

## src/components/MapDisplay.vue

```
1 <script setup>
2 import { ref, computed, watch, onMounted, onUnmounted } from 'vue';
3
4 const props = defineProps({
5   mapImageUrl: String,
6   locaisOnFloor: Array,
7   userPosition: Object, // { x, y } | null
8   routeSegments: Array, // Array de { x, y, length, angle }
9   debugWaypoints: Array, // Array de waypoints para mostrar
10  selectedLocalId: [Number, String, null],
11  popupInfo: Object, // { visible, x, y, text }
12  mapScale: Number,
13  loading: Boolean,
14 });
15
16 const emit = defineEmits([
17   'map-click',
18   'marker-click',
19   'marker-mouseenter',
20   'marker-mouseleave',
21   'update:mapDimensions' // Emite as dimensões do mapa para o pai
22 ]);
23
24 const mapImageRef = ref(null); // Referência para o elemento .map-image
25 const mapDimensions = ref({ width: 0, height: 0 }); // Dimensões em pixels
26
27 const mapStyle = computed(() => ({
28   backgroundImage: `url(${props.mapImageUrl || ''})`,
29   transform: `scale(${props.mapScale || 1})`,
30   // Adicionar transform-origin se não estiver no CSS global
31   // transformOrigin: 'center center'
32 }));
33
34 const handleMapClick = (event) => {
```

```
35 // Emite o evento original e a referência do elemento para cálculo no pai/composable
36 emit('map-click', event, mapImageRef.value?.getBoundingClientRect());
37 };
38
39 const selectLocalFromMarker = (local) => {
40     emit('marker-click', local);
41 };
42
43 const showPopupOnEnter = (local) => {
44     emit('marker-mouseenter', local);
45 };
46
47 const hidePopupOnLeave = () => {
48     emit('marker-mouseleave');
49 };
50
51 // Observador para atualizar dimensões quando o elemento do mapa for montado ou redimensionado
52 const resizeObserver = ref(null);
53 onMounted(() => {
54     if (mapImageRef.value) {
55         resizeObserver.value = new ResizeObserver(entries => {
56             for (let entry of entries) {
57                 mapDimensions.value = {
58                     width: entry.contentRect.width,
59                     height: entry.contentRect.height,
60                 };
61                 // Emite as dimensões atualizadas para o componente pai/composable
62                 emit('update:mapDimensions', mapDimensions.value);
63             }
64         });
65         resizeObserver.value.observe(mapImageRef.value);
66         // Define dimensões iniciais
67         const rect = mapImageRef.value.getBoundingClientRect();
68         mapDimensions.value = { width: rect.width, height: rect.height };
69         emit('update:mapDimensions', mapDimensions.value);
70     }
71 }
```

```

72 });
73
74 onUnmounted(( ) => {
75     if (resizeObserver.value && mapImageRef.value) {
76         resizeObserver.value.unobserve(mapImageRef.value);
77     }
78 });
79
80 // Recalcula dimensões se a escala mudar (embora o ResizeObserver deva pegar)
81 watch(( ) => props.mapScale, ( ) => {
82     if (mapImageRef.value) {
83         const rect = mapImageRef.value.getBoundingClientRect();
84         mapDimensions.value = { width: rect.width / props.mapScale, height: rect.height / props.mapScale };
85         emit('update:mapDimensions', mapDimensions.value);
86     }
87 }, { immediate: true });
88
89
90 </script>
91
92 <template>
93     <div class="map-container">
94         <div
95             class="map-image"
96             :style="mapStyle"
97             ref="mapImageRef"
98             @click="handleMapClick"
99         >
100         <div
101             v-for="local in locaisOnFloor"
102             :key="`loc-${local.id}`"
103             class="location-marker"
104             :style="{ left: `${local.x}%`, top: `${local.y}%` }"
105             @click.stop="selectLocalFromMarker(local)"
106             @mouseenter="showPopupOnEnter(local)"
107             @mouseleave="hidePopupOnLeave"
108             :title="local.nome"

```

```
109 ></div>
110
111 <div
112     v-if="userPosition"
113     class="user-marker"
114     :style="{ left: `${userPosition.x}%`, top: `${userPosition.y}%` }"
115     title="Sua localização estimada"
116 ></div>
117
118 <div v-if="routeSegments && routeSegments.length > 0">
119     <div
120         v-for="(segment, index) in routeSegments"
121         :key="`route-${index}`"
122         class="path-line"
123         :style="{
124             left: `${segment.x}%`,
125             top: `${segment.y}%`,
126             width: `${segment.length}px`,
127             transform: `rotate(${segment.angle}deg)`
128         }"
129     ></div>
130 </div>
131
132 <div v-if="debugWaypoints && debugWaypoints.length > 0">
133     <div
134         v-for="wp in debugWaypoints"
135         :key="`wp-${wp.id}`"
136         class="waypoint-marker"
137         :style="{ left: `${wp.x}%`, top: `${wp.y}%` }"
138         :title="`Waypoint: ${wp.id}`"
139     ></div>
140 </div>
141
142 <div
143     v-if="popupInfo && popupInfo.visible"
144     class="location-popup"
145     :style="{ left: `${popupInfo.x}%`, top: `${popupInfo.y}%` }"
```

```
146     >
147     {{ popupInfo.text }}
148 </div>
149
150 </div>
151
152 <div v-if="loading" class="loading">
153     Carregando mapa...
154 </div>
155
156 </div>
157 </template>
158
159 <style scoped>
160 /*
161 /* Estilos específicos do mapa/marcadores já estão em main.css
162 .map-container {
163
164 }
165 .map-image {
166
167 }
168 .waypoint-marker {
169
170 }*/
171
172 /* Garante que o loading fique sobre o mapa*/
173 .loading {
174     z-index: 40; /* Acima do mapa, abaixo dos controles/popup */
175 }
176 </style>
```