**Projek Kecil IB Spatial Database**

**IF4040 PEMODELAN DATA LANJUT**

Trapsilo Pramudya Bumi / 13510052 Hanif Eridaputra / 13510091

# DESKRIPSI PERSOALAN

Angkutan Kota (Angkot), merupakan salah satu transportasi yang paling banyak dan paling sering digunakan oleh orang Indonesia salah satunya adalah di Bandung. Setiap angkot di Bandung tentunya memiliki jenis dan rute masing-masing sehingga untuk mencapai suatu tempat tertentu terkadang harus menaiki dua atau lebih angkot. Permasalahannya adalah angkot di Bandung cukup banyak dan tidak semua orang mengetahui rute yang dilewati oleh angkot tersebut. Keterangan rute yang tertera pada angkot biasanya hanya menuliskan terminal awal dan akhir yang dilewati angkot tersebut. Maka dari itu, dibutuhkan sebuah aplikasi yang dapat menyimpan data-data spatial dari rute angkot yang ada di bandung sehingga masyarakat bandung dapat mencari rute untuk mencapai suatu tempat tertentu dengan menggunakan angkot.

# TEKNOLOGI YANG DIGUNAKAN

1. **PostgreSQL**

Salah satu basis data relasional yang biasa digunakan.

1. **PostGIS**

Estensi dari PostgreSQL yang mengurusi masalah basis data spatial dan query-query spatial.

1. **Google Maps Javascript API v3**

Tampilan map dari Google.

1. **PHP**

Bahasa standar yang biasa digunakan untuk aplikasi berbasis web.

# DESAIN UMUM APLIKASI

Aplikasi yang dibuat adalah aplikasi berbasis Web yang memanfaatkan bahasa PHP dan basis data yang dikelola menggunakan PostgreSQL. Aplikasi ini akan memiliki beberapa fitur utama yaitu sebagai berikut:

1. User dapat menemukan angkot apa saja yang harus digunakan untuk mencapai suatu tujuan tertentu dengan menunjuk tempat asal dan tempat tujuan pada peta yang telah ditentukan.
2. User dapat melihat peta rute dari angkot yang akan digunakan.
3. User dapat mengetahui jarak tempuh dari tempat asal ke tempat tujuan.

# RUTE ANGKOT BANDUNG

Data sekitar 38 jenis angkot yang ada, hanya sebanyak 16 jenis yang dimasukkan ke dalam basis data yaitu sebagai berikut:

1. ABDUL MUIS – CICAHEUM VIA BINONG
2. ABDUL MUIS – DAGO
3. ABDUL MUIS – LEDENG
4. CICAHEUM – LEDENG
5. CICAHEUM – CIROYOM
6. CICAHEUM – CIBADUYUT
7. STASIUN HALL – DAGO
8. STASIUN HALL – CIUMBEULEUIT VIA CIHAMPELAS
9. STASIUN HALL – SARIJADI
10. STASIUN HALL – GUNUNG BATU
11. MARGAHAYU RAYA – LEDENG
12. CIROYOM – SARIJADI VIA SUKAJADI
13. SEDERHANA – CIJERAH
14. CIWASTRA – UJUNG BERUNG
15. ABDUL MUIS – MENGGER
16. CIBOGO ATAS – HALTEU ANDIR

# SKEMA BASIS DATA

Basis data yang digunakan sangat sederhana dan hanya memiliki satu buah tabel seperti gambar di bawah. Atribut geom merupakan atribut basis data spatial yang akan menyimpan data spatial dalam bentuk garis rute angkot.



# QUERY POSTGIS YANG DIGUNAKAN

geometry ST\_GeomFromText(text WKT)

Mengembalikan bentuk geometry dari masukan teks dalam bentuk Well-Known Text (WKT)

Implementasi: Untuk membuat geometry dari masukan koordinat, digunakan query

SELECT ST\_GeomFromText('POINT($koordinat\_asal)')

dimana $koordinat\_asal adalah koordinat bujur-lintang dalam bentuk desimal

text ST\_AsGeoJSON(geometry A)

Mengembalikan bentuk GeoJSON dari masukan geometri

Implementasi: Untuk mengembalikan GeoJSON sebagai hasil kalkulasi rute angkot, digunakan query

SELECT ST\_AsGeoJSON('$rute\_akhir')

dimana $rute\_akhir adalah geometry yang akan dikembalikan dalam bentuk GeoJSON.

boolean ST\_Intersects(geometry A, geometry B)

