Pilot application deployment –

Pilot 06 - Evacuation (Routing) Service

Author(s)/Organisation(s):
Christos Charmatzis; Anestis Trypitsidis
Work package / Task:
WP7 – Pilot Applications T7.3 - Scenario Application deployment
References:

Short description:
This document provides information about the development of the routing service within the context of eENVplus project –Pilot 6
Keywords:
Pilots, application, workflow, Routing Service, PgRouting, Dijkstra,

Table of Contents

1 Overwiew ... 3
1.1 Routing System Architecture ... 4
1.2 Barriers technologies ... 5
1.3 Requirements ... 5
1.4 Client ... 5
1.5 Storage ... 5
1.6 Server ... 5
2 Quick Start ... 6
2.1 DB and Application installation ... 6
2.2 Data Configuration ... 6
3 Development of the routing service ... 8
3.1 PgRouting ... 8
3.2 Dataset preparation ... 8
3.3 Routing ... 8
3.4 Routing with Barriers ... 8
3.5 Visualization of the results ... 9
4 Developing the Barriers Service ... 12
4.1 Get Study Areas Query ... 12
4.2 Fire Simulation Maps List ... 13
4.3 Forecast Time Map ... 13
4.4 Convex Hull Query ... 13
5 Examples ... 16

Table of Figures

Figure 1: Routing with sublines 3
Figure 2: Routing with barriers and sublines 4
Figure 3: Routing system architecture 4
Figure 4: Barriers technologies 5
Figure 5: INSPIRE Road Network 6
Figure 6: DB Manager 7
Figure 7: Routing service main interface 16
Figure 8: Routing service parameters 16
Figure 9: Routing service results 17

List of tables

Table 1: Parameters routingwithsublines 10
Table 1: Parameters routingwithBarriersWithSubLines 10

1 Overwiew

The eENVplus routing service is compiled from two OGC WMS standard spatial services providing the quickest route from INSPIRE road network.

The first WMS Service is called **routingWithSubLines** and visualizes the quickest route from start point A to end point B. A simple example is reference in the above link:
http://epsilonportal.cloudapp.net/geoserver/EPSILON/wms?service=WMS&version=1.1.0&request=GetMap&layers=EPSILON:routingWithSubLines&styles=&bbox=411404.975963788,4496783.19968117,413708.412383858,4501296.11432828&width=261&height=512&srs=EPSG:2100&format=application/openlayers

