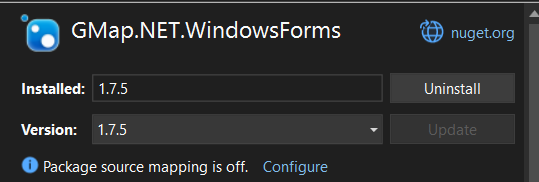
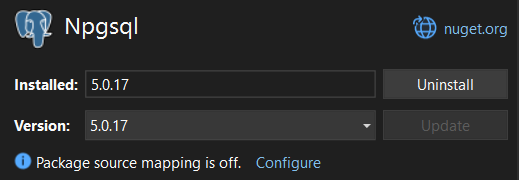
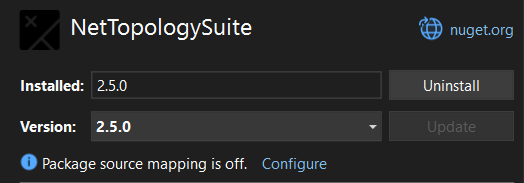
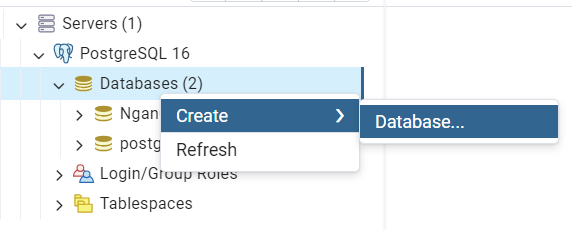
Display data (Points, Lines, Polygons) from database to Winform Map

# 1. Cài đặt

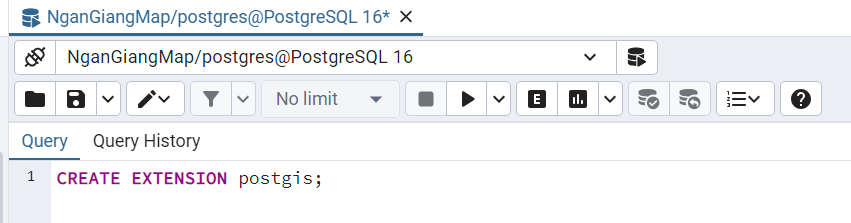
* Cài đặt PostgreSQL tại đây: <https://www.postgresql.org/>
* Hướng dẫn cài đặt: <https://www.youtube.com/watch?v=WFEsT60q30k>
* Cài đặt file dữ liệu chứa dữ liệu shapefile
* Cài đặt thư viện trong dự án winform
* 
* 
* 

# 2. Hướng dẫn đẩy dữ liệu shapefile vào CSDL Postgresql

Mở **PgAdmin4** và tiến hành tạo các thông tin cơ bản trong lần đầu. Tiếp theo tạo mới Database như hình dưới



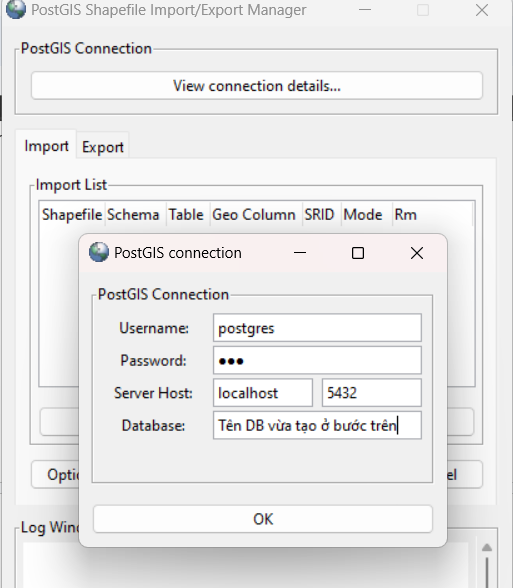
Mở trình soạn câu lệnh truy vấn chạy câu lệnh như bên dưới



Chọn App như hình bên dưới:



Tiến hành kết nối



# 3. Hướng dẫn chi tiết mã nguồn

## 3.1. Giao diện

## 

## 3.2. Nhập các thư viện

| using System;  using System.Collections.Generic;  using System.Data;  using System.Windows.Forms;  using GMap.NET;  using GMap.NET.MapProviders;  using GMap.NET.WindowsForms;  using GMap.NET.WindowsForms.Markers;  using Npgsql;  using NetTopologySuite.Geometries;  using NetTopologySuite.IO; |
| --- |

