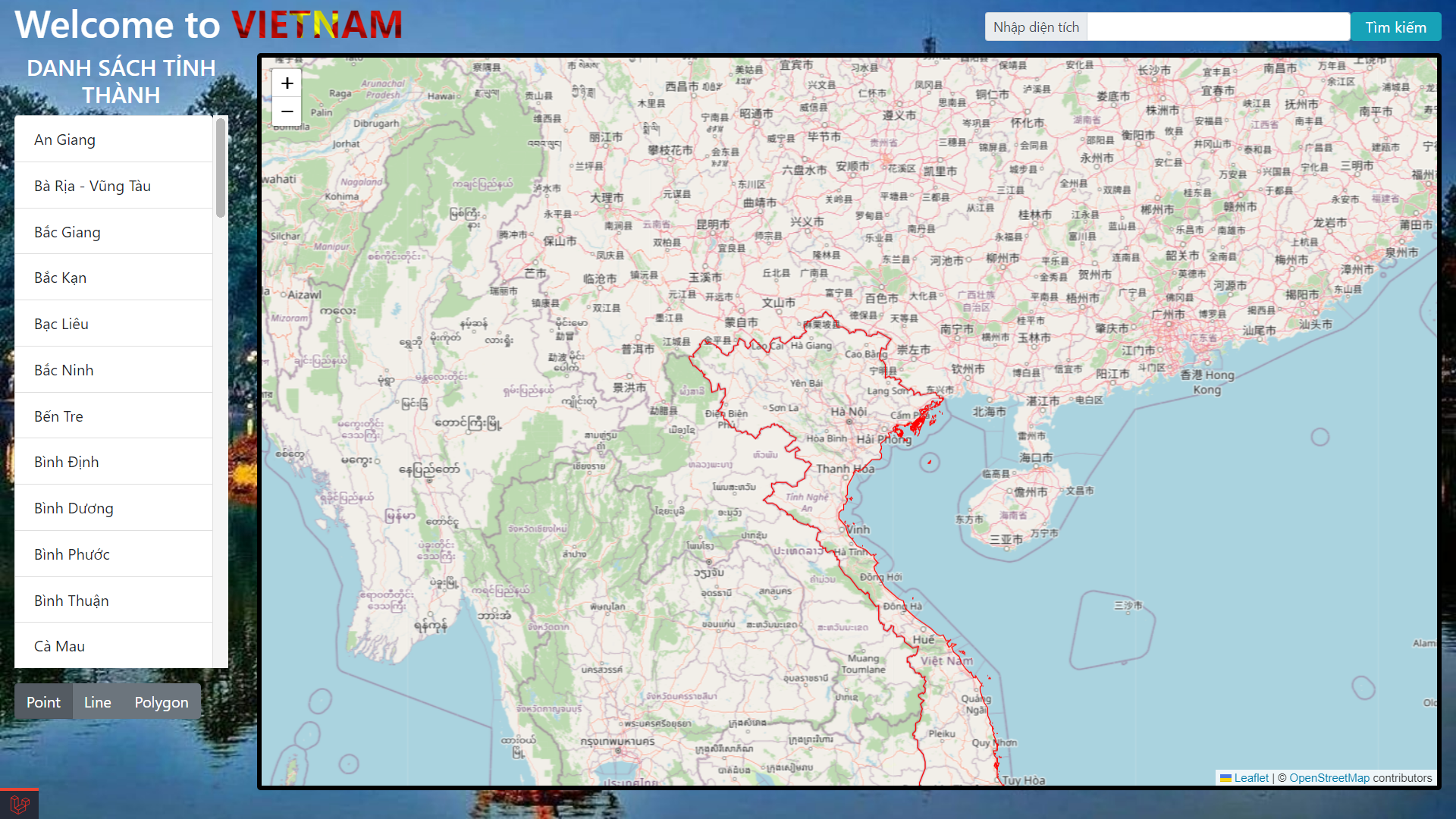
Đọc dữ liệu vùng không gian từ PostGreSQL và hiển thị trên bản đồ với Laravel

