Escola Universitària
Politècnica de Mataró

Enginyeria Tècnica en Informàtica de Gestió

YouRoutes.com
Aplicació amb Google Maps

Domènec Corbella Boada
Joan Jou Majó

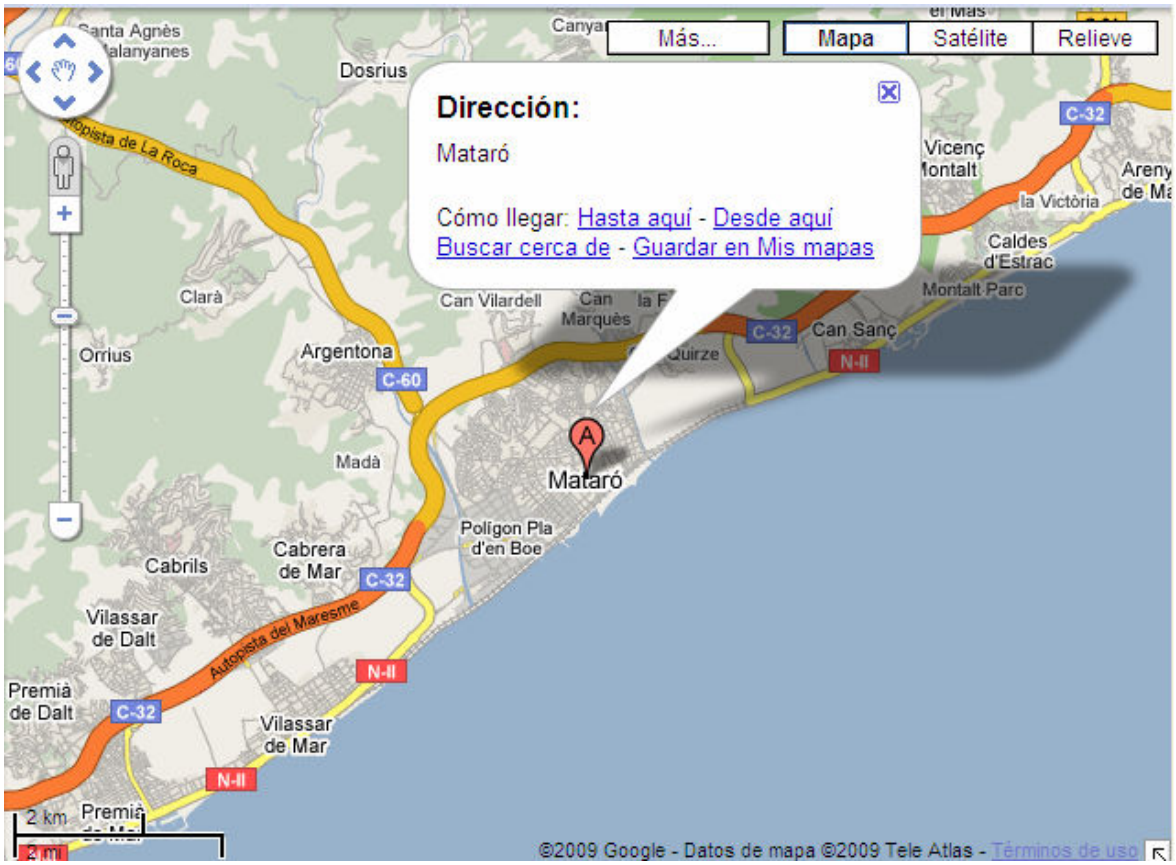
TARDOR 2008

ÍNDEX

1.- Introducció

Google maps és un servidor d'aplicacions de mapes a la web. Ens ofereix imatges en vectorial, en satèl·lit i més recentment en forma de tour virtual [1] (Street View). Al mateix temps tenim l'opció de fer cerques de: carrers, ciutats, pobles i de buscar rutes entre diferents ubicacions. Google ofereix aquest servei gratuït des del 6 d'octubre de 2005.

El llenguatge principal de les funcions que utilitza Google Maps és el Javascript [2], que s'executa a la part del client. Per tant el client fa les peticions al servidor i aquestes es visualitzen a la màquina del client. Per exemple: Si un usuari fa una cerca d'una població aquesta surt indicada amb un pin; es tracta d'una imatge en format PNG [3] que és transparent i que es sobreposa a la del mapa.



Per tal d'aconseguir una sincronia amb el servidor Google va aplicar l'ús de la tecnologia AJAX [4].

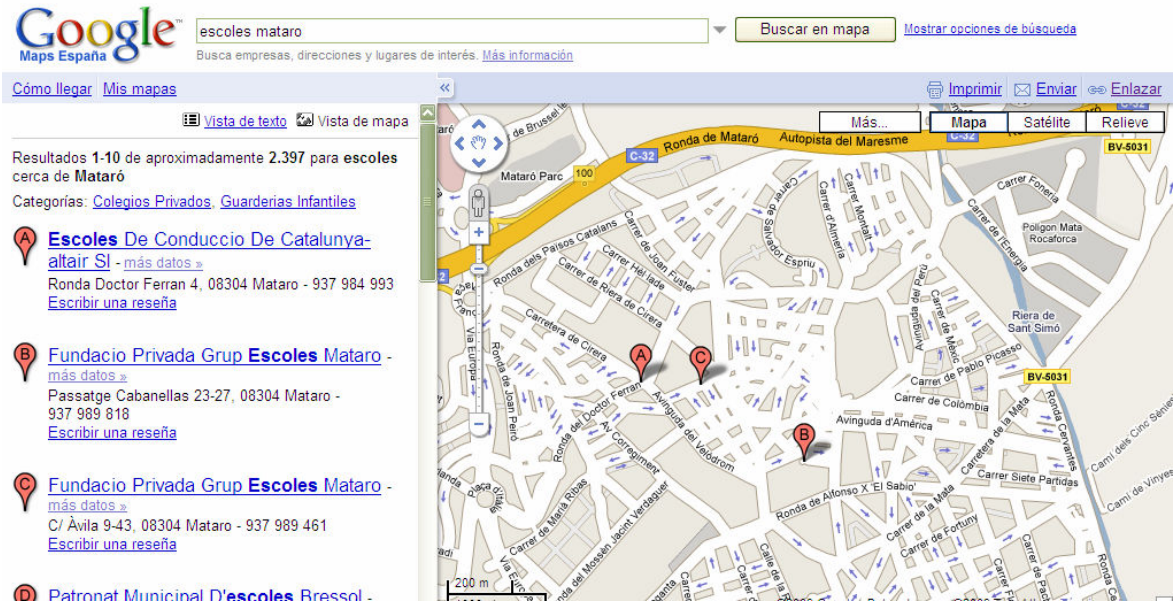
1.1. Característiques bàsiques

Google maps ofereix les següents possibilitats:

1. El control sobre el mapa: translació en totes les direccions i zoom. És interessant remarcar que el control del mapa es pot realitzar tant amb el ratolí com amb les fletxes del teclat i els símbols "+", "-" en el cas del zoom.

2. La cerca: els usuaris poden introduir el nom d'un carrer, una població o la intersecció, gràcies a l'opció de google local podem restringir els resultats de la cerca a una zona. Últimament Google Maps també permet la cerca de negocis, empreses, institucions, serveis, etc..similar al concepte de PDI (Punts d'interès) que tenen els GPS tipus TOMTOM o similar.

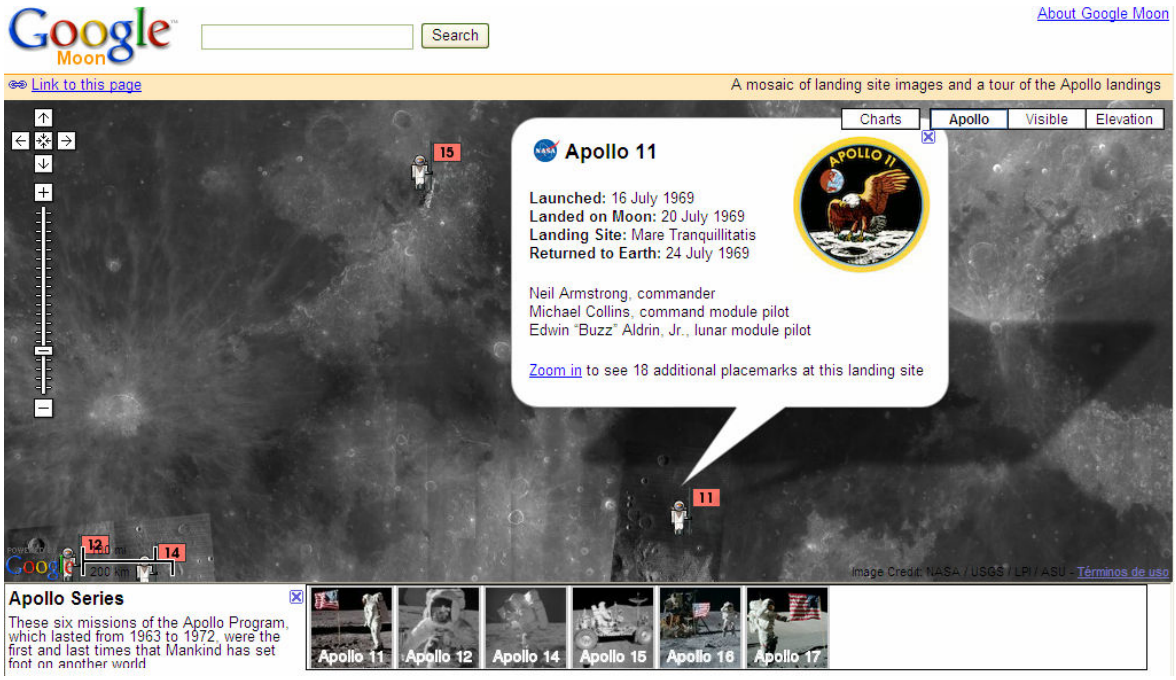
   Per tant també es preveu una clara evolució de Google Maps cap a una versió digital de les pàgines grogues amb l'afegit que ens ubicarà en el mapa el concepte que busquem més proper a la nostra ubicació i a més a més ens mostrarà com arribar-hi.



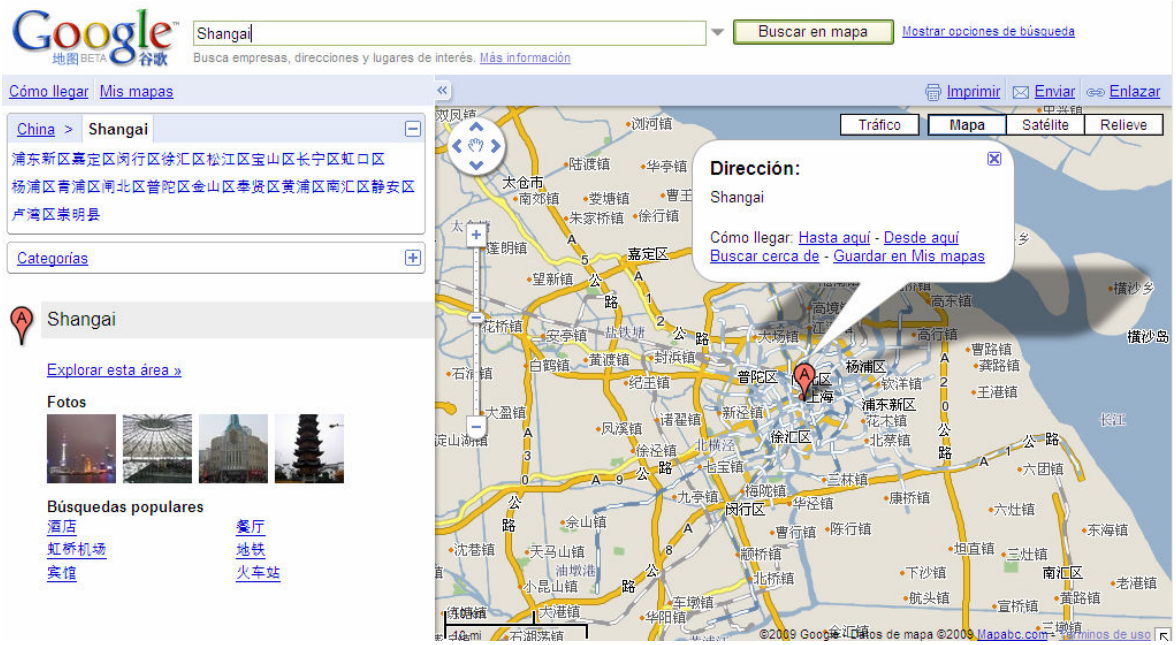
3. Com a resultat de cerca Google Maps també ofereix: l'enllaç de l'entitat que s'ha buscat, de forma que ens permet ampliar la informació d'allò que busquem.

4. Ride finder: és un experiment que es troba en fase de desenvolupament a les ciutats d'Atlanta, San Francisco i Washington consisteix en una localització via GPS dels taxis de la ciutat, aquests es representen en forma de pin dins el mapa del Google Maps.

   Posteriorment Google Maps ha aprofitat la llibreria per desenvolupar diferents variants com per exemple: google moon, google mars, google bendi (la versió de Google Maps per la Xina) i google earth que és un software que permet l'ús més personalitzat i en local, sense tenir la necessitat d'estar connectat a Internet.



Exemple de google moon.



Exemple de la versió Xinesa de Google Maps.

5. Com a última novetat Google Maps ens ofereix l'API per tal que puguem treballar amb ella des d'Adobe Flash [5] directament. Això obre un nou món de possibilitats amb el Flash que interacciona i s'entén perfectament amb Javascript ja que el llenguatge de programació en Flash és l'actionScript [6].

1.2. Possibilitats d'utilització de Google Maps

L'API de Google Maps (el codi font és obert; lliure i accessible per tothom que ho vulgui) ha permès gran varietat d'aplicacions que exploten diferents variants del codi i personalitzacions. Un bon exemple d'elles les podem trobar a la següent adreça: ref. [7]

Entre elles hi ha una clara tendència a l'explotació de terminals mòbils GPS vinculats a l'API i, aprofitar el posicionament i seguiment.

Aquesta tendència bé donada per l'evolució de la telefonia mòbil, cada cop més els telèfons van enfocats a convertir-se en mini ordinadors de butxaca, incorporen noves

funcionalitats i una d'aquestes és l'estrella; el GPS està fent furor i obre un nou món d'aplicacions que explotin aquesta tecnologia, pot tenir múltiples utilitats com ara: el seguiment de flotes de transport de mercaderies, seria una forma econòmica de tenir localitzat en tot moment el transportista, per poder fer planificacions i modificacions de ruta. Una altra possibilitat seria la de realitzar seguiment de persones i tenir-les controlades en tot moment, podria ésser útil per persones amb malalties mentals, adolescents o inclús podria tenir aplicacions en l'espionatge o per controlar persones perilloses.

La localització del GPS en l'API del Google Maps també seria d'utilitat en aplicacions esportives; podria servir com a mètode d'entrenament per controlar els recorreguts i velocitats mitjanes i/o fer les representacions automàtiques al mapa en competicions esportives com ara el París – Dakar o competicions de vela; el fet que donaria la possibilitat de veure l'evolució de la carrera a temps real, compliment de la ruta etc...(Aquestes tipus d'idees a nivell esportiu guanyaran força en el moment en que s'integri Internet a les televisions de casa).

Segur que a mesura que l'accés a Internet sigui més accessible des de la telefonia mòbil sorgiran aplicacions més inversemblants; del tipus "busca singles" o d'altres per l'estil.

El funcionament bàsic per la localització consistiria en una aplicació que fos capaç d'adquirir les coordenades actuals del GPS i enviar-les a un servidor d'aplicacions via GPRS en format "txt" amb un interval determinat, de totes maneres si es disposés de tarifa plana es podrien enviar dades en l'interval que interessés per segons o minuts.

L'aplicació web en Google Maps hauria d'anar fent lectures de l'arxiu text i plasmar les posicions sobre el mapa i unir els punts mitjançant línies. El resultat seria un mapa de línies on es podria observar el recorregut de les últimes x hores fins el moment actual. Es podria utilitzar diferents colors per diferenciar la temporalitat dels desplaçaments.

2.- Objectius

Els objectius d'aquest projecte són:

1.- Realitzar un estudi teòric - pràctic de les possibilitats que tenim a l'hora de desenvolupar una aplicació que combini visual web JSF i Google Maps.

Per tant aprofundir en un bon coneixement de l'API de Google Maps estudiar i testejar quines possibilitats actuals tenim per crear aplicacions amb Java i la integració de Google Maps. Al mateix temps veure quina és la forma de treballar amb visual web JSF; desenvolupar integrament el disseny, maquetació i programació únicament amb l'eina Netbeans.

Visual web JSF, és tracta d'un nou marc de treball en Java que ens permet generar pàgines web d'una forma visual; de manera que es pot desenvolupar una pàgina a l'estil "drag and drop" i editar les característiques dels components des de la pestanya propietats de l'editor, que inclou a banda de cridar funcions en Javascript, també ens permet editar les CSS (fulls d'estil) de la pàgina.

2.- Desenvolupar una aplicació que demostri que la simbiosi entre visual web JSF i Google Maps és possible. El software s'ha de desenvolupar integrament utilitzant l'eina Netbeans. L'usuari s'haurà de poder registrar i un cop registrat se li ha de permetre modificar les seves dades o donar-se de baixa. Un cop l'usuari hagi ingressat a la web podrà veure un mapa de Google Maps i realitzar cerques dins d'aquest.

Per tal de portar a terme la integració de Google Maps dins el visual JSF s'haurà d'escollir la tecnologia més pertinent de les estudiades.

3.- Estudi de les característiques de l'API Google Maps

3.1. Javascript, llibreria Google Maps

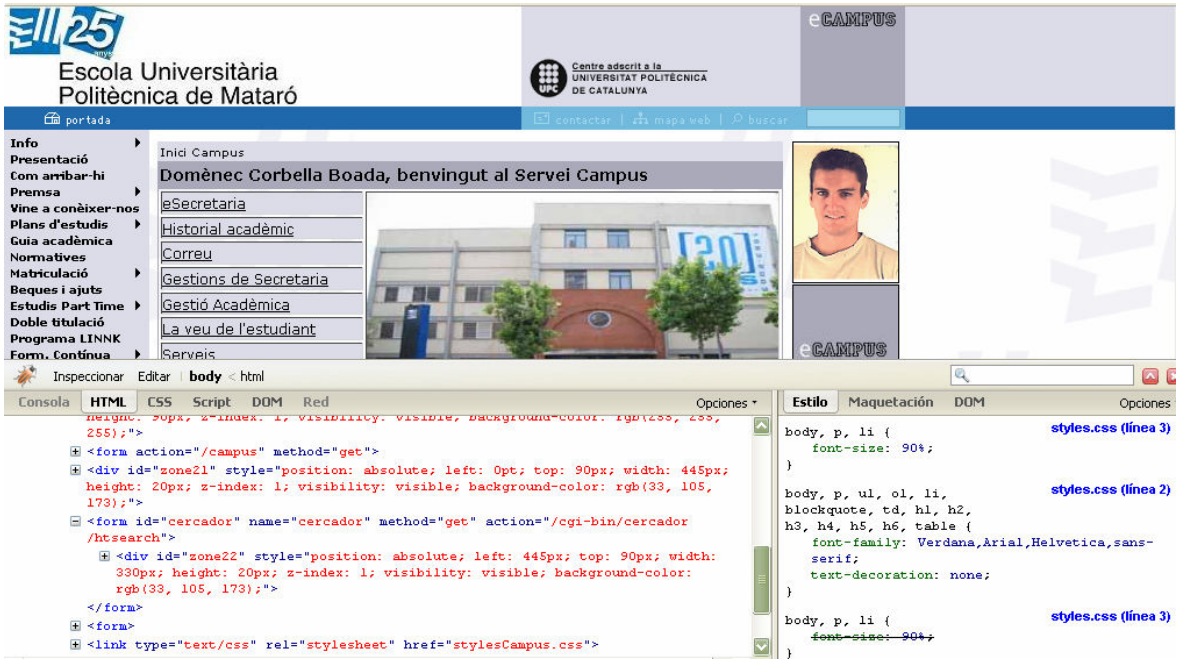
Tenim multitud de funcions de personalització de l'API de Google Maps. A continuació detallarem les més importants adjuntant el propi codi en Javascript.