## 3.3. Khai báo, khởi tạo và đóng Form

| private void Form1\_FormClosing(object sender, FormClosingEventArgs e)  {  // Giải phóng bản đồ khi form đóng  if (map != null)  {  map.Dispose();  }  } |
| --- |

| private void Form1\_Load(object sender, EventArgs e)  {  string query = "SELECT name\_1, hasc\_1 FROM cmr\_adm1";  DataTable dt = new DataTable();  using (NpgsqlConnection conn = new NpgsqlConnection(connectionString))  {  conn.Open();  using (NpgsqlCommand cmd = new NpgsqlCommand(query, conn))  {  NpgsqlDataAdapter adapter = new NpgsqlDataAdapter(cmd);  adapter.Fill(dt);  dtgv.DataSource = dt;  }  }  } |
| --- |

| private void Form1\_FormClosing(object sender, FormClosingEventArgs e)  {  // Giải phóng bản đồ khi form đóng  if (map != null)  {  map.Dispose();  }  } |
| --- |

## 3.4. Hàm xử lý vẽ Line

| private void AddLine(string connectionString, string searchItem)  {  map.Overlays.Clear();  using (NpgsqlConnection conn = new NpgsqlConnection(connectionString))  {  conn.Open();  string query = "SELECT ST\_ASTEXT(cmr\_roads.geom) " +  "FROM cmr\_roads " +  "WHERE EXISTS (" +  "SELECT 1 FROM cmr\_adm1 " +  "WHERE ST\_Within(cmr\_roads.geom, cmr\_adm1.geom) " +  "AND cmr\_adm1.name\_1 = @searchItem " +  ");";  using (NpgsqlCommand cmd = new NpgsqlCommand(query, conn))  {  cmd.Parameters.AddWithValue("@searchItem", searchItem);  using (NpgsqlDataReader reader = cmd.ExecuteReader())  {  var geomReader = new WKTReader();  GMapOverlay overlay = new GMapOverlay("my\_overlay");  while (reader.Read())  {  string geomText = reader.GetString(0);  Geometry geom = geomReader.Read(geomText);  if (geom is MultiLineString multiLineString)  {  foreach (LineString lineString in multiLineString.Geometries)  {  List<PointLatLng> points = GetPointsFromLineString(lineString);  GMapRoute route = new GMapRoute(points, "A road");  overlay.Routes.Add(route);  }  }  }  map.Overlays.Add(overlay);  map.SetPositionByKeywords("Cameroon");  map.Position = map.Position;  }  }  }  }  private List<PointLatLng> GetPointsFromLineString(LineString lineString)  {  List<PointLatLng> points = new List<PointLatLng>();  foreach (Coordinate coordinate in lineString.Coordinates)  {  points.Add(new PointLatLng(coordinate.Y, coordinate.X)); // Swap X and Y for GMap compatibility  }  return points;  } |
| --- |

## 3.5. Hàm xử lý vẽ điểm và vùng

Hàm hiển thị dữ liệu lên bản đồ

| private void DisplayDataOnMap(string searchItem)  {  using (NpgsqlConnection conn = new NpgsqlConnection(connectionString))  {  conn.Open();  string query = "SELECT ST\_AsText(geom) AS geom\_text FROM cmr\_adm1 " +  "WHERE name\_1 = @searchItem";  using (NpgsqlCommand cmd = new NpgsqlCommand(query, conn))  {  cmd.Parameters.AddWithValue("@searchItem", searchItem);  using (NpgsqlDataReader reader = cmd.ExecuteReader())  {  if (reader.Read())  {  var geomReader = new WKTReader();  GMapOverlay overlay = new GMapOverlay("my\_overlay");  string geomText = reader.GetString(0);  Geometry geom = geomReader.Read(geomText);  if (geom is MultiPolygon multiPolygon)  {  foreach (Polygon polygon in multiPolygon.Geometries)  {  List<PointLatLng> points = GetPointsFromPolygon(polygon);  if (rbPoint.Checked)  {  map.Overlays.Clear();  PointLatLng center = GetCenterPoint(points);  GMapMarker marker = new GMarkerGoogle(center, GMarkerGoogleType.red\_dot);  GMapOverlay markersOverlay = new GMapOverlay("markers");  markersOverlay.Markers.Add(marker);  map.Overlays.Add(markersOverlay);  }  if (rbPolygon.Checked)  {  map.Overlays.Clear();  GMapPolygon gmapPolygon = new GMapPolygon(points, "Polygon");  overlay.Polygons.Add(gmapPolygon);  }  }  map.Overlays.Add(overlay);  map.SetPositionByKeywords(searchItem + ", Japan");  map.Position = map.Position;  }  }  }  }  }  } |
| --- |