# 1. View



File **resources\views\index.blade.php**

| <main>  <div class="container-fluid">  <div class="row" style="height: 90vh;">  <div class="col-lg-2 h-100">  <div class="provinces-list h-100">  <h4 class="text-center text-white">DANH SÁCH TỈNH THÀNH</h4>  <ul id="Provinces\_List" class="list-group h-75 overflow-auto">  @foreach ($provinces as $each)  <li class="list-group-item">{{ $each->name\_1 }}</li>  @endforeach  </ul>  <div class="text-center mt-3 btn-group btn-group-toggle" data-toggle="buttons">  <label class="btn btn-secondary active">  <input type="radio" name="GIS\_Type" id="Point" autocomplete="off"  checked>Point  </label>  <label class="btn btn-secondary">  <input type="radio" name="GIS\_Type" id="Line" autocomplete="off"> Line  </label>  <label class="btn btn-secondary">  <input type="radio" name="GIS\_Type" id="Polygon" autocomplete="off"> Polygon  </label>  </div>  </div>  </div>  <div class="col-lg-10 h-100">  <div id="mapid" class="w-100 h-100 border rounded"  style="border: 5px solid #000 !important;"></div>  </div>  </div>  </div>  </main> |
| --- |

# 2. Điều hướng

File **routes\web.php**

| <?php  use Illuminate\Support\Facades\Route;  use Illuminate\Support\Facades\DB;  use App\Http\Controllers\HomeController;  Route::get('/', [HomeController::class, 'index']);  Route::get('/get-data', [HomeController::class, 'getData']);  Route::get('/get-area', [HomeController::class, 'getArea']); |
| --- |

# 3. Controller

File **app\Http\Controllers\HomeController.php**

| <?php  namespace App\Http\Controllers;  use Illuminate\Http\Request;  use Illuminate\Support\Facades\DB; // Import namespace của lớp DB  class HomeController extends Controller  {  public function index(){  // Thực hiện truy vấn để lấy tọa độ địa lý Vietnam  $vietnam = DB::table('gadm41\_vnm\_0')->select('country')->selectRaw('ST\_AsGeoJSON(geom) AS geometry')->get();  // Thực hiện truy vấn để lấy tọa độ của các tỉnh từ bảng 'gadm41\_VNM\_1'  $provinces = DB::select("  SELECT name\_1, ST\_X(ST\_Centroid(geom)) AS lon, ST\_Y(ST\_Centroid(geom)) AS lat, ST\_AsGeoJSON(geom) AS geometry  FROM gadm41\_VNM\_1  ");  return view('index', compact('vietnam', 'provinces')); // Truyền dữ liệu $provinces vào view  }  public function getData(Request $request)  {  $lat = $request->input('lat');  $lng = $request->input('lng');  // Thực hiện truy vấn để lấy dữ liệu từ DB  $provinces\_geom = DB::select("  SELECT name\_1, ST\_AsGeoJSON(geom) AS geometry  FROM gadm41\_vnm\_1  WHERE ST\_Contains(geom, ST\_MakePoint($lng, $lat))  ");    return response()->json($provinces\_geom);  }  public function getArea(Request $request)  {  $area = $request->input('area');  $provinces\_geom = DB::select("  SELECT name\_1, ST\_Area(geom) AS area  FROM gadm41\_vnm\_1  WHERE ST\_Area(geom) > $area;  ");    return response()->json($provinces\_geom);  }  } |
| --- |

# 4. Mã nguồn xử lý GIS

## 4.1. Khởi tạo các thông tin cơ bản

| // Khởi tạo bản đồ:  var mymap = L.map('mapid').setView([21.0285, 105.8542], 6); // Vị trí Hà Nội  // Thêm lớp tile (lớp dạng ô vuông) từ OpenStreetMap  L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {  attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'  }).addTo(mymap);  // Lấy dữ liệu địa lý của Việt Nam và các tỉnh:  var vietnam = {!! json\_encode($vietnam) !!};  var provinces = {!! json\_encode($provinces) !!};  // Tạo các nhóm lớp (Layer Group) cho các tỉnh và Việt Nam:  var provincesLayerGroup = L.layerGroup().addTo(mymap);  var vietnamLayerGroup = L.layerGroup().addTo(mymap);  // Hiển thị biên của Việt Nam trên bản đồ:  var geomVNM = JSON.parse(vietnam[0].geometry);  geomVNM.coordinates.forEach(function(polygonCoords) {  var border = L.polyline(polygonCoords[0].map(function(coord) {  return [coord[1], coord[0]];  }), {  color: 'red',  weight: 1  });  border.addTo(vietnamLayerGroup);  }); |
| --- |