· Personalització d'idioma de tota la informació textual (noms, controls, avisos, drets autor i rutes) En cas que volguéssim la visualització en català hauríem d'incloure el següent script: `&hl=ca` dins la següent etiqueta: `<script type="text/javascript" src="http://maps.google.com/maps?file=api&v=2&key=abcdefg&hl=ca"` en cas que ho volguéssim en castellà hauríem de substituir el `"ca"` per `"es"`.

· Actualitzacions de l'API: Actualment l'API va per la versió 2. Dins la pàgina de Google Maps trobem com es pot fer la migració d'una versió cap a la següent. La versió es determina amb el paràmetre `"v"` hi ha versions més actuals com la `v=2.75` però recomanaríem, ara per ara treballar amb la `v2` que és la més estable.

· Solució de problemes: Depurar o detectar una errada en el nostre codi en molts casos pot arribar a ser molt difícil i lent de trobar per tant recomanem fer un anàlisi sistemàtic del codi que consistiria en els següents passos:

1. Assegurar-nos que tenim una Key correctament ben adreçada.
2. Buscar errors del tipus tipogràfic; Javascript diferència majúscules de minúscules.
3. Utilitzar el depurador d'errors de Javascript. Recomanem fer les proves en Firefox i amb complement Firebug.



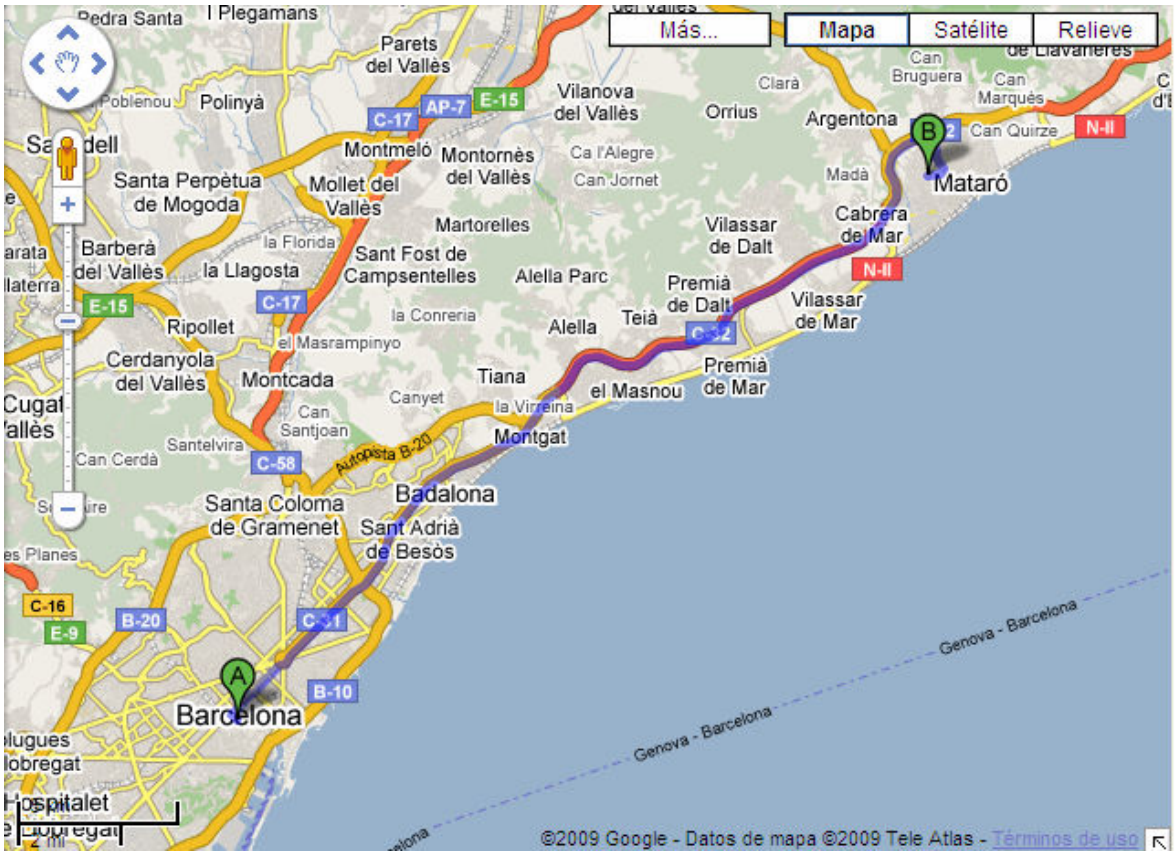
Exemple debug de l'intranet de l'Escola Universitària Politècnica de Mataró.

4. Si amb aquests tres passos no hem aconseguit resoldre el nostre problema podríem recórrer al fòrum de programadors de l'API de Google Maps.

5. Com última solució podríem recórrer a recursos de tercers programadors.

· Reducció de fugues de la memòria del navegador, es donen sobretot en el IExplorer bàsicament degut a que Google Maps fomenta l'ús de tancament de funcions juntament amb el sistema de gestió d'esdeveniments de l'API `(GEvent)`. Per tal d'evitar-ho podem cridar la funció `GUnload()` de la següent forma: `<body onunload="GUnload()">`.
Per detectar les fugues de memòria disposem de l'eina Drip

· Format XHTML Serà imprescindible utilitzar-lo quan vulguem representar polilínies dins el nostre mapa.



Exemple de traç amb polilínies; en aquest cas des de Mataró fins Barcelona.

La capçalera és la següent:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:v="urn:schemas-microsoft-com:vml">
```

Objectes bàsics del mapa:

Com ha primer pas per tal de representar el mapa dins la nostra aplicació el que haurem de fer és registrar la nostra pròpia clau [8].

5 passos per veure el nostre mapa a la web:

1.- Incloure el codi javascript de l'API dins l'etiqueta script, com es pot observar el valor de la key que ens proporciona google des de la seva pàgina, té molts caràcters.

```
<script
src="http://maps.google.com/maps?file=api&v=2&key=ABQIAAAA7PVwCvwFmVYlR6Y4cF97gRRnxs
HGuCM8Y1VDNSbQgXi4Z6cV7RQIWrgpC5yyprfPf-VMxZMCtOFmfQ"
```

```
 type="text/javascript">
</script>
```

2.- Ens creem una capa "div" on allotjarem el mapa:

```
var map = new GMap2(document.getElementById("map_canvas"));// Objecte
elemental; mapa en sí mateix.
<div id="map_canvas" style="width: 500px; height: 300px"></div>
```

3.- Escriure la funció en Javascript de la creació del mapa.

4.- Centrem el mapa a les coordenades que vulguem latitud i longitud, juntament amb el nivell de zoom:

```
map.setCenter(new GLatLng(37.4419, -122.1419), 13);
```

5.- Inicialitzem el mapa amb l'event Javascript onload dins el body.

```
<body onload="initialize()" onunload="GUnload()">
```

Com a recomanació sempre és millor treballar amb variables a l'hora de la creació del mapa; és a dir tenir els valors de latitud, longitud i zoom passats en variables.

Atributs de mapa: Per defecte, l'API de Google Maps ens mostra el mapa en format d'imatges vectorial, un format semblant al de les guies de carrers de les ciutats, però ho podem canviar amb el mètode setMapType();

```
var map = new GMap2(document.getElementById("map_canvas"));
 map.setMapType(G_SATELLITE_MAP);
```

Les opcions que tenim són les següents:

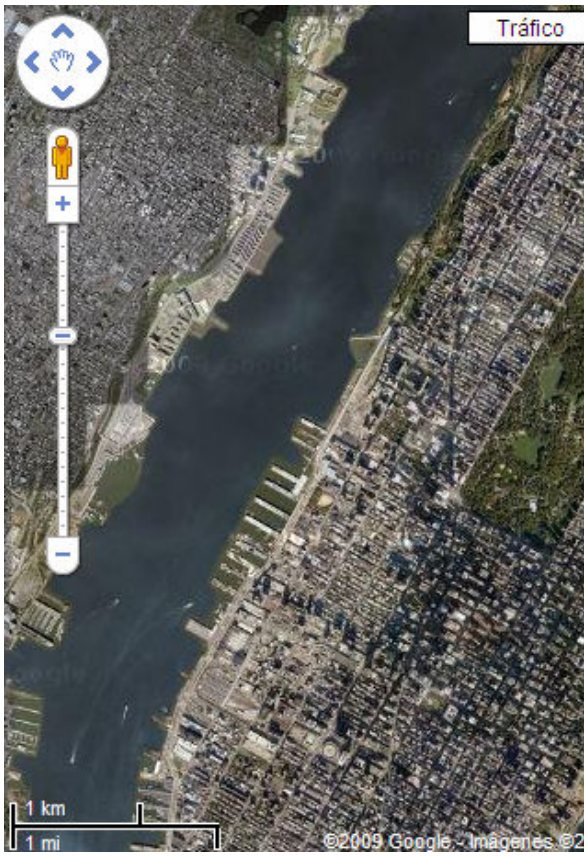
`G_NORMAL_MAP:` Vista predeterminada.

`G_SATELLITE_MAP:` Imatges en satèl·lit; tal i com es veu en el googleEarth.

`G_HYBRID_MAP:` Barreja de les dues anteriors.

`G_DEFAULT_MAP_TYPES:` Barreja de tres tipus.

Exemple: imatge en vista de mapa.



Exemple: imatge en vista satèl·lit.

Exemple: imatge en vista mixta.

Exemple: imatge en vista de relleu.

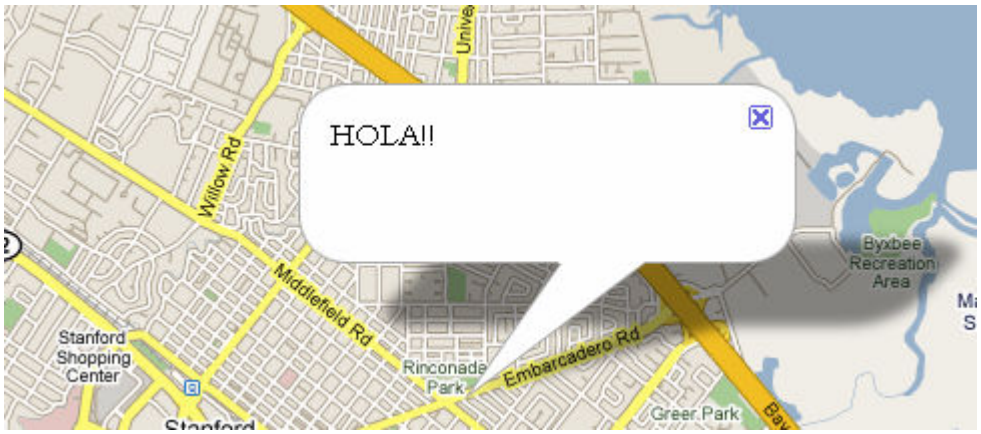
Finestres d'informació:

Les finestres d'informació són bombolles de text tipus vinyeta que apareixen en el punt on nosaltres li indiquem; ens permet donar el nom del lloc; descripció, afegir-hi hipervincles i inclús contingut d'imatges o integrar-hi flash o vídeo.(imatge).

Per tal d'inserir una etiqueta, partint que hem creat una variable map, es faria de la següent forma:

```
var map = new GMap2(document.getElementById("map_canvas"));
map.setCenter(new GLatLng(37.4419, -122.1419), 13);
map.openInfoWindow(map.getCenter(),document.createTextNode("HOLA!!"));
```

El resultat seria el següent:



Inserció de marca:

Per inserir un pin o marca ho hem de realitzar cridant a la funció de mapa addOverlay i crear un nou Gmarker(point). El codi resultant seria el següent:

```
map.addOverlay(new GMarker(point));
```

Esdeveniments (events) del mapa:

Els esdeveniments són accions que pot prendre l'usuari i que permet interactuar amb l'API de Google Maps, tenint un feedback. Consultar [9] per obtenir més informació sobre els esdeveniments de l'objecte GMap2. Els més típics són: click, dbclick, move etc...

Per registrar els esdeveniments utilitzem el mètode estàtic.

```
GEvent.addListener().
```

En aquest mètode li hem de passar tres paràmetres: l'objecte en el nostre cas serà el mapa, els esdeveniments que volem detectar, per exemple, "click" i, finalment la funció que volem que es cridi quan es succeeixi el succés. Un exemple podria ser el següent:

```
GEvent.addListener(map, "click", function() {
  alert("Has fet click en el mapa.");
});
```

Cal tenir en compte que tots els esdeveniments són provocats per les interaccions que realitza l'usuari mitjançant el teclat o el ratolí es propaguen en el DOM [10].

Tal com hem vist, el model de l'API de Google Maps crea i gestiona els seus propis esdeveniments, no obstant el DOM també en té de propis; l'API ens permet detectar i vincular-los sense la necessitat d'utilitzar codi personalitzat. Per accedir als esdeveniments del DOM podem utilitzat els mètodes estàtics: `GEvent.addDomListener()` i `GEvent.bindDom()`

És convenient eliminar els detectors d'esdeveniments un cop han deixat de ser necessaris observem el següent exemple; podem observar que el propi mètode GEvent s'encarrega de l'eliminació.

```
function MyApplication() {
  this.counter = 0;
  this.map = new GMap2(document.getElementById("map")); //creem el mapa.
  this.map.setCenter(new GLatLng(37.4419, -122.1419), 13);//determinem la posició de
la vista
  var myEventListener = GEvent.bind(this.map, "click", this, function(marker,point)
{
   //guardem l'event click de l'usuari en una variable i mirem el seu valor
   if (this.counter == 0) {
      if (point) {
```

```
            this.map.addOverlay(new GMarker(point)) //afegim un nou marcador
            this.counter++;

          } else {
            this.removeOverlay(marker) //si no s'ha fet click s'esborra el marcador.
          }
        } else {
          GEvent.removeListener(myEventListener); // eliminem el captador d'events.
        }
      });
    }
```

Controls del mapa: L'API incorpora una sèrie de controls que permeten interactuar amb l'usuari:

`GLargeMapControl` Ens fa un zoom màxim i un desplaçament a l'extrem superior esquerra.

`GSmallMapControl` Igual que el anterior però amb el zoom i el desplaçament més petit possible.

`GSmallZoomControl` Fa un zoom d'un nivell.

`GScaleControl` Una escala pel mapa.

`GMapTypeControl` Són els botons que permeten alternar diferents tipus de mapes.

`GHierarchicalMapTypeControl` Selecció de botons i menú.

`GOverviewMapControl` Ens mostra un mapa de vista general que es pot contraure.

Tots aquests controls estan basats en l'objecte `GControl`.

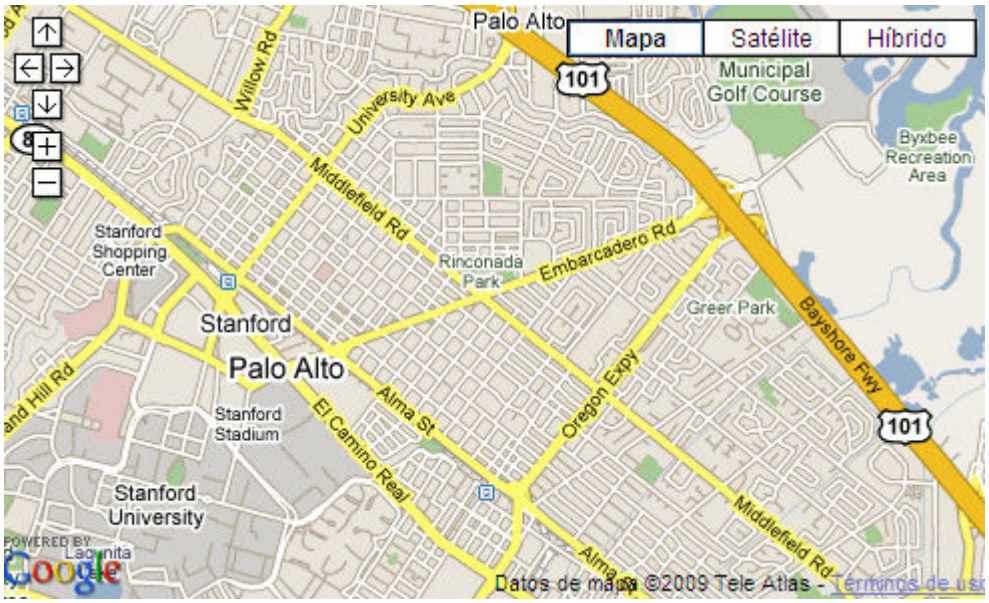
En cas que volguéssim afegir nous controls, la forma de fer-ho és la següent; podem afegir tants controls com vulguem:

```
map.addControl(new GLargeMapControl());
```

El mètode `addControl` té un segon paràmetre opcional: `GControlPosition`; que ens permet escollir on volem disposar el control que afegim. Les opcions són les següents:

```
G_ANCHOR_TOP_RIGHT
G_ANCHOR_TOP_LEFT
G_ANCHOR_BOTTOM_RIGHT
G_ANCHOR_BOTTOM_LEFT
```

El resultat seria aquest:




```
map.addControl(new GLargeMapControl(), bottomRight);
```

L'API de google Maps ens permet incorporar controls personalitzats, per exemple, fer un zoom de forma textual (apropar/allunyar) amb botons personalitzats etc...

Superposicions del mapa:

Les superposicions són objectes vinculats al mapa amb unes coordenades de latitud i longitud, per tant, si fem qualsevol tipus d'interacció amb el mapa aquestes també es veuran afectades. Les més habituals són:

Marcadors: `GMarker` també pot utilitzar el `GIcon` per personalitzar.

Polilínies: `Gpolyline` (es representen com un conjunt de punts).

Polígons: Semblant a les polilínies; també es representa com un conjunt de punts.

Superposició de terreny.

Superposició de mosaics: El propi mapa ja es representa com una superposició de mosaic, el podem modificar mitjantçant `GTileLayerOverlay.`

Superposicions personalitzades.

Els marcadors:

Els marcadors per defecte porten la típica icona de Google Maps `G_DEFAULT_ICON` però si ens interessa, la podem personalitzar. Els marcadors estan pensats per ser interactius; per tant, poden rebre per part del usuari esdeveniments del tipus `click`.

En concret els marcadors que es poden arrastrar utilitzen els següents esdeveniments: `dragstart`, `drag` i `dragend`. Abans de res els haurem d'inicialitzar com a `draggable=true`. Els marcadors desplaçables tenen una animació predeterminada que també la podem canviar amb l'opció `bouncy=false` i d'aquesta manera el marcador no tindrà animació. (`bouncy=true` suposa una animació de la icona del tipus efecte rebot).

Exemple de creació de marcador desplaçable:

```
var marker = new GMarker(center, {draggable: true});
```

Exemple de personalització d'icona:

```
var bike = new GIcon(G_DEFAULT_ICON);
bike.image = "http://www.youroutes.com/img/bike.png";
```

El format de les imatges ha de permetre la transparència, per tant, es recomanable utilitzar el format d'imatge amb extensió .png de 24 bits amb transparència alfa; Si volem que el nostre marcador també tingui ombra ho haurem d'especificar; cal tenir en compte que l'ombra es projecta a 45 graus.

El codi resultant seria de la manera següent:

```
var youRoute = new GIcon();
youRoute.image = "http://www.youroutes.com/img/youroute.png";
youRoute.shadow = "http://www.youroutes.com/img/youroute_shadow.png";
youRoute.iconSize = new GSize(12, 20);
youRoute.shadowSize = new GSize(22, 20);
youRoute.iconAnchor = new GPoint(6, 20);
youRoute.infoWindowAnchor = new GPoint(5, 1);
```

`MarkerManagerOptions:` Ens permet ajustar vàries opcions dels marcadors, tals com ara:

`maxZoom:` Especifiquen el valor màxim de zoom que podem fer sobre el nostre marcador.

`borderPadding:` Especifiquem el valor màxim de desplaçament que podem fer fora dels contorns del mapa, els valors van en píxels.

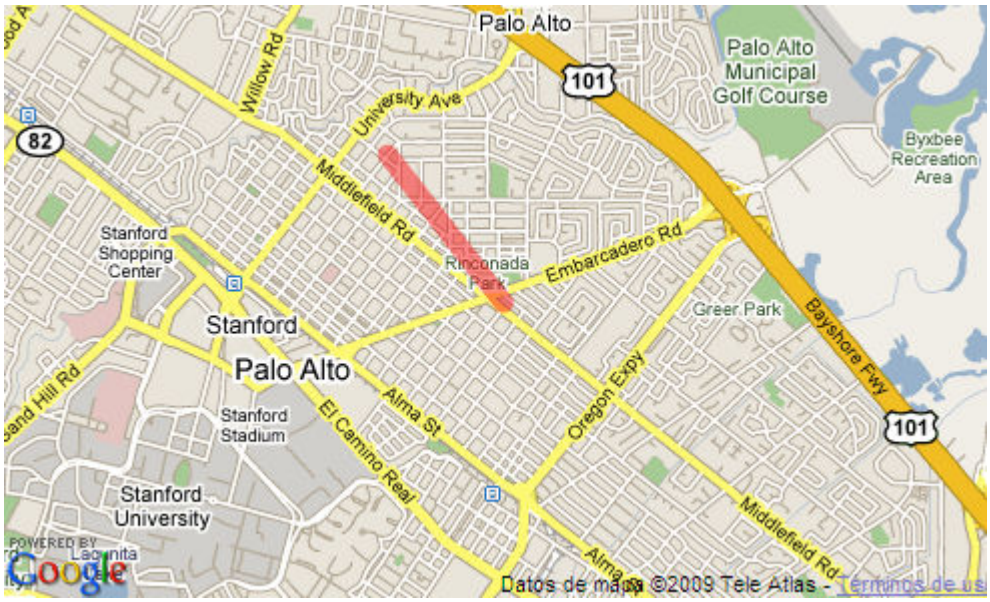
`trackMarkers:` amb l'opció `setPoint()` a `true` especifiquen aquell marcador com administrador i es realitzarà un seguiment dels moviments d'aquell marcador. De forma predeterminada està a `false`. Podrem afegir marcador amb la funció `addMarkers()` aquests no es mostraran al mapa fins que no fem un `refresh()` del `MarkerManager` amb aquest mètode podem obtenir un camí amb tot de marcadors.

Polilínies:

Les polilínies es representen en el mapa com una sèrie de segments rectes; Podem ajustar el color, el gruix i la opacitat.

