Introducció als Sistemes de Informació Geogràfica

Joan Manuel González Febrer
ETIG
Jordi Ferrer Duran
18/06/2004

Introducció als Sistemes de Informació Geogràfica

Resum

Aquest document ens aproxima als sistemes d'informació geogràfica donant una definició del que fa, descrivint que són els objectes geogràfics, entitats amb informació geomètrica i temàtica així com la problemàtica de la seva representació. També ens mostra quines són les passes a fer en un sistema d'informació geogràfica, des de l'introducció de les dades fins a la presentació de resultats, passant per la consulta i l'anàlisi. El document presenta algun que altre exemple en concret per facilitar la comprensió. No tracta ni de les possibles anàlisis que es poden fer ni tampoc del tractament gràfic dels objectes geogràfics. En la segona part es planteja un projecte en concret com és el cas de la distribució d'aigua potable. Es fa un anàlisi exhaustiu del cicle de l'aigua urbana i es detallen els elements, el que abans havíem dit objectes geogràfics, per passar llavors a situacions que poden ser resoltes per un sistema d'informació geogràfica. Finalment es presenta un petit programari en fase inicial amb un objectiu totalment didàctic i sense cap interès que no sigui posar en pràctica els coneixement adquirits.

Introducció

Justificació del TFC i context en el qual es desenvolupa: punt de partida i aportació del TFC

L'ús dels GIS ha anat evolucionant molt ràpidament en els darrers anys i és una eina
molt utilitzada en diferents àmbits: gestió cadastral i tributària, localització de serveis
públics, cartografia,....
El treball estarà format per tres parts: una teòrica , una pràctica i una addicional també pràctica però de naturalesa diferent a l'anterior. En la part teòrica del treball es fa una descripció de les característiques fonamentals d'un GIS i de les seves principals aplicacions.
En la primera part pràctica es planteja un projecte GIS on es defineix l'estructura de la base de dades d'informació d'elements , en concret sobre la representació de la
xarxa pública d'aigua d'una població qualsevol.
En la tercera part s'intenta posar en pràctica aquelles funcionalitats que ha de fer un programari gis.

Objectius

Conèixer les característiques fonamentals dels Sistemes d'Informació Geogràfica.
Saber plantejar un projecte GIS.
Saber estructurar la base de dades d'informació no geogràfica i la seva connexió amb les dades geogràfiques.
Tractar la problematica de la representació gràfica dels objects geogràfics així com les seves relacions.

Enfocament i mètode seguit.

S'ha donat un enfocament divulgatiu en tot moment sense perdre rigor. El mètode seguit ha estat la consulta bibliogràfica i telemàtica.

Que és un GIS, que resol ?

Concepte

Un Sistema de Informació Geogràfica (GIS) és un conjunt format per una base de dades geogràfica, un programari que gestiona aquesta base, una interfície que permet als usuaris a manipular les dades que resolen la seva problemàtica específica i per últim l'equip informàtic necessari. Concretament el NCCGIA (National Center for Geographic Information and Analysis), ho defineix com sistema de maquinari, programari i procediments dissenyats per a capturar, emmagatzemar, manipular, analitzar i modelar dades georeferenciades. Una distinció que es fan dels GIS és en la manera de tractar les dades geogràfiques depenent si usa el model vectorial o el model ràster. En aquest document es tracta només del model vectorial.

Així necessitem:

- unes dades geogràfiques
- un programari de gestió de les dades
- una interfície que permeti l'entrada i anàlisi
- equip informàtic adequat

Capes

La realitat del terreny és molt diversa: podem descriure el relleu (corbes de nivell), el tipus de sòl (litologia), la presència d'aigua (hidrografia), els assentaments (poblacions, cases, edificis), els usos del sòl, la xarxa de carreteres, etc. Tot això es pot veure com si fossin capes. Per a cada una d'aquestes capes es fan divisions més petites (fulls) on així no cal treballar amb tot el mapa. En cada capa hi situarem els nostres objectes geogràfics (cases, carreteres, etc) representats per objectes geomètrics (punts, línies,etc).

D'aquests apartats en parlarem a continuació.

Dades geogràfiques

Entenem com a dada geogràfica un fet de la realitat amb una posició espacial definida en el que es poden mesurar diferents característiques.