## 4.2. Các hàm bắt sự kiện và xử lý validate

| **// Bắt sự kiện khi người dùng nhấp vào các tỉnh trên danh sách:**  $('#Provinces\_List').on('click', 'li', function(event) {  var target = $(this).text();  var pointRadioButton = $('#Point');  var lineRadioButton = $('#Line');  var polygonRadioButton = $('#Polygon');  if (polygonRadioButton.is(':checked')) {  renderPolygon(target);  } else if (pointRadioButton.is(':checked')) {  renderPoint(target);  }  });  **// Bắt sự kiện khi người dùng nhấp vào bản đồ để lấy dữ liệu tọa độ:**  mymap.on('click', function(e) {  getData(e.latlng.lat, e.latlng.lng);  });  **// Bắt sự kiện khi người dùng thay đổi loại hình học (điểm, đường, hoặc đa giác):**  $('input[name="GIS\_Type"]').change(function() {  $('label.btn').removeClass('active');  $(this).closest('label.btn').addClass('active');  });  **// Bắt sự kiện khi người dùng nhấp vào một mục trên danh sách:**  $('.list-group-item').on('click', function() {  $('.list-group-item').removeClass('active');  $(this).addClass('active');  });  **// Thực hiện validate:**  $("#AreaInput").on("input", function() {  var inputValue = $(this).val();  if (!/^\d\*\.?\d\*$/.test(inputValue)) {  alert("Vui lòng chỉ nhập số hoặc số thập phân.");  $(this).val("");  }  });  **// Kiểm tra và hạn chế nhập liệu cho ô nhập diện tích:**  $("#AreaSearch").click(function() {  var inputValue = $("#AreaInput").val();  searchArea(inputValue);  }); |
| --- |

## 4.3. Xử lý khi người dùng click vào tỉnh thành trên danh sách

| function **renderPolygon**(target) {  provincesLayerGroup.clearLayers();  provinces.forEach(function(province) {  if (province.name\_1 === target) {  var geom = JSON.parse(province.geometry);  L.geoJSON(geom, {  style: {  fillColor: 'yellow',  fillOpacity: 0.7,  color: 'black',  weight: 2  }  }).addTo(provincesLayerGroup).bindPopup(province.name\_1);  }  });  }  function **renderPoint**(target) {  provincesLayerGroup.clearLayers();  provinces.forEach(function(province) {  if (province.name\_1 === target) {  var lat = parseFloat(province.lat);  var lon = parseFloat(province.lon);  L.marker([lat, lon]).addTo(provincesLayerGroup).bindPopup(province.name\_1);  }  });  } |
| --- |

## 4.4. Hàm xử lý khi người dùng click vào điểm bất kỳ trên bản đồ

| function **getData**(lat, lng) {  $.ajax({  url: '/get-data',  type: 'GET',  data: {  lat: lat,  lng: lng  },  success: function(response) {  provincesLayerGroup.clearLayers();  var provinceName = response[0].name\_1;  provinces.forEach(function(province) {  if (province.name\_1 === provinceName) {  var geom = JSON.parse(province.geometry);  L.geoJSON(geom, {  style: {  fillColor: 'yellow',  fillOpacity: 0.7,  color: 'black',  weight: 2  }  }).addTo(provincesLayerGroup).bindPopup(province.name\_1)  .bindTooltip(province.name\_1, {  permanent: true,  direction: 'top'  });  }  });  },  error: function(xhr) {  console.log('Error:', xhr);  }  });  } |
| --- |

## 4.5. Hàm xử lý tô màu tỉnh có diện tích lớn hơn diện tích được nhập

| function **searchArea**(inputValue) {  $.ajax({  url: '/get-area',  type: 'GET',  data: {  area: inputValue  },  success: function(response) {  provincesLayerGroup.clearLayers();  var provinceName = response;  provinceName.forEach(function(pro) {  console.log(pro.name\_1);  provinces.forEach(function(province) {  if (province.name\_1 === pro.name\_1) {  var geom = JSON.parse(province.geometry);  L.geoJSON(geom, {  style: {  fillColor: 'yellow',  fillOpacity: 0.7,  color: 'black',  weight: 2  }  }).addTo(provincesLayerGroup)  .bindTooltip(province.name\_1, {  permanent: true,  direction: 'top'  });  }  });  });  },  error: function(xhr) {  console.log('Error:', xhr);  }  });  } |
| --- |