```
var polyline = new GPolyline([
  new GLatLng(37.4419, -122.1419),
  new GLatLng(37.4519, -122.1519)
], "#ff0000", 10);
map.addOverlay(polyline);
```

Aquest exemple realitza una línia recte de color vermell de 10 píxels d'ample entre els dos punts especificats:



Polilínies geodèsiques:

Són línies rectes respecte la projecció actual; es mostren com a rectes i no respecten la curvatura de la Terra. Si volem passar una polilínia a geodèsica és tant senzill com posar el paràmetre `geodesic:trae`

```
var polyOptions = {geodesic:true};
var polyline = new GPolyline([
  new GLatLng(40.65642, -73.7883),
  new GLatLng(61.1699849, -149.944496)
  ], "#ff0000", 10, 1, polyOptions);
  map.addOverlay(polyline);
```

Polilínies codificades:

Les línies codificades es representen com una sèrie de punts una línia llarga pot representar molt punts i això suposarà un gran cost de memòria i temps per representar-les. Una línia codificada senzilla pot tenir el següent aspecte:

```
var encodedPolyline = new GPolyline.fromEncoded({
    color: "#FF0000",
    weight: 10,
    points: "yzocFzynhVq}@n}@o}@nzD",
    levels: "BBB",
    zoomFactor: 32,
    numLevels: 4
});
map.addOverlay(encodedPolyline);
```

Com podeu observar els punts porten associats la informació latituds i longituds i van codificats en ASCII d'aquesta manera no es visualitzen de la forma tradicional.

També existeix una utilitat interactiva que permet passar la informació de longitud i latitud a punts. [11]

Polígons:

Consisteix en una sèrie de punts amb una seqüència enllaçada. Cal precisar que els polígons tenen els extrems tancats i igual que les línies les podem personalitzar. Tant el GPolyline com el GPolygon fan les seves representacions en vectorial; per tant, encara que ampliem el nivell de zoom no es veuran píxels.

```
GEvent.addListener(map, 'click', function(overlay, latlng) {
  var lat = latlng.lat();
  var lon = latlng.lng();
  var latOffset = 0.01;
  var lonOffset = 0.01;
  var polygon = new GPolygon([
    new GLatLng(lat, lon - lonOffset),
    new GLatLng(lat + latOffset, lon),
    new GLatLng(lat, lon + lonOffset),
    new GLatLng(lat - latOffset, lon),
    new GLatLng(lat, lon - lonOffset)
  ], "#f33f00", 5, 1, "#ff0000", 0.2);
  map.addOverlay(polygon);
});
```

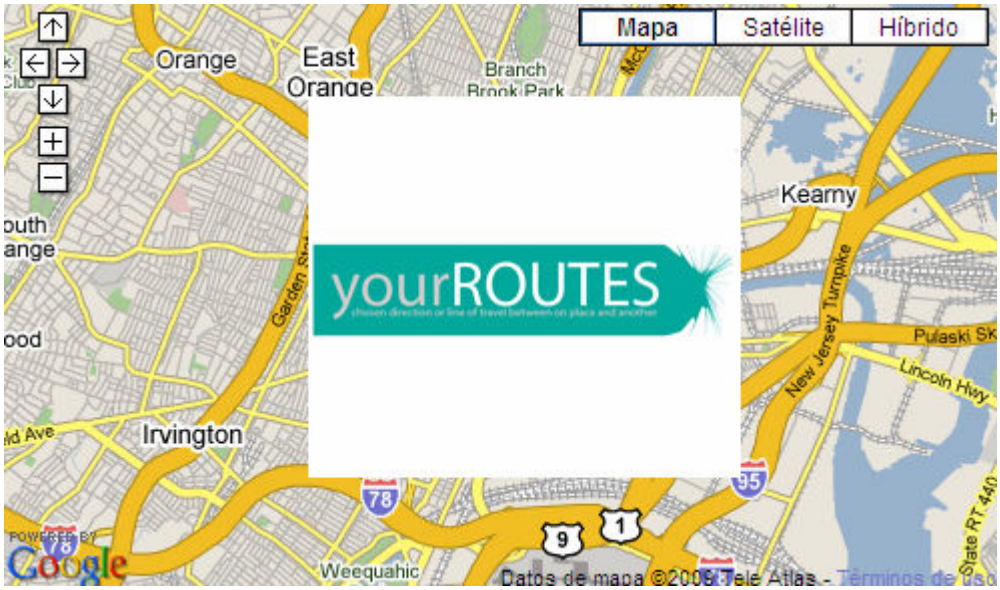
Aquest exemple realitza un quadrat de 10 píxels entre els 4 punts el polígon i es tanca retornant les coordenades inicials. Sempre s'han de tancar els punts per tal d'evitar comportaments imprevistos.

Superposicions del sòl:

En el cas que tinguem que superposar alguna imatge, en aquest cas es quan utilitzarem l'opció de superposició del sòl. Per fer-ho hem d'utilitzar l'objecte `GGroundOverlay`, li hem de passar la direcció de la imatge igual que passa amb el `Gmarker` juntament amb l'objecte GLatLngBounds.

```
var boundaries = new GLatLngBounds(new GLatLng(40.716216,-74.213393), new
GLatLng(40.765641,-74.139235));
var logo = new GGroundOverlay("http://www.yourotues.com/images/logo.jpg",
boundaries);
//enmagatzem a la variable logo la ruta de l'imatge que volem superposar
map.addControl(new GSmallMapControl());
map.addControl(new GMapTypeControl());
map.addOverlay(logo); //superposem l'imatge sobre el mapa.
```

El resultat d'aquest exemple és la superposició d'un mapa antic sobre l'actual:



Superposicions de mosaics:

El mapa tal com el veiem es divideix entre diferents quadrants formant un mosaic, aquests quadrats estan numerats del extrem Nord Oest a l'extrem Sud Est. Trobem moltes zones on aquests mosaics (quadrats) el nivell de zoom que tenim no és el desitjat ja sigui perquè es tracta de zones de poc interès, menys població, zones com ara la zona de l'Oceà Pacífic on Google Maps no es permet fer grans aproximacions amb el zoom. Cada nivell d'aproximació que fem divideix el quadrant en 4 mosaics més de manera que el que tenim és una quadrícula on els quadrats cada cops es fan més petits aportant més resolució i zoom. Per exemple, si fem un nivell d'aproximació de 2; voldrà dir que tenim el món dividit en una quadrícula de 4x4 amb un total de 16 quadrats. Si volem modificar la informació d'aquests quadrats l'API de Google Maps ens dóna dues opcions:

1. Implementa el nostre propi quadrat sobre el existent mitjançant `GTileLayerOverlay`.

2. Implementa el nostre tipus de mapa mitjançant `GMapType`.

El 1er cas és més senzill d'implementar que el segon, per tant, com que aquest tipus de modificació només es donarà en aplicacions molt concretes i específiques només detallarem el primer cas.

Per fer una superposició de capa en el mosaic el primer que hem de fer és crear un objecte del tipus: `GCopyrightCollection` i adjuntar-lo a la capa del mosaic per tal d'indicar els permisos que utilitzen les imatges.

Exemple: Els passos serien els següents: indicar els drets d'autor de l'imatge i a continuació implementar-la amb els tres mètodes abstractes:

```
// Indica l'informació sobre els drets d'autor.
// Las imágenes utilizadas deben indicar los permisos de derechos de autor.
var myCopyright = new GCopyrightCollection("© ");
myCopyright.addCopyright(new GCopyright('Demo',
  new GLatLngBounds(new GLatLng(-90,-180), new GLatLng(90,180)),
  0,'©2009 Google'));
// Crea la superposició de la capa de mosaic i implementa els tres mètodes abstractes.
var tilelayer = new GTileLayer(myCopyright);
tilelayer.getTileUrl = function() { return "../include/tile_crosshairs.png"; };
tilelayer.isPng = function() { return true;};
tilelayer.getOpacity = function() { return 1.0; }
var myTileLayer = new GTileLayerOverlay(tilelayer);
var map = new GMap2(document.getElementById("map_canvas"));
map.setCenter(new GLatLng(37.4419, -122.1419), 13);
map.addOverlay(myTileLayer);
```

Serveis del mapa:

L'API de Google Maps cada cop implementarà més serveis, bàsicament, perquè és una llibreria de "open source" que està en plena evolució. Detallaré els serveis més bàsics i útils.

El servei per excel·lència és el de **codificació geogràfica**; és el que permet donada una adreça i un número trobar les coordenades latitud longitud dins del mapa i indicar-les amb un marcador. Podem accedir directament mitjançant sol·licituds HTML o amb l'objecte `GClientGeocoder.`

S'ha de tenir en compte que aquest procés consumeix temps i recursos i per tant no convé fer-ne un ús excessiu. Els resultats s'emmagatzemen a una memòria cau o "caché" anomenada cau de codificació geogràfica (aquesta cau memoritza els resultats buscats i permet retrobar-los de forma més ràpida). Per tant, sempre que sigui possible farem les sol·licituds geogràfiques a través de HTML.

Per fer-ho enviem la sol·licitud directament a http://maps.google.com/maps/geo? Amb els següents paràmetres:

`q`: El valor de la direcció que volem codificar.
`key`: La clau de l'API del nostre mapa.
`output`: El format en el que volem que ens retorni la resposta: xml, kml, csv o json.

Exemple de petició:

```
http://maps.google.com/maps/geo?q=1600+Amphitheatre+Parkway,+Mountain+View,+CA&output=xml&k
ey=ABQIAAAA7PVwCvwFmVYlR6Y4cF97gRRnxsHGuCM8Y1VDNSbQgXi4Z6cV7RQIWrgpC5yyprfPf-VMxZMCtOFmfQ
```

La resposta que obtindríem vindría donada en format XML i KML.

Si el que volem és obtenir una resposta més breu el que podem fer és sol·licitar un output en format CSV; La resposta que obtenim consta de 4 números:

1. Codi d'estat de HTTP
2. Precisió
3. Latitud
4. Longitud

L'aspecte seria el següent: `200,6,42.730070,-73.690570`

Ús d'objectes Street View:

Street View ens proporciona vistes panoràmiques a 360º. Per tal d'obtenir la visualització es necessita tenir instal·lat el complement de Flash al navegador.

A data d'avui no és possible veure totes les ciutats, només les importants i no de tots els països amb aquest format, doncs Street View suposa un gran cost, ja que suposa la realització de un gran número de fotografies amb unes òptiques ull de peix. Per tal de preservar l'anonimat de la gent es difuminen les cares i les matrícules de les persones i dels cotxes que hi surten, respectivament. (Street View, ha generat un gran debat sobre la preservació de la intimitat).

Les imatges s'ofereixen a través de l'objecte `GStreetviewPanorama` ens proporciona la interfície de l'API en un visor Flash. Si volem incorporar street view a la nostra aplicació em de seguir els pasos següents:

1.- Crear un contenidor; normalment crearem una capa o "layer" .

2.- Crear l'objecte i `GStreetviewPanorama` col·locar-lo al contenidor.

3.- Inicialitzar l'objecte i dir-li que faci referència a una posició inicial.

4.- Verificar si el client disposa del component Flash al seu navegador [12].

Exemple de la vista que s'obté amb el Street View.

Definició de vista panoràmica:

La ubicació del Street View la definim amb la posició, però un altre paràmetre que s'ha de tenir en compte amb el mode Street View és la posició de la càmera; d'això s'encarrega l'objecte `GPov` que té tres propietats:

1.- `yaw`: defineix l'angle de la càmera el Nord absolut (de la terra) representa 0º; Est 90º els angles van en el sentit horari del rellotge.

2.- `pitch`: defineix la inclinació de la càmera en sentit vertical la rotació es mou en un interval de 90º / -90º els valors negatius els prendrà quan la inclinació sigui cap amunt i positius amb graus positius; en llenguatge cinematogràfic si agaféssim com a referència un objecte seria l'equivalent al picat i contrapicat.

3.- `zoom`: indica el nivell d'apropament, per defecte és 0 que és el valor de la càmera sense zoom. Leds transicions de zoom (apropament) generalment són lentes el format de les imatges és un CGPJ progressiu de forma que a mesura que es va carregant la imatge aquesta va guanyant amb definició, inicialment es veu pixelada sense definició.

Així doncs, podem, a través del paràmetre `GStreetviewPanoramaOptions`, definir la vista que vulguem. Exemple:

```
mataroPark = new GLatLng(42.345573,-71.098326);
myPOV = {yaw:370.64659986187695,pitch:-20};
svOpts = {latlng:mataroPark, pov:myPOV};
var myPano = new GStreetviewPanorama(document.getElementById("pano"), svOpts);
```

També la podem definir a posteriori amb el mètode `setLocationAndPov()` un cop tenim creat l'objecte `GStreetViewPanorama`.

```
var myPano = new GStreetviewPanorama(document.getElementById("pano"));
mataroPark = new GLatLng(42.345573,-71.098326);
myPOV = {yaw:370.64659986187695,pitch:-20};
myPano.setLocationAndPOV(mataroPark, myPOV);
```

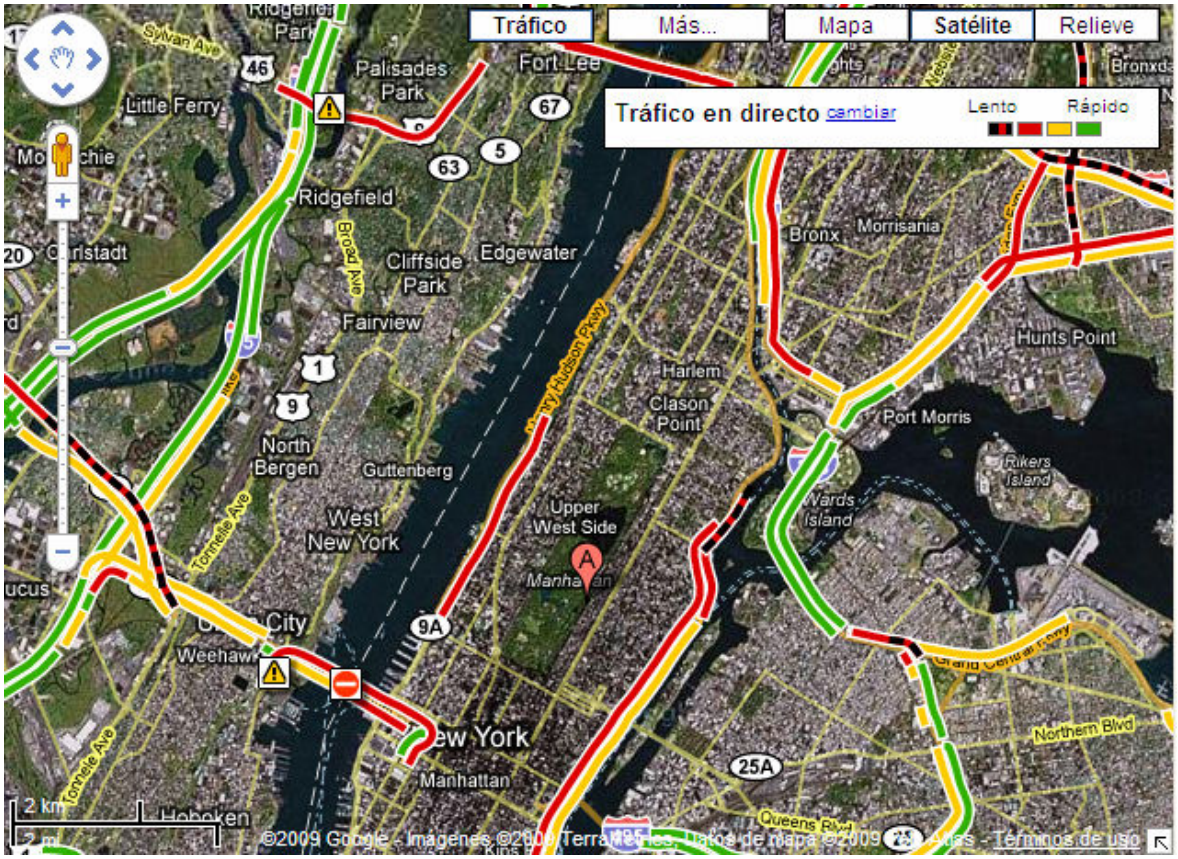
Superposicions de tràfic:

L'API Google Maps ens permet afegir informació del tràfic mitjançant l'objecte `GTrafficOverlay` que implementa l'interfície `GOverlay` la informació s'afegeix a través del mètode `GMap2.addOverlay()`.

Cal tenir en compte que aquesta informació només es veurà en les ciutats que ho admeten, igual que passa amb Street View. (bàsicament en ciutat dels EEUU com ara NY, Sant Francisco etc…)

Exemple:

```
var map;
var trafficInfo = new GTrafficOverlay();
function initialize() {
  map = new GMap2(document.getElementById("map_canvas"));
  map.setCenter(new GLatLng(49.496675,-102.65625), 3);
  map.addOverlay(trafficInfo);
}
```



Exemple del traffic View a la ciutat de NY

Rutes

Podem afegir rutes mitjançant l'objecte `GDirections`. Aquest objecte rep les rutes com a cadena de caràcters: origen – destí, com ara: ("Barcelona-Mataró") o també podem fer-ho a través de les coordenades de latitud i longitud: ("40876 -34887 a 67984 -33447"). Les rutes es sol·liciten a través del mètode: `GDirections.load()`.

Aquest mètode necessita la cadena de consulta i una sèrie de paràmetres opcionals `GDirectionsOptions`

Depenent de com haguem construït l'objecte GDirections obtindrem resultat d'una manera o altre:

1. Si `GDirection` l'hem construït a través de l'objecte GMap2: El resultat de la consulta contindrà una superposició de polilínea. Per recuperar l'objecte ho podem fer amb el mètode: `GDirections.getPolyline()`.

2. Si `GDirection` l'hem construït amb un element `DIV`. El resultat contindrà un objecte del tipus `GRoute`. Format al mateix temps per conjunt d'objectes `GStep.` Si la ruta consta de varis punts el resultat seran varis objectes `Groute` amb els seus corresponents `GStep`.

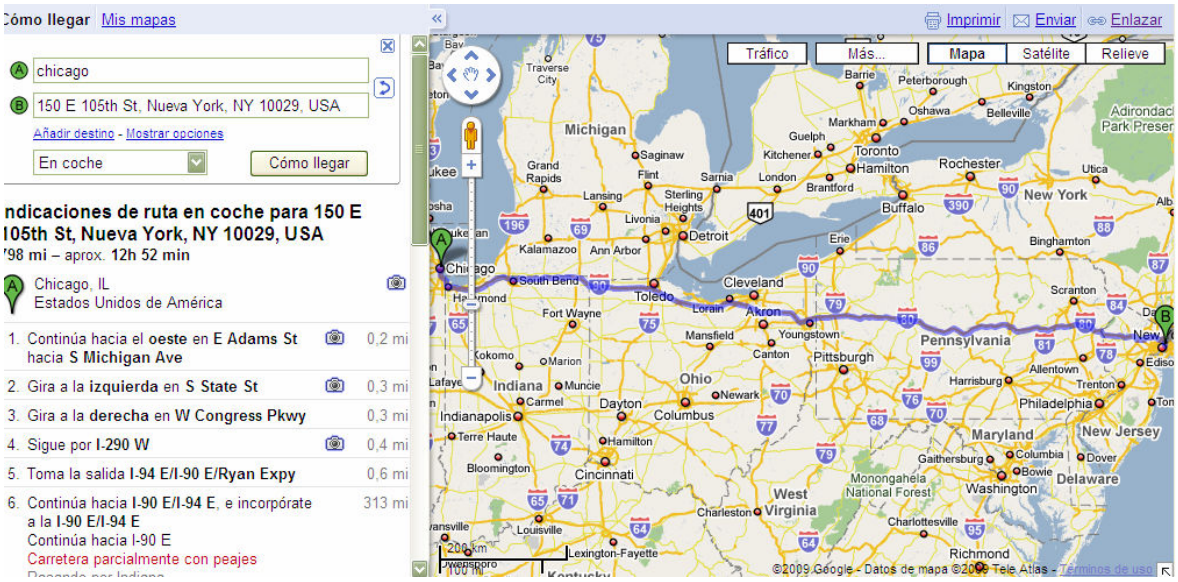
Per recuperar l'objecte ho podem fer amb el mètode:

`GDirections.getRoute(i:Number)` I cada pas el podem recuperar amb `GRoute.getStep(i:Number)` finalment el resum de cada pas (descripció) el recuperem amb `GStep.getDescriptionHtml()`.

El següent exemple crea un objecte de ruta i enregistra un mapa el DIV allotja les rutes calculades.