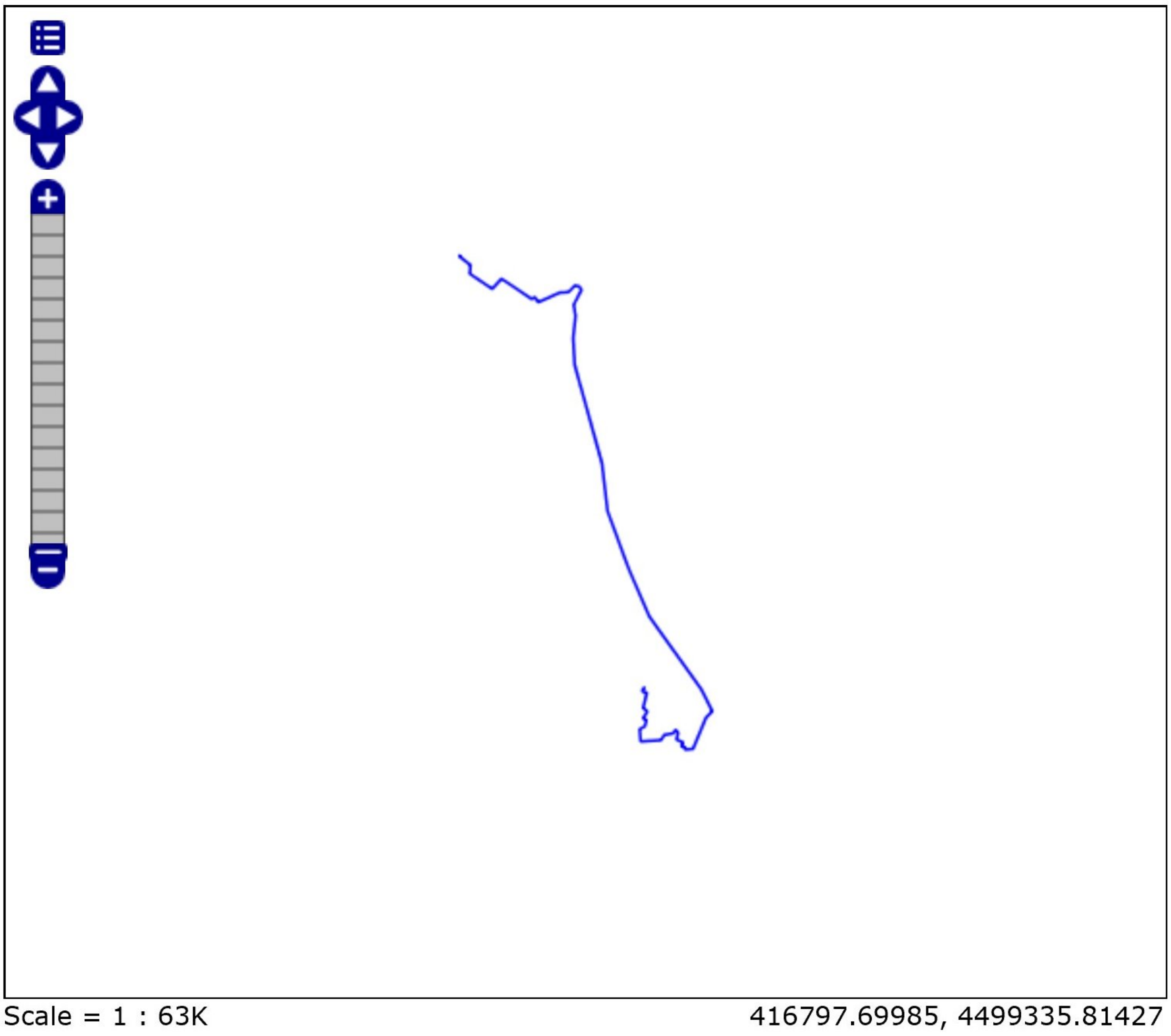


Figure 1: Routing with sublines

The second WMS service is called **routingwithBarriersWithSubLines** and visualizes the route from start point A to end point B without passing from an area which is formed from a polygon. In the above link there is an example.
http://epsilonportal.cloudapp.net/geoserver/EPSILON/wms?service=WMS&version=1.1.0&request=GetMap&layers=EPSILON:routingwithBarriersWithSubLines&styles=&bbox=411404.975963788,4496783.19968117,413708.412383858,4501287.39972855&width=261&height=512&srs=EPSG:2100&format=application/openlayers

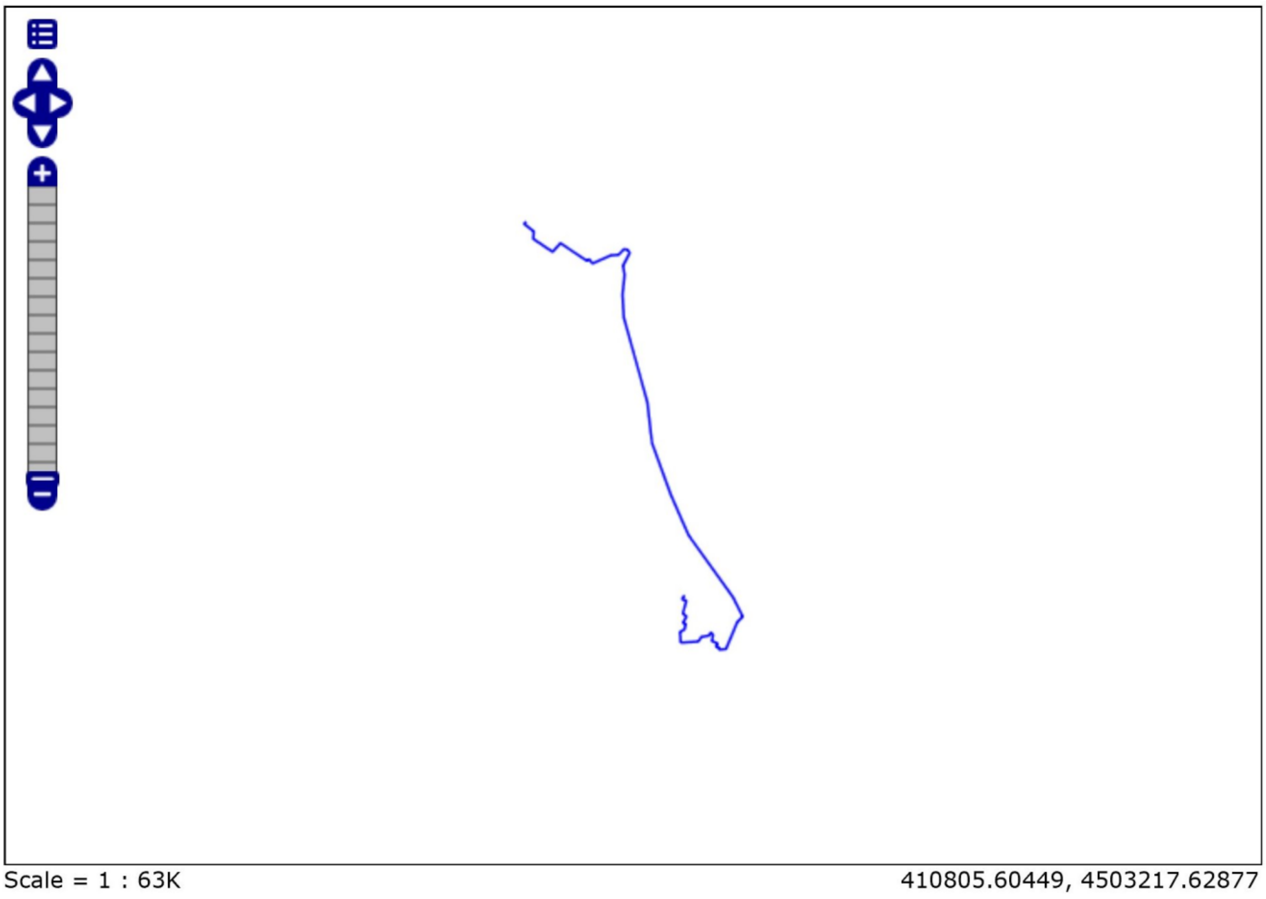


Figure 2: Routing with barriers and sublines

1.1 Routing System Architecture

The basic eENVplus Routing Service is based in a 3-Tier Architecture: PostgreSQL for the Database; GeoServer as the application; GeoEXT & OpenLayers as the web-client

