src/stores/locationStore.js

```
import { defineStore } from 'pinia';
import { ref, computed } from 'vue';
import * as LocationService from '../services/LocationService';

export const useLocationStore = defineStore('location', () => {
  // Estado
  const locais = ref([]);
  const waypoints = ref([]);
  const floors = ref([]);
  const selectedLocalId = ref(null);
  const currentFloorId = ref('terreo');
  const loading = ref(false);
  const error = ref(null);

  // Getters computados
  const selectedLocal = computed(() => {
    return locais.value.find(local => local.id === selectedLocalId.value) || null;
  });

  const locaisAtualAndar = computed(() => {
    return locais.value.filter(local => local.andar === currentFloorId.value);
  });

  const currentFloorData = computed(() => {
    return floors.value.find(f => f.id === currentFloorId.value) || null;
  });

  // Ações
  async function fetchInitialData(forceRefresh = false) {
    loading.value = true;
    error.value = null;

    try {
      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;
```

```
    } else {
      console.warn(`Tentativa de mudar para um andar inexistente: ${floorId}`);
    }
  }

  function clearSelectedLocal() {
    selectedLocalId.value = null;
  }

  // Funções CRUD para administração
  async function createLocal(localData) {
    loading.value = true;
    error.value = null;

    try {
      const success = await LocationService.createLocation(localData);
      if (success) {
        await fetchInitialData(true);
        return true;
      } else {
        error.value = 'Falha ao criar local.';
        return false;
      }
    } catch (e) {
      error.value = `Erro ao criar local: ${e.message}`;
      return false;
    } finally {
      loading.value = false;
    }
  }

  async function updateLocal(localData) {
    loading.value = true;
    error.value = null;

    try {
      const success = await LocationService.updateLocation(localData);
```

```
      if (success) {
        await fetchInitialData(true);
        return true;
      } else {
        error.value = 'Falha ao atualizar local.';
        return false;
      }
    } catch (e) {
      error.value = `Erro ao atualizar local: ${e.message}`;
      return false;
    } finally {
      loading.value = false;
    }
  }

  async function deleteLocal(id) {
    loading.value = true;
    error.value = null;

    try {
      const success = await LocationService.deleteLocation(id);
      if (success) {
        await fetchInitialData(true);
        // Se o local excluído for o selecionado, limpa a seleção
        if (selectedLocalId.value === id) {
          clearSelectedLocal();
        }
        return true;
      } else {
        error.value = 'Falha ao excluir local.';
        return false;
      }
    } catch (e) {
      error.value = `Erro ao excluir local: ${e.message}`;
      return false;
    } finally {
      loading.value = false;
```

```
    }
  }

  return {
    // Estado
    locais,
    waypoints,
    floors,
    selectedLocalId,
    currentFloorId,
    loading,
    error,

    // Getters
    selectedLocal,
    locaisAtualAndar,
    currentFloorData,

    // Ações
    fetchInitialData,
    selectLocal,
    changeFloor,
    clearSelectedLocal,
    createLocal,
    updateLocal,
    deleteLocal
  };
});
```