

```
1 <script setup>
2 import { ref, computed, watch, onMounted, onUnmounted } from 'vue';
3 import { useMapRouting } from '../composables/useMapRouting';
4
5 const props = defineProps({
6   waypoints: {
7     type: Array,
8     default: () => []
9   },
10  locations: {
11    type: Array,
12    default: () => []
13  },
14  currentFloor: {
15    type: String,
16    required: true
17  },
18  userPosition: {
19    type: Object,
20    default: null
21  },
22  selectedLocation: {
23    type: Object,
24    default: null
25  },
26  mapScale: {
27    type: Number,
28    default: 1
29  },
30  loading: {
31    type: Boolean,
32    default: false
33  }
34 });
35
36 const emit = defineEmits([
```

```
37   'map-click',
38   'marker-click',
39   'marker-mouseenter',
40   'marker-mouseleave',
41   'update:mapDimensions'
42 ]);
43
44 const mapRef = ref(null);
45 const mapImageRef = ref(null);
46 const mapDimensions = ref({ width: 0, height: 0 });
47
48 // Filtra locais do andar atual
49 const currentFloorLocations = computed(() => {
50   return props.locations.filter(loc => loc.andar === props.currentFloor);
51 });
52
53 // Configuração do sistema de rotas
54 const waypointsRef = ref(props.waypoints); // Ref para os waypoints
55 const { routeSegments, debugWaypoints } = useMapRouting(
56   ref(props.userPosition),
57   ref(props.selectedLocation),
58   ref(props.currentFloor),
59   ref(props.mapScale),
60   mapDimensions,
61   waypointsRef
62 );
63
64 // Atualiza os waypoints quando mudam
65 watch(() => props.waypoints, (newWaypoints) => {
66   waypointsRef.value = newWaypoints;
67 }, { deep: true });
68
69 // Estilo do mapa
70 const mapStyle = computed(() => ({
71   transform: `scale(${props.mapScale})`
72 }));
73
```

```
74 // Handlers de eventos
75 const handleMapClick = (event) => {
76   if (!mapRef.value) return;
77
78   const rect = mapRef.value.getBoundingClientRect();
79   const x = ((event.clientX - rect.left) / rect.width) * 100;
80   const y = ((event.clientY - rect.top) / rect.height) * 100;
81
82   emit('map-click', { x, y });
83 };
84
85 const handleLocationClick = (location) => {
86   emit('marker-click', location);
87 };
88
89 const handleLocationHover = (location) => {
90   emit('marker-mouseenter', location);
91 };
92
93 const handleLocationLeave = () => {
94   emit('marker-mouseleave');
95 };
96
97 // Atualização das dimensões do mapa
98 const updateMapDimensions = () => {
99   if (!mapRef.value) return;
100
101   const rect = mapRef.value.getBoundingClientRect();
102   mapDimensions.value = {
103     width: rect.width,
104     height: rect.height
105   };
106   emit('update:mapDimensions', mapDimensions.value);
107 };
108
109 // Lifecycle hooks
110 onMounted(() => {
```

```
111     updateMapDimensions();
112     window.addEventListener('resize', updateMapDimensions);
113
114     // Configurar ResizeObserver para mapImageRef se necessário
115     if (mapImageRef.value) {
116         const resizeObserver = new ResizeObserver(() => {
117             updateMapDimensions();
118         });
119         resizeObserver.observe(mapImageRef.value);
120     }
121 });
122
123 onUnmounted(() => {
124     window.removeEventListener('resize', updateMapDimensions);
125
126     // Limpar ResizeObserver se foi configurado
127     if (mapImageRef.value && resizeObserver) {
128         resizeObserver.disconnect();
129     }
130 });
131 </script>
132
133 <template>
134     <div
135         ref="mapRef"
136         class="map-container"
137         @click="handleMapClick"
138     >
139         
146
147         <!-- Marcadores de locais -->
```

```

148 <div
149   v-for="location in currentFloorLocations"
150   :key="location.id"