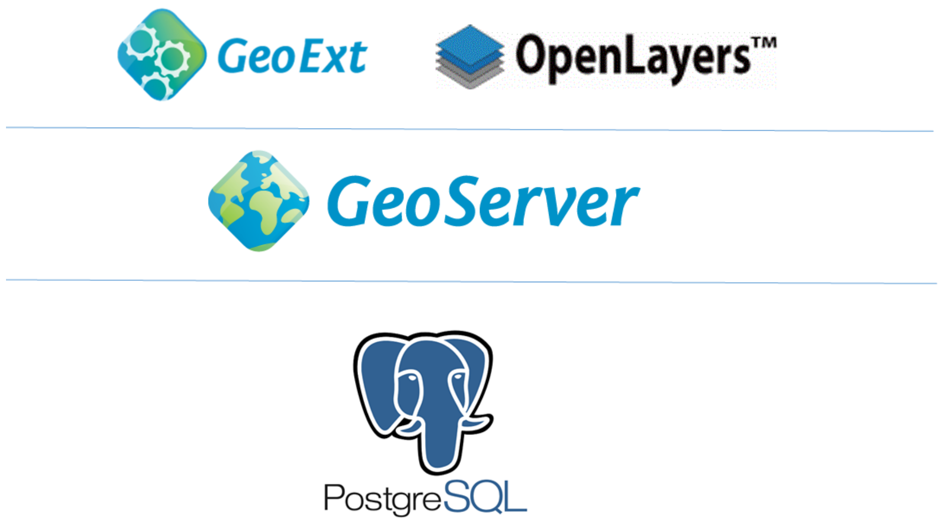


Figure 3: Routing system architecture

1.2 Barriers technologies

To create the polygon barrier it was developed an ASP.NET MVC Controller that manipulates data from a SQL EXPRESS LocalDB and returns JSON Data.

Figure 4: Barriers technologies

1.3 Requirements

For the development of the Routing Service the following technologies used:

1. DB PostgreSQL 9.3 + PostGIS 2.0
2. Apache Tomcat 8.0
3. GeoServer 2.6.0
4. ASP.NET 4.5

1.4 Client

To construct the http queries and visualize the WMS, needs to apply a map client. In our case OpenLayers 2.13 is used to show the images that served from the GeoServer.

1.5 Storage

The eENVplus service uses a SQL EXPRESS LocalDB for storing all the results that saved from the Fire Simulation Service. In this way, the Routing form can retrieve data and process them through JavaScript to form the HTTP Get query.

The road network is stored in a table in the PostgreSQL.

1.6 Server

eENVplus GeoServer processes the query through the WMS parameters and executes a SQL function where the ways table has all the data from the road network.

2 Quick Start

2.1 DB and Application installation

- First, it must be installed the PostgreSQL + PostGis in a Server machine.
- Second, it must be installed a GeoServer in a Server machine that can be connected through the PostgreSQL's client to the previous DB.

2.2 Data Configuration

In order to achieve the quickest way to import spatial data inside the PostgreSQL, we used QGIS. In our case and as it is depicted in the following image, an INSPIRE compliant Road Network is imported to the DB. In this regard, the following two layers are necessary for the routing service:

