src/components/LocationAdmin.vue

```vue
<template>
  <div class="admin-container">
    <router-link to="/">Voltar para o Mapa</router-link> <h1>Admin de Locais</h1>

    <div v-if="localLoading" class="loading">Carregando...</div>
    <div v-if="localError" class="error">{{ localError }}</div>
    <div v-if="localSuccessMessage" class="success">{{ localSuccessMessage }}</div>

    <button @click="showAddForm" :disabled="localLoading || showForm" class="btn btn-primary">Adicionar Novo Local</button>

    <div v-if="showForm" class="form-section">
      <h2>{{ isEditing ? 'Editar Local' : 'Adicionar Local' }}</h2>
      <form @submit.prevent="handleSubmit">
        <div class="form-group" v-if="isEditing">
          <label for="id">ID:</label>
          <input type="text" id="id" :value="formData.id" disabled>
        </div>
        <div class="form-group">
          <label for="nome">Nome:</label>
          <input type="text" id="nome" v-model="formData.nome" required>
        </div>
        <div class="form-group">
           <label for="endereco">Endereço/Descrição (para geocodificação no backend, se houver):</label>
          <input type="text" id="endereco" v-model="formData.endereco" required>
        </div>
        <div class="form-group">
          <label for="andar">Andar (ex: terreo, primeiro):</label>
          <input type="text" id="andar" v-model="formData.andar" required>
        </div>
        <div class="form-actions">
          <button type="submit" :disabled="localLoading" class="btn btn-success">{{ isEditing ? 'Salvar Alterações' :
'Criar Local' }}</button>
          <button type="button" @click="cancelForm" :disabled="localLoading" class="btn btn-secondary">Cancelar</button>
        </div>
      </form>
```

```
    </div>

    <h2>Locais Existentes</h2>
    <div class="table-container">
      <table class="locations-table">
        <thead>
          <tr>
            <th>ID</th>
            <th>Nome</th>
            <th>Andar</th>
            <th>Endereço</th>
            <th>X%</th>
            <th>Y%</th>
            <th>Ações</th>
          </tr>
        </thead>
        <tbody>
           <tr v-if="locations.length === 0 && !localLoading && !localError">
            <td colspan="7" style="text-align: center;">Nenhum local cadastrado.</td>
          </tr>
           <tr v-if="locations.length === 0 && localError">
                <td colspan="7" style="text-align: center; color: red;">Falha ao carregar locais.</td>
            </tr>
          <tr v-for="loc in locations" :key="loc.id">
            <td>{{ loc.id }}</td>
            <td>{{ loc.nome }}</td>
            <td>{{ loc.andar }}</td>
            <td>{{ loc.endereco || 'N/A' }}</td>
             <td>{{ loc.x?.toFixed ? loc.x.toFixed(2) : loc.x }}</td>
            <td>{{ loc.y?.toFixed ? loc.y.toFixed(2) : loc.y }}</td>
            <td>
              <button @click="showEditForm(loc)" :disabled="localLoading || showForm" class="btn btn-warning btn-
sm">Editar</button>
              <button @click="confirmDelete(loc.id)" :disabled="localLoading || showForm" class="btn btn-danger btn-
sm">Excluir</button>
            </td>
          </tr>
```

```
        </tbody>
      </table>
    </div>
  </div>
</template>

<script setup>
import { ref, onMounted } from 'vue';
import { useRouter } from 'vue-router';
// Importa as funções CRUD do serviço centralizado
import {
    createLocation,
    updateLocation,
    deleteLocation,
    fetchAdminLocations // Usa a função específica para buscar locais para o admin
} from '@/services/LocationService.js';

const router = useRouter();
const locations = ref([]);
// Estados locais para feedback dentro do componente admin
const localLoading = ref(false);
const localError = ref(null);
const localSuccessMessage = ref(null);

const showForm = ref(false);
const isEditing = ref(false);
const formData = ref({
  id: null,
  nome: '',
  endereco: '',
  andar: '',
  // x: null, // Descomente se adicionar inputs X/Y
  // y: null, // Descomente se adicionar inputs X/Y
});

// Limpa mensagens locais após um tempo
function clearLocalMessages() {
```

```
  setTimeout(() => {
    localError.value = null;
    localSuccessMessage.value = null;
  }, 5000); // Limpa após 5 segundos
}

// --- Funções de Interação com o Serviço ---

async function loadLocations() {
  localLoading.value = true;
  localError.value = null; // Limpa erro anterior
  locations.value = []; // Limpa locais antes de buscar
  try {
    const fetchedLocais = await fetchAdminLocations(); // Usa a função do serviço
    if (fetchedLocais) {
      locations.value = fetchedLocais;
    } else {
       // Se fetchAdminLocations retornar null, o erro já foi setado no serviço,
       // mas podemos colocar uma msg genérica aqui se quisermos.