Aquests fets poden ser naturals, on la referència espacial és intrínseca al propi fet observat, com per exemple la subdivisió de l'espai per tipus de terreny (cultiu, bosc, urbà, industrial, etc.) o també artificials, on la referència espacial és extrínseca, com la divisió de l'espai en unitats administratives. Les fronteres entre països, entre comunitats autònomes són exemples d'aquesta última visió.

El problema de la representació de les dades geogràfiques

Com s'ha dit abans la dada geogràfica representa un fet de la realitat situada en algun lloc. Com es pot veure existeix per una part la ubicació d'aquest fet (la georeferenciació) i per una altra la descripció d'aquest fet.

Representar correctament les dades espacials implica geocodificar les dades i descriure digitalment les característiques espacials. En la primera part s'ha d'assignar una "etiqueta" que identifica la seva posició espacial respecte a algun punt comú o marc de referència, i en el segon apartat descriure la posició geomètrica i establir les relacions espacials respecte altres objectes.

Per codificar les dades geogràfiques podem usar dos procediments.

Directament usant un sistema d'eixos coordenats. A cada lloc li donem una posició absoluta.

Indirectament , donant una adreça o referència relativa. Aquesta opció permet establir la posició relativa respecte als demés objectes. Per exemple, si diem que visc al carrer Balmes, 5 de Barcelona, estic donant unes referències del meu domicili indirectament. Primer, cal esbrinar on és el carrer Balmes, i aquest queda determinat per la referència a altres llocs, com per exemple a prop de plaça Catalunya, etc, el nombre ....

Si ens decantem per la referència directa usarem els sistemes de coordenades, els quals poden ser planes o esfèriques.

De les planes s'utilitza sovint el sistema UTM (Universal Transversal Mercator).

En les esfèriques, un lloc geogràfic queda posicionat per dos coordenades ,a nomenades longitud i latitud (λ i ϕ). El primer mostra quin meridià prenem mesurat amb l'angle a partir del meridià de Greenwich . Es mesura com 180 E o 180 W. La latitud mostra el paral·lel i es mesura a partir de l'Equador prenent l'angle a partir d'ell. Es mesura entre 90N i 90S.

Per a passa d'un sistema a l'altre podem usar les fórmules següents:

$$\begin{cases} x = \lambda \\ y = \ln\left(\tan\left(\frac{\pi}{4} + \frac{\phi}{2}\right)\right) \end{cases}$$

La majoria de GIS permeten realitzar transformacions dels mapes representats, ja que no sempre treballem en un sistema de coordenades donat, així que cal saber les noves coordenades dels llocs referenciats en els nous. En el sistema antic usarem (x_0,y_0) i en el sistema nou (x'_0,y'_0).

Transformacions

Translació

Les noves coordenades s'obtenen a partir de la translació segons un vector donat $\vec{v}$.

$$\begin{pmatrix} x'_0 \\ y'_0 \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} v_x \\ v_y \end{pmatrix}$$

Gir

El gir té en compte l'angle usat. La transformació és la següent:

$$\begin{pmatrix} x'_0 \\ y'_0 \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

Canvi d'escala

$$\begin{pmatrix} x'_0 \\ y'_0 \end{pmatrix} = \begin{pmatrix} k & 0 \\ 0 & t \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

Però no totes les transformacions són simples, així que podem combinar-les obtenint de més complexes.

Si volem referenciar un punt sota un altre sistema de coordenades qualsevol usarem combinacions diferents.

Exemple:

Gir + translació + canvi d'escala.

$$\begin{pmatrix} x'_0 \\ y'_0 \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

$$\begin{pmatrix} x''_0 \\ y''_0 \end{pmatrix} = \begin{pmatrix} x'_0 \\ y'_0 \end{pmatrix} + \begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} v_x \\ v_y \end{pmatrix}$$

$$\begin{pmatrix} x'''_0 \\ y'''_0 \end{pmatrix} = \begin{pmatrix} k & 0 \\ 0 & t \end{pmatrix} \begin{pmatrix} x''_0 \\ y''_0 \end{pmatrix} = \begin{pmatrix} k & 0 \\ 0 & t \end{pmatrix} \begin{pmatrix} x'_0 + v_x \\ y'_0 + v_y \end{pmatrix} = \begin{pmatrix} k & 0 \\ 0 & t \end{pmatrix} \left(\begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} v_x \\ v_y \end{pmatrix} \right)$$

Totes aquestes operacions donen lloc a una transformació lineal, on les rectes són rectes. Si es volen aproximacions no lineals es pot recórrer a funcions quadràtiques aproximades per mínims quadrats per exemple.