- RoadLinks
- SpeedLimit

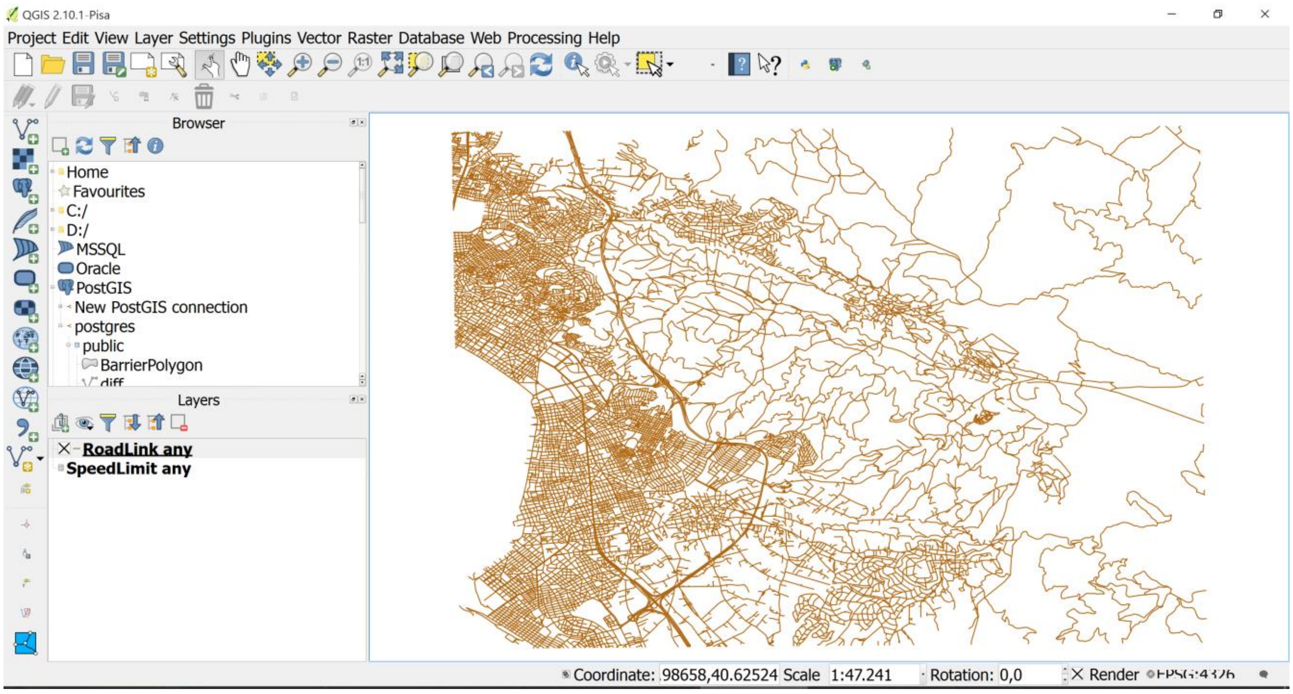
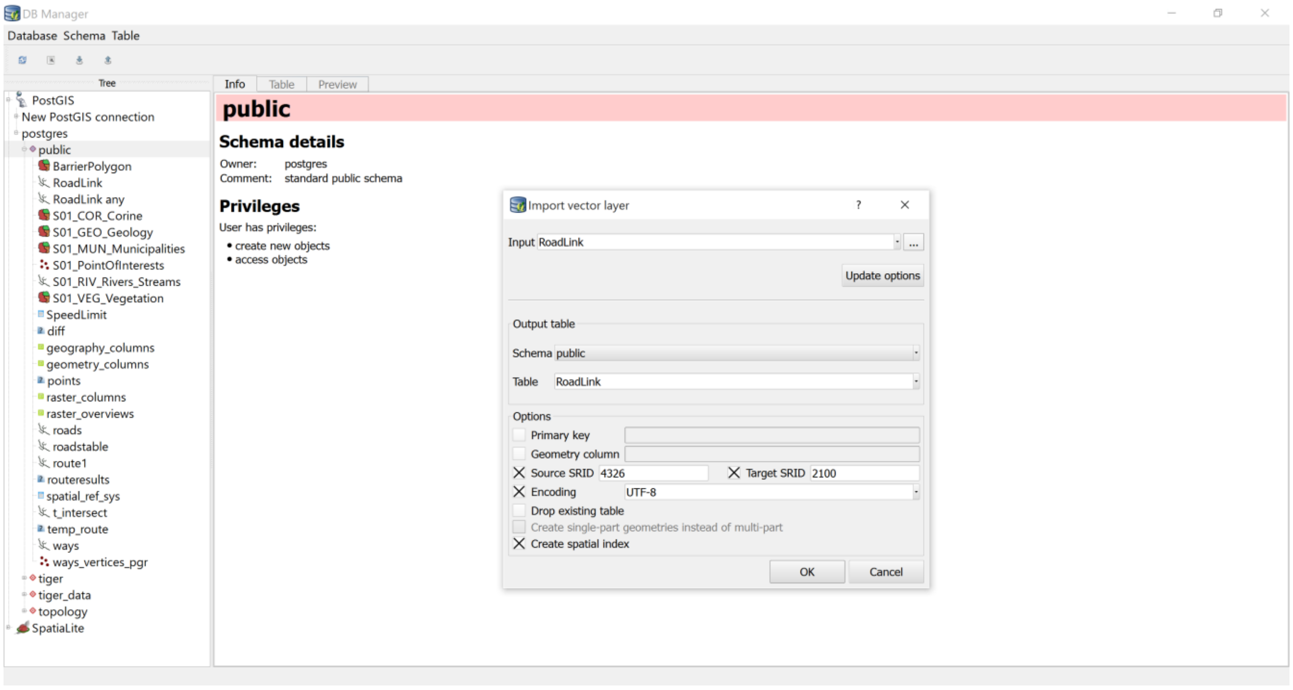


Figure 5: INSPIRE Road Network

To import them we use the DB Manager extension to import from the INSPIRE compliment GML road network to tables RoadLink and SpeedLimit. The configurations are:

- output schema is the "public",
- table same name as the Layer,
- Source SRID:4326 (Only for the RoadLink Layer)
- Target SRID 2100 (Only for the RoadLink Layer),
- Encoding: UTF-8,
- Create Spatial Index (Only for the RoadLink Layer)