```
var map;
var directionsPanel;
var directions;

function initialize() {
  map = new GMap2(document.getElementById("map_canvas"));
  directionsPanel = document.getElementById("my_textual_div");
  map.setCenter(new GLatLng(49.496675,-102.65625), 3);
  directions = new GDirections(map, directionsPanel);
  directions.load("New York, NY to Chicago, IL");
}
```



L'exemple quedaria d'aquesta manera.

L'objecte GDIrections també permet crear rutes des de varis punts; no només un origen i un final. Per fer-ho tenim el mètode `GDirections.loadFromWaypoints()` li hem de passar una matriu de punts latitud longitud. Cada punt es processa com una ruta independent i es retorna un objecte `GRoute` independent. Amb els correspondents `GStep`.

L'objecte `GDirections` activa 3 esdeveniments que es poden interceptar:

1.- `Load`: s'activa quan es retorna el resultat de la ruta correctament.

2.- `Addoverlay`: s'activa quan es mostra el resultat en forma de recorregut sobre el mapa del div.

3.- `Error`: s'activa quan la ruta ha generat algun error.

3.2. JSP i la llibreria específica de Google Maps.

La llibreria Google Maps JSP "taglibrary" ens proveeix la capacitat de treballar amb Google Maps sense la necessitat d'utilitzar AJAX ni Javascript.

Per fer-ho hem de descarregar una llibreria específica que a base d'etiquetes del tipus: <googlemaps:key> ens permet manipular certes propietats de l'API que hem vist en el apartat anterior. És una llibreria que es troba en evolució, per tant, cada cop es va enriquint més i incorporant noves propietats. A la URLgrafia [13] trobareu un enllaç amb totes les funcionalitats disponibles i l'enllaç per descarregar-se la llibreria.

A data d'avui les funcions més comuns són les possibles; és a dir: podríem carregar el nostre mapa, posar-hi els marcadors que volguéssim, inserir bombolles de text, podríem dibuixar línies representant rutes, visualitzar les superposicions de tràfic, superposar una imatge que nosaltres volguéssim, modificar el nivell de zoom etc...

En cas que la nostra aplicació hagi de treballar explícitament amb Street View és quan seria convenient treballar directament amb l'API de Google Maps i fer les crides directament en Javascript. La llibreria ens permet incorporar Javascript.

3.2.1. Característiques principals. JSP i la llibreria específica de Google Maps.

La llibreria té una estructura jeràrquica i per invocar certes funcions abans hem de fer la crida d'altres funcions que les podríem anomenar pare.

La funció pare seria <googlemaps:map> a partir d'aquí en podem cridar de moltes altres.

Aquest exemple ajudarà a entendre el funcionament de la llibreria Google Maps per JSP. El resultat és la visualització del mapa amb un simple marcador.

```
<googlemaps:map id="map" width="250" height="300" version="2" type="STREET"
            zoom="12">
  <googlemaps:key domain="localhost" key="xxxx"/>
  <googlemaps:point id="point1" address="74 Connors Lane" city="Elkton"
                    state="MD" zipcode="21921" country="US"/>
  <googlemaps:marker id="marker1" point="point1"/>
</googlemaps:map>
```

Tot i que la llibreria ha estat pensada per ésser compatible amb tots els navegadors, en el cas que haguem de treballar amb IExplorer i haguem de dibuixar polilínies, caixes o polígons per tal d'obtenir un funcionament correcte haurem d'incloure les etiquetes `<!DOCTYPE>` i `<html>` de la següent forma:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:v="urn:schemas-microsoft-com:vml">
```

3.2.2. Avantatges i inconvenients. JSP i la llibreria específica de Google Maps.

Una dels principals avantatges és que no tenim cap tipus de limitació; és a dir podem fer qualsevol tipus d'implementació explotant al màxim la potència de Google Maps (més endavant veurem que depenen de l'opció que escollim no sempre tindrem totes les opcions.

Per poder treballar-hi correctament necessitem un bon coneixement de l'API de Google Maps i tenir certes nocions en Javascript, un altre avantatge és que disposem de molts exemples, molta i bona documentació i molts fòrums de suport.

L'inconvenient és que no s'integra visualment dins el Netbeans; per tant, només veurem el codi i fins que no compilem no obtindrem el resultat de com queda la nostra aplicació.

La pàgina de codi resultant de l'aplicació pot arribar a ser enrevessada d'entendre i poc intel·ligible per algú que no hi estigui habituat.

3.3.- JSF i la llibreria de Google Maps per JSF.

3.3.1. Característiques principals. JSF i la llibreria de Google Maps per JSF.

La llibreria de Google Maps per JSF la trobem disponible a la següent pàgina: *http://ode.google.com/p/gmaps4jsf* a data d'avui té unes funcionalitats limitades, que depenent del tipus de projecte poden ser més que suficients. S'ha de remarcar que tot i que es pot integrar dins el Netbeans sense cap tipus de problema si treballem amb pàgines Visual web no veurem el mapa fins que no compilem el projecte. Per tant a nivell de maquetació o disseny haurem de fer-ho amb el codi dins la pestanya JSF que tenim al Netbeans.

Les possibilitats que ens permet aquesta llibreria són les següents:

Situar un punt en el mapa passant la latitud i longitud o bé l'adreça.

- Afegir marcadors.
- Afegir informació de text.
- Afegir controls al mapa del tipus: cursors o zoom.
- Create event listener(s) on the map objects.
- Dibuixar polilínies sobre el mapa.
- Dibuixar polígons sobre el mapa.
- Afegir groundOverlay(s) sobre el mapa.
- Personalitzar diferents operacions com ara el nivell de zoom del mapa, o els butons pels diferents tipus de mapa, ...etc.
- Crear un Streetview Panorama e integrar-lo al mapa.

3.3.2. Configuració. JSF i la llibreria de Google Maps per JSF.

Per tal de poder treballar des de Netbeans amb la llibreria Google Maps per JSF, el primer que hem de fer es descarregar-nos la core: gmaps4jsf-core-1.1.jar l'haurem d'incloure dins el nostre projecte a la carpeta corresponent de les llibreries (lib) dins la carpeta WEB-INF.

Un cop fet això dins la pàgina que vulguem incloure el mapa em d'inserir el següent codi:

1. Hem de cridar la referència de la llibreria:

```
<%@ taglib uri="http://code.google.com/p/gmaps4jsf/" prefix="m" %>
```

2. Hem d'introduir la nostra Key dins el següent script:

```
<script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;
key=ABQIAAAAxrVS1QxlpJHXxQ2Vxg2bJBT2yXp_ZAY8_ufC3CFXhHIE1NvwkxS9AOPy_YJl48ifAy4m
q6I8SgK8fg"
type="text/javascript">
</script>
```

3. Cridem a la llibreria **m** i mostra el mapa amb la longitud i latitud que nosaltres volguem:

```
<m:map width="500px" height="500px" latitude="30.01" longitude="31.14" />
```

4. Creem un mapa amb un marcador e informació de text:

```
<m:map width="500px" height="500px" latitude="30.01" longitude="31.14">
        <m:marker latitude="30.01" longitude="31.14"/>
        <m:htmlInformationWindow latitude="30.01" longitude="31.14"
        htmlText="<B>Egypt</B>"/>
</m:map>
```

5. Creem un mapa amb l'adreça i posem un marcador:

```
  <m:map address="#{addressBean.address}">
         <m:marker/>
         <m:htmlInformationWindow htmlText="#{addressBean.address}"/>
  </m:map>
```

6. Afegim controls al nostre mapa:

```
<m:map width="500px" height="500px" latitude="30.01" longitude="31.14">
     <m:mapControl name="GLargeMapControl"
       position="G_ANCHOR_BOTTOM_RIGHT"/>
     <m:mapControl name="GMapTypeControl"/>
</m:map>
```

7. Creem un esdeveniment d'escolta al nostra mapa per interactuar-hi:

```
<m:map width="90%" height="90%" latitude="24" longitude="15">
    <m:eventListener eventName="moveend"    jsFunction="mapMoveEndHandler"/>
</m:map>
```

8. Creem un polígon amb els diferents punts:

```
<m:polygon lineWidth="4">
                <m:point latitude="30.01" longitude="31.14"/>
                <m:point latitude="-33" longitude="19"/>
<m:point latitude="39" longitude="-101"/>
                <m:point latitude="30.01" longitude="31.14"/>
        </m:polygon>
```

9. Creem una línia amb diferents punts:

```
<m:polyline>
                <m:point latitude="30.01" longitude="31.14"/>
                <m:point latitude="-33" longitude="19"/>
<m:point latitude="39" longitude="-101"/>
        </m:polyline>
```

10. Afegim una imatge al nostre mapa:

```
<m:map width="90%" height="90%" latitude="24" longitude="15" zoom="2">
  <m:groundoverlay imageURL="http://www.youroutes.com/img/logo.png"
                 startLatitude="7" endLatitude="23"
                 startLongitude="-54" endLongitude="84" />
 </m:map>
```

11. Creem un street view:

```
<m:streetViewPanorama width="500px" height="500px"
                latitude="42.345573" longitude="-71.098326" />
</m:streetViewPanorama>
```

Com es pot observar treballar amb la llibreria JSF resulta molt senzill mostrar un mapa per fer aquestes manipulacions senzilles possiblement sigui inclús més fàcil que treballar directament amb l'API de Google Maps i en Javascript.

Per tant en aquells casos on trobem un exemple que s'adapti a les nostres necessitats, és molt recomanable utilitzar aquesta llibreria.

La llibreria també ens permet treballar en Javascript en cas que sigui necessari, per això disposem de la jsVariable que és un atribut de la llibreria.

Amb GMaps4JSF podem treballar amb Facelets (és un llenguatge de plantilles que ens permet construir un arbre de components, ens permet una gran reutilització, podem crear etiquetes lògiques a mida, creació de llibreries de components i un desenvolupament més agradable de cara al dissenyador) i Portlets (components modulars d'interfície d'usuari gestionades i visualitzades en un portal web, un portlet tenen certa similitud amb els servlets; Els portlets són manipulats per un contenidor especialitzat, generen contingut dinàmic, el cicle de vida del portlet és controlat pel contenidor, interactuen amb el client amb request/response)

3.3.3. Avantatges i inconvenients. JSF i la llibreria de Google Maps per JSF.

És fàcil de crear i fer petites variacions sobre el mapa, és ràpid i amb el sistema de suggeriments de Netbeans pot resultar intuïtiu.

Un altre avantatge és que no cal tenir coneixements amplis de Javascript, és com si treballéssim amb una classe de Google Maps que té una sèrie de funcions.

Al tractar-se d'un llenguatge etiquetat es diferencia bé el codi, per tant, és fàcil de localitzar.

Com inconvenients destacaria que no té interfície visual dins el Netbeans per tant totes les propietats hauran d'ésser a través del codi.

La documentació que hi ha és poc completa i molt limitada.

Hi ha pocs recursos i poca gent que estigui treballant amb aquesta llibreria (comparat amb la gent que treballa directament amb l'API) pot arribar a ser un handicap el fet de quedar-nos encallats i no saber cap on tirar.

No totes les funcionalitats de l'API estan implementades dins la llibreria.

3.3.4. Limitacions a tenir en compte. JSF i la llibreria de Google Maps per JSF.

Amb la llibreria Gmaps no disposem al 100% del que podríem arribar a fer explotant directament el mapa amb l'API de Google Maps. És una limitació molt important que em de tenir en compte a l'hora d'escollir amb quina tecnologia volem desenvolupar el nostre projecte. Per tant haurem d'estudiar bé el que volem fer i si amb aquesta llibreria en tenim suficient.

3.4. JSF visual web amb components Google Maps.

3.4.1. Característiques principals. JSF visual web amb components Google Maps.

En la versió del Netbeans 5.5 i el visual web pack instal·lat tenim l'opció de treballar amb el mapa de Google Maps directament com un element de la paleta, de manera que es molt fàcil d'implementar. Això és possible gràcies al component BluePrints Ajax Component Library.

Dins la paleta tenim una secció anomenada BluePrints AJAX Suport Beans amb una sèrie de components entre ells trobem el mapa de Google Maps, servei de geocoder (localització dins el mapa), punt geogràfic, etc..

L'avantatge és que es treballa tot de forma visual, s'arrastra el component de Google Maps sobre l'espai de treball i ja tenim el mapa. Ara només s'ha d'anar a la pàgina de Google Maps, aconseguir la "key" i posar-la dins les propietats. D'aquesta manera tant senzilla tenim el mapa visible dins la nostra aplicació. En el supòsit que volguéssim fer una aplicació on l'objectiu és buscar una ciutat o un lloc dins el mapa hauríem d'arrastrar el geocoder dins la pàgina de disseny a l'acció del botó hem de passar-li el texfield a l'objecte geocode.

3.4.2. Configuració. JSF visual web amb components Google Maps.

La configuració és senzilla i ràpida però només funciona a la versió 5.5 del Netbeans, per versions superiors aquests components no estan disponibles.

L'únic que hem de fer és instal·lar el visual web pack, fer un update i seleccionar el component BluePrints de la llista de disponibles.

Reiniciem l'IDE del Netbeans i ja tenim disponibles els objectes a la nostra paleta.

3.4.3. Avantatges i inconvenients. JSF visual web amb components Google Maps.

L'avantatge principal és la facilitat i la rapidesa amb la que podem crear una aplicació senzilla que utilitzi Google Maps.

Un altre avantatge és que podem modificar les propietats del mapa a través del menú de propietats un cop tenim seleccionat el component.

Podem crear un mapa sense la necessitat de manipular res del codi.

Veiem el component gràficament dins el Netbeans, això ens permet posicionar-lo al lloc que vulguem i fer un disseny adequat de la pàgina.

Per contra tenim que només serveix per una versió obsoleta com és la 5.5. Actualment es treballa amb la 6.5 que és la última versió més estable.

El component BluePrints no ha tingut una continuïtat dins Netbeans; per tant les seves funcionalitats son limitades.

Només podem crear un mapa de forma visual, l'altre opció que tenim és crear un geocoder que s'encarregarà de fer les localitzacions dins el mapa; si volem fer una altre aplicació que dibuixi polilínies o polígons hauríem de buscar alguna altre forma de fer-ho i no ens serviria el component, de manera que tot el que hem guanyat en rapidesa inicial després ho perdríem.

3.4.4. Limitacions a tenir en compte. JSF visual web amb components Google Maps.

Únicament serveix per Netbeans 5.5 i el component ens mostra el mapa i un localitzador.

Per altres possibles aplicacions el cost d'investigació és més gran apart de no tenir garantia d'èxit.

3.5.- JSF visual web amb components ICEFaces.

3.5.1. Característiques principals. JSF visual web amb components ICEFaces.

ICEFaces ha desenvolupat una col·lecció de components basades en JavaServer Faces (JSF) aporten al Netbeans el concepte de (RIA) Rich Internet Application, faciliten sobretot el treball a temps real ja que utilitzen la tecnologia AJAX Push.

Des de la versió 1.7. de ICEFaces s'integra perfectament amb Netbeans. Al mateix temps ICEFaces també ha desenvolupat altres frameworks per altres IDE's com ara Eclipse.

La versió que he provat ha estat ICEFaces 1.7.2-SP1 que s'integra perfectament a la versió 6.5 del Netbeans [14].

La particularitat d'aquest marc de treball es que incorpora un mapa de google, a part dels típics components.

3.5.2. Configuració. JSF visual web amb components ICEFaces.

Per tal d'aconseguir El marc de treball ens dirigim a la pàgina de ICEFaces i ens descarreguem el mòdul per Netbeans 6.5 el descomprimim i des de Netbeans importem el pluguin, reiniciem l'IDE i ja tenim la paleta d'objectes, El mapa el podem veure des d' un projecte visual web a nivell gràfic dins de l'IDE.

És fàcil, ràpid i còmode incorporar el Google Maps dins la nostra aplicació, tot i que no té una paleta de propietats on poguem posar valor com la Key cosa que si teníem en l'opció BluePrints.

Ara per ara hi ha molt poca documentació de les possibilitats que ens ofereix, hi ha pocs exemples, per tant treballar amb aquest component comporta experimentar-hi i dedicar-hi hores. Per altra banda es previsible que s'incorporin noves funcionalitats i, a priori no es previsible que s'extingeixi aquest component si que tingui una continuïtat i una evolució i que no passi com el blueprints que només es va desenvolupar per una versió del Netbeans i no ha evolucionat més..

3.5.3. Avantatges i inconvenients. JSF visual web amb components ICEFaces.

El principal avantatge de ICEFaces és igual que el component de BluePrints. ICEFaces s'integra dins Netbeans sense donar cap tipus d'error i crea un mapa on només cal arrastrar el component dins el nostre espai de treball.

Serveix per les versions més actuals de Netbeans i és una llibreria que està en evolució, per tant es previsible que surtin noves versions actualitzades.

Es pot concloure, que en un futur serà una bona opció per treballar de forma visual des de Netbeans tot el que siguin aplicacions basades en Google Maps, existeix un acord de col·laboració entre Netbeans i ICEFaces.

Com inconvenients a data d'avui hi ha pocs projectes desenvolupats amb l'objecte de Google Maps que incorpora ICEFaces, hi ha poca documentació inclús en la pròpia pàgina del proveïdor.

