src/layouts/MapLayout.vue

```vue
<template>
    <div class="app-container">
      <button class="menu-toggle-button" @click="toggleMenu" title="Abrir/Fechar Menu">☰</button>

      <AppSidebar
        :is-open="isMenuOpen"
        :all-locais="allLocais"
        :floors="floors"
        :current-floor-id="currentFloorId"
        :selected-local-id="selectedLocal?.id"
        :loading="loading.initialData" :error="error.locais" @update:currentFloorId="changeFloor"
        @select-local="handleLocalSelect"
        @close="closeMenu" />

      <div class="main-content" style="flex-grow: 1; position: relative;">
        <MapDisplay
          :map-image-url="currentFloorMap"
          :locais-on-floor="locaisAtualAndar"
          :user-position="userPosition"
          :route-segments="routeSegments"
          :debug-waypoints="debugWaypoints" :selected-local-id="selectedLocal?.id"
          :popup-info="popupInfo"
          :map-scale="mapScale"
          :loading="loading.map" @map-click="handleMapClick"
          @marker-click="handleLocalSelect"
          @marker-mouseenter="showPopup"
          @marker-mouseleave="hidePopup"
          @update:map-dimensions="mapDimensions = $event" />

        <MapControls
          @zoom-in="zoomIn"
          @zoom-out="zoomOut"
          @update-location="triggerAutoLocation"
          @set-location-manual="triggerManualLocationMode" />
```

```
35
36          <div v-if="overallLoading && !displayError" class="loading">
37              <span v-if="loading.initialData">Carregando dados do mapa...</span>
38              <span v-else-if="isGettingLocation">Obtendo localização...</span>
39              <span v-else>Carregando...</span>
40           </div>
41
42
43          <div v-if="displayError" class="error">{{ displayError }}</div>
44
45          <div v-if="settingUserLocationMode" class="manual-location-prompt">
46             {{ manualModeMessage }}
47          </div>
48
49          <router-view />
50        </div>
51      </div>
52    </template>
53
54    <script setup>
55    import { ref, computed, onMounted, watch } from 'vue';
56    import AppSidebar from '@/components/AppSidebar.vue';
57    import MapDisplay from '@/components/MapDisplay.vue';
58    import MapControls from '@/components/MapControls.vue';
59    import { useGeolocation } from '@/composables/useGeolocation.js';
60    import { useMapInteraction } from '@/composables/useMapInteraction.js';
61    import { useMapRouting } from '@/composables/useMapRouting.js';
62    // Importa a função unificada do serviço
63    import { fetchInitialData, locationServiceState } from '@/services/LocationService.js';
64
65    // --- Estado Principal ---
66    const allLocais = ref([]);
67    const floors = ref([]);
68    const waypoints = ref([]); // Novo ref para waypoints
69    const selectedLocal = ref(null);
70    const currentFloorId = ref('terreo'); // Andar inicial padrão
71    const isMenuOpen = ref(false);
```

```
  const mapDimensions = ref({ width: 0, height: 0 });

  // Estados de Loading e Erro - mais detalhados
  const loading = ref({
      initialData: true, // Combina loading de locais, waypoints, andares
      map: true, // Pode ser redundante se o mapa só carrega após initialData
  });
  const error = ref({
      initialData: null, // Erro ao carregar dados gerais
      locais: null, // Mantido para erro específico da sidebar, se necessário
      routing: null, // Vindo do useMapRouting
      geolocation: null // Vindo do useGeolocation
  });
  const manualModeMessage = ref(''); // Mensagem para modo manual

  // --- Composables ---
  const {
    userPosition,
    geolocationError,
    isGettingLocation,
    getUserLocationAuto,
    setUserLocationManually
  } = useGeolocation();

  const {
    mapScale,
    popupInfo,
    settingUserLocationMode,
    zoomIn,
    zoomOut,
    showPopup,
    hidePopup,
    handleMapClick: handleMapClickInteraction,
    enableSetUserLocationMode,
  } = useMapInteraction();

  // Passa o ref de waypoints para o composable de roteamento
```

```
  const {
    routeSegments,
    hasRoute,
    debugWaypoints, // Recebe waypoints de debug
    routingError: routeCalcError,
    calculateRoute // Pode não ser mais necessário chamar manualmente
  } = useMapRouting(userPosition, selectedLocal, currentFloorId, mapScale, mapDimensions, waypoints);