PostGIS
New PostGIS connection
postgres
public
BarrierPolygon
RoadLink
RoadLink any
S01_COR_Corine
S01_GEO_Geology
S01_MUN_Municipalities
S01_PointOfInterests
S01_RIV_Rivers_Streams
S01_VEG_Vegetation
SpeedLimit
diff
geography_columns
geometry_columns
points
raster_columns
raster_overviews
roads
roadstable
route1
routeresults
spatial_ref_sys
t_intersect
temp_route
ways
ways_vertices_pgr
tiger
tiger_data
topology
SpatiaLite

public

Schema details

Owner: postgres
Comment: standard public schema

Privileges

User has privileges:
- create new objects
- access objects

Input RoadLink

Output table
Schema public
Table RoadLink

Options
Primary key
Geometry column
[✓ Source SRID] 4326 [✓ Target SRID] 2100
[✓ Encoding] UTF-8
Drop existing table
Create single-part geometries instead of multi-part
[✓ Create spatial index]

Figure 6: DB Manager

3 Development of the routing service

The routing service is divided into two sub-applications:

- Routing, where the shortest route of two points is visualized
- Routing with barriers, where again the shortest route of two points is visualized but with an additional parameter (the route does not intersect specific/burned areas)

3.1 PgRouting

"Shortest Path Dijkstra" is used for the calculation of the shortest route, from the pgrouting package algorithms which are included in PostGIS.

3.2 Dataset preparation

First we create an new table called "ways" which is an inner join of the tables "RoadLink" and "SpeadLimit". Combined these two tables we have all the necessary information for the "Djikstra" algorithm that needs to have the road network table. Then with the `pgr_createTopology` function which actually builds a new table "ways_verticles_pgr" with source and target information (topology and attributes).

3.3 Routing

A wrapper function was developed for the calculation of the shortest path between two points. Specifically, we use **`pgr_fromAtoBWithSubLines`** function within Postgres schema. The parameters of this function are the following: X1 (double precision); Y1 (double precision); X2 (double precision); Y2 (double precision). "X", "Y" are the start and end points of the route. The coordinate system we are using is EGSA '87 (EPSG: 2100 GGRS87).

Basically, the above mentioned function is using the `pgr_dijkstra()` function. **`pgr_dijkstra`** function calculates the path between two nodes of the road network.

In the case where the above mentioned start and end points are not nodes of the road network, the following logic was used:

- The shortest parts of the road network are taken into consideration
- Using `ST_ClosestPoint()` function the closest points are calculated over the previous parts of the road network
- The sub-part of the road network is calculated from the nearest node by using the following two functions `ST_Line_Substring()` & `ST_LineLocatePoint()`
- Then we are using those nodes as an input to the `pgr_dijkstra()` function. The output of this calculation is the shortest path.
- At the end if the shortest path contains or not the sub-part of the road network then this sub-part is removed or is added respectively. For the last step we are using the following functions: `ST_DWithin(),ST_Difference(),ST_Union()`.

NOTE: The output of this function is a multi LineString geometry

3.4 Routing with Barriers

A wrapper function was developed for the calculation of the shortest path between two points. Specifically, we use **`pgr_fromAtoBwithBarriersWithSubLines`** function within Postgres schema. The parameters of this function are the following: X1 (double precision); Y1 (double precision); X2 (double precision); Y2 (double precision); Barriers (text) – which is the polygon (burned area) in WKT (well-known text) format.

The steps for the calculation are the following:

- The shortest parts of the road network are calculated from start-end points.
- Using `ST_ClosestPoint()` function the closest points are calculated over the previous parts of the road network
- The sub-part of the road network is calculated from the nearest node by using the following two functions `ST_Line_Substring()` & `ST_LineLocatePoint()`
- From the polygon (burned area) parameter the geometry is inserted to a table with the `ST_GeometryFromText()` function
- Then, we select the road network which is not contain the polygon geometry, using `ST_Intersects()` function & `left join`
- At the end if the shortest path contains or not the sub-part of the road network then this sub-part is removed or is added respectively. For the last step we are using the following functions: `ST_DWithin(),ST_Difference(),ST_Union()`.

NOTE: The output of this function is a multi LineString geometry

3.5 Visualization of the results

For the visualization of routing results we are using GeoServer and create SQL Views, where it can be used via SQL functions from the Store.

In order to implement this, two new layers were created:

- eENVYPlus:routingWithSubLines, which is using **`pgr_fromAtoBWithSubLines`** algorithm (http://epsilonportal.cloudapp.net/geoserver/eENVYPlus/wms?service=WMS&version=1.1.0&request=GetMap&layers=eENVYPlus:routingWithSubLines&styles=&bbox=410882.8549,4497022.3775,413099.168246801,4501296.11433119&width=265&height=512&srs=EPSG:2100&format=application/openlayers)
- eENVYPlus:routingwithBarriersWithSubLines, which is using **`pgr_fromAtoBwithBarriersWithSubLines`** algorithm. (http://epsilonportal.cloudapp.net/geoserver/eENVYPlus/wms?service=WMS&version=1.1.0&request=GetMap&layers=eENVYPlus:routingwithBarriersWithSubLines&styles=&bbox=410882.8549,4497022.3775,413099.168246801,4501296.11433119&width=265&height=512&srs=EPSG:2100&format=application/openlayers)