      localError.value = "Não foi possível carregar os locais.";
    }
  } catch (err) {
    console.error("Erro não capturado ao carregar locais:", err);
    localError.value = `Falha ao carregar dados: ${err.message}`;
  } finally {
    localLoading.value = false;
    if(localError.value) clearLocalMessages(); // Limpa msg de erro tbm
  }
}

async function handleAddLocation() {
  localLoading.value = true;
  localError.value = null;
  localSuccessMessage.value = null;
  try {
     // Prepara os dados do formData
     const dataToSend = {
```

```
        nome: formData.value.nome,
        endereco: formData.value.endereco,
        andar: formData.value.andar,
        // x: formData.value.x, // Envia se for editável
        // y: formData.value.y, // Envia se for editável
     };
    const success = await createLocation(dataToSend); // Usa a função do serviço
    if (success) {
      localSuccessMessage.value = "Local adicionado com sucesso!"; // Mensagem local
      await loadLocations(); // Recarrega a lista
      cancelForm(); // Esconde o formulário
      clearLocalMessages();
    } else {
      // Se retornou false, o erro provavelmente está no estado global do serviço
      // Poderíamos buscá-lo ou mostrar uma msg genérica
      localError.value = "Falha ao adicionar local.";
      clearLocalMessages();
    }
  } catch (err) {
     console.error("Erro ao adicionar local:", err);
     localError.value = `Erro ao criar: ${err.message}`;
     clearLocalMessages();
  } finally {
    localLoading.value = false;
  }
}

async function handleUpdateLocation() {
  localLoading.value = true;
  localError.value = null;
  localSuccessMessage.value = null;
   try {
      const dataToSend = {
        id: formData.value.id,
        nome: formData.value.nome,
        endereco: formData.value.endereco,
        andar: formData.value.andar,
```

```
        // x: formData.value.x, // Envia se for editável
        // y: formData.value.y, // Envia se for editável
      };
    const success = await updateLocation(dataToSend); // Usa a função do serviço
    if (success) {
      localSuccessMessage.value = "Local atualizado com sucesso!"; // Mensagem local
      await loadLocations(); // Recarrega a lista
      cancelForm(); // Esconde o formulário
      clearLocalMessages();
    } else {
      localError.value = "Falha ao atualizar local.";
      clearLocalMessages();
    }
  } catch (err) {
     console.error("Erro ao atualizar local:", err);
     localError.value = `Erro ao atualizar: ${err.message}`;
     clearLocalMessages();
   } finally {
    localLoading.value = false;
  }
}

async function handleDeleteLocation(id) {
  localLoading.value = true;
  localError.value = null;
  localSuccessMessage.value = null;
   try {
    const success = await deleteLocation(id); // Usa a função do serviço
    if (success) {
      localSuccessMessage.value = "Local excluído com sucesso!"; // Mensagem local
      await loadLocations(); // Recarrega a lista
      clearLocalMessages();
    } else {
      localError.value = "Falha ao excluir local.";
      clearLocalMessages();
    }
   } catch (err) {
```

```
      console.error("Erro ao excluir local:", err);
      localError.value = `Erro ao excluir: ${err.message}`;
      clearLocalMessages();
    } finally {
     localLoading.value = false;
  }
}


// --- Funções de UI ---

function resetForm() {
  formData.value = { id: null, nome: '', endereco: '', andar: '' /*, x: null, y: null */ };
}

function showAddForm() {
  resetForm();
  isEditing.value = false;
  showForm.value = true;
  localError.value = null;
  localSuccessMessage.value = null;
}

function showEditForm(location) {
  isEditing.value = true;
  // Preenche o form com os dados do local. Certifique-se que a API retorna todos necessários.