Per tant en cas de trobar-te encallat en algun punt és difícil tirar endavant, doncs tens pocs punts de referència on consultar els dubtes.

3.5.4. Limitacions a tenir en compte. JSF visual web amb components ICEFaces.

La principal limitació és la falta de documentació del component, tot i que és previsible i d'esperar que estigui disponible properament.

ICEFaces està encara força verge, per tant és difícil de trobar manuals, tutorials o exemples que ajudin una mica.

Per tant és necessària una fase d'investigació i desenvolupament més gran doncs requerirà de certa experimentació abans d'aconseguir el resultat final desitjat.

4.- Aplicació

4.1. Objectius

En l'aplicació s'ha realitzat amb les tecnologies estudiades, per tant el projecte consisteix en la combinació de dues tecnologies: Visual JSF i la integració de Google Maps, tal com s'ha vist a l' estudi tenim vàries opcions per treballar amb Google Maps, per tant s'ha fet una tria sospesant els punts a favor i els punts en contra.

S'ha apostat per una tecnologia de futur com és la combinació del visual web JSF amb la paleta de ICEFaces que tal com hem vist integra perfectament l'eina de Google Maps dins el Netbeans.

L'aplicació la podríem separar en dues parts:

Una part consistiria en realitzar la implementació del registre d'usuari on hi ha les operacions d'inserció, modificació i esborrat de les dades a la base de dades, un cop l'usuari està registrat té l'opció de : visualitzar el mapa de Google Maps i fer cerques on apareix un punter que mostra l'ubicació de la cerca.

La idea ha estat crear una aplicació on un usuari es pugui donar d'alta mitjançant un registre; autenticar-se, modificar les seves dades si ho troba necessari fins i tot donar-se de baixa. Tota aquesta part s'ha desenvolupat amb visual JSF.

Per explorar una mica la potència del ICEFaces s'ha implementat un mapa que permet fer una cerca d'un país, adreça o ciutat que vulguem.

4.2. Tecnologia i metodologia

S'ha utilitzat com eina de desenvolupament el Netbeans en la seva versió 6.5.

El projecte s'ha realitzat amb visual web JSF i ICEFaces. D'aquesta manera s'ha aconseguit treballar amb Google Maps i crear tota una interfície i disseny única i exclusivament utilitzant el Netbeans, doncs amb el Visual web integra les seves pròpies CSS. Únicament s'ha utilitzat Illustrator CS3 pel disseny la maquetació ha estat realitzada amb el Netbeans.

4.3. Planificació i pressupost

Es realitza una planificació del temps, dividida on es destinen 200 hores d'estudi i investigació pràctica i altres 300 hores pel desenvolupament de l'aplicació. Donat que el projecte ha evolucionat durant 4 mesos ha suposat aproximadament una dedicació de 4 hores diàries de mitja.

Es comptabilitzen al voltant de unes 500 hores, de les quals aproximadament unes 200 han estat destinades al procés d'investigació i estudi, amb les altres 300h s'ha desenvolupat l'aplicació i la documentació del projecte.

Donat que el cost d'hores d'estudis té un cost substancialment més baix que les hores de desenvolupament el pressupost fina queda de la següent forma:

Cost del Estudi Java i Google Maps (200h)	4000.00€
Desenvolupament d'apliació (300h)	9000.00€

Total: 13000.00€

16% IVA: 2080.00€

Cost total: 15080.00€

4.4. Anàlisi de requeriments

Els requeriments funcionals d'aquesta aplicació són els següents:

L'usuari, per poder accedir als contingut de l'aplicació realitzada en Google Maps, haurà de completar un formulari amb unes dades bàsiques, de les quals unes seran obligatòries.

Un cop realitzat el registre introduint el seu usuari i contrasenya, podrà autenticar-se i d'aquesta forma accedir als continguts de l'aplicació.

Un cop l'usuari és autenticat per l'aplicació podrà realitzar les següents operacions:

Modificar el seu perfil, si ho considera pertinent.

Donar-se de baixa com usuari de l'aplicació, si ho desitja.

Realitzar consultes d'ubicacions en un mapa que implementa la tecnologia de Google Maps.

Després de l'ús de l'aplicació l'usuari tindrà l'opció de sortir de l'aplicació.

Els requeriments no funcionals d'aquesta aplicació són els següents:

L'usuari que no estigui registrat a l'aplicació no tindrà accés a realitzar consultes en el mapa, és a dir no podrà interrogar l'API de Google Maps sobre la situació de ciutats, pobles, direccions, etc...

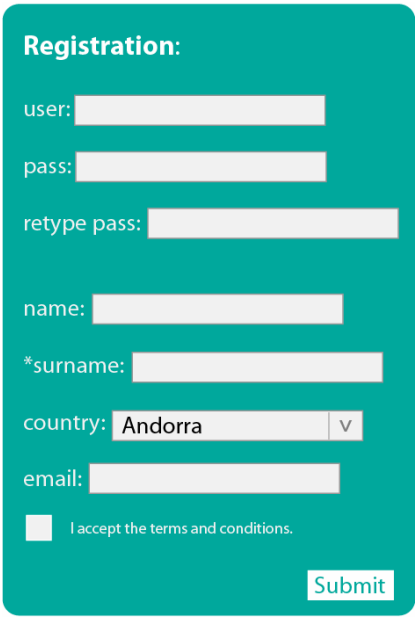
4.5. Casos d'ús

Gestió d'usuari:

Alta d'usuari:

L'usuari s'haurà de registrar a través d'un formulari que inclourà els següents camps:

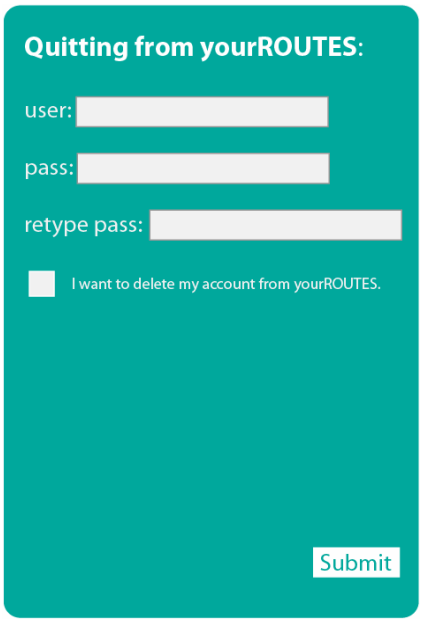
- User: l'usuari introduirà el seu nick en aquest camp, serà el nom amb el que voldrà iniciar la sessió a la web, els users no es poden estar repetits, es tindrà en compte si es tracta de majúscules o minúscules.
- Password: com a mínim haurà de constar de 6 lletres i es recomana que contingui majúscules minúscules i números per un tema de seguretat. Es demanarà a l'usuari que l'escrigui dos cops per tal d'evitar escriure errors a la BBDD, una funció en Javascript s'ocuparà de validar que els dos camps siguin iguals. En cas contrari mostrarà un missatge d'error indicant que els camps no són iguals i la necessitat de reescriure'ls un altre cop.
- Name: l'usuari introduirà el seu nom.
- Surnames: introduirà el seu cognom/s. És tracta d'un camp opcional.
- Country: en aquest camp el formulari disposarà d'un "combo box"; desplegable que inclourà el llistat de països, d'aquesta manera ens estalviem errors i evitem tenir països repetits; uns amb minúscules i d'altres amb majúscules.
- Email: El usuari haurà d'escriure el seu correu, es necessari que estigui actiu doncs el registre no finalitzarà fins que l'usuari entri dins la seva compte de correu i faci "click" sobre un enllaç de la web. Això ens permet verificar que el correu és correcte.



Vista del formulari de registre.

Baixa d'usuari:

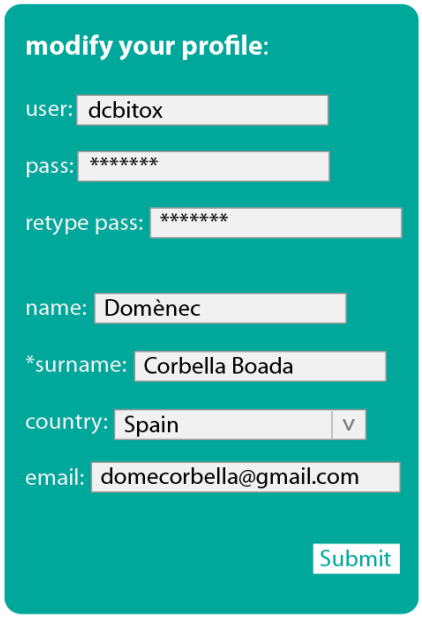
Pel cas d'ús de baixa d'usuari serà una opció que trobarà l'usuari dins el menú un cop aquest s'hagi identificat dins l'aplicació. Un cop faci "click" a donar-se de baixa trobarà un formulari on li tornarà a demanar el nom d'usuari i contrasenya i un "check box" una casella a marcar amb l'advertiment: està segur que es vol donar de baixa de yourruotes? Si l'usuari "click" el botó de "enter" el sistema donarà de baixa aquell usuari i es perdran totes les rutes que hagi pogut crear.



Vista del formulari de baixa d'usuari.

Modificació de les dades d'usuari:

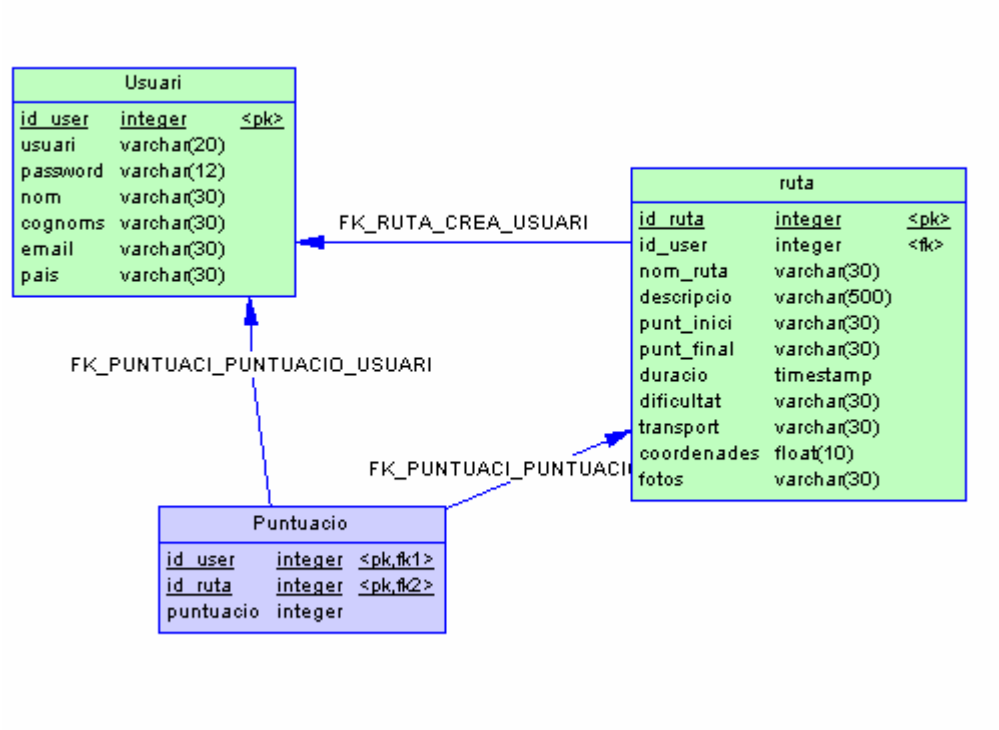
Aquesta opció igual que en cas de baixa d'usuari, es trobarà un cop l'usuari s'hagi identificat. Es mostrarà un formulari igual que el del registre però amb les dades corresponents al usuari que va omplir en el dia del registre; un cop allà l'usuari podrà fer les modificacions que vulgui. Un cop accepti igual que en el cas de registre es verificarà que la contrasenya sigui la mateixa i tingui un mínim de 6 lletres i que el camp user en cas d'estar modificat no correspongui al mateix d'un altre usuari.



Vista del formulari de modificació de dades

4.6. Disseny

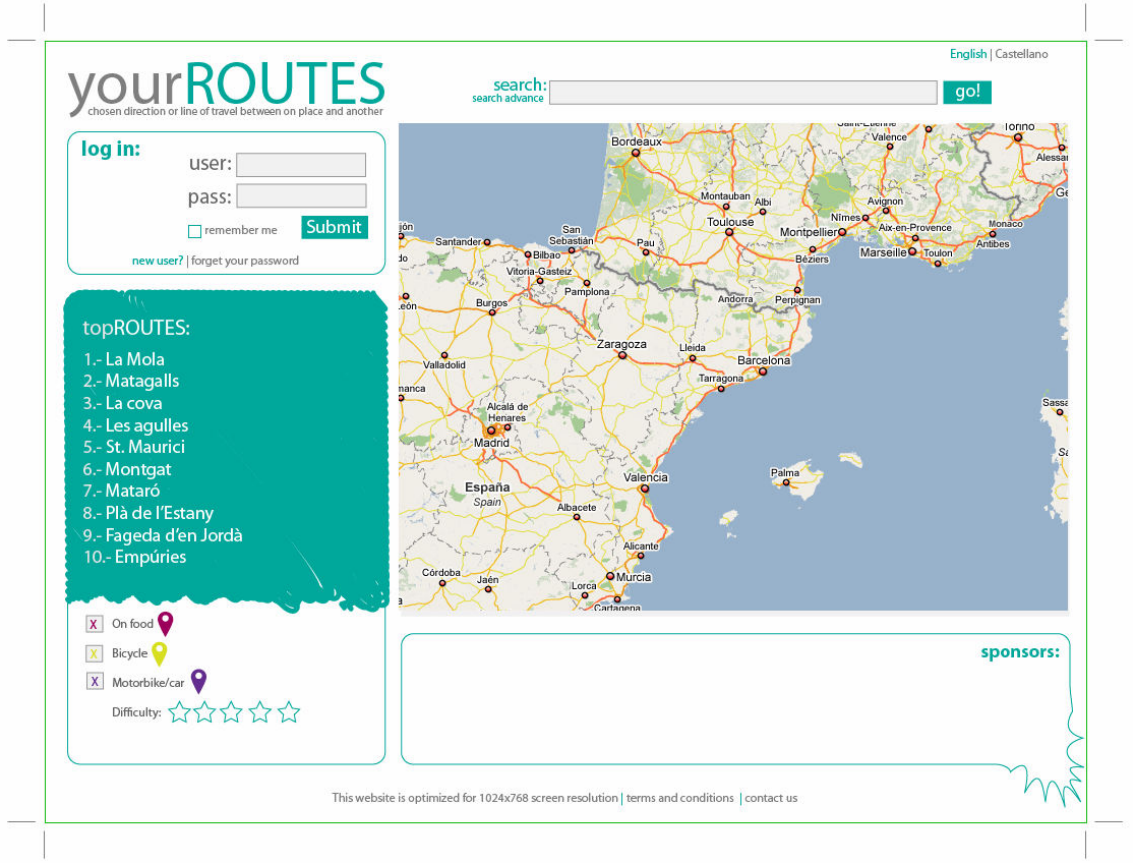
4.6.1. Disseny base de dades

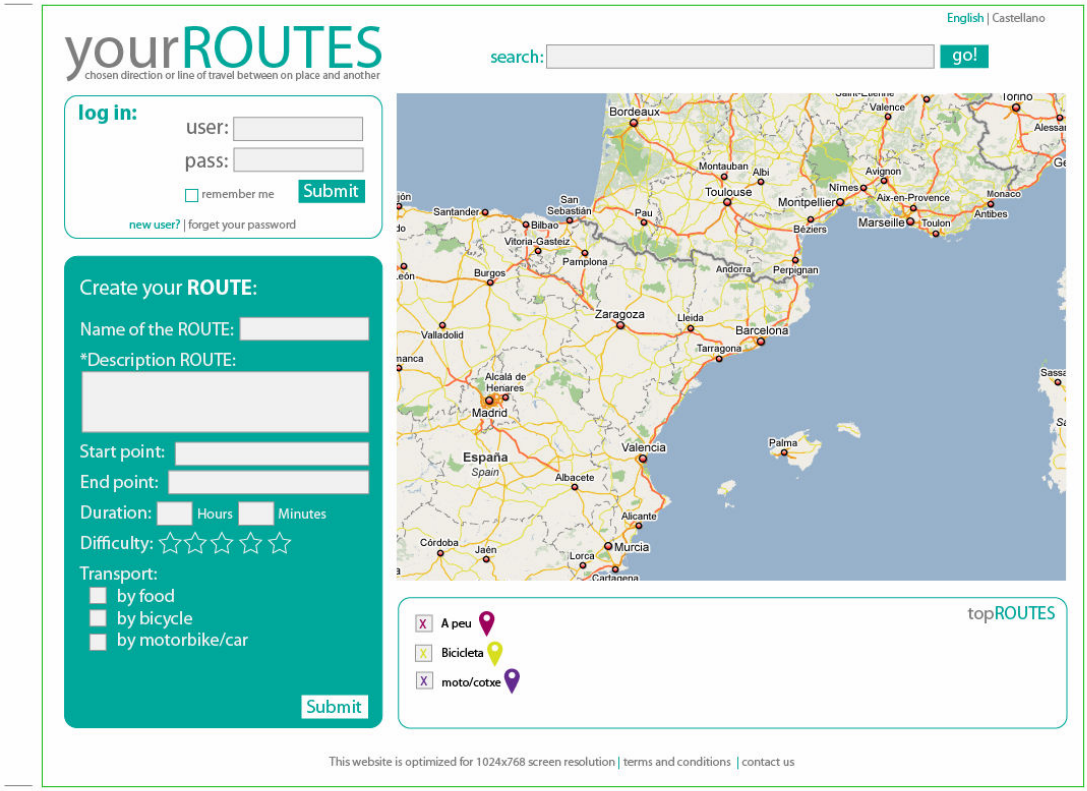


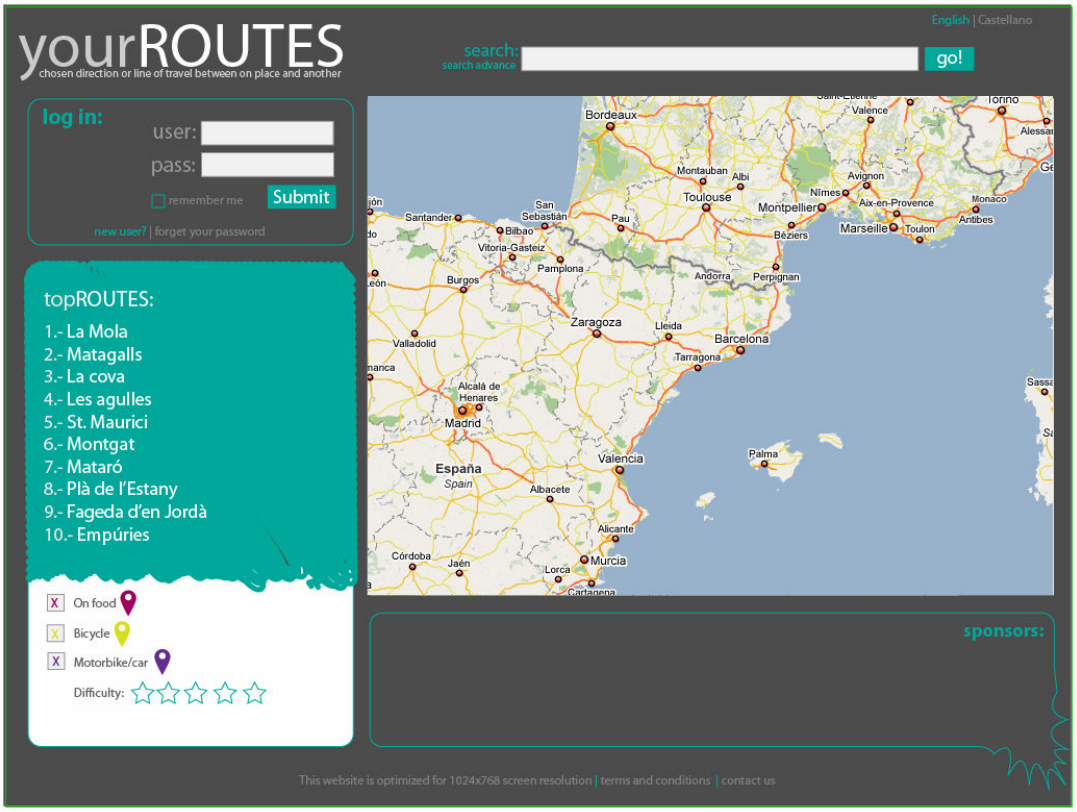
Esquema físic de l'estructura de la base de dades.