The visualization is being made via GeoExt & OpenLayers, though the results of the routing service are visualized as a WMS Layer

The following parameters are used for the routingWithSubLines layer:

LAYERS	**eENVYPlus:routingWithSubLines**
FORMAT	image/png
TRANSPARENT	True
TILED	True
VIEWPARAMS	x1:414792.16342970496; y1:4497772.444403767; x2:416177.13051993295;

	y2:4497706.444372113
SERVICE	WMS
VERSION	1.1.1
REQUEST	GetMap
STYLES	
SRS	EPSG:900913
BBOX	2563392.1802148,4958011.4019995,2564615.1726672,4959234.3944519
WIDTH	256
HEIGHT	256

Table 1: Parameters routingwithsublines

NOTE: VIEWPARAMS field is using EGSA ’87 (EPSG: 2100 GGRS87) as the coordinate referenced system for the start-end points.

The following parameters are used for the routingwithBarriersWithSubLines layer:

LAYERS	**eENVYPlus:routingwithBarriersWithSubLines**
FORMAT	image/png
TRANSPARENT	true
TILED	true
VIEWPARAMS	x1:414759.98618824413; y1:4497691.742116936; x2:416133.3631826089; y2:4497717.025092335; barriers:POLYGON((415059.99999982566 4497919.999999897\,415059.9999998258 4497959.9999998985\,415159.9999998264 4498339.999999899\,415179.99999982776 4498379.9999998985\,415479.9999998313 4498679.9999998985\,415519.9999998318 4498699.999999898\,415559.99999983236 4498699.999999899\,415939.9999998374 4498639.999999899\,415899.9999998368 4498119.999999902\,415739.99999983515 4497939.999999899\,415559.99999983224 4497759.999999901\,415519.99999983143 4497739.999999898\,415379.9999998305 4497759.999999899\,415299.99999982887 4497779.999999898\,415099.9999998267 4497879.9999998985\,415059.99999982566 4497919.999999897))
SERVICE	WMS
VERSION	1.1.1
REQUEST	GetMap
STYLES	
SRS	EPSG:900913
BBOX	2558500.2104053,4955565.4170947,2559723.2028577,4956788.4095471
WIDTH	256
HEIGHT	256

Table 2: Parameters routingwithBarriersWithSubLines

NOTE: VIEWPARAMS field is using EGSA '87 (EPSG: 2100 GGRS87) as the coordinate referenced system for the start-end points and the same goes for the polygon in WKT format.

4 Developing the Barriers Service

For the Routing Service with the area avoiding, before the final HTTP GET query in the GeoServer, there are steps that are needed to compile right the query. These steps are GET queries in an ASP.NET Controller and the responses are json files that are bind through ExtJS object models to retrieve the data.

4.1 Get Study Areas Query

With above query we get all the study areas stored in our database http://epsilonportal.cloudapp.net/FireAccessTimeSimulation/Geoprocessing/GetStudyAreas , where the response is a json file which includes all the areas add some features that are needed to process and visualize the result. An example of the response is presented adove

```
[
   {
      "Id":6,
      "Area":"001",
      "AreaName":"Σεϊχ Σου",
      "xllcorner":409900,
      "yllcorner":4491500,
      "xUpperllcorner":422800,
      "yUpperllcorner":4503200,
      "ncols":645,
      "nrows":585
   },
   {
      "Id":7,
      "Area":"002",
      "AreaName":"Στενή Δίρφυος",
      "xllcorner":485500,
      "yllcorner":4267900,
      "xUpperllcorner":492000,
      "yUpperllcorner":4274400,
      "ncols":325,
      "nrows":325
   },
   {
      "Id":8,
      "Area":"003",
      "AreaName":"Φολόη",
      "xllcorner":295436,
      "yllcorner":4177650,
      "xUpperllcorner":304436,
      "yUpperllcorner":4185650,
      "ncols":450,
      "nrows":400
   },
   {
      "Id":9,
      "Area":"004",
      "AreaName":"Πάρνηθα",
      "xllcorner":466850,
      "yllcorner":4216709,
      "xUpperllcorner":486050,
      "yUpperllcorner":4233109,
      "ncols":960,
      "nrows":820
   }
]
```

4.2 Fire Simulation Maps List

From the selected area when it is necessary to retrieve the Fire Simulations models that have been save in the DB. The query link is the above with the selected area the area of «Σεϊχ Σου», http://epsilonportal.cloudapp.net/FireAccessTimeSimulation/Geoprocessing/FireSimulationMapsList?area=001 . The response is a json file with an array of the simulation

```
[
  {
    "Id":1,
    "Name":"001_20150731_000_000_FAT.asc "
  }
]
```