  // --- Dados Computados ---
  const currentFloorData = computed(() => floors.value.find(f => f.id === currentFloorId.value));
  const currentFloorMap = computed(() => currentFloorData.value?.image || '');
  const locaisAtualAndar = computed(() => allLocais.value.filter(local => local.andar === currentFloorId.value));
  const overallLoading = computed(() => loading.value.initialData || isGettingLocation.value);
  // Combina erros para exibição principal (pode priorizar)
  const displayError = computed(() =>
      error.value.initialData ||
      error.value.geolocation ||
      error.value.routing ||
      locationServiceState.error.value // Observa erro global do serviço tbm
  );


  // --- Métodos / Manipuladores de Eventos ---
  const toggleMenu = () => { isMenuOpen.value = !isMenuOpen.value; };
  const closeMenu = () => { isMenuOpen.value = false; };

  const loadInitialData = async () => {
    loading.value.initialData = true;
    loading.value.map = true; // Assume que mapa depende dos dados
    error.value.initialData = null;
    error.value.locais = null; // Limpa erro específico também
    try {
      const initialData = await fetchInitialData(); // Chama a função unificada
      if (initialData) {
        allLocais.value = initialData.locais;
        floors.value = initialData.floors;
        waypoints.value = initialData.waypoints; // Atualiza o ref de waypoints
```

```
146
147         // Define andar padrão se necessário (após carregar andares)
148         if (floors.value.length > 0 && !floors.value.some(f => f.id === currentFloorId.value)) {
149           currentFloorId.value = floors.value[0].id;
150         }
151          // Verifica se há locais, andares ou waypoints
152           if (allLocais.value.length === 0) console.warn("Nenhum local carregado da API.");
153           if (floors.value.length === 0) console.warn("Nenhum andar carregado da API.");
154           if (waypoints.value.length === 0) console.warn("Nenhum waypoint carregado da API.");
155
156
157       } else {
158           // Erro já deve estar em locationServiceState.error ou error.value do serviço
159           error.value.initialData = locationServiceState.error.value || "Falha ao carregar dados iniciais.";
160       }
161     } catch (err) {
162        // Captura erros inesperados no processo
163       console.error("Erro crítico ao carregar dados iniciais:", err);
164       error.value.initialData = `Erro inesperado: ${err.message}`;
165     } finally {
166       loading.value.initialData = false;
167       loading.value.map = false; // Permite que o mapa seja exibido
168     }
169   };
170
171   const changeFloor = (newFloorId) => {
172       if (currentFloorId.value !== newFloorId) {
173           console.log("Mudando para andar:", newFloorId);
174           currentFloorId.value = newFloorId;
175           selectedLocal.value = null; // Desseleciona local ao mudar de andar
176           hidePopup(); // Esconde popup
177           // A rota será recalculada automaticamente pelo watcher em useMapRouting
178       }
179   };
180
181
182   const handleLocalSelect = (local) => {
```

```
    if (!local) return;
    selectedLocal.value = local;
    // Se o local selecionado estiver em outro andar, muda para ele
    if (local.andar !== currentFloorId.value) {
       changeFloor(local.andar); // Usa a função changeFloor
    }
    // Fecha o menu ao selecionar um local
    closeMenu();
  };

  // Função chamada pelo MapDisplay quando o mapa é clicado
  const handleMapClick = (event, mapRect) => {
    // Passa para o composable de interação, fornecendo callbacks
    handleMapClickInteraction(
      event,
      mapRect,
      (xPercent, yPercent) => { // Callback para definir localização manual
        setUserLocationManually (xPercent, yPercent);
        error.value.geolocation = null; // Limpa msg de erro de geo
        manualModeMessage.value = ''; // Limpa msg de modo manual
      },
      () => { // Callback para tentar fechar o menu
        // Fecha o menu somente se estiver aberto e o clique não foi no botão toggle
        if (isMenuOpen.value && !event.target.closest('.menu-toggle-button')) {
          closeMenu();
        }
      }
    );

    // Se NÃO estiver no modo de definir localização, um clique no mapa deseleciona o local?