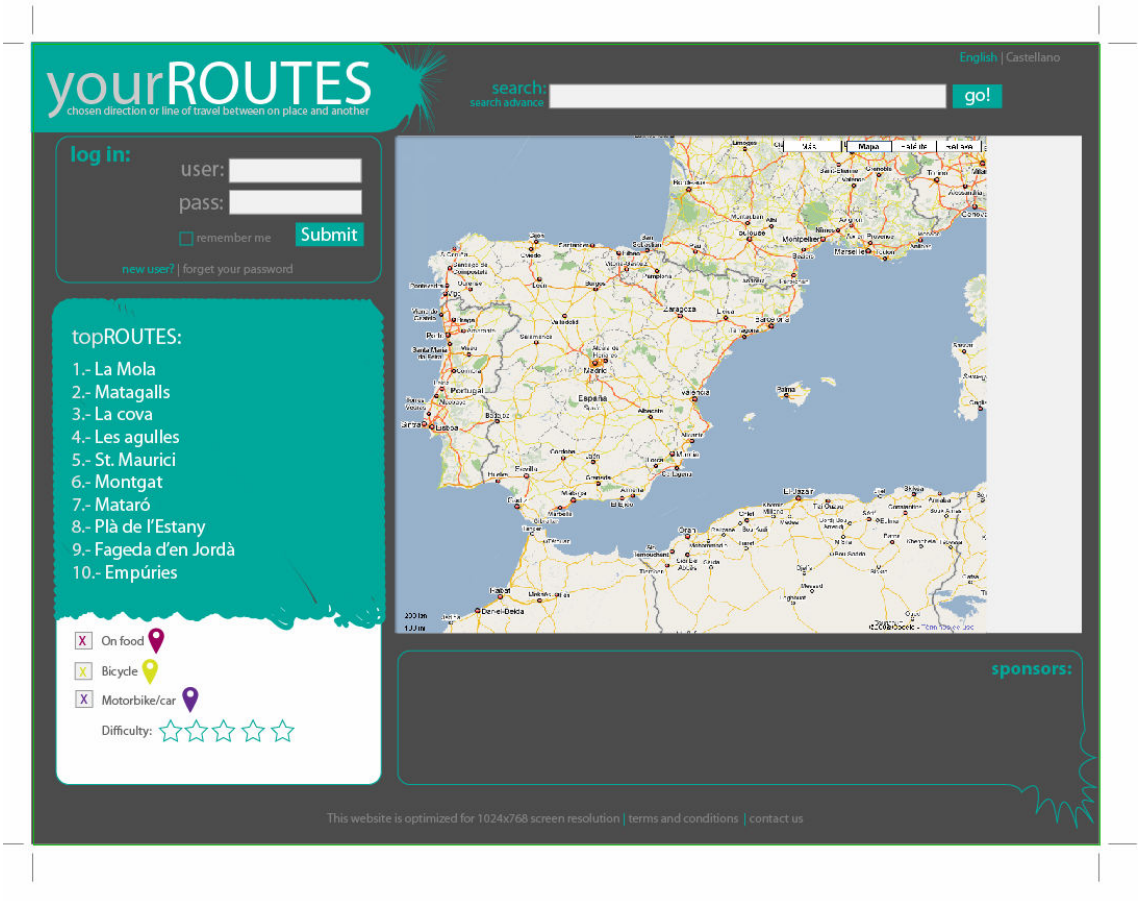
4.6.2. Disseny de l'interfície

A continuació es mostren diferents propostes de disseny:



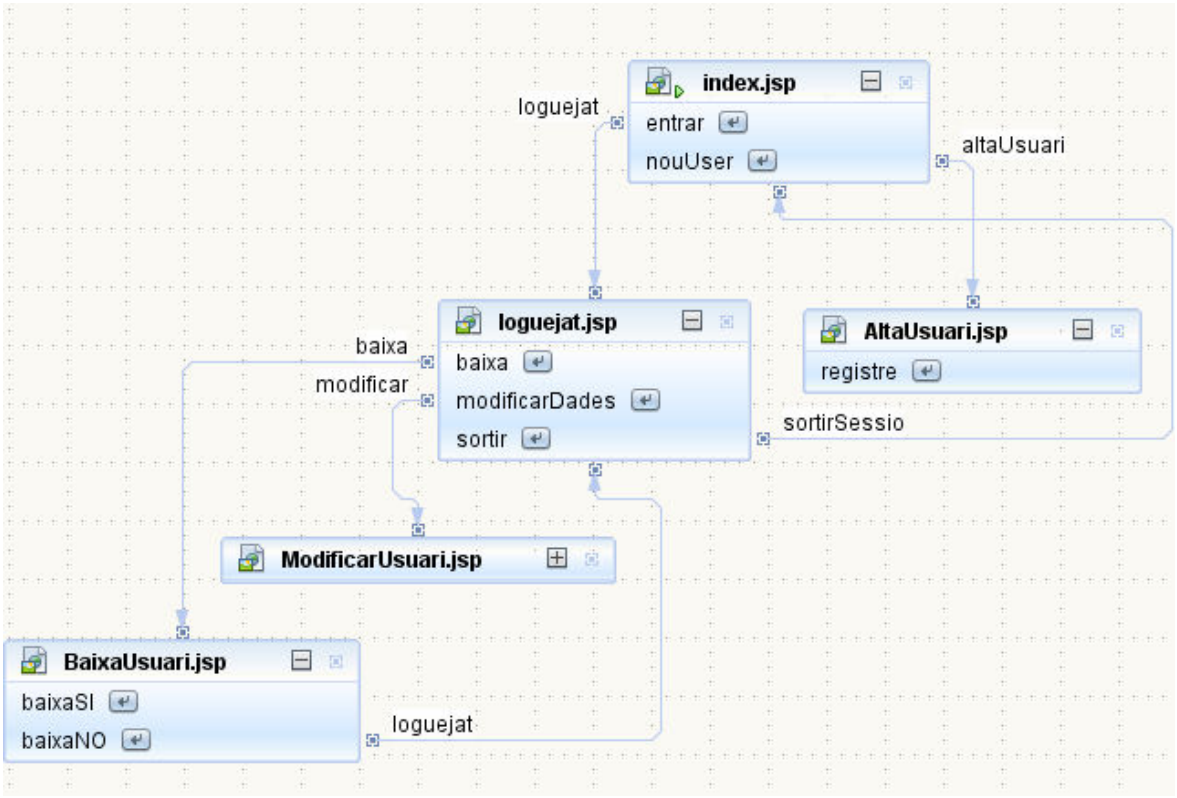

yourROUTES
chosen direction or line of travel between on place and another
English | Castellano
search:
go!
log in:
user:
pass:
remember me
Submit
new user? | forget your password
Create your ROUTE:
Name of the ROUTE:
*Description ROUTE:
Start point:
End point:
Duration: Hours Minutes
Difficulty:
Transport:
by food
by bicycle
by motorbike/car
Submit
topROUTES
A peu
Bicicleta
moto/cotxe
This website is optimized for 1024x768 screen resolution | terms and conditions | contact us


yourROUTES
chosen direction or line of travel between on place and another
English | Castellano
search:
search advance
go!
log in:
user:
pass:
remember me
Submit
new user? | forget your password
topROUTES:
1.- La Mola
2.- Matagalls
3.- La cova
4.- Les agulles
5.- St. Maurici
6.- Montgat
7.- Mataró
8.- Plà de l'Estany
9.- Fageda d'en Jordà
10.- Empúries
On food
Bicycle
Motorbike/car
Difficulty:
sponsors:
This website is optimized for 1024x768 screen resolution | terms and conditions | contact us



La versió definitiva és la superior realitzada sobre un fons blanc enlloc de gris fosc.

4.6.3. Graf de navegació



Relació de pàgines que entren en interacció dins el projecte.

4.6.4 Disseny del logotip

A continuació es mostren diferents propostes del disseny del logotip, finalment hem escollit l'últim de tots, és el que dona més dinamisme i es relaciona millor el concepte de mobilitat que volem transmetre a la nostra pàgina.

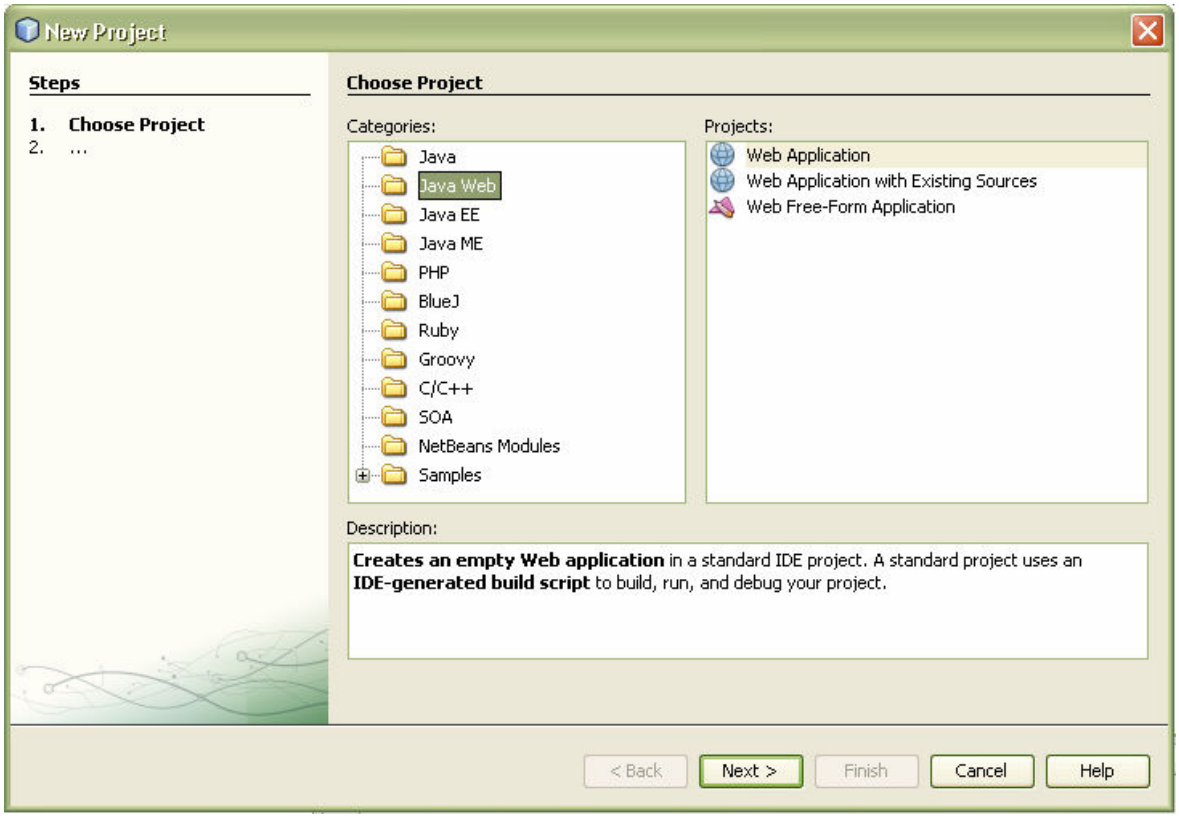
5.- Desenvolupament

Per desenvolupar aquesta aplicació s'ha utilitzat el següent software:

- Netbeans 6.5.
- ICEfaces-1.7.2-SP1-NetBeans-6.5-modules.
- Servidor MySQL.
- MySQL Administrator.
- Power designer (disseny conceptual de la BBDD).
- Adobe Illustrator CS3. (desenvolupament del disseny).

Un cop tenim el Netbeans 6.5 i el servidor MySQL instal·lats el següent que farem serà importar els components ICEfaces al Netbeans i arrancar els serveis.

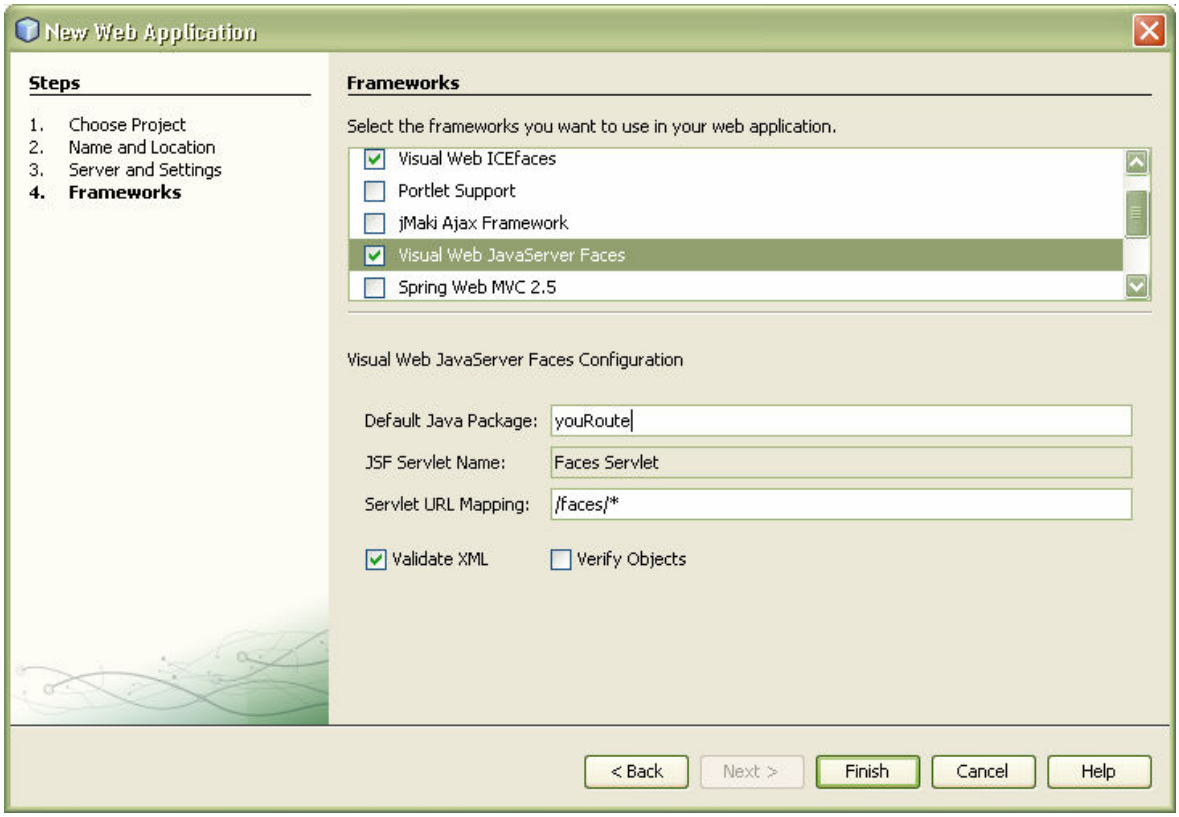
Arribat aquest punt crearem el nostre projecte; farem un new-project i escollirem dins la categoria de Java Web, el projecte Web Application.



A continuació posarem un nom al nostre projecte i a la següent finestra escollirem el servidor: en el cas d'aquesta aplicació s'ha escollit l'Apache, (però també es podria haver escollit el servidor Glassfish, s'ha de tenir en compte que si agafem aquesta última opció a l'hora de compilar el nostre projecte el procés de compilació serà més llarg d'efectuar.).

Finalment, arribats a la darrera pestanya escollirem el marc de treball amb el qual volem treballar, serà el moment d'escollir:

Visual Web ICEfaces i Visual Web JavaServer Faces

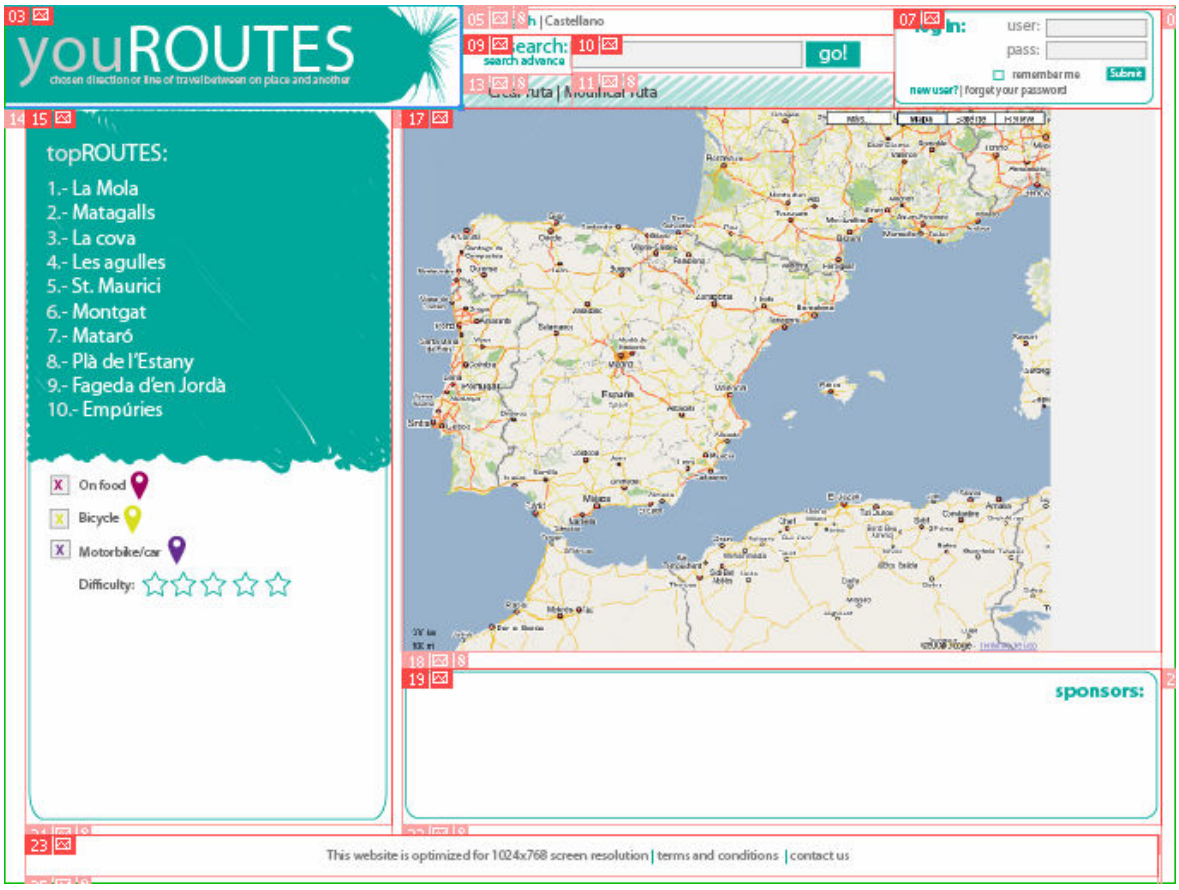


Quan creem un projecte podem escollir tants marcs de treball com necessitem, però seria un absurd seleccionar-los tots. El resultat seria que el nostre projecte pesaria molt.

Ja podem posar-nos a desenvolupar l'aplicació.

Un cop tenim el disseny de la nostra pàgina realitzat ja sigui amb Adobe Photoshop o Adobe Illustrator, és el moment de diferenciar les zones de tall de la nostra pàgina. Creant les zones de tall, el que aconseguim, és exportar totes les imatges de la nostra pàgina per separat. La idea és crear un puzle amb diferents peces i més petites.