4.3 Forecast Time Map

Then from the Id of the Simulation we must retrieve data of the time classification of the Fire Simulation. That is also a query http://epsilonportal.cloudapp.net/FireAccessTimeSimulation/Geoprocessing/ForecastTimeMap?IdMap=1 . The result is a json file with the Id and the Simulation Time

```
[
  {
    "Id":1,
    "SimulationTime":0
  },
  {
    "Id":1,
    "SimulationTime":30
  },
  {
    "Id":1,
    "SimulationTime":60
  },
  {
    "Id":1,
    "SimulationTime":90
  },
  {
    "Id":1,
    "SimulationTime":120
  },
  {
    "Id":1,
    "SimulationTime":150
  },
  {
    "Id":1,
    "SimulationTime":180
  }
]
```

4.4 Convex Hull Query

Finally, the query that returns points of a Convex Hull polygon of a certain Fire Simulation Model Id and the time simulation. The query is the above http://epsilonportal.cloudapp.net/FireAccessTimeSimulation/Geoprocessing/ConvexHull?Id=1&Forecast=180. The result is a json file of points with the X, Y coordinates in the Greek Grid Geographic System (EPSG: 2100)

```
[
  {
    "X":414540,
    "Y":4498280
  },
  {
    "X":414540,
    "Y":4498340
  },
  {
    "X":414580,
    "Y":4498580
  },
  {
    "X":414640,
    "Y":4498760
  },
  {
    "X":414660,
    "Y":4498800
  },
  {
    "X":414700,
    "Y":4498860
  },
  {
    "X":414760,
    "Y":4498900
  },
  {
    "X":414840,
    "Y":4498940
  },
  {
    "X":414860,
    "Y":4498940
  },
  {
    "X":415120,
    "Y":4498760
  },
  {
    "X":415140,
    "Y":4498700
  },
  {
    "X":415140,
    "Y":4498580
  },
  {
    "X":415000,
    "Y":4498340
  },
  {
    "X":414960,
    "Y":4498300
  },
  {
    "X":414900,
    "Y":4498260
  },
  {
    "X":414760,
    "Y":4498200
  },
  {
    "X":414620,
    "Y":4498200
  }
]
```

5 Examples

The following steps must be followed in order a user/stakeholder to use properly the routing service:

I. User navigate to http://epsilonportal.cloudapp.net/portal/
II. User click tab “Forest Fire Management”
III. User click on «Evacuation-Routing Service» button and the following window appears

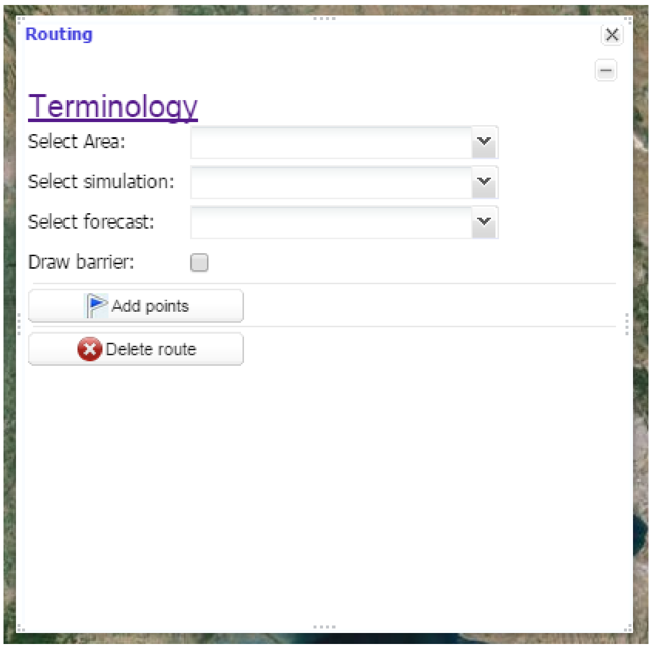
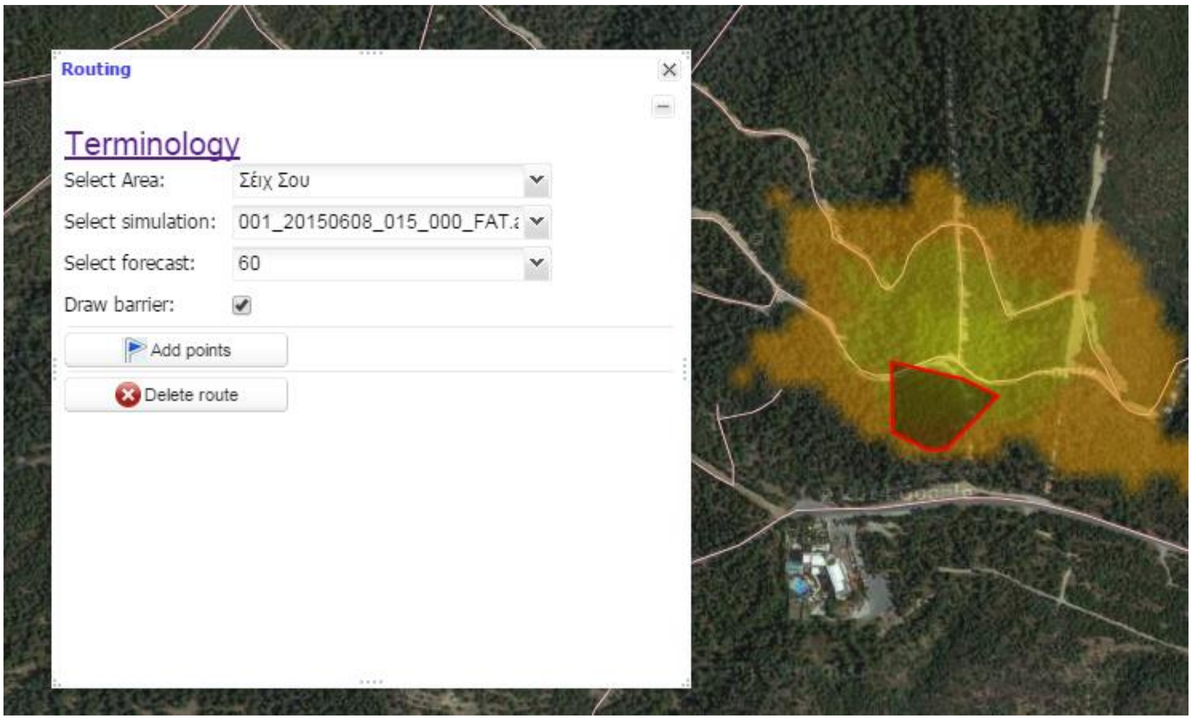