Mengembalikan nilai boolean true jika geometry A berpotongan dengan geometry B

Implementasi: Untuk mengembalikan seluruh rute angkot yang berpotongan dengan titik awal digunakan query

SELECT geom FROM rute WHERE ST\_Intersects('$titik\_asal', geom)

dimana $titik\_asal adalah geometry Point dan geom adalah geometry rute angkot

boolean ST\_Overlaps(geometry A, geometry B)

Mengembalikan nilai boolean true jika geometry A memiliki sebagian ruang yang sama dengan geometry B, berdimensi sama, tetapi tidak ada satu geometry yang melingkupi yang lain

Implementasi: Untuk mencari jurusan angkot yang melingkupi suatu garis, digunakan query

SELECT jurusan FROM rute WHERE ST\_Overlaps(geom, '$bagian\_rute\_akhir')

dimana jurusan adalah nama jurusan angkot, geom adalah geometry rute angkot, dan $bagian\_rute\_akhir adalah garis hasil kalkulasi sementara rute angkot yang harus digunakan

boolean ST\_Equals(geometry A, geometry B)

Mengembalikan nilai boolean true jika geometry A mewakili ruang yang sama dengan geometry B

Implementasi: Untuk mencari rute lain yang berpotongan dengan suatu rute angkot, namun tidak sama dengan rute angkot itu sendiri digunakan query

SELECT geom FROM rute WHERE ST\_Intersects('$rute\_proses', geom) and not ST\_Equals('$rute\_proses', geom)

dimana geom adalah geometry rute angkot dan $rute\_proses adalah rute angkot yang sedang diproses

geometry ST\_Buffer(geometry A, float radius)

Mengembalikan geometry ruang di sekeliling geometry A sejauh radius

Implementasi: Untuk membentuk lingkaran dari sebuah titik dengan radius sebesar 0,001 digunakan query

ST\_Buffer('$titik\_asal', 0,001)

Dengan $titik\_asal adalah titik asal naik angkot

geometry ST\_LineSubstring(geometry a\_linestring, float start, float end)

Mengembalikan geometry yang merupakan bagian dari LineString a\_linestring, dengan titik awal start dan titik akhir end yang merupakan pecahan dari total panjang A

Implementasi: Untuk menyeleksi garis setelah pengguna naik angkot digunakan query

SELECT ST\_LineSubstring('$inter\_geom', $location\_point\_awal, 1)

dimana $inter\_geom adalah rute angkot yang berpotongan dengan titik awal, $location\_point\_awal adalah angka desimal yang menunjukkan lokasi titik awal relatif terhadap panjang rute

geometry ST\_Intersection(geometry A, geometry B)

Mengembalikan geometry perpotongan geometry A dan geometry B

Implementasi: Untuk menyeleksi perpotongan antara dua rute angkot, digunakan query

SELECT ST\_Intersection('$rute\_akhir', '$rute\_awal')

dimana $rute\_akhir dan $rute\_awal adalah geometry rute angkot

geometry ST\_StartPoint(geometry a\_linestring)

mengembalikan Point yang merupakan titik paling awal dari sebuah LineString a\_linestring

Implementasi: Untuk menyeleksi titik awal sebuah garis rute angkot, digunakan query

SELECT ST\_StartPoint('$lokasi\_potong')

dimana $lokasi\_potong adalah sebuah garis perpotongan rute angkot

float ST\_LineLocatePoint(geometry a\_linestring, geometry a\_point)

Mengembalikan angka desimal antara 0 dan 1 yang merupakan lokasi titik terdekat pada LineString a\_linestring dengan Point a\_point

Implementasi: untuk mengembalikan lokasi perpotongan titik dalam suatu rute angkot digunakan query

SELECT ST\_LineLocatePoint('$acuan', '$titik\_potong')

dimana $acuan adalah garis rute angkot yang dijadikan acuan dan $titik\_potong adalah titik perpotongan rute angkot dengan rute lainnya

text ST\_GeometryType(geometry A)

Mengembalikan tipe geometry A

Implementasi:

SELECT ST\_GeometryType('$titik\_potong\_geom')

integer ST\_NumGeometries(geometry A)

Mengembalikan jumlah geometry yang terkandung dalam geometry A jika A adalah koleksi geometry, 1 jika A adalah geometry tunggal

Implementasi:

SELECT ST\_NumGeometries('$titik\_potong\_geom')

geometry ST\_GeometryN(geometry A, integer N)

Mengembalikan geometry ke-N jika A adalah koleksi geometry

Implementasi:

SELECT ST\_GeometryN('$titik\_potong\_geom', $i)