Un cop tenim les peces diferenciades serà el moment de tornar al Netbeans i agafar les imatges per composar la pàgina altre cop, però aquesta vegada ho combinarem amb els components, en molts casos ens pot anar bé situar l'imatge de fons i col·locar els components a sobre per ser el més fidels possible al disseny de l'interfície.



Imatge de l'interfície, on els requadres en vermell representen els talls.

Ara arriba el moment de composar la nostra pàgina amb el Netbeans:

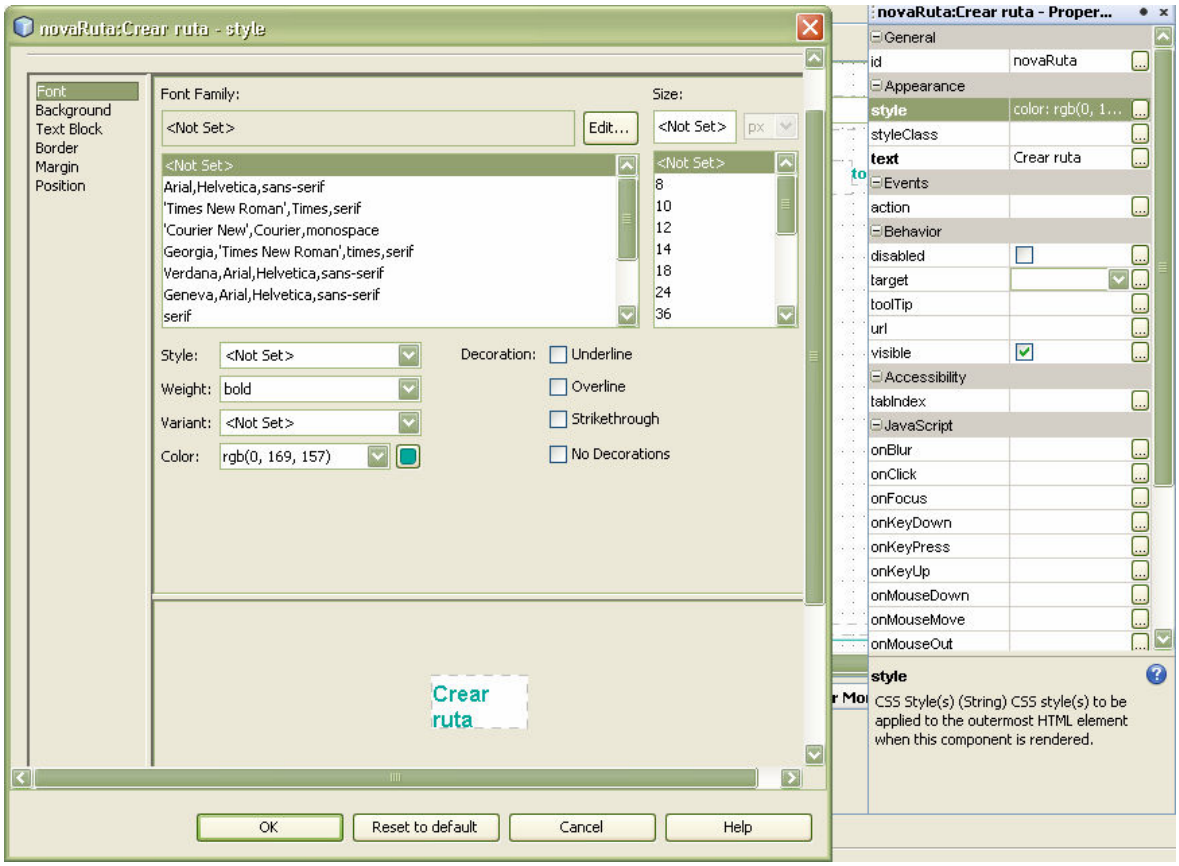
Utilitzarem la paleta d'objectes per anar inserint les imatges, els texts, la creació dels formularis, botons etc...

El que hem d'aconseguir és crear la pàgina amb Netbeans amb el mateix resultat de disseny que havíem obtingut amb l' Illustrator.

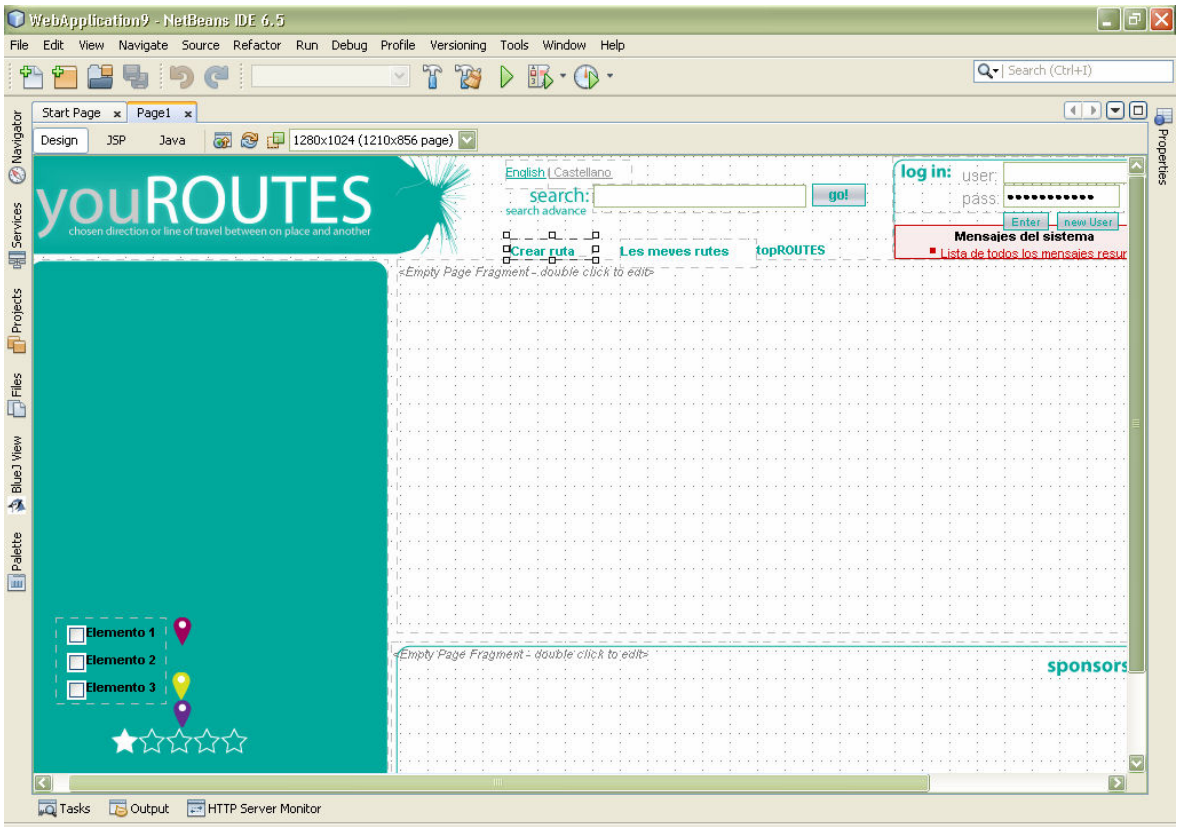
Un cop inserim qualsevol component en el nostre espai de treball, si seleccionem l'objecte i anem a la finestra de propietats visualitzarem una col·lecció de paràmetres que podrem configurar. Entre ells n'hi ha un de determinant per configurar l'aspecte. És dins la pestanya de "Apperance ➔ Style o Style class" Si cliquem se'ns obrirà una finestra on podrem configurar tot el relacionat amb l'aspecte del component. Està composat pels següents apartats: "font, blackground, text block, border, margin, position".

Si estem acostumats a treballar amb altres IDE's com ara el Adobe Dreamweaver notarem que l'edició de les CSS o estils igual de ben aconseguits. Possiblement el desenvolupament sigui més lent al principi però a mesura que ens hi anem familiaritzant serem capaços de donar l'aspecte que vulguem a la nostra web d'una manera molt ràpida.

L'avantatge de definir Style Class és que qualsevol modificació que fem sobre el Style Class es veurà reflexada sobre tots els components que el tinguin assignat, la qual cosa ens fa guanyar temps.



Finestra de estils.



Resultat del disseny implementat amb el Netbeans.

Un cop tenim la composició gràfica realitzada és el moment de programar els components per tal que interaccionin de la forma que nosaltres tenim prevista.

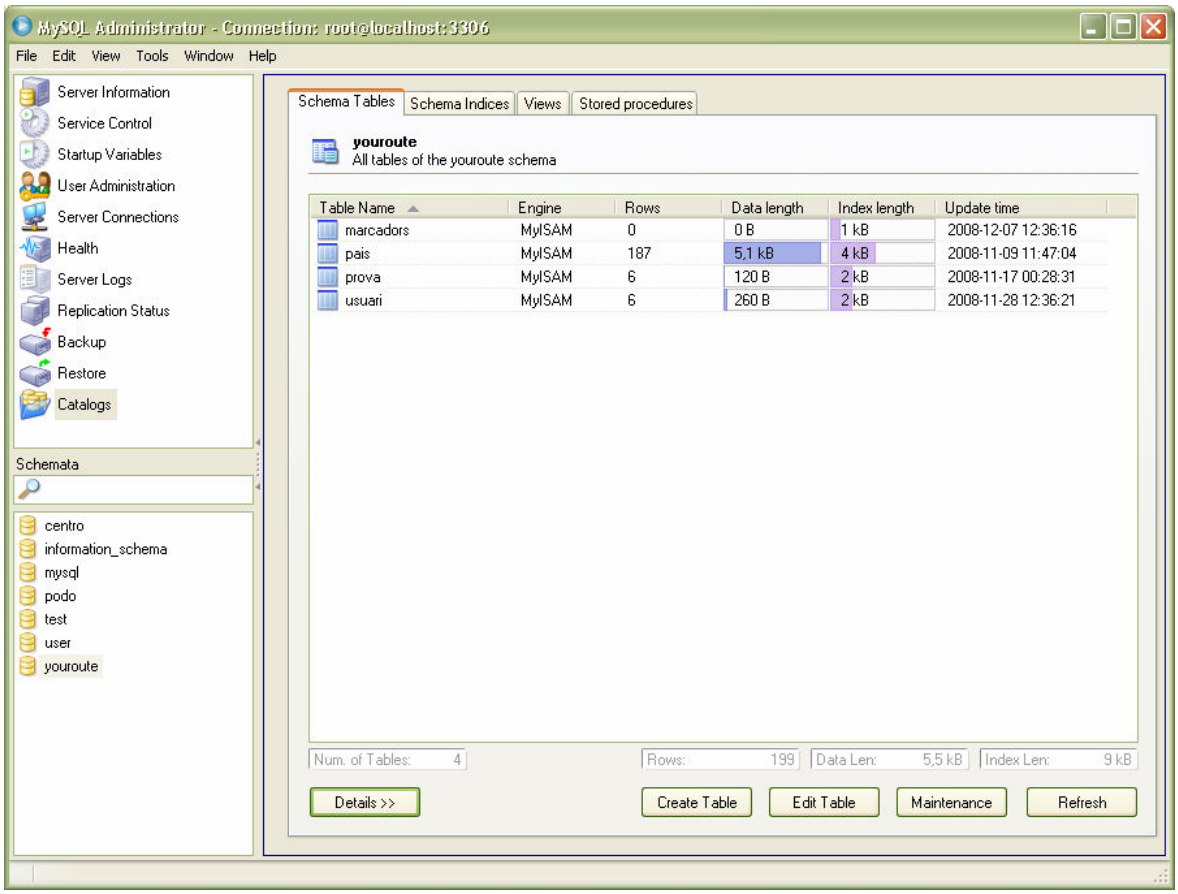
Però abans hem de remarcar que en el disseny de l'interfície s'ha utilitzat un sistema de capes que incorpora el Visual WEB JSF anomenat Visual web JSF page fragment (si ens fixen en la figura superior podem veure amb lletra gris dins un requadre puntejat gris el següent text : <Empty Page Fragment: double click to edit>), indica que es tracta d'una "page fragment".

Una de les avantatges importants de Visual web JSF page fragment és que ens permet alliberar de codi la pàgina. Doncs cada fragment es tracta com una nova pàgina, té els seu propi codi, estils, comportaments, etc. de forma que la nostra pàgina tal com la veiem és la composició de la suma de les diferents pàgines.

Una vegada hem composat la maquetació i tenim el disseny conceptual i físic de la base de dades. Crearem la base de dades amb els següents passos: arrancant el servidor de MySQL amb el EasyPHP i creant el nostre esquema i les taules de la base de dades amb el MySQL Administrator.

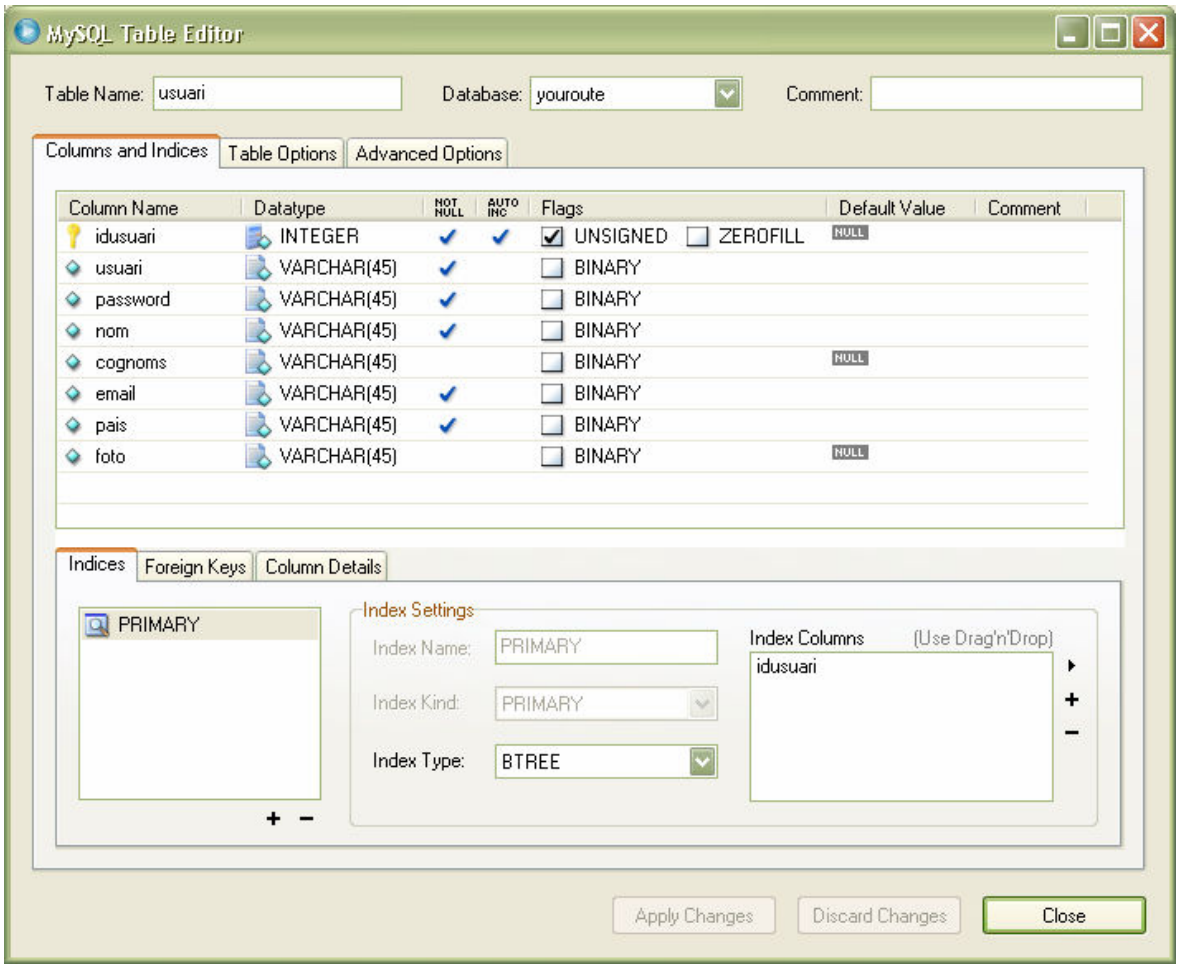
El MySQL Administrator disposa d'una interfície intuïtiva, ens permet crear backups amb molta facilitat. Es recomanable crear el disseny de les taules finals amb MySQL Administrator i no amb IDE's del tipus Power Designer, que poden estar molt bé per crear el disseny conceptual, però un cop fan l'exportació a MYSQL ens pot donar problemes.

L'esquema de la nostra aplicació té el següent aspecte:



Esquema de les taules vist amb MySQL Administrator.

Com es pot veure en la següent figura l'edició dels camps de les taules és força visual.

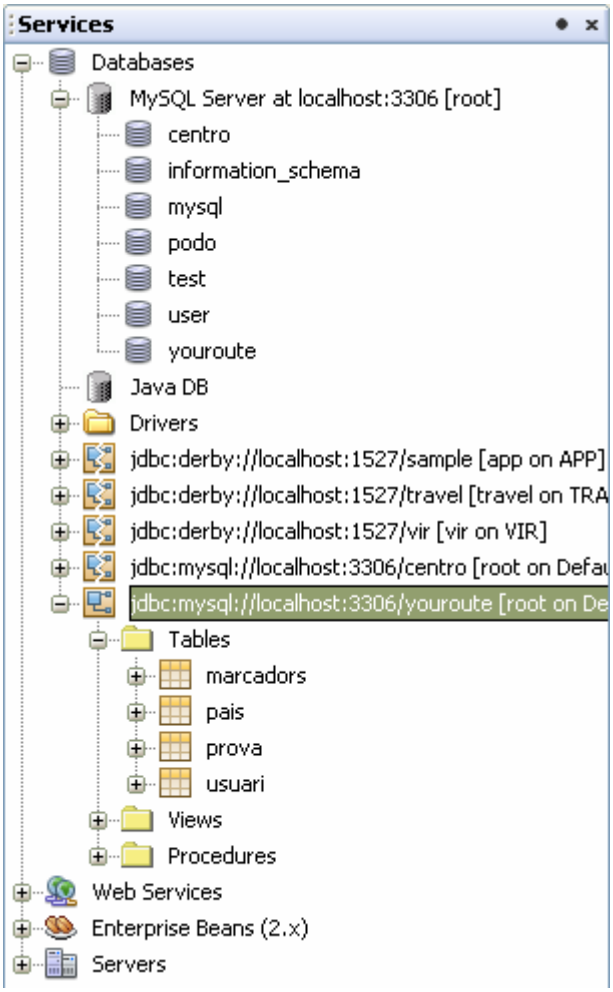


Vista del disseny de la taula d'usuari.

Un cop tenim creada la base de dades Netbeans se'ns dóna l'opció de visualitzar-la des del propi IDE, per fer-ho anirem a la pestanya de serveis i on posa base de dades crearem una nova connexió (botó dret).

Cal complimentar el tipus de connector: en el nostre cas el MYSQL (connector/Jdriver), host: localhost, port: (el que tinguem configurat en el servidor MySQL) en aquest cas era el 3306, "Database": posarem el nom de la base de dades i finalment l'usuari i la contrasenya. Introduïm els valors que tinguem configurats; en el nostre cas "root" com usuari i sense contrasenya.

Un cop creada la nova connexió ens hi podem connectar (botó dret) i veure les taules.



Vista de les taules des de Netbeans.

Visual web JSF en ocasions fa intuïtiu el sincronisme amb la base de dades i la capa de presentació, un exemple d'això és quan hem de mostrar els valors de la base de dades a la nostra web.

Visual web JSF per sí mateix s'ocupa automàticament de la persistència de la base de dades, de totes maneres es poden aplicar diferents patrons de disseny per tal de reforçar-ho, sobretot en casos on podem preveure que la nostra base de dades tindrà molta concurrència.

Un bon exemple del sincronisme de la base de dades amb la capa de presentació l'hem pogut veure en el moment de realitzar el formulari d'alta. Hi ha un camp desplegable on es mostra tot el llistat de països que es troben emmagatzemats a la base de dades.

