolve => setTimeout(resolve, 500));
167
168         // Mapeia os dados para formato adequado ao mapa
169         const locais = await locationService.getAllLocations();
170         const waypoints = await locationService.getAllWaypoints();
171         const andares = await locationService.getFloors();
```

```
172
173 // Adiciona informações adicionais para o funcionamento do mapa
174 const mappedFloors = andares.map(andar => ({
175   ...andar,
176   id: andar.id.toString(), // Garante que ID é string
177   nome: andar.nome,
178   image: `/maps/${andar.id === 1 ? 'terreo' : andar.id === 2 ? 'primeiro' : 'segundo'}.svg`
179 }));
180
181 // Mapeia locais para incluir coordenadas x, y relativas para o SVG
182 const mappedLocations = locais.map(local => ({
183   ...local,
184   andar: local.andar.toString(), // Converte andar para string para compatibilidade
185   // Valores x, y são calculados para posicionamento no mapa (hipotético)
186   x: 40 + Math.random() * 60, // Posição x aleatória entre 40% e 60% da largura do mapa
187   y: 30 + Math.random() * 50 // Posição y aleatória entre 30% e 80% da altura do mapa
188 }));
189
190 // Atualiza o timestamp de última atualização
191 locationServiceState.lastUpdate.value = new Date();
192
193 return {
194   locais: mappedLocations,
195   waypoints,
196   floors: mappedFloors
197 };
198 } catch (error) {
199   console.error('Erro ao buscar dados iniciais:', error);
200   locationServiceState.error.value = `Erro ao carregar dados: ${error.message}`;
201   return null;
202 } finally {
203   locationServiceState.loading.value = false;
204 }
205
206
207 export default locationService;
```

src/stores/location.js

```
35     if (floors.value.length > 0 && !selectedFloor.value) {
36         selectedFloor.value = floors.value[0];
37     }
38
39     return true;
40 } else {
41     error.value = 'Não foi possível carregar os dados do mapa.';
42     return false;
43 }
44 } catch (e) {
45     console.error('Erro na store ao carregar dados:', e);
46     error.value = `Erro ao carregar dados: ${e.message}`;
47     return false;
48 } finally {
49     isLoading.value = false;
50 }
51 }
52
53 function selectLocal(local) {
54     selectedLocal.value = local;
55
56     // Se o local selecionado estiver em outro andar, muda para esse andar
57     if (local && selectedFloor.value?.id !== local.andar) {
58         const floor = floors.value.find(f => f.id === local.andar);
59         if (floor) {
60             selectFloor(floor);
61         }
62     }
63 }
64
65 function selectFloor(floor) {
66     selectedFloor.value = floor;
67 }
68
69 function clearSelectedLocal() {
70     selectedLocal.value = null;
71 }
```

```
72
73 // Funções CRUD para administração
74 async function createLocation(localData) {
75   isLoading.value = true;
76   error.value = null;
77
78   try {
79     const success = await LocationService.createLocation(localData);
80     if (success) {
81       await loadInitialData(true);
82       return true;
83     } else {
84       error.value = 'Falha ao criar local.';
85       return false;
86     }
87   } catch (e) {
88     error.value = `Erro ao criar local: ${e.message}`;
89     return false;
90   } finally {
91     isLoading.value = false;
92   }
93 }
94
95 async function updateLocation(localData) {
96   isLoading.value = true;
97   error.value = null;
98
99   try {
100     const success = await LocationService.updateLocation(localData);
101     if (success) {
102       await loadInitialData(true);
103       return true;
104     } else {
105       error.value = 'Falha ao atualizar local.';
106       return false;
107     }
108   } catch (e) {
```

```
109     error.value = `Erro ao atualizar local: ${e.message}`;
110     return false;
111 } finally {
112     isLoading.value = false;
113 }
114 }
115
116 async function deleteLocation(id) {
117     isLoading.value = true;
118     error.value = null;
119
120     try {
121         const success = await LocationService.deleteLocation(id);
122         if (success) {
123             await loadInitialData(true);
124             // Se o local excluído for o selecionado, limpa a seleção
125             if (selectedLocal.value?.id === id) {
126                 clearSelectedLocal();
127             }
128             return true;
129         } else {
130             error.value = 'Falha ao excluir local.';
131             return false;
132         }
133     } catch (e) {
134         error.value = `Erro ao excluir local: ${e.message}`;
135         return false;
136     } finally {
137         isLoading.value = false;
138     }
139 }
140
141 // Utilitários para navegação e rotas
142 function findPathBetween(startPoint, endPoint) {
143     // Implementação básica - conectar dois pontos em linha reta
144     // Em uma implementação real, usariamos os waypoints e algoritmo A* ou similar
145     if (!startPoint || !endPoint) return [];
```