151   class="location-marker"
152   :class="{ 'selected': location === props.selectedLocation }"
153   :style="{
154     left: `${location.x}%`,
155     top: `${location.y}%`
156   }"
157   @click.stop="handleLocationClick(location)"
158   @mouseenter="handleLocationHover(location)"
159   @mouseleave="handleLocationLeave"
160 >
161   <span class="marker-label">{{ location.nome }}</span>
162 </div>
163
164 <!-- Waypoints de debug (se necessário) -->
165 <div
166   v-for="waypoint in debugWaypoints"
167   :key="waypoint.id"
168   class="waypoint-marker"
169   :style="{
170     left: `${waypoint.x}%`,
171     top: `${waypoint.y}%`
172   }"
173 />
174
175 <!-- Segmentos da rota -->
176 <div
177   v-for="(segment, index) in routeSegments"
178   :key="index"
179   class="route-segment"
180   :style="{
181     left: `${segment.x}%`,
182     top: `${segment.y}%`,
183     width: `${segment.length}%`,
184     transform: `rotate(${segment.angle}deg)`

```

```
185     }"
186   />
187
188   <!-- Marcador de posição do usuário -->
189   <div
190     v-if="userPosition"
191     class="user-marker"
192     :style="{
193       left: `${userPosition.x}%`,
194       top: `${userPosition.y}%`
195     }"
196   />
197
198   <!-- Indicador de carregamento -->
199   <div v-if="loading" class="loading-overlay">
200     <div class="loading-spinner"></div>
201   </div>
202 </div>
203 </template>
204
205 <style scoped>
206 .map-container {
207   position: relative;
208   width: 100%;
209   height: 100%;
210   overflow: hidden;
211 }
212
213 .map-image {
214   width: 100%;
215   height: 100%;
216   object-fit: contain;
217   transform-origin: center center;
218 }
219
220 .location-marker {
221   position: absolute;
```

```
222 width: 20px;
223 height: 20px;
224 background-color: #2196F3;
225 border-radius: 50%;
226 transform: translate(-50%, -50%);
227 cursor: pointer;
228 z-index: 2;
229 }
230
231 .location-marker.selected {
232   background-color: #4CAF50;
233   box-shadow: 0 0 0 4px rgba(76, 175, 80, 0.3);
234 }
235
236 .marker-label {
237   position: absolute;
238   bottom: 100%;
239   left: 50%;
240   transform: translateX(-50%);
241   white-space: nowrap;
242   background-color: rgba(0, 0, 0, 0.7);
243   color: white;
244   padding: 2px 6px;
245   border-radius: 4px;
246   font-size: 12px;
247 }
248
249 .waypoint-marker {
250   position: absolute;
251   width: 8px;
252   height: 8px;
253   background-color: rgba(255, 0, 0, 0.5);
254   border-radius: 50%;
255   transform: translate(-50%, -50%);
256   z-index: 1;
257 }
258
```

```
259 .route-segment {
260     position: absolute;
261     height: 4px;
262     background-color: #4CAF50;
263     transform-origin: left center;
264     z-index: 1;
265 }
266
267 .user-marker {
268     position: absolute;
269     width: 16px;
270     height: 16px;
271     background-color: #FF5722;
272     border-radius: 50%;
273     border: 2px solid white;
274     transform: translate(-50%, -50%);
275     z-index: 3;
276     box-shadow: 0 0 4px rgba(0, 0, 0, 0.3);
277 }
278
279 .loading-overlay {
280     position: absolute;
281     top: 0;
282     left: 0;
283     right: 0;
284     bottom: 0;
285     background-color: rgba(255, 255, 255, 0.7);
286     display: flex;
287     justify-content: center;
288     align-items: center;
289     z-index: 4;
290 }
291
292 .loading-spinner {
293     width: 40px;
294     height: 40px;
295     border: 4px solid #f3f3f3;
```



```
296     border-top: 4px solid #3498db;
297     border-radius: 50%;
298     animation: spin 1s linear infinite;
299 }
300
301 @keyframes spin {
302     0% { transform: rotate(0deg); }
303     100% { transform: rotate(360deg); }
304 }
305 </style>
```