

## 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
60 function selectLocal(localId) {
61   selectedLocalId.value = localId;
62
63   // Se o local selecionado estiver em outro andar, muda para esse andar
64   if (selectedLocal.value && selectedLocal.value.andar !== currentFloorId.value) {
65     changeFloor(selectedLocal.value.andar);
66   }
67 }
68
69 function changeFloor(floorId) {
70   if (floors.value.some(f => f.id === floorId)) {
71     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 }
123
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 });
```