```
146
147     return [
148         startX: startPoint.x,
149         startY: startPoint.y,
150         endX: endPoint.x,
151         endY: endPoint.y,
152         distance: Math.sqrt(
153             Math.pow(endPoint.x - startPoint.x, 2) +
154             Math.pow(endPoint.y - startPoint.y, 2)
155         ),
156         angle: Math.atan2(
157             endPoint.y - startPoint.y,
158             endPoint.x - startPoint.x
159         ) * 180 / Math.PI
160     ];
161 }
162
163 return {
164     // Estado
165     locations,
166     waypoints,
167     floors,
168     selectedLocal,
169     selectedFloor,
170     isLoading,
171     error,
172
173     // Getters
174     currentFloorLocations,
175
176     // Ações
177     loadInitialData,
178     selectLocal,
179     selectFloor,
180     clearSelectedLocal,
181     createLocation,
182     updateLocation,
```

```
183     deleteLocation,  
184  
185     // Utilitários  
186     findPathBetween  
187     );  
188 });
```

src/stores/locationStore.js

```
1 import { defineStore } from 'pinia';
2 import { ref, computed } from 'vue';
3 import * as LocationService from '../services/LocationService';
4
5 export const useLocationStore = defineStore('location', () => {
6     // Estado
7     const locais = ref([]);
8     const waypoints = ref([]);
9     const floors = ref([]);
10    const selectedLocalId = ref(null);
11    const currentFloorId = ref('terreo');
12    const loading = ref(false);
13    const error = ref(null);
14
15    // Getters computados
16    const selectedLocal = computed(() => {
17        return locais.value.find(local => local.id === selectedLocalId.value) || null;
18    });
19
20    const locaisAtualAndar = computed(() => {
21        return locais.value.filter(local => local.andar === currentFloorId.value);
22    });
23
24    const currentFloorData = computed(() => {
25        return floors.value.find(f => f.id === currentFloorId.value) || null;
26    });
27
28    // Ações
29    async function fetchInitialData(forceRefresh = false) {
30        loading.value = true;
31        error.value = null;
32
33        try {
34            const data = await LocationService.fetchInitialData(forceRefresh);
```

```
35
36     if (data) {
37         locais.value = data.locais || [];
38         waypoints.value = data.waypoints || [];
39         floors.value = data.floors || [];
40
41         // Verificar se o andar atual ainda existe
42         if (floors.value.length > 0 && !floors.value.some(f => f.id === currentFloorId.value)) {
43             currentFloorId.value = floors.value[0].id;
44         }
45
46         return true;
47     } else {
48         error.value = 'Não foi possível carregar os dados do mapa.';
49         return false;
50     }
51 } catch (e) {
52     console.error('Erro na store ao carregar dados:', e);
53     error.value = `Erro ao carregar dados: ${e.message}`;
54     return false;
55 } finally {
56     loading.value = false;
57 }
58 }

59 function selectLocal(localId) {
60     selectedLocalId.value = localId;
61
62     // Se o local selecionado estiver em outro andar, muda para esse andar
63     if (selectedLocal.value && selectedLocal.value.andar !== currentFloorId.value) {
64         changeFloor(selectedLocal.value.andar);
65     }
66 }
67 }

68 function changeFloor(floorId) {
69     if (floors.value.some(f => f.id === floorId)) {
70         currentFloorId.value = floorId;
```

```
72 } else {
73     console.warn(`Tentativa de mudar para um andar inexistente: ${floorId}`);
74 }
75 }
76
77 function clearSelectedLocal() {
78     selectedLocalId.value = null;
79 }
80
81 // Funções CRUD para administração
82 async function createLocal(localData) {
83     loading.value = true;
84     error.value = null;
85
86     try {
87         const success = await LocationService.createLocation(localData);
88         if (success) {
89             await fetchInitialData(true);
90             return true;
91         } else {
92             error.value = 'Falha ao criar local.';
93             return false;
94         }
95     } catch (e) {
96         error.value = `Erro ao criar local: ${e.message}`;
97         return false;
98     } finally {
99         loading.value = false;
100    }
101 }
102
103 async function updateLocal(localData) {
104     loading.value = true;
105     error.value = null;
106
107     try {
108         const success = await LocationService.updateLocation(localData);
```

```
109     if (success) {
110         await fetchInitialData(true);
111         return true;
112     } else {
113         error.value = 'Falha ao atualizar local.';
114         return false;
115     }
116 } catch (e) {
117     error.value = `Erro ao atualizar local: ${e.message}`;
118     return false;
119 } finally {
120     loading.value = false;
121 }
122 }

124 async function deleteLocal(id) {
125     loading.value = true;
126     error.value = null;
127
128     try {
129         const success = await LocationService.deleteLocation(id);
130         if (success) {
131             await fetchInitialData(true);
132             // Se o local excluído for o selecionado, limpa a seleção
133             if (selectedLocalId.value === id) {
134                 clearSelectedLocal();
135             }
136             return true;
137         } else {
138             error.value = 'Falha ao excluir local.';
139             return false;
140         }
141     } catch (e) {
142         error.value = `Erro ao excluir local: ${e.message}`;
143         return false;
144     } finally {
145         loading.value = false;
```

```
146    }
147 }
148
149 return {
150   // Estado
151   locais,
152   waypoints,
153   floors,
154   selectedLocalId,
155   currentFloorId,
156   loading,
157   error,
158
159   // Getters
160   selectedLocal,
161   locaisAtualAndar,
162   currentFloorData,
163
164   // Ações
165   fetchInitialData,
166   selectLocal,
167   changeFloor,
168   clearSelectedLocal,
169   createLocal,
170   updateLocal,
171   deleteLocal
172 };
173});
```

src/types/index.d.ts