Figure 7: Routing service main interface

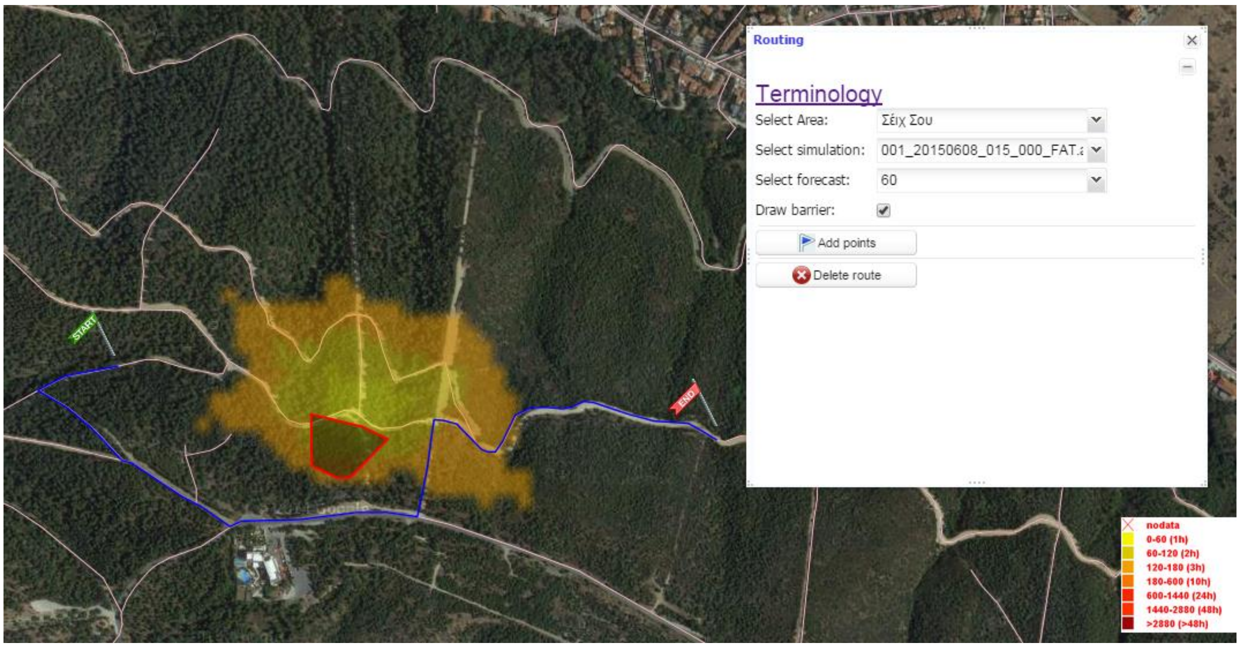
IV. User selects the area (Seih-Sou), the specific simulation that he/she made previously, and then the forecast – which is actually the isochrone line. **NOTE:** when the user checks draw barrier the corresponded isochrone line is presented



Select Area: Σέιχ Σου
Select simulation: 001_20150608_015_000_FAT.
Select forecast: 60
Draw barrier: ✓

Figure 8: Routing service parameters

V. User select “Add Point” the user can select the start and the end point



Terminology

Select Area: Σέιχ Σου

Select simulation: 001_20150608_015_000_FAT.

Select forecast: 60

Draw barrier: ✓

✗	nodata
🟨	0-60 (1h)
🟨	60-120 (2h)
🟧	120-180 (3h)
🟧	180-600 (10h)
🟥	600-1440 (24h)
🟥	1440-2880 (48h)
🟥	>2880 (>48h)

Figure 9: Routing service results

VI. User with the button “Delete route” can delete his route.

NOTE: The routing service is applicable also without selecting any simulation. In this case the routing service will give back the shortest route