Lấy điểm từ vùng

| private List<PointLatLng> GetPointsFromPolygon(Polygon polygon)  {  List<PointLatLng> points = new List<PointLatLng>();  foreach (Coordinate coordinate in polygon.Coordinates)  {  points.Add(new PointLatLng(coordinate.Y, coordinate.X)); // Swap X and Y for GMap compatibility  }  return points;  } |
| --- |

Lấy tọa độ trung tâm từ các tọa độ địa lý được truyền vào

| private PointLatLng GetCenterPoint(List<PointLatLng> points)  {  double totalLat = 0;  double totalLng = 0;  foreach (var point in points)  {  totalLat += point.Lat;  totalLng += point.Lng;  }  double avgLat = totalLat / points.Count;  double avgLng = totalLng / points.Count;  return new PointLatLng(avgLat, avgLng);  } |
| --- |

## 3.6. Hàm xử lý sự kiện click vào ô trong Datagridview

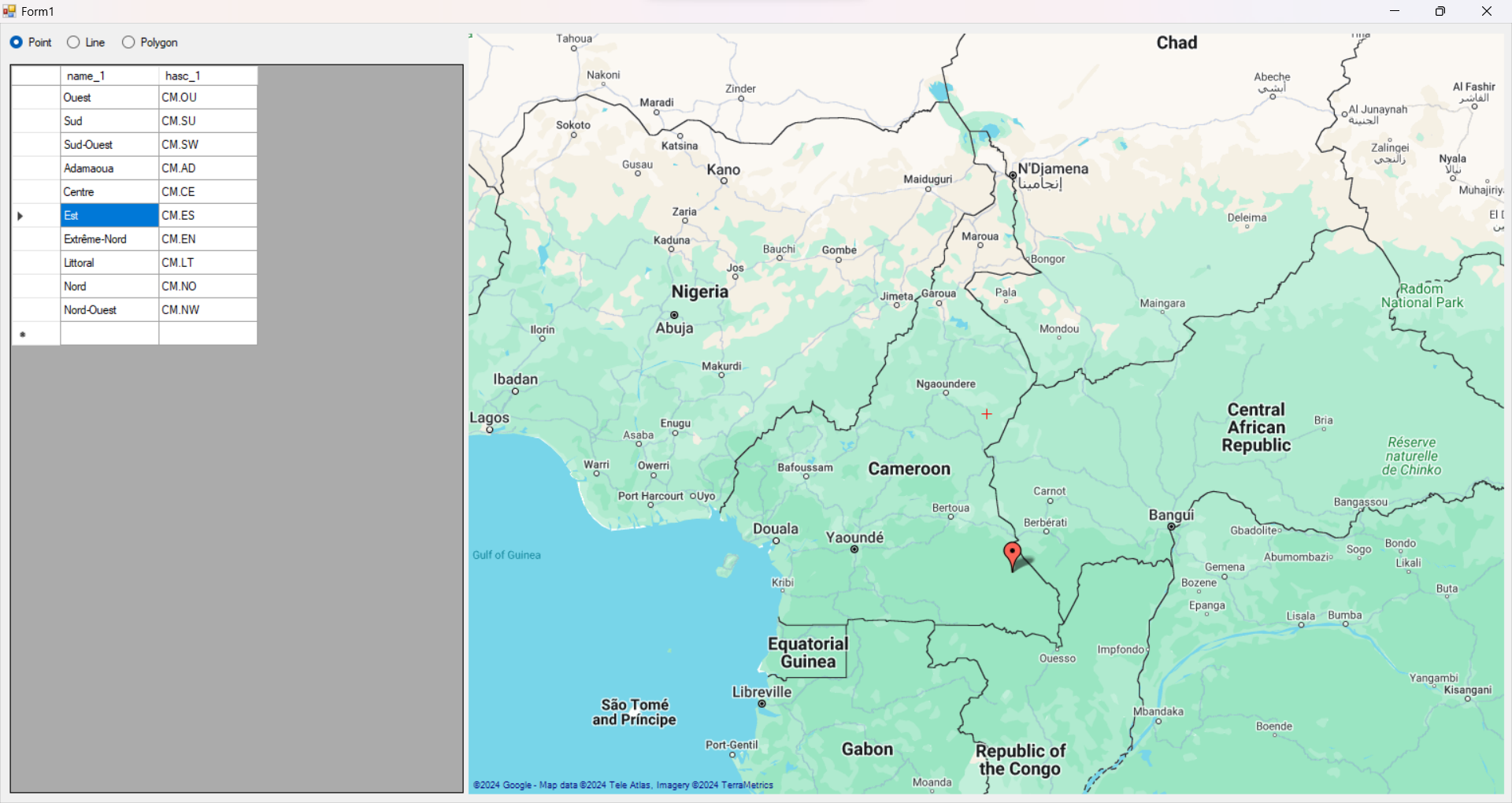
| private void dtgv\_CellClick(object sender, DataGridViewCellEventArgs e)  {  // Kiểm tra xem ô đã bấm có phải là ô dữ liệu không và không phải là ô header  if (e.RowIndex >= 0 && e.ColumnIndex >= 0)  {  object cellValue = dtgv.Rows[e.RowIndex].Cells[0].Value;  // Kiểm tra giá trị của ô không rỗng  if (cellValue != null)  {  string searchItem = cellValue.ToString();  // Kiểm tra xem người dùng đã chọn loại dữ liệu nào để hiển thị trên bản đồ  if (rbPolygon.Checked || rbPoint.Checked)  DisplayDataOnMap(searchItem);  if (rbLine.Checked)  AddLine(connectionString, searchItem);  }  }  } |
| --- |

# 4. Mã nguồn đầy đủ

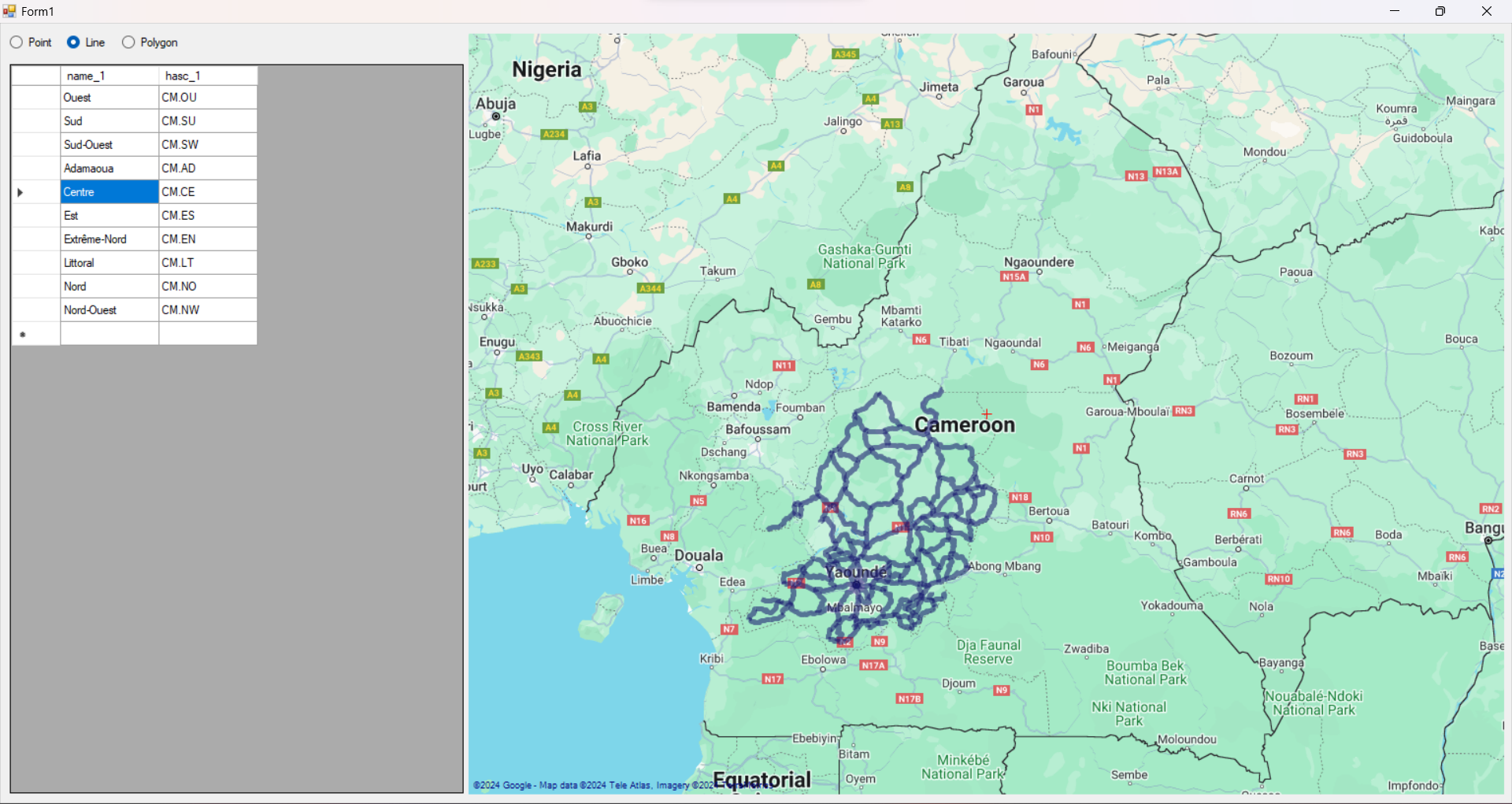
| using System;  using System.Collections.Generic;  using System.Data;  using System.Windows.Forms;  using GMap.NET;  using GMap.NET.MapProviders;  using GMap.NET.WindowsForms;  using GMap.NET.WindowsForms.Markers;  using Npgsql;  using NetTopologySuite.Geometries;  using NetTopologySuite.IO;  namespace Job28  {  public partial class Form1 : Form  {  private GMapControl map;  private string connectionString = "Host=localhost;Port=5432;Database=NganGiangMap;Username=postgres;Password=123;";  public Form1()  {  InitializeComponent();  InitializeMap();  }  private void InitializeMap()  {  // Khởi tạo control GMapControl  map = new GMapControl();  // Đặt thuộc tính Dock của GMapControl thành Fill để nó lấp đầy Panel1  map.Dock = DockStyle.Fill;  //map.MapProvider = GoogleMapProvider.Instance;  map.SetPositionByKeywords("Cameroon"); // Đặt vị trí mặc định của bản đồ  GMaps.Instance.Mode = AccessMode.ServerOnly; // Thiết lập bản đồ truy cập là máy chủ không phải máy khách  map.MapProvider = GMapProviders.GoogleMap;  //map.MapProvider = GMapProviders.GoogleSatelliteMap;  //map.Manager.Mode = AccessMode.ServerOnly; // Đảm bảo rằng bạn sử dụng chế độ server để có thể hiển thị bản đồ giao thông  panelMap.Controls.Add(map);  map.MinZoom = 1;  map.MaxZoom = 20;  map.Zoom = 6;  }  private void Form1\_Load(object sender, EventArgs e)  {  string query = "SELECT name\_1, hasc\_1 FROM cmr\_adm1";  DataTable dt = new DataTable();  using (NpgsqlConnection conn = new NpgsqlConnection(connectionString))  {  conn.Open();  using (NpgsqlCommand cmd = new NpgsqlCommand(query, conn))  {  NpgsqlDataAdapter adapter = new NpgsqlDataAdapter(cmd);  adapter.Fill(dt);  dtgv.DataSource = dt;  }  }  }  private void Form1\_FormClosing(object sender, FormClosingEventArgs e)  {  // Giải phóng bản đồ khi form đóng  if (map != null)  {  map.Dispose();  }  }  private void dtgv\_CellClick(object sender, DataGridViewCellEventArgs e)  {  // Kiểm tra xem ô đã bấm có phải là ô dữ liệu không và không phải là ô header  if (e.RowIndex >= 0 && e.ColumnIndex >= 0)  {  object cellValue = dtgv.Rows[e.RowIndex].Cells[0].Value;  // Kiểm tra giá trị của ô không rỗng  if (cellValue != null)  {  string searchItem = cellValue.ToString();  // Kiểm tra xem người dùng đã chọn loại dữ liệu nào để hiển thị trên bản đồ  if (rbPolygon.Checked || rbPoint.Checked)  DisplayDataOnMap(searchItem);  if (rbLine.Checked)  AddLine(connectionString, searchItem);  }  }  }  private void AddLine(string connectionString, string searchItem)  {  map.Overlays.Clear();  using (NpgsqlConnection conn = new NpgsqlConnection(connectionString))  {  conn.Open();  string query = "SELECT ST\_ASTEXT(cmr\_roads.geom) " +  "FROM cmr\_roads " +  "WHERE EXISTS (" +  "SELECT 1 FROM cmr\_adm1 " +  "WHERE ST\_Within(cmr\_roads.geom, cmr\_adm1.geom) " +  "AND cmr\_adm1.name\_1 = @searchItem " +  ");";  using (NpgsqlCommand cmd = new NpgsqlCommand(query, conn))  {  cmd.Parameters.AddWithValue("@searchItem", searchItem);  using (NpgsqlDataReader reader = cmd.ExecuteReader())  {  // Đối tượng WKTReader từ thư viện GeoAPI (chuyển đổi dữ liệu địa lý từ dạng văn bản định dạng  // (WKT - Well-Known Text) sang các đối tượng hình học trong mã lập trình.  // Trong trường hợp này, nó được sử dụng để đọc dữ liệu văn bản định dạng WKT của các hình học từ cơ sở dữ liệu.  var geomReader = new WKTReader();  GMapOverlay overlay = new GMapOverlay("my\_overlay");  while (reader.Read())  {  string geomText = reader.GetString(0);  Geometry geom = geomReader.Read(geomText);  if (geom is MultiLineString multiLineString)  {  foreach (LineString lineString in multiLineString.Geometries)  {  List<PointLatLng> points = GetPointsFromLineString(lineString);  GMapRoute route = new GMapRoute(points, "A road");  overlay.Routes.Add(route);  }  }  }  map.Overlays.Add(overlay);  map.SetPositionByKeywords("Cameroon");  map.Position = map.Position;  }  }  }  }  private List<PointLatLng> GetPointsFromLineString(LineString lineString)  {  List<PointLatLng> points = new List<PointLatLng>();  foreach (Coordinate coordinate in lineString.Coordinates)  {  points.Add(new PointLatLng(coordinate.Y, coordinate.X)); // Swap X and Y for GMap compatibility  }  return points;  }  private void DisplayDataOnMap(string searchItem)  {  using (NpgsqlConnection conn = new NpgsqlConnection(connectionString))  {  conn.Open();  string query = "SELECT ST\_AsText(geom) AS geom\_text FROM cmr\_adm1 " +  "WHERE name\_1 = @searchItem";  using (NpgsqlCommand cmd = new NpgsqlCommand(query, conn))  {  cmd.Parameters.AddWithValue("@searchItem", searchItem);  using (NpgsqlDataReader reader = cmd.ExecuteReader())  {  if (reader.Read())  {  var geomReader = new WKTReader();  GMapOverlay overlay = new GMapOverlay("my\_overlay");  string geomText = reader.GetString(0);  Geometry geom = geomReader.Read(geomText);  if (geom is MultiPolygon multiPolygon)  {  foreach (Polygon polygon in multiPolygon.Geometries)  {  List<PointLatLng> points = GetPointsFromPolygon(polygon);  if (rbPoint.Checked)  {  map.Overlays.Clear();  PointLatLng center = GetCenterPoint(points);  GMapMarker marker = new GMarkerGoogle(center, GMarkerGoogleType.red\_dot);  GMapOverlay markersOverlay = new GMapOverlay("markers");  markersOverlay.Markers.Add(marker);  map.Overlays.Add(markersOverlay);  }  if (rbPolygon.Checked)  {  map.Overlays.Clear();  GMapPolygon gmapPolygon = new GMapPolygon(points, "Polygon");  overlay.Polygons.Add(gmapPolygon);  }  }  map.Overlays.Add(overlay);  map.SetPositionByKeywords(searchItem + ", Japan");  map.Position = map.Position;  }  }  }  }  }  }  private List<PointLatLng> GetPointsFromPolygon(Polygon polygon)  {  List<PointLatLng> points = new List<PointLatLng>();  foreach (Coordinate coordinate in polygon.Coordinates)  {  points.Add(new PointLatLng(coordinate.Y, coordinate.X)); // Swap X and Y for GMap compatibility  }  return points;  }  private PointLatLng GetCenterPoint(List<PointLatLng> points)  {  double totalLat = 0;  double totalLng = 0;  foreach (var point in points)  {  totalLat += point.Lat;  totalLng += point.Lng;  }  double avgLat = totalLat / points.Count;  double avgLng = totalLng / points.Count;  return new PointLatLng(avgLat, avgLng);  }  }  } |
| --- |

# 5. Kết quả nhận được

Point



Line



Polygon