  formData.value = {
    id: location.id,
    nome: location.nome || '',
    endereco: location.endereco || '', // Necessita que a API retorne 'endereco'
    andar: location.andar || '',
    // x: location.x, // Preenche se for editável
    // y: location.y, // Preenche se for editável
  };
  showForm.value = true;
  localError.value = null;
  localSuccessMessage.value = null;
```

```
}

function cancelForm() {
  showForm.value = false;
  isEditing.value = false;
  resetForm();
}

function handleSubmit() {
  // Validação básica
  if (!formData.value.nome || !formData.value.endereco || !formData.value.andar) {
    localError.value = "Por favor, preencha nome, endereço e andar.";
    clearLocalMessages();
    return;
  }
  // Adicione validação para X/Y se forem editáveis

  if (isEditing.value) {
    if(!formData.value.id) {
      localError.value = "ID do local ausente para edição.";
      clearLocalMessages();
      return;
    }
    handleUpdateLocation();
  } else {
    handleAddLocation();
  }
}

function confirmDelete(id) {
  if (window.confirm(`Tem certeza que deseja excluir o local com ID "${id}"? Esta ação não pode ser desfeita.`)) {
    handleDeleteLocation(id);
  }
}

// Carrega locais iniciais ao montar o componente
onMounted(() => {
```

```
  loadLocations();
});
</script>

<style scoped>
/* Estilos específicos para o container geral da página admin */
.admin-container {
  padding: 25px;
  max-width: 900px; /* Limita a largura para melhor leitura */
  margin: 20px auto; /* Centraliza */
  font-family: sans-serif;
  background-color: #fff;
  border-radius: 8px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}

/* Estilos para links e botões */
a {
  color: #007bff;
  text-decoration: none;
  margin-bottom: 15px;
  display: inline-block;
}
a:hover {
  text-decoration: underline;
}

h1, h2 {
  color: #333;
  margin-bottom: 15px;
}

.loading, .error, .success {
  padding: 12px 15px;
  margin-bottom: 20px;
  border-radius: 5px;
  text-align: center;
```

```
  font-size: 0.95em;
}
.loading { background-color: #eef; color: #333; }
.error { background-color: #f8d7da; color: #721c24; border: 1px solid #f5c6cb; }
.success { background-color: #d4edda; color: #155724; border: 1px solid #c3e6cb; }

.btn {
  padding: 10px 18px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  margin: 5px;
  font-size: 1em;
  transition: background-color 0.2s ease, box-shadow 0.2s ease;
  box-shadow: 0 1px 3px rgba(0,0,0,0.1);
}
.btn:hover:not(:disabled) {
  opacity: 0.9;
  box-shadow: 0 2px 5px rgba(0,0,0,0.15);
}
.btn-primary { background-color: #007bff; color: white; }
.btn-success { background-color: #28a745; color: white; }
.btn-warning { background-color: #ffc107; color: #212529; border: 1px solid #d39e00;}
.btn-danger { background-color: #dc3545; color: white; }
.btn-secondary { background-color: #6c757d; color: white; }
.btn-sm { padding: 6px 12px; font-size: 0.85em; }
.btn:disabled { background-color: #e9ecef; color: #6c757d; cursor: not-allowed; box-shadow: none; }


/* Estilos para o formulário */
.form-section {
  margin-top: 25px;
  margin-bottom: 30px;
  padding: 20px;
  border: 1px solid #ddd;
  border-radius: 8px;
  background-color: #fdfdfd;
```

```
}
.form-group {
  margin-bottom: 15px;
}
.form-group label {
  display: block;
  margin-bottom: 6px;
  font-weight: bold;
  color: #555;
}
.form-group input[type="text"],
.form-group input[type="number"], /* Estilo para inputs numéricos (X/Y) */
.form-group select {
  width: 100%; /* Full width */
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 4px;
  box-sizing: border-box; /* Include padding in width */
  font-size: 1em;
}
.form-group input[disabled] {
  background-color: #e9ecef;
  cursor: not-allowed;
}
.form-actions {
  margin-top: 20px;
  text-align: right; /* Alinha botões à direita */
}

/* Estilos para a tabela */
.table-container {
  overflow-x: auto; /* Permite rolagem horizontal em telas pequenas */
}
.locations-table {
  width: 100%;
  margin-top: 25px;
  border-collapse: collapse;
```

```
  font-size: 0.9em;
}
.locations-table th,
.locations-table td {
  border: 1px solid #dee2e6;
  padding: 10px 12px;
  text-align: left;
  vertical-align: middle;
}
.locations-table th {
  background-color: #f8f9fa;
  font-weight: bold;
  color: #495057;
}
.locations-table tr:nth-child(even) {
  background-color: #f8f9fa;
}
.locations-table tr:hover {
    background-color: #e9ecef;
}
.locations-table td button {
    margin-right: 5px; /* Espaço entre botões de ação */
}
</style>
```