# 5. Mã nguồn chi tiết

File **resources\views\index.blade.php**

| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Main Page</title>  <link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css"  integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">  <style>  body {  background-image: url('https://static.vinwonders.com/production/ho-hoan-kiem-2.jpg');  background-size: cover;  background-position: center;  background-repeat: no-repeat;  }  .list-group-item:hover {  background-color: rgb(153, 213, 251);  cursor: pointer;  }  .list-group-item:active {  background-color: rgb(60, 180, 255);  }  .text-with-image {  font-weight: bold;  color: transparent;  background-image: url('https://st.quantrimang.com/photos/image/2021/02/04/Hinh-nen-Quoc-Ky-VN-9.jpg');  background-size: cover;  -webkit-background-clip: text;  background-clip: text;  background-position: center center;  }  </style>  </head>  <body>  <div id="app">  <div id="Header">  <div class="container-fluid">  <div class="row align-items-center h-100">  <div class="col-lg-4">  <h1 class="text-white">Welcome to <span class="text-with-image">VIETNAM</span></h1>  </div>  <div class="col-lg-4"></div>  <div class="col-lg-4 d-flex align-items-center justify-content-center">  <div class="input-group input-group-sm">  <div class="input-group-prepend">  <span class="input-group-text" id="inputGroup-sizing-sm">Nhập diện tích</span>  </div>  <input id="AreaInput" type="text" class="form-control" aria-label="Small"  aria-describedby="inputGroup-sizing-sm">  </div>  <button id="AreaSearch" style="height: 30px; width: 120px" type="button"  class="btn btn-info m-0 p-0">Tìm kiếm</button>  </div>  </div>  </div>  </div>  <main>  <div class="container-fluid">  <div class="row" style="height: 90vh;">  <div class="col-lg-2 h-100">  <div class="provinces-list h-100">  <h4 class="text-center text-white">DANH SÁCH TỈNH THÀNH</h4>  <ul id="Provinces\_List" class="list-group h-75 overflow-auto">  @foreach ($provinces as $each)  <li class="list-group-item">{{ $each->name\_1 }}</li>  @endforeach  </ul>  <div class="text-center mt-3 btn-group btn-group-toggle" data-toggle="buttons">  <label class="btn btn-secondary active">  <input type="radio" name="GIS\_Type" id="Point" autocomplete="off"  checked>Point  </label>  <label class="btn btn-secondary">  <input type="radio" name="GIS\_Type" id="Line" autocomplete="off"> Line  </label>  <label class="btn btn-secondary">  <input type="radio" name="GIS\_Type" id="Polygon" autocomplete="off"> Polygon  </label>  </div>  </div>  </div>  <div class="col-lg-10 h-100">  <div id="mapid" class="w-100 h-100 border rounded"  style="border: 5px solid #000 !important;"></div>  </div>  </div>  </div>  </main>  <div id="Footer">  </div>  </div>  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>  <script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>  <script>  $(document).ready(function() {  // Khởi tạo bản đồ:  var mymap = L.map('mapid').setView([21.0285, 105.8542], 6); // Vị trí Hà Nội  // Thêm lớp tile (lớp dạng ô vuông) từ OpenStreetMap  L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {  attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'  }).addTo(mymap);  // Lấy dữ liệu địa lý của Việt Nam và các tỉnh:  var vietnam = {!! json\_encode($vietnam) !!};  var provinces = {!! json\_encode($provinces) !!