# ALGORITMA RUTE ANGKOT

Algoritma pencarian rute angkot adalah seperti gambar berikut:

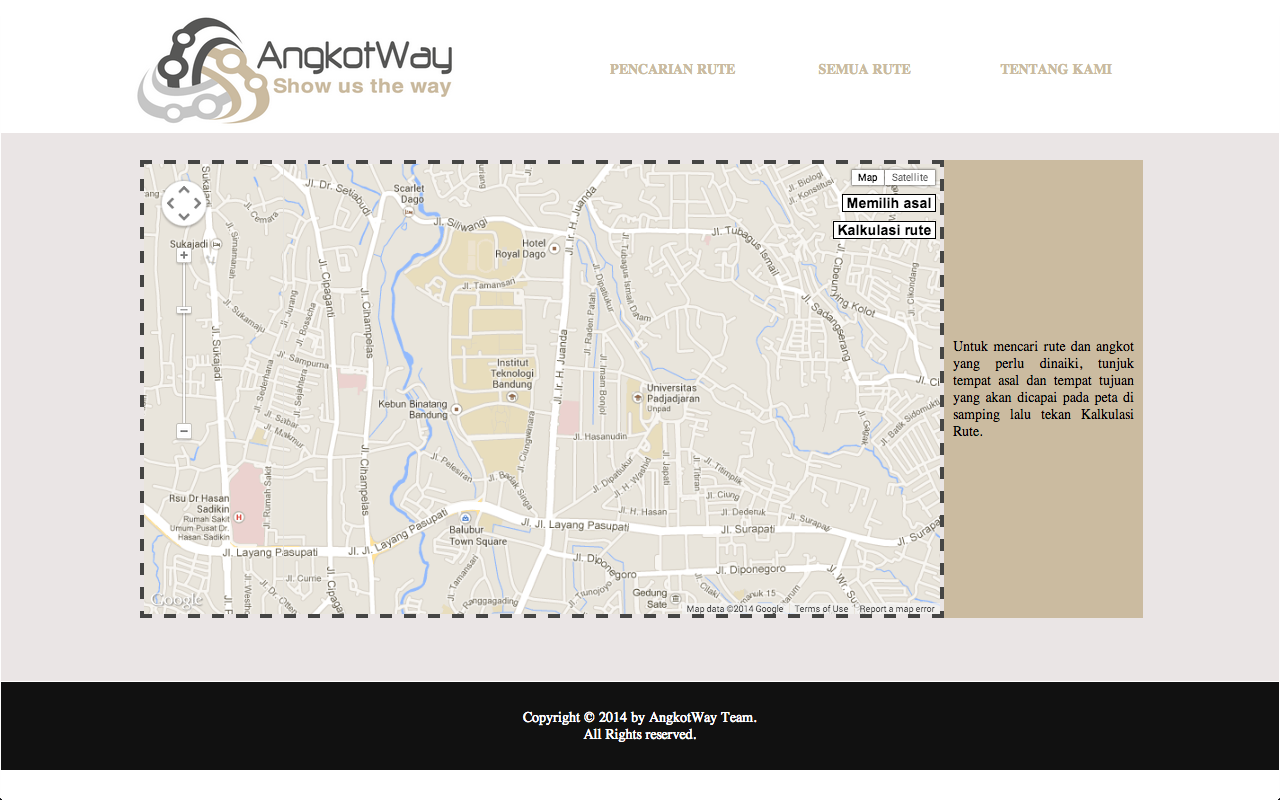
# SARAH:Users:haniferidaputra:Sites:pdl_ruteangkot:algoritma rute angkot.png