En moltes ocasions cal saber quina és la transformació entre dos sistemes de coordenades, degut a que es tenen mapes de diferent escala i orientacions diferents. Si més no existeixen altres transformacions, la lineal és la més acceptada.

Representació vectorial de la informació espacial

Un dels problemes que cal resoldre és: com representem la informació espacial, quines estructures de dades són necessàries ?

Durant molt temps les propostes de representació han anat variant fins arribar a les representacions arc/node. Aquest tipus de representació és el que resol de manera satisfactòria la representació d'objectes irregulars així com el tractament de la topologia, aspecte aquest molt important en un GIS.

Però primer de tot veiem quins han estat els model proposats, la seva evolució i la seva problemàtica.

llista de coordenades

El més senzill, però es repeteixen punts i no respecta la topologia. Es registra el nom, nombre de vèrtexs i es descriuen els vèrtexs. Si la figura és tancada es repeteix el primer vèrtex. En l'annex 1 es troba una codificació del tractament d'un fitxer d'aquest tipus.

El següent quadre mostra aquest tipus d'estructura

```
poligonA,9 1,6.5 5,6.5 5,8 6,8 6,9 8,9 8,11 1,11 1,6.5 puntA,1 1,2.3 linia1,3
2,4 4,5.6 2,7
```

poligonA,9 1,6.5 5,6.5 5,8 6,8 6,9 8,9 8,11 1,11 1,6.5

En vermell tenim el nom de la figura i el nombre de vèrtexs.
Subratllat les 8 coordenades dels vèrtexs que la formen.
En blau i repetit el primer vèrtex.

diccionari de vèrtexs

Format per dos fitxers. Un, amb tots els vèrtexs i l'altre amb totes les figures i els vèrtexs que el formen. Això evita la redundància dels vèrtexs, però no resol el tema de la topologia. Veieu l'annex 2 per més detall.

Fitxer de vèrtexs

```
1,6.5 5,6.5 5,8 6,8 6,9 8,9
11,9
8,11
11,9 11,11 1,11 1,1 8,1
8,5
```

```
5,5
11,1
```

En el quadre anterior hi ha 16 vèrtexs, al primer (1,6.5) li correspon l'identificatiu 1 i a l'últim l'identificatiu (11,1) 16. La disposició dels vèrtexs és irrellevant.

Diccionari de vèrtexs

```
puntA 11
puntB 12
poligonA 1 2 3 4 5 6 8 10 1
poligonB 6 7 9 8 6
poligonC 1 2 3 5 1
liniaA 1 3 6
```

Com s'observa al polígonA li corresponen els vèrtexs amb identificadors 1,2,3,4,5,6,8,10 i 1. Altre cop es repeteix l'últim vèrtex per denotar un polígon.

fitxers dime

Utilitza tres fitxers. Semblant a l'estructura anterior usem un fitxer de coordenades de vèrtexs, un fitxer de segments on s'emmagatzema el vèrtex origen, el vèrtex final i els polígons a dreta / esquerra (si existeixen) dels esmentats segments. Finalment, s'usa un fitxer que descriu els polígons, indicant els segments que conté.

Aquest sistema ja inclou la topologia. Aquest concepte es veu clarament en paraules com polígon a dreta o a esquerra d'un segment i permetrà resoldre per exemple quants polígons envolten un de donat. Aquestes i d'altres qüestions són les que fan referència al concepte de topologia.

En la taula següent es mostra el contingut del fitxer de segments.

segment	pol a dreta	pol a esquerra	vèrtex origen	vèrtex final
1	D	A	1	2
2	C	A	2	3
...	...	....	...	...