};  // Tạo các nhóm lớp (Layer Group) cho các tỉnh và Việt Nam:  var provincesLayerGroup = L.layerGroup().addTo(mymap);  var vietnamLayerGroup = L.layerGroup().addTo(mymap);  // Hiển thị biên của Việt Nam trên bản đồ:  var geomVNM = JSON.parse(vietnam[0].geometry);  geomVNM.coordinates.forEach(function(polygonCoords) {  var border = L.polyline(polygonCoords[0].map(function(coord) {  return [coord[1], coord[0]];  }), {  color: 'red',  weight: 1  });  border.addTo(vietnamLayerGroup);  });  // Bắt sự kiện khi người dùng nhấp vào các tỉnh trên danh sách:  $('#Provinces\_List').on('click', 'li', function(event) {  var target = $(this).text();  var pointRadioButton = $('#Point');  var lineRadioButton = $('#Line');  var polygonRadioButton = $('#Polygon');  if (polygonRadioButton.is(':checked')) {  renderPolygon(target);  } else if (pointRadioButton.is(':checked')) {  renderPoint(target);  }  });  // Bắt sự kiện khi người dùng nhấp vào bản đồ để lấy dữ liệu tọa độ:  mymap.on('click', function(e) {  getData(e.latlng.lat, e.latlng.lng);  });  // Bắt sự kiện khi người dùng thay đổi loại hình học (điểm, đường, hoặc đa giác):  $('input[name="GIS\_Type"]').change(function() {  $('label.btn').removeClass('active');  $(this).closest('label.btn').addClass('active');  });  // Bắt sự kiện khi người dùng nhấp vào một mục trên danh sách:  $('.list-group-item').on('click', function() {  $('.list-group-item').removeClass('active');  $(this).addClass('active');  });  // Thực hiện validate:  $("#AreaInput").on("input", function() {  var inputValue = $(this).val();  if (!/^\d\*\.?\d\*$/.test(inputValue)) {  alert("Vui lòng chỉ nhập số hoặc số thập phân.");  $(this).val("");  }  });  // Kiểm tra và hạn chế nhập liệu cho ô nhập diện tích:  $("#AreaSearch").click(function() {  var inputValue = $("#AreaInput").val();  searchArea(inputValue);  });  // Xử lý khi người dùng click vào tỉnh thành trên danh sách  function renderPolygon(target) {  provincesLayerGroup.clearLayers();  provinces.forEach(function(province) {  if (province.name\_1 === target) {  var geom = JSON.parse(province.geometry);  L.geoJSON(geom, {  style: {  fillColor: 'yellow',  fillOpacity: 0.7,  color: 'black',  weight: 2  }  }).addTo(provincesLayerGroup).bindPopup(province.name\_1);  }  });  }  function renderPoint(target) {  provincesLayerGroup.clearLayers();  provinces.forEach(function(province) {  if (province.name\_1 === target) {  var lat = parseFloat(province.lat);  var lon = parseFloat(province.lon);  L.marker([lat, lon]).addTo(provincesLayerGroup).bindPopup(province.name\_1);  }  });  }  // Hàm xử lý và render Polygon khi người dùng click vào điểm trên bản đồ  function getData(lat, lng) {  $.ajax({  url: '/get-data',  type: 'GET',  data: {  lat: lat,  lng: lng  },  success: function(response) {  provincesLayerGroup.clearLayers();  var provinceName = response[0].name\_1;  provinces.forEach(function(province) {  if (province.name\_1 === provinceName) {  var geom = JSON.parse(province.geometry);  L.geoJSON(geom, {  style: {  fillColor: 'yellow',  fillOpacity: 0.7,  color: 'black',  weight: 2  }  }).addTo(provincesLayerGroup).bindPopup(province.name\_1)  .bindTooltip(province.name\_1, {  permanent: true,  direction: 'top'  });  }  });  },  error: function(xhr) {  console.log('Error:', xhr);  }  });  }  // Xử lý tô màu tỉnh có diện tích lớn hơn diện tích được nhập  function searchArea(inputValue) {  $.ajax({  url: '/get-area',  type: 'GET',  data: {  area: inputValue  },  success: function(response) {  provincesLayerGroup.clearLayers();  var provinceName = response;  provinceName.forEach(function(pro) {  console.log(pro.name\_1);  provinces.forEach(function(province) {  if (province.name\_1 === pro.name\_1) {  var geom = JSON.parse(province.geometry);  L.geoJSON(geom, {  style: {  fillColor: 'yellow',  fillOpacity: 0.7,  color: 'black',  weight: 2  }  }).addTo(provincesLayerGroup)  .bindTooltip(province.name\_1, {  permanent: true,  direction: 'top'  });  }  });  });  },  error: function(xhr) {  console.log('Error:', xhr);  }  });  }  });  </script>  </body>  </html> |
| --- |

# 