```
1 // Definição dos tipos de dados principais do aplicativo
2
3 // Interface para um local/ponto de interesse
4 export interface Local {
5   id: string;
6   nome: string;
7   andar: string;
8   x: number;
9   y: number;
10  tipo?: string;
11  descricao?: string;
12  endereco?: string;
13  cor?: string;
14 }
15
16 // Interface para um andar
17 export interface Floor {
18   id: string;
19   nome: string;
20   image?: string;
21   descricao?: string;
22 }
23
24 // Interface para um waypoint (ponto de rota)
25 export interface Waypoint {
26   id: string;
27   andar: string;
28   x: number;
29   y: number;
30   connections: string[]; // IDs dos waypoints conectados
31 }
32
33 // Interface para um segmento de rota
34 export interface RouteSegment {
```

```
35  startX: number;
36  startY: number;
37  endX: number;
38  endY: number;
39  length: number;
40  angle: number;
41 }
42
43 // Interface para a posição do usuário
44 export interface UserPosition {
45   andar: string;
46   x: number;
47   y: number;
48 }
49
50 // Interface para informações de popup
51 export interface PopupInfo {
52   x: number;
53   y: number;
54   content: string;
55   visible: boolean;
56 }
57
58 // Interface para dimensões do mapa
59 export interface MapDimensions {
60   width: number;
61   height: number;
62 }
63
64 // Interface para estado de carregamento
65 export interface LoadingState {
66   initialData: boolean;
67   map: boolean;
68 }
69
70 // Interface para estado de erro
71 export interface ErrorState {
```

```
72  initialData: string | null;
73  locais: string | null;
74  routing: string | null;
75  geolocation: string | null;
76 }
77
78 // Declaração de módulos para importar arquivos não-JavaScript
79 declare module "*.png" {
80   const value: string;
81   export default value;
82 }
83
84 declare module "*.jpg" {
85   const value: string;
86   export default value;
87 }
88
89 declare module "*.svg" {
90   const value: string;
91   export default value;
92 }
93
94 // Extensão das interfaces do Vue
95 declare module 'vue' {
96   interface ComponentCustomProperties {
97     $locais: Local[];
98     $floors: Floor[];
99   }
100 }
```

src/views/AdminPanel.vue

```
1 <template>
2   <div class="admin-panel">
3     <LocationAdmin />
4   </div>
5 </template>
6
7 <script setup>
8 import LocationAdmin from '@/components/LocationAdmin.vue';
9 </script>
10
11 <style scoped>
12   .admin-panel {
13     padding: 20px;
14     max-width: 1200px;
15     margin: 0 auto;
16     background: white;
17     min-height: 100vh;
18   }
19 </style>
```

src/views/MapHome.vue

```
1 <template>
2   <!-- Este componente está vazio propositalmente, pois o conteúdo do mapa já é renderizado pelo MapLayout -->
3   <div class="map-home-container">
4     <!-- Todo o conteúdo do mapa é gerenciado pelo layout -->
5   </div>
6 </template>
7
8 <script setup>
9 // Não é necessário lógica adicional aqui, pois o MapLayout já gerencia tudo
10 </script>
11
12 <style scoped>
13 .map-home-container {
14   width: 100%;
15   height: 100%;
16 }
17 </style>
```

src/App.vue

```
1 // src/App.vue
2 <template>
3   <router-view />
4 </template>
5
6 <script setup>
7 // Não precisamos mais de refs, computed, watchers, ou imports de componentes/composables do mapa aqui.
8 // Toda essa lógica agora reside em MapLayout.vue (ou nas views específicas).
9 import { onMounted } from 'vue';
10
11 onMounted(() => {
12   console.log("App.vue montado. O Router se encarregará do conteúdo.");
13 });
14
15 </script>
16
17 <style scoped>
18 /* Remova ou mova quaisquer estilos que eram específicos para
19  elementos que não existem mais neste template (como .main-content, .error, .loading).
20  Os estilos relevantes agora devem estar em MapLayout.vue ou main.css.
21  Pode manter estilos globais aqui se aplicável a TODAS as rotas,
22  mas geralmente main.css é melhor para isso.
23 */
24
25 /* Exemplo: se você tinha um estilo global para o container do app */
26 /* #app {
27   font-family: Avenir, Helvetica, Arial, sans-serif;
28   -webkit-font-smoothing: antialiased;
29   -moz-osx-font-smoothing: grayscale;
30   text-align: center;
31   color: #2c3e50;
32 }
33 */
34 </style>
```

src/main.js

```
1 import { createApp } from 'vue'  
2 import { createPinia } from 'pinia'  
3 import App from './App.vue'  
4 import router from './router' // Importa o roteador  
5 import './assets/styles/main.css'  
6  
7 const app = createApp(App)  
8 app.use(createPinia())  
9 app.use(router)  
10 app.mount('#app')
```