Estructura arc/node

Aquest tipus d'estructura surt com a resposta a l'estructura de fitxers DIME en referència a la representació d'objectes geogràfic no exactament rectangulars. Aquesta estructura es basa en l'arc (cadena de segments rectes que tenen la mateixa topologia), els nodes, vèrtexs on concorren tres o més arcs, o lloc terminal d'una línia.
En principi els nodes responen de forma natural a la intersecció de tres o més arcs, però hi ha situacions en que s'ha de reflectir un canvi en la naturalesa de l'arc, com per exemple el fet que una canonada de conducció d'aigua estigui composta per diversos materials.

En mateixa topologia volem dir que mantenen a dreta i a esquerra del segment els mateixos polígons.

El següent mètode implementat en Java dóna la topologia d'un arc en un context on les figures són polígons.

```
/**
 * Mètode que posa el polígon passat com a paràmetre a l'esquerra o dreta d'aquest arc.
 * Aquest mètode funciona bé si el centroide és dins el polígon, ja que usa el producte
 * vectorial com a eina per a decidir l'orientació de l'arc.
 *
 * @param p Poligon
 */
  public void topologia(Poligon p)
  {
    //D'aquest arc consideram el segment
    Vector2D v2D = new Vector2D(p.centroide(),this.nodeInicial());
    Vector2D w2D= new Vector2D(this.nodeInicial(),this.nodeFinal());
    if (v2D.sentit(w2D)<0) aDreta=p; //arc sentit horari respecte el polígon
    if (v2D.sentit(w2D)>0) aEsquerra=p; //arc sentit antihorari respecte el polígon
  }
```

En l'annex 2 s'hi troba molta més informació al respecte.

Elements d'un sig vectorial

Fins ara s'ha tractat el tema de la informació geogràfica en el que respecte a la georeferenciació, però hi ha la part temàtica de les dades, aquella que havíem dit reflecteix la realitat indicant les seves característiques.

La pregunta natural que sorgeix és, com organitzem la informació espacial i temàtica ?

Models

A priori ens trobem amb dos models:

El que incorpora separadament la base de dades espacial i la base de dades temàtica. Aquest respondria al nom de model híbrid.
Cada objecte geogràfic es troba representat en ambdues bases de dades amb una etiqueta identificadora que els vincula . El programari ARC/INFO n'és un exemple.

L'altre, és considerar les dues bases de dades en una única sola. Diguem-ne, compacte.

Tant en un cas com en l'altra existeix un concepte subjacent que és el que les bases de dades han de seguir un model per a decidir quines dades s'han

d'emmagatzemar i quines són les relacions entre elles. El model entitat- relació és sens dubte el més acceptat actualment.

Una altra opció en el model híbrid seria tractar la informació pròpiament espacial en fitxers, separadament de la informació temàtica que es tractaria mitjançant un sistema gestor de base de dades (sgbd).
De fet avui dia, existeixen ja sgbd que incorporen els aspectes espacials. En el cas del sgbd MySQL (versió 4.1), hi ha una referència al model espacial. Veieu l'annex 2 per més informació. En aquesta referència s'explica com aconseguir , emmagatzemar i analitzar els aspectes geomètrics del objectes geogràfics. En aquesta referència s'explica el model geomètric proposat per OpenGIS (Consorci GIS), estructures de dades per representar dades espacials i com usar-ho en MySQL

Exemple que mostra com crear una taula en MySQL amb informació temàtica i espacial.

Per usar un polígon (en aquest cas es declara el polígon exterior i l'interior), o qualsevol altre objecte geogràfic s'ha de seguir una sintaxi. En l'exemple següent declarem un quadrat.

POLYGON((0 0,10 0,10 10,0 10,0 0),(5 5,7 5,7 7,5 7, 5 5))

Cream una taula que incorpori tant dades temàtiques com espacials. En MySQL s'usa el tipus GEOMETRY per referenciar qualsevol tipus d'objecte geogràfics, però hi ha també els tipus POINT, POLYGON, LINESTRING.

```
mysql> CREATE TABLE bosc (g POLYGON, nombre_arbres INT, tipus_arbre
VARCHAR(20));
```

inserim en la taula informació

```
mysql> SET @g = 'POLYGON((0 0,10 0,10 10,0 10,0 0),(5 5,7 5,7 7,5 7, 5 5))';
mysql> INSERT INTO bosc VALUES (PolygonFromText(@g), 100,'alzina');
```

Fixeu-vos en les funcions que s'usen per introduir el valor correctament. En aquest cas s'usa PolygonFromText

I ara demanen la seva àrea.

```
mysql> SET @h ='PolygonFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),(5
5,7 5,7 7,5 7, 5 5)));

mysql> SELECT Area(@h);
```

Què podem de fer en un SIG ?