# LANGKAH-LANGKAH MENJALANKAN APLIKASI

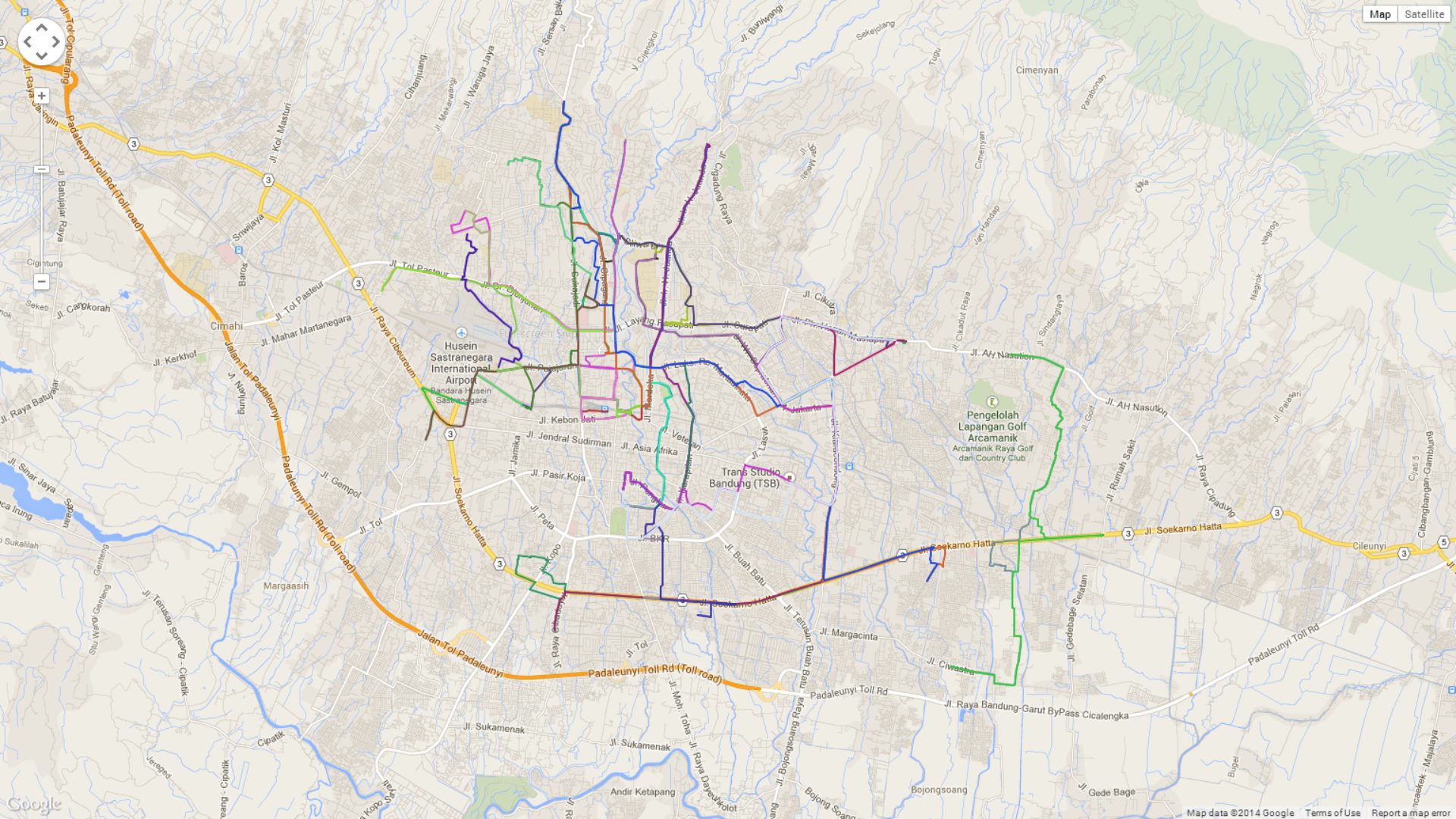
1. Pengguna memasukkan koordinat asal dan koordinat tujuan ke peta
2. Aplikasi(php) melakukan query ke postgre/postgis
3. Postgis mengembalikan jurusan angkot yang harus ditempuh
4. Peta menampilkan jurusan angkot yang dikembalikan