     if (!settingUserLocationMode.value) {
         selectedLocal.value = null;
         hidePopup();
     }
  };

  // Inicia a busca automática de localização (acionado pelo botão)
```

```
  const triggerAutoLocation = () => {
    getUserLocationAuto(); // Chama a função do composable
    closeMenu(); // Fecha o menu ao tentar obter localização
  };

  // Ativa o modo de definição manual (acionado pelo botão)
  const triggerManualLocationMode = () => {
    enableSetUserLocationMode(); // Ativa o modo no composable de interação
    manualModeMessage.value = "Clique no mapa para definir sua localização."; // Informa o usuário
    error.value.geolocation = null; // Limpa outros erros de geo
    closeMenu();
  };

  // --- Watchers ---

  // Observa erros dos composables para atualizar o estado de erro local
  watch(geolocationError, (newError) => {
      // Só atualiza o erro de geolocalização se NÃO estiver no modo de definição manual
      if (!settingUserLocationMode.value) {
          error.value.geolocation = newError;
          manualModeMessage.value = ''; // Garante que a msg manual seja limpa se houver erro real
      }
  });
  watch(routeCalcError, (newError) => { error.value.routing = newError; });

  // Observa mudança no andar (já tratado em changeFloor, mas pode ter lógica adicional aqui se necessário)
  // watch(currentFloorId, (newFloor, oldFloor) => { ... });


  // Limpa a seleção e o popup se o local selecionado não pertencer mais ao andar atual
  // (Embora changeFloor já deselecione, este watcher pode ser útil se currentFloorId mudar por outros meios)
  watch(selectedLocal, (newLocal) => {
      if (newLocal && newLocal.andar !== currentFloorId.value) {
          // Isso não deveria acontecer se changeFloor deseleciona, mas como segurança:
          selectedLocal.value = null;
          hidePopup();
      }
```

```
});


// Observa o estado de erro global do LocationService
watch(() => locationServiceState.error.value, (newServiceError) => {
    if (newServiceError && !error.value.initialData) {
        // Mostra erro do serviço se não houver erro mais específico já sendo mostrado
        // Poderia ser mais granular
        console.error("Erro detectado no LocationService:", newServiceError);
    }
});


// --- Ciclo de Vida ---
onMounted(() => {
  loadInitialData(); // Carrega locais, waypoints, andares da API
  getUserLocationAuto(); // Tenta obter localização ao iniciar
});

</script>

<style scoped>
/* Estilos específicos para o layout, se necessário */
.main-content {
  width: 100%;
  height: 100vh;
}

/* Ajuste para sobrepor o erro/loading global se necessário */
.error, .loading {
  z-index: 1100; /* Acima de outros elementos */
}

/* Estilo para a mensagem do modo manual */
.manual-location-prompt {
    position: absolute;
    top: 10%;
```

```
      left: 50%;
      transform: translateX(-50%);
      background: lightblue;
      color: black;
      border: 1px solid blue;
      padding: 10px 15px;
      border-radius: 5px;
      z-index: 1100;
      box-shadow: 0 2px 5px rgba(0,0,0,0.2);
      font-weight: bold;
      text-align: center;
  }

  /* Estilos para o botão de toggle (pode estar em main.css) */
  .menu-toggle-button {
    position: absolute;
    top: 15px;
    left: 15px;
    z-index: 1001; /* Acima de tudo */
    background-color: var(--button-bg-color, #f0f0f0);
    color: var(--text-color, #333);
    border: 1px solid var(--border-color, #ccc);
    border-radius: 4px;
    padding: 8px 12px;
    font-size: 1.5em;
    cursor: pointer;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
    transition: background-color 0.2s ease;
  }
  .menu-toggle-button:hover {
    background-color: var(--button-hover-bg-color, #e0e0e0);
  }

  /* Container principal (pode estar em main.css) */
  .app-container {
    position: relative;
    width: 100vw;
```

```
    height: 100vh;
    display: flex;
    overflow: hidden;
  }
  </style>
```