Bàsicament són quatre les fases que en un gis s'han de realitzar.

Entrada de dades.

Depenent del model de sig usat haurem d'introduir les dades espacials, és a dir, indicar les coordenades i aspectes topològics dels objectes.
A continuació introduirem les dades temàtiques dels objectes geogràfics.

Recerca/extracció de la informació.

Aquí faríem les consultes contra les dades, be demanant en quin lloc passa tal cosa, bé demanant quines coses passen en tal lloc.

Anàlisi de dades.

Si ens centrem en el model vectorial , caldria parlar d'un anàlisi estadístic a partir de les dades temàtiques (les contingudes en la base de dades) i una anàlisi diguem-ne espacial , contra les dades geomètriques (podríem fer anàlisi dels punts, línies i polígons)

En l'ampli tema de tractament de dades, val a dir que gairebé la majoria de sigs incorporen eines de tractament estadístic i geomètric.

En l'apartat d'anàlisi, s'apliquen tècniques estadístiques prenent com base els atributs temàtics.

En l'aspecte geomètric s'incorporen eines que permeten analitzar el tipus de capa tractada. Així en una capa on els objectes són línies i en concret una xarxa (interconnexió d'arestes que formen circuits) podem obtenir informació referent a camins òptims, grau de compacitat, sinuositat de la xarxa, etc.

Per exemple pels camins òptims s'apliquen algoritmes de cerca en profunditat que resolen el problema. El camí òptim és justament el subgraf arbre que enganxa el node de sortida (l'arrel) fins el node d'arribada (la fulla) de l'arbre generat usant l'algorisme anterior.

En altres aspectes com matrius d'adjacències entre els nodes s'aplica l'algorisme anterior però donant pesos a les arestes d'una unitat. Així es coneix quin és el nombre mínim d'arestes que calen per anar d'un punt a un altre.

Presentació a l'usuari.

Ja sigui en forma de taules, gràfics o la més clara, a través de la cartografia, les dades amb les recerques i anàlisis efectuades han de poder ser mostrades a l'usuari. Si volem crear un mapa que mostri gràficament les poblacions de les ciutats podríem procedir de la següent manera: Per a cada ciutat (punt) dibuixaríem un cercle amb radi proporcional a la seva població (atribut habitants).

Etiquetat dels objectes geogràfics.

Un aspecte important és l'etiquetatge dels objectes geogràfics. És clar que cada objecte definit en el seu propi context té unes referències geomètriques que el fan descriure unívocament, però aquesta no és la manera més adient, sinó usant identificadors.

S'usen dues parts, el nom primari i el secundari.

Pel primari s'usa el nom oficial en cas de entitats de caràcter administratiu (la província de Madrid, n'és un exemple). En el cas d'entitats naturals (algunes taques de bosc, per exemple) s'han d'assignar un nom a aquestes taques, com per exemple pinar1, pinar2, llac1,llac2,llac3,carretera1,carretera2,etc.

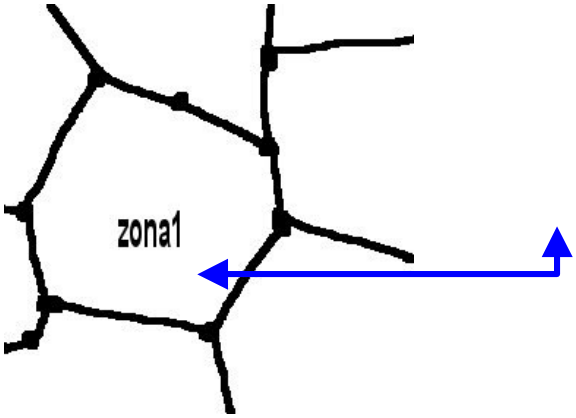
El secundari s'usa per codificar altres característiques. En el cas d'objectes espacials de tipus administratiu, per a designar el codi del municipi de Barcelona tindríem:
01 08 54 231, on 01 indica la comunitat autònoma, 08 la província, 