src/components/LocationAdmin.vue

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

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

      <button @click="showAddForm" :disabled="loading || 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):</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="loading" class="btn btn-success">{{ isEditing ? 'Salvar Alterações' : 'Criar
Local' }}</button>
            <button type="button" @click="cancelForm" :disabled="loading" 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 && !loading">
              <td colspan="7" style="text-align: center;">Nenhum local cadastrado.</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(2) }}</td>
              <td>{{ loc.y?.toFixed(2) }}</td>
              <td>
                <button @click="showEditForm(loc)" :disabled="loading || showForm" class="btn btn-warning btn-
sm">Editar</button>
                <button @click="confirmDelete(loc.id)" :disabled="loading || 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'; // Para navegação programática se necessário

  // !!! IMPORTANTE: Use a URL /exec da sua implantação FINAL, não a /dev !!!
  // A URL /dev é para testes, use a URL que termina em /exec para produção.
  const API_URL = "https://script.google.com/macros/s/AKfycbwibpdB71MCww6TX3HbA0mL8H-ELtUvEcsUbBHGJXE/dev"; // Substitua se tiver a URL /exec

  const router = useRouter();
  const locations = ref([]);
  const loading = ref(false);
  const error = ref(null);
  const successMessage = ref(null);
  const showForm = ref(false);
  const isEditing = ref(false);
  const formData = ref({
    id: null,
    nome: '',
    endereco: '',
    andar: ''
  });

  // Limpa mensagens após um tempo
  function clearMessages() {
      setTimeout(() => {
          error.value = null;
          successMessage.value = null;
      }, 5000); // Limpa após 5 segundos
  }

  // --- Funções da API ---

  // Função genérica para chamar a API Apps Script via GET com parâmetros
```

```
  async function callApi(params = {}) {
    loading.value = true;
    error.value = null;
    successMessage.value = null;
    let url = `${API_URL}?`;
    const queryParams = [];
    for (const key in params) {
        // Só adiciona o parâmetro se ele tiver um valor (útil para update parcial)
       if (params[key] !== undefined && params[key] !== null && params[key] !== '') {
          queryParams.push(`${encodeURIComponent(key)}=${encodeURIComponent(params[key])}`);
       }
    }
    url += queryParams.join('&');

    console.log("Chamando API:", url); // Para depuração

    try {
      const response = await fetch(url, { method: 'GET', mode: 'cors' }); // GET é o padrão, mas explícito aqui. mode: 'cors' pode ser necessário.

      // Tenta analisar como JSON, mas prepara para texto simples em caso de redirecionamento
       let result;
      try {
          result = await response.json();
      } catch (jsonError) {
          // Se falhar ao analisar JSON, pode ser um redirecionamento ou erro HTML
          console.error("Falha ao analisar JSON:", jsonError);
          const textResponse = await response.text();
          throw new Error(`Resposta inesperada da API (não-JSON): Status ${response.status}. Resposta: ${textResponse.substring(0, 100)}...`);
      }


      console.log("Resposta da API:", result); // Para depuração

      if (!response.ok || result.error || result.status === 'error') {
        const errorMessage = result?.message || result?.error || `Erro HTTP ${response.status}`;
```

```
        throw new Error(errorMessage);
      }

      // Retorna os dados para a função de leitura ou a mensagem de sucesso para outras
      if (params.action === undefined || params.action === 'read') {
          return result.locais || [];
      } else {
          successMessage.value = result.message || 'Operação realizada com sucesso!';
          clearMessages();
          return true; // Indica sucesso para operações CRUD
      }

    } catch (err) {
      console.error(`Erro na API action=${params.action || 'read'}:`, err);
      error.value = `Falha na operação: ${err.message}`;
      clearMessages();
      return params.action === undefined || params.action === 'read' ? [] : false; // Retorna array vazio na leitura, false
no CRUD
    } finally {
      loading.value = false;
    }
  }

  async function fetchLocations() {
    // A API Apps Script precisa retornar a coluna 'endereco' para que ela apareça na tabela/edição
    const fetchedLocais = await callApi(); // Chama sem parâmetros para 'read'
    locations.value = fetchedLocais;
  }

  async function addLocation() {
    const params = {
      action: 'create',
      nome: formData.value.nome,
      endereco: formData.value.endereco,
      andar: formData.value.andar
    };
    const success = await callApi(params);
```

```
    if (success) {
      await fetchLocations(); // Recarrega a lista
      cancelForm(); // Esconde o formulário
    }
  }

  async function updateLocation() {
    const params = {
      action: 'update',
      id: formData.value.id,
      // Envia apenas os campos preenchidos para permitir atualização parcial no Apps Script (se ele suportar)
      ...(formData.value.nome && { nome: formData.value.nome }),
      ...(formData.value.endereco && { endereco: formData.value.endereco }),
      ...(formData.value.andar && { andar: formData.value.andar }),
    };
    const success = await callApi(params);
     if (success) {
      await fetchLocations(); // Recarrega a lista
      cancelForm(); // Esconde o formulário
    }
  }

  async function deleteLocation(id) {
   const params = {
      action: 'delete',
      id: id
    };
    const success = await callApi(params);
    if (success) {
      await fetchLocations(); // Recarrega a lista
    }
  }

  // --- Funções de UI ---

  function resetForm() {
     formData.value = { id: null, nome: '', endereco: '', andar: '' };
```

```
  }

  function showAddForm() {
    resetForm();
    isEditing.value = false;
    showForm.value = true;
    error.value = null; // Limpa erros ao abrir form
    successMessage.value = null;
  }

  function showEditForm(location) {
    isEditing.value = true;
    // Preenche o form com os dados do local.
    // Certifique-se que a API 'read' está retornando 'endereco'
    formData.value = {
        id: location.id,
        nome: location.nome || '',
        endereco: location.endereco || '', // Necessita que a API retorne 'endereco'
        andar: location.andar || ''
    };
    showForm.value = true;
    error.value = null;
    successMessage.value = null;
  }

  function cancelForm() {
    showForm.value = false;
    isEditing.value = false;
    resetForm(); // Limpa os dados do formulário
  }

  function handleSubmit() {
    if (!formData.value.nome || !formData.value.endereco || !formData.value.andar) {
        error.value = "Por favor, preencha todos os campos obrigatórios.";
        clearMessages();
        return;
    }
```

```
    if (isEditing.value) {
      if(!formData.value.id) {
          error.value = "ID do local ausente para edição.";
          clearMessages();
          return;
      }
      updateLocation();
    } else {
      addLocation();
    }
  }

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

  // Carrega locais iniciais ao montar o componente
  onMounted(() => {
    fetchLocations();
  });

  </script>

  <style scoped>
  /* Estilos 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 select { /* Adicionado select se usar */
  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>
```