D'aquesta manera aconseguim uns valors persistents per la base de dades; és a dir ens estalviem: que l'usuari ens pugui introduir a la nostra base de dades valors erronis gramaticalment parlant, països erronis o fins i tot inventats i al mateix temps ens assegurem que tots els valors seran iguals i ens estalviem la repetició de valors pel simple fet que uns van amb majúscules i els altres van en minúscules, es tractarà d'un camp que ja no haurem de validar perquè el facilitem als nostres usuaris de forma ja validada.

Per aconseguir que es mostrin els països dins el desplegable simplement cal arrastrar la taula de la base de dades sobre el component desplegable, automàticament ens apareixerà la llista. Ara només haurem de seleccionar el camp que ens interessi mostrar.

El resultat té un aspecte en JSP tal com aquest:

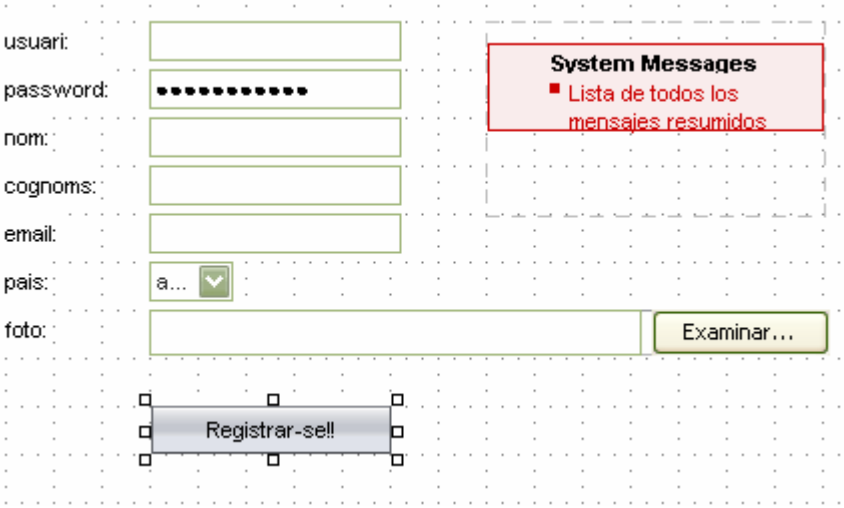
```
<webuijsf:dropDown //Indica el tipus de component; en aquest cas un desplegable.
binding="#{AltaUsuari.pais}" converter="#{AltaUsuari.paisConverter}" id="pais"
 items="#{AltaUsuari.paisDataProvider.options['pais.PAI_PK,pais.PAI_ISO2']}"   //ens
agafa del paisDataProvider el camp PAI_ISO2 que correspón al nom dels paissos. A la
base de dades guardarem la clau primària que és un número.
style="left: 120px; top: 216px; position: absolute"/> //especifiquem els estils de
disseny.
```

Igualment, en el supòsit que ens interessés mostrar una sèrie de resultats de la base de dades en una taula, el procediment seria el mateix: arrastrar la taula de la base de dades i deixar-la anar sobre el component taula, en la nostra pàgina de disseny del Netbeans.

5.1. Inserció a la base de dades:

En l'aplicació ens trobem amb aquesta necessitat en el moment que hem de donar d'alta a un nou usuari; creem el formulari amb els components, és recomanable posar el nom de cada component a la finestra de propietats; dins el camp id. D'aquesta manera tindrem associat cada component.

Un cop tenim disposats tots els components ens disposarem a realitzar la programació de botó de registre.



Esquema del formulari d'alta d'usuari.

Un cop obtinguem el comportament desitjat ens disposarem a integrar-lo amb el disseny conjunt de la pàgina.

Quan vulguem programar un component fem dos clics sobre aquest i automàticament el Netbeans ens traslladarà a la finestra de Java.

Per inserir dades a la base de dades cal: fer l'acció de "set" sobre el dataprovider, de l'usuari. Com a tècnica; si el que volem es crear un dataprovider de la taula d'usuari hem d'arrastrar la taula dins l'espai de treball i deixar-la anar en un espai en blanc, acte seguit veure'm a la finestra de navegació com se'ns crea el dataprovider de la taula que hem arrastrat.

El següent pas, és anar a la sessionBean i modificar la sentència SQL del rowset en qüestió, en aquest cas la de l'usuari. A la sentència canviarem el select per un update, delete o insert, segons es tracti d'una consulta, una modificació, un esborrat o una inserció a la base de dades.

Així doncs, tornem al nostre botó ens creem un "rowKey", l'obrim i inserim un per un tots els camps del formulari. Fem un "commit" i un "refresh". De forma que ens quedarà el següent codi:

```
public String registre_action() {
        try{
            RowKey rk = usuariDataProvider.appendRow();
            usuariDataProvider.setCursorRow(rk);
//a continuació fem els sets dels valors dels camps del formulari. Posem la parella
que fa referència a la BBDD i el getText del camp.
            usuariDataProvider.setValue("usuari.usuari", this.usuari.getText());
            usuariDataProvider.setValue("usuari.password", this.password.getText());
            usuariDataProvider.setValue("usuari.nom", this.nom.getText());
            usuariDataProvider.setValue("usuari.cognoms", this.cognoms.getText());
            usuariDataProvider.setValue("usuari.email", this.email.getText());
            usuariDataProvider.setValue("usuari.pais", this.pais.getSelected());
            usuariDataProvider.setValue("usuari.foto", this.foto.getText());
        try{
            usuariDataProvider.commitChanges();
//fem commit, sino resulta saltarà l'excepció.
            usuariDataProvider.refresh();

        } catch (DataProviderException dataProviderException) {
                    error("Commit error: "  + dataProviderException);
                }

            } catch (Exception e) {
                error("Cannot update with row key "  + e);
            }
        return null;
    }
```

Un cop ja estem registrats el següent pas lògic serà fer el login a l'aplicació.

5.2. Autenticació d'usuari a la base de dades:

El que hem de fer és capturar els camps de text de l'usuari, guardar-los en dues variables i buscar a la base de dades si existeix la parella usuari/contrasenya.

El codi del botó d'entrada és el següent:

```
public String entrar_action() {
//guardem a les variables user/pass els camps introduïts per l'usuari
        String user = (String)this.usuari.getText();
        String pass = (String)this.password.getText();

        String [] param = {"USUARI","PASSWORD"};
        Object [] values = {user, pass};
        RowKey row = this.usuariDataProvider.findFirst(param, values);
        if(row==null){
           error ("error autenticació");
           return null;
        }
//busquem en dataprovider el camp usuari que és diferent per cada usuari.
        getUsuariDataProvider().refresh();
        row = getUsuariDataProvider().findFirst(
                new String[] {"usuari.usuari"},
                new Object[] {user}
        );
        if (row == null) {
            error("No es pot fer login de user '" + user + "'. ");
            return null;
        }
//recuperem els valors de l'usuari i els guardem a la sessió per si s'han
d'utilitzar en un futur.
        getUsuariDataProvider().setCursorRow(row);
        getSessionBean1().setLogin(user);
        getSessionBean1().setContrasenya(pass);
        getSessionBean1().setNom(
                getUsuariDataProvider().getValue("usuari.nom")
                + " "
                + getUsuariDataProvider().getValue("usuari.cognoms")
                + " "
                + getUsuariDataProvider().getValue("usuari.email")
                + " "
                + getUsuariDataProvider().getValue("usuari.pais")


        );
//si tot anat correctament retornem loguejat.
        return "loguejat";
    }
```

Un cop ens hem identificat dins l'aplicatiu a nivell d'usuari tenim vàries opcions:

Una d'elles és la de donar-nos de baixa i l'altre és la de modificar les dades del nostre perfil.

5.3. Donar-se de baixa a la base de dades:

Donar-se de baixa, és l'acció que consisteix en esborrar de la base de dades les dades de l'usuari. Hem de recordar que l'usuari ja es troba autenticat a l'aplicació, per tant tenim a la sessió totes les dades de l'usuari que navega.

El mecanisme de donar-se de baixa a la base de dades serà semblant al d'entrada a la pàgina; és a dir: 1er. buscarem a la base de dades les tuples corresponents de l'usuari que tenim emmagatzemat a la sessió i un cop el tenim localitzat farem el "delete", el corresponent commit i refresh.

El codi del botó d'eliminar usuari queda de la següent forma:

```
public String baixaSI_action() {
    // TODO: Process the action. Return value is a navigation
    // case name where null will return to the same page.
  RowKey rowKey = getUsuariDataProvider().findFirst(
            new String[] {"usuari.usuari"},
            new Object[] {getSessionBean1().getLogin()}
    );
    getUsuariDataProvider().setCursorRow(rowKey);
    getUsuariDataProvider().removeRow(rowKey);
    getUsuariDataProvider().commitChanges();
    return null;
 }
```

Pel que respecte a l'usuari l'altre tasca que podria realitzar seria la de modificar les dades del seu perfil.

5.4. Modificació de dades dins la base de dades:

Aquesta acció és substancialment diferent però respon al mateix esquema: és a dir mostrar les dades de l'usuari que tenim a la sessió i un cop l'usuari hagi fet la correcció o modificació que ell consideri oportunes, tornar a emmagatzemar a la base de dades les modificacions realitzades, sobre la mateixa tupla; és a dir fer un update.

Per mostrar les dades de l'usuari es pot aprofitar el disseny del formulari d'alta. Però en aquest cas els camps mostrats seran dinàmics i no estàtics com passava a l'alta.

El codi dels camps seria el següent:

```
<webuijsf:textField binding="#{ModificarUsuari.nom}" id="nom" style="position: absolute;
left: 144px; top: 120px" text="#{ModificarUsuari.usuariDataProvider.value['usuari.NOM']}"/>
```

El codi del botó que s'encarrega de fer l'update es detalla a continuació:

```
String nomU = (String) this.nom.getText();
        String loginU = (String) this.usuari.getText();
        String cognomsU = (String) this.cognoms.getText();
        String emailU = (String) this.email.getText();
        RowKey rowKey;
        String pass = (String) this.password.getText();
        rowKey=getUsuariDataProvider().getCursorRow();
        rowKey = getUsuariDataProvider().findFirst(
                new String[] {"usuari.usuari"},
                new Object[] {getSessionBean1().getLogin()}
        );
        if (rowKey ==  null) {
            error("No es poden actualitzar les dades");
            return null;
        }

        getUsuariDataProvider().setValue("usuari.usuari", rowKey, loginU);
        getUsuariDataProvider().setValue("usuari.nom", rowKey, nomU);
        getUsuariDataProvider().setValue("usuari.cognoms", rowKey, cognomsU);
        getUsuariDataProvider().setValue("usuari.email", rowKey, emailU);
        getUsuariDataProvider().commitChanges();
        getUsuariDataProvider().refresh();
```

```
//un cop hem actualitzat les dades, actualitzem les dades a la sessió:

        getSessionBean1().setLogin(loginU);
        getSessionBean1().setContrasenya(pass);
        getSessionBean1().setNom(
                getUsuariDataProvider().getValue("usuari.nom")
                + " "
                + getUsuariDataProvider().getValue("usuari.cognoms")
                + " "
                + getUsuariDataProvider().getValue("usuari.email")
                + " "
                + getUsuariDataProvider().getValue("usuari.pais")
        );
        return null;
    }
```

Fins aquí s'ha detallat tot el que fa referència al cas d'ús de gestió d'usuari.

5.5. Integració de Google Maps:

Pel que respecte la integració de Google Maps dins la pàgina visual web JSF, s'ha portat a terme vàries proves, s'han provat totes les possibilitat estudiades, la limitació ha vingut donada pel visual web JSF. Si el que volem és seguir desenvolupant una aplicació 100% visual només tenim dues opcions:

1.- Utilitzar el component Blueprint i la versió 5.5 del Netbeans.

2.- Utilitzar els components ICEfaces i la versió 6.5 del Netbeans,és l'última, més estable.

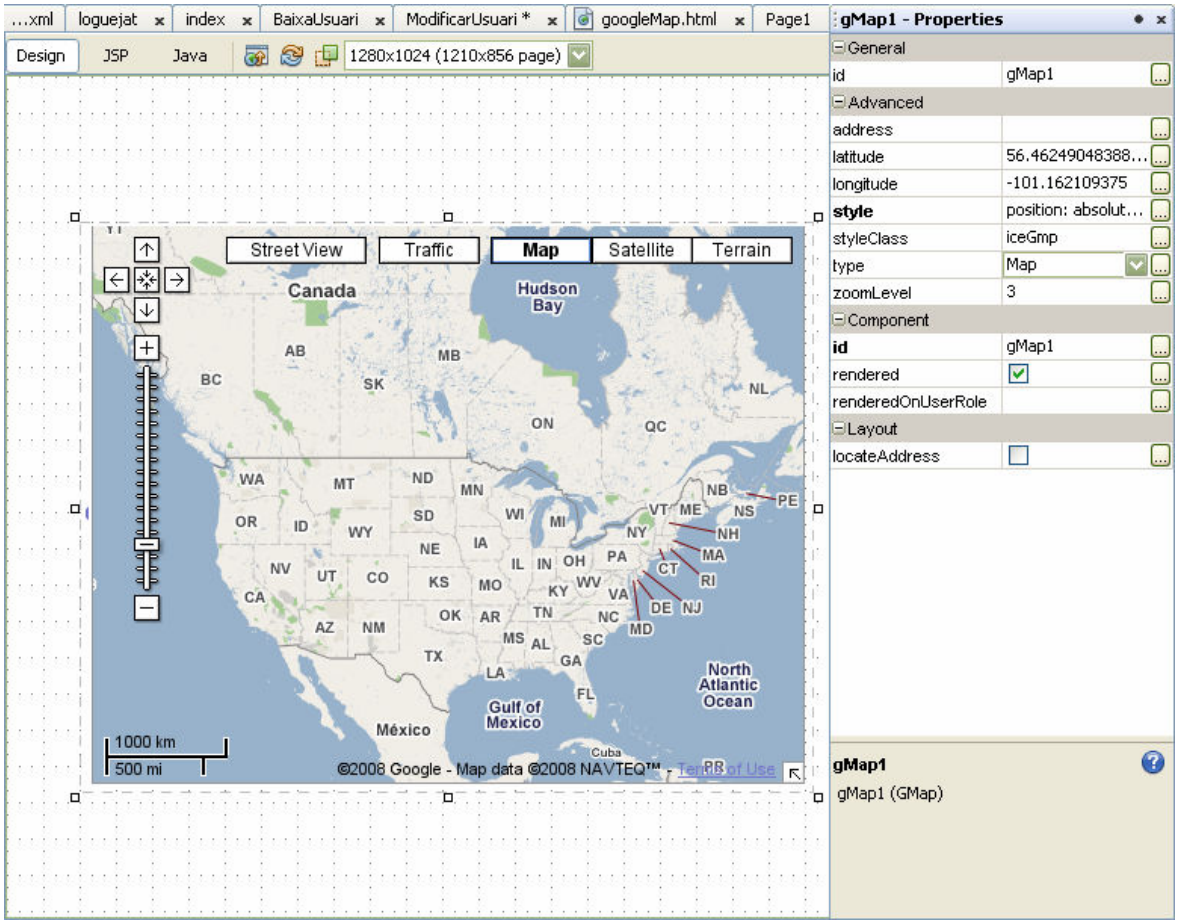
Les altres opcions de la integració de Google Maps en Java també ens permetran veure el mapa però no visualment dins el Netbeans, haurem de compilar el projecte perquè sigui visible.

Tant si treballem amb la llibreria gmaps2 o amb la llibreria JSF el codi l'haurem d'implementar dins la finestra de la pestanya JSP del Netbeans.

Finalment s'ha decidit seguir amb el disseny visual al 100%. Degut a diferents problemes de funcionament s'ha hagut de passar de la versió 6.01 del Netbeans a la 6.5, per tan s'ha desestimat la versió 5.5 i Blueprints. S'ha escollit implementar el Google Maps amb el component ICEFaces.

Per treballar el component Google Maps de ICEfaces, hem d'anar a la paleta i seleccionar el mapa de google, un cop seleccionat marquem un quadre dins l'àrea de treball, automàticament ens apareixerà el mapa amb les mides que hem dibuixat el requadre.

De manera que la vista que tindrem serà aquesta:



Vista del component de mapa Google Maps del ICEFaces al Netbeans.

Com es pot observar a la figura anterior a la finestra de propietats tenim una sèrie de paràmetre que es poden configurar com ara la latitud, longitud de la vista inicial, el zoom, el estil, la posició d'emplaçament entre d'altres.

El següent pas és obtenir la key de la pàgina de google i introduir-la de manera que el nostre mapa quedarà amb el següent codi:

```
<div style="-rave-layout: grid; width: 400px; height: 200px"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:ice="http://www.icesoft.com/icefaces/component">
    <f:subview id="map">
        <ice:gMap id="gMap1"
key="ABQIAAAA7PVwCvwFmVYlR6Y4cF97gRQRVgfb9GIaWGgwoK3g4v9ermARCRRlRH6MT66cOJ3_GbscFZl
MuUeddQ" style="position: absolute; left: 48px; top: 96px; width: 264px; height:
216px"/>
    </f:subview>
</div>
```

6.- Conclusions

De la realització d'aquest projecte se'n poden treure vàries conclusions:

1.- Google Maps és una tecnologia amb moltes possibilitats d'explotació, que està en constant evolució i que permet crear aplicacions WEB 2.0. és a dir pàgines web interactives on qui pren la iniciativa és l'usuari. Com ja comentat en el projecte a curt termini prendran molta força aquelles aplicacions que combinin serveis de GPS amb la potència de l'API de Google Maps. Degut, bàsicament a l'evolució de la tecnologia de la telefonia mòbil que té una clara tendència a l'integració de Internet i GPS en tots els terminals.

2.- Si el que volem és fer una aplicació que exploti al 100% els recursos de l'API el més pràctic serà treballar directament sobre l'API en Javascript però això suposarà deixar de banda el Visual web JSF i implicarà treballar amb JSP i Javascript.

3.- Si la nostra prioritat és treballar amb visual web JSF i hem de fer una aplicació bàsica on no s'hagin de fer gaires manipulacions o consultes al mapa, en aquest cas la solució òptima és la que hem utilitzat per desenvolupar aquesta aplicació visual web JSF i ICEFaces.

4.- Com a punt entremig entre la conclusió 2 i la conclusió 3, és a dir volem crear una aplicació que exploti els recursos de l'API a un 70% la millor solució suposaria altre cop deixar de banda el visual web JSF i desenvolupar l'aplicació amb JSP i la llibreria específica de google maps. De totes maneres abans seria convenient estudiar la llibreria per tal de veure si amb l'implementació actual seria suficient per cobrir els nostres requisits. L'avantatge principal és que no haurem de fer crides en Javascript, sino que treballarem amb la llibreria com si es tractés d'un objecte d'una classe.

En conclusió depenent del que vulguem desenvolupar escollirem una de les opcions, però tot el que suposi sortir-se de treballar amb l'API de google maps directament en Javascript suposarà davant qualsevol imprevist una dificultat afegida trobar-hi solució, degut a la falta de documentació i recursos existents, fet que amb el temps de ben segur que quedarà pal·liés.

7.- Annex I Cd de l'aplicació

8.- URLgrafia

Les pàgines web consultades són les següents:

[1].- http://en.wikipedia.org/wiki/Virtual_tour

[2].- http://es.wikipedia.org/wiki/JavaScript

[3].-http://es.wikipedia.org/wiki/PNG

[4].- http://es.wikipedia.org/wiki/AJAX

[5].- http://es.wikipedia.org/wiki/Adobe_Flash

[6].- http://es.wikipedia.org/wiki/ActionScript

[7].- http://googlemapsmania.blogspot.com

[8].- http://code.google.com/apis/maps/signup.html (aconseguim la key de google).

[9].- http://code.google.com/intl/es/apis/maps/documentation/reference.html#Events_GMap //(Events GMap2).

[10].- http://es.wikipedia.org/wiki/DOM

[11].-http://code.google.com/intl/es/apis/maps/documentation/polylinealgorithm.html (codificacio linia).

[12].- http://www.adobe.com/es/products/flashplayer/

[13].- http://www.lamatek.com/GoogleMaps/

[14].-http://component-showcase.icefaces.org/