# SCREENSHOT APLIKASI ANGKOTWAY

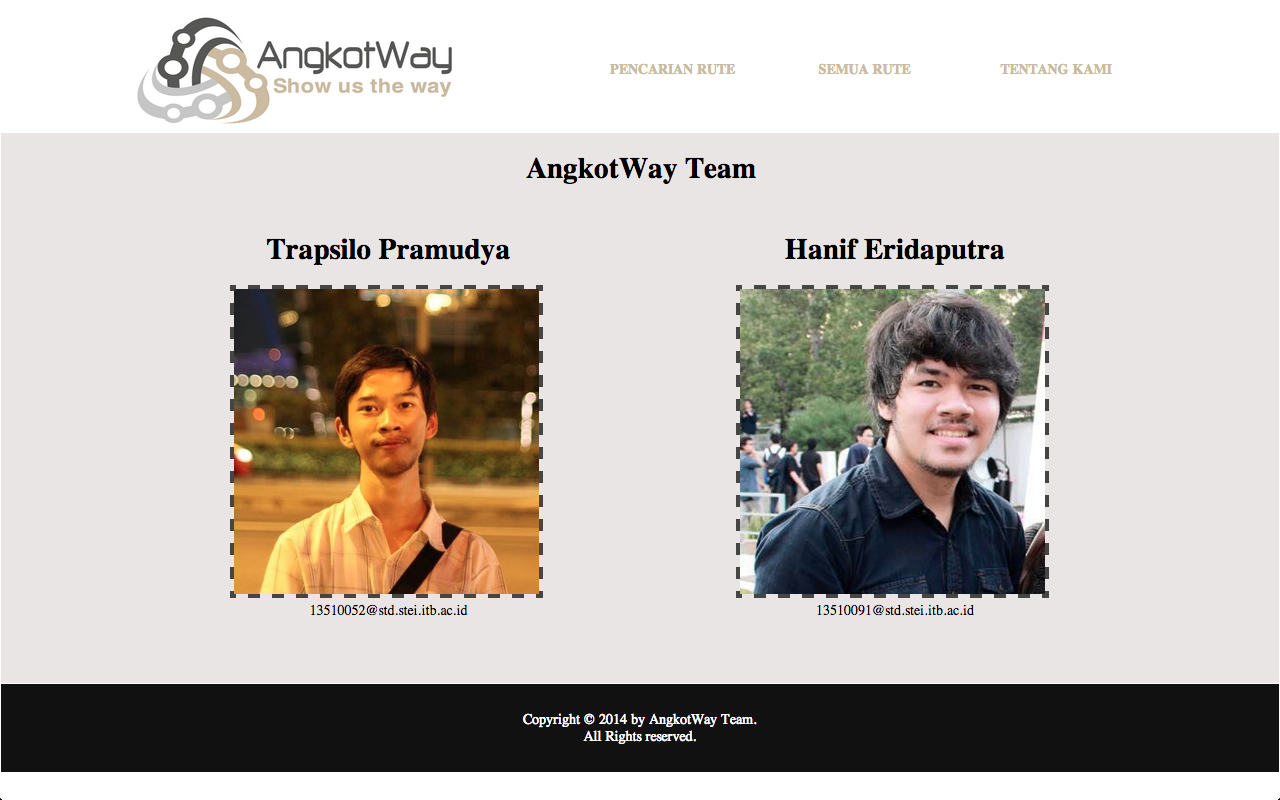
Antar muka halaman pencarian rute angkot:



Antar muka halaman semua rute:



Antar muka halaman tentang kami:



# KESIMPULAN

# REFERENSI

* http://gilangdawous.wordpress.com/2012/03/11/rute-angkot-bandung/
* http://postgis.net/docs/manual-2.1/reference.html
* https://developers.google.com/maps/documentation/javascript/3.exp/reference
* http://www.php.net/manual/en/ref.pgsql.php
* http://www.postgresql.org/docs/9.3/static/index.html

# LAMPIRAN

query.php (semua query dan fungsi dimasukkan ke dalam satu file).

|  |
| --- |
| <?php  define('TOLERANCE', 0.001);  // koneksi ke PostgreSQL  $dbconn = pg\_connect("host=localhost dbname=rute\_angkot user=postgres password=root") or die("Could not connect.");  $koordinat\_asal = $\_GET["titik\_asal"];  $koordinat\_tujuan = $\_GET["titik\_tujuan"];  $titik\_asal = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_GeomFromText('POINT($koordinat\_asal)')"), 0);  $titik\_tujuan = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_GeomFromText('POINT($koordinat\_tujuan)')"), 0);  $rute\_antrian = array();  $intersection = pg\_fetch\_all(pg\_query($dbconn, sprintf("SELECT geom FROM rute WHERE ST\_Intersects(ST\_Buffer('$titik\_asal', %f), geom)", TOLERANCE)));  foreach ($intersection as $inter) {  $inter\_geom = $inter["geom"];  $location\_point\_awal = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_LineLocatePoint('$inter\_geom', '$titik\_asal')"), 0);  $inter\_substring = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_LineSubstring('$inter\_geom', $location\_point\_awal, 1)"), 0);  $rute\_antrian[] = [$inter\_substring];  }  while (count($rute\_antrian) > 0) {  $array\_rute\_proses = array\_shift($rute\_antrian);  $rute\_proses = $array\_rute\_proses[count($array\_rute\_proses)-1];  $sudah\_sampai\_tujuan = pg\_fetch\_result(pg\_query($dbconn, sprintf("SELECT ST\_Intersects('$rute\_proses', ST\_Buffer('$titik\_tujuan', %f))", TOLERANCE)), 0);  if ($sudah\_sampai\_tujuan == 't') {  $all\_routes = array();  for ($i=0; $i < count($array\_rute\_proses); $i++) {  $bagian\_rute\_akhir = $array\_rute\_proses[$i];  if ($i == 0) {  $location\_point\_awal = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_LineLocatePoint('$bagian\_rute\_akhir', '$titik\_asal')"), 0);  } else {  $rute\_sebelumnya = $array\_rute\_proses[$i-1];  $location\_point\_awal = cari\_lokasi\_titik\_potong($rute\_sebelumnya, $bagian\_rute\_akhir, $bagian\_rute\_akhir);  }  if ($i == count($array\_rute\_proses)-1) {  $location\_point\_akhir = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_LineLocatePoint('$bagian\_rute\_akhir', '$titik\_tujuan')"), 0);  } else {  $rute\_sesudahnya = $array\_rute\_proses[$i+1];  $location\_point\_akhir = cari\_lokasi\_titik\_potong($bagian\_rute\_akhir, $rute\_sesudahnya, $bagian\_rute\_akhir);  }  $rute\_substring = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_AsGeoJSON(ST\_LineSubstring('$bagian\_rute\_akhir', $location\_point\_awal, $location\_point\_akhir))"), 0);  $decode\_routes = json\_decode($rute\_substring, true);  $decode\_routes["jurusan"] = pg\_fetch\_result(pg\_query($dbconn, "SELECT jurusan FROM rute WHERE ST\_Overlaps(geom, '$bagian\_rute\_akhir')"), 0);  $all\_routes[] = $decode\_routes;  }  $json\_routes = json\_encode($all\_routes);  echo $json\_routes;  break;  } else {  $rute\_lain\_yg\_intersect = pg\_fetch\_all(pg\_query($dbconn, "SELECT geom FROM rute WHERE ST\_Intersects('$rute\_proses', geom) and not ST\_Equals('$rute\_proses', geom)"));  foreach ($rute\_lain\_yg\_intersect as $rt) {  $rt\_geom = $rt["geom"];  $location\_point\_titik\_potong = cari\_lokasi\_titik\_potong($rute\_proses, $rt\_geom, $rt\_geom);  $inter\_substring = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_LineSubstring('$rt\_geom', $location\_point\_titik\_potong, 1)"), 0);  array\_push($array\_rute\_proses, $inter\_substring);  $rute\_antrian[] = $array\_rute\_proses;  array\_pop($array\_rute\_proses);  }  }  }  function cari\_lokasi\_titik\_potong($rute\_awal, $rute\_akhir, $acuan) {  global $dbconn;  $titik\_potong\_geom = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_Intersection('$rute\_akhir', '$rute\_awal')"), 0);  $type = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_GeometryType('$titik\_potong\_geom')"), 0);  if ($type == "ST\_GeometryCollection" || $type == "ST\_MultiPoint") {  $num\_geoms = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_NumGeometries('$titik\_potong\_geom')"), 0);  for ($i=1; $i <= $num\_geoms; $i++) {  $lokasi\_potong = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_GeometryN('$titik\_potong\_geom', $i)"), 0);  if (pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_GeometryType('$lokasi\_potong')"), 0) == "ST\_Point") {  $location\_point\_calon\_titik\_potong = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_LineLocatePoint('$acuan', '$lokasi\_potong')"), 0);  } else {  $lokasi\_potong\_awal = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_StartPoint('$lokasi\_potong')"), 0);  $location\_point\_calon\_titik\_potong = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_LineLocatePoint('$acuan', '$lokasi\_potong\_awal')"), 0);  }  if (!isset($location\_point\_titik\_potong) || (int)$location\_point\_calon\_titik\_potong < (int)$location\_point\_titik\_potong) {  $location\_point\_titik\_potong = $location\_point\_calon\_titik\_potong;  }  }  } else if ($type == "ST\_MultiLineString") {  $num\_geoms = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_NumGeometries('$titik\_potong\_geom')"), 0);  for ($i=1; $i <= $num\_geoms; $i++) {  $line = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_GeometryN('$titik\_potong\_geom', $i)"), 0);  $calon\_titik\_potong = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_StartPoint('$line')"), 0);  $location\_point\_calon\_titik\_potong = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_LineLocatePoint('$acuan', '$calon\_titik\_potong')"), 0);  if (!isset($location\_point\_titik\_potong) || (int)$location\_point\_calon\_titik\_potong < (int)$location\_point\_titik\_potong) {  $location\_point\_titik\_potong = $location\_point\_calon\_titik\_potong;  }  }  } else if ($type == "ST\_LineString") {  $titik\_potong = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_StartPoint('$titik\_potong\_geom')"), 0);  $location\_point\_titik\_potong = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_LineLocatePoint('$acuan', '$titik\_potong')"), 0);  } else if ($type == "ST\_Point") {  $location\_point\_titik\_potong = pg\_fetch\_result(pg\_query($dbconn, "SELECT ST\_LineLocatePoint('$acuan', '$titik\_potong\_geom')"), 0);  } else {  var\_dump($type);  die("Unknown intersection type.");  }  return $location\_point\_titik\_potong;  }  pg\_close($dbconn);  ?> |

peta.html (file html utama yang digunakan untuk mencari rute)

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />  <style type="text/css">  html { height: 100% }  body { height: 100%; margin: 0; padding: 0 }  #map-canvas { height: 100% }  </style>  <script type="text/javascript"  src="http://maps.googleapis.com/maps/api/js?sensor=false">  </script>  <script type="text/javascript">  var asal = null;  var tujuan = null;  var lines = [];  function getRandomColor() {  var letters = '0123456789ABCDEF'.split('');  var color = '#';  for (var i = 0; i < 6; i++ ) {  color += letters[Math.round(Math.random() \* 15)];  }  return color;  }  function emptyLines() {  for (var i = 0; i < lines.length; i++) {  lines[i].setMap(null);  };  lines = [];  }  PilihAsalTujuanControl.prototype.toggleAsal = true;    function PilihAsalTujuanControl(controlDiv, map) {  var control = this;  // Set CSS styles for the DIV containing the control  // Setting padding to 5 px will offset the control  // from the edge of the map  controlDiv.style.padding = '5px';  // Set CSS for the control border  var controlUI = document.createElement('div');  controlUI.style.backgroundColor = 'white';  controlUI.style.borderStyle = 'solid';  controlUI.style.borderWidth = '1px';  controlUI.style.cursor = 'pointer';  controlUI.style.textAlign = 'center';  controlUI.title = 'Klik untuk berubah antara memilih titik asal dan titik tujuan';  controlDiv.appendChild(controlUI);  // Set CSS for the control interior  var controlText = document.createElement('div');  controlText.style.fontFamily = 'Arial,sans-serif';  controlText.style.fontSize = '16px';  controlText.style.paddingLeft = '4px';  controlText.style.paddingRight = '4px';  controlText.innerHTML = '<b>Memilih asal</b>';  controlUI.appendChild(controlText);  google.maps.event.addDomListener(controlUI, 'click', function() {  if (!control.toggleAsal) {  control.toggleAsal = true;  controlText.innerHTML = '<b>Memilih asal</b>';  }  else {  control.toggleAsal = false;  controlText.innerHTML = '<b>Memilih tujuan</b>';  }  });  google.maps.event.addListener(map, 'click', function(e) {  if (control.toggleAsal) {  if (asal != null) {  asal.setMap(null);  asal = null;  }  asal = new google.maps.Marker({  position: e.latLng,  map: map,  animation: google.maps.Animation.DROP,  });  console.log("Asal: " + asal.getPosition().lng() + " " + asal.getPosition().lat());  } else {  if (tujuan != null) {  tujuan.setMap(null);  tujuan = null;  }  tujuan = new google.maps.Marker({  position: e.latLng,  map: map,  animation: google.maps.Animation.DROP,  });  console.log("Tujuan: " + tujuan.getPosition().lng() + " " + tujuan.getPosition().lat());  }  })  }  function CalculateRouteControl(controlDiv, map) {  // Set CSS styles for the DIV containing the control  // Setting padding to 5 px will offset the control  // from the edge of the map  controlDiv.style.padding = '5px';  // Set CSS for the control border  var controlUI = document.createElement('div');  controlUI.style.backgroundColor = 'white';  controlUI.style.borderStyle = 'solid';  controlUI.style.borderWidth = '1px';  controlUI.style.cursor = 'pointer';  controlUI.style.textAlign = 'center';  controlUI.title = 'Klik untuk memulai kalkulasi rute';  controlDiv.appendChild(controlUI);  // Set CSS for the control interior  var controlText = document.createElement('div');  controlText.style.fontFamily = 'Arial,sans-serif';  controlText.style.fontSize = '16px';  controlText.style.paddingLeft = '4px';  controlText.style.paddingRight = '4px';  controlText.innerHTML = '<b>Kalkulasi rute</b>';  controlUI.appendChild(controlText);  google.maps.event.addDomListener(controlUI, 'click', function() {  var xmlhttp = new XMLHttpRequest();  xmlhttp.onreadystatechange = function() {  if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {  console.log(xmlhttp.responseText);  if (xmlhttp.responseText.indexOf("<") != -1) {  alert("error.");  return;  }  emptyLines();  var jsonstring = eval(xmlhttp.responseText);  var jurusan = "";  for (var h = 0; h < jsonstring.length; h++) {  var lineStringCoords = [];  var GeoJSON = jsonstring[h];  for (var i = 0; i < GeoJSON.coordinates.length; i++) {  lineStringCoords.push(new google.maps.LatLng(GeoJSON.coordinates[i][1], GeoJSON.coordinates[i][0]));  };  var lineString = new google.maps.Polyline({  path: lineStringCoords,  map: map,  strokeColor: getRandomColor(),  strokeOpacity: 1.0,  strokeWeight: 2  });  lines.push(lineString);  jurusan += GeoJSON.jurusan + " -> ";  }  jurusan = jurusan.substr(0, jurusan.length-4);  alert("Jurusan yang harus dinaiki: " + jurusan);  }  };  var querystring = "query.php?titik\_asal="+asal.getPosition().lng() + "%20" + asal.getPosition().lat()  +"&titik\_tujuan="+tujuan.getPosition().lng() + "%20" + tujuan.getPosition().lat();  console.log(querystring);  xmlhttp.open("GET", querystring, true);  xmlhttp.send();  });  }  function initialize() {  var itebe = new google.maps.LatLng(-6.892578,107.610397);  var mapOptions = {  center: itebe,  zoom: 15,  streetViewControl: false,  };  var map = new google.maps.Map(document.getElementById("map-canvas"),  mapOptions);  var pilihAsalTujuanControlDiv = document.createElement('div');  var pilihAsalTujuanControl = new PilihAsalTujuanControl(pilihAsalTujuanControlDiv, map);  pilihAsalTujuanControlDiv.index = 1;  map.controls[google.maps.ControlPosition.RIGHT\_TOP].push(pilihAsalTujuanControlDiv);    var calculateRouteDiv = document.createElement('div');  var calculateRouteControl = new CalculateRouteControl(calculateRouteDiv, map);  calculateRouteDiv.index = 1;  map.controls[google.maps.ControlPosition.RIGHT\_TOP].push(calculateRouteDiv);  }  google.maps.event.addDomListener(window, 'load', initialize);  </script>  </head>  <body>  <div id="map-canvas" />  </body>  </html> |

showallroutes.html (file html yang digunakan untuk menampilkan semua rute)

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />  <style type="text/css">  html { height: 100% }  body { height: 100%; margin: 0; padding: 0 }  #map-canvas { height: 100% }  </style>  <script type="text/javascript"  src="http://maps.googleapis.com/maps/api/js?sensor=false">  </script>  <script type="text/javascript">  var asal = null;  var tujuan = null;  function getRandomColor() {  var letters = '0123456789ABCDEF'.split('');  var color = '#';  for (var i = 0; i < 6; i++ ) {  color += letters[Math.round(Math.random() \* 15)];  }  return color;  }  function initialize() {  var bandung = new google.maps.LatLng(-6.908704023900703,107.61880874633789);  var mapOptions = {  center: bandung,  zoom: 13,  streetViewControl: false,  };  var map = new google.maps.Map(document.getElementById("map-canvas"),  mapOptions);  // --- kode dibawah untuk menampilkan hasil query postgis berbentuk GeoJSON ---  var xmlhttp = new XMLHttpRequest();  var jsonstring = "";  xmlhttp.onreadystatechange = function() {  if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {  jsonstring = eval(xmlhttp.responseText);  for (var h = 0; h < jsonstring.length; h++) {  var lineStringCoords = [];  var GeoJSON = jsonstring[h];  for (var i = 0; i < GeoJSON.coordinates.length; i++) {  lineStringCoords.push(new google.maps.LatLng(GeoJSON.coordinates[i][1], GeoJSON.coordinates[i][0]));  };  var lineString = new google.maps.Polyline({  path: lineStringCoords,  map: map,  strokeColor: getRandomColor(),  strokeOpacity: 1.0,  strokeWeight: 2  });  }  }  }  xmlhttp.open("GET", "queryallroutes.php", true);  xmlhttp.send();  }  google.maps.event.addDomListener(window, 'load', initialize);  </script>  </head>  <body>  <div id="map-canvas" />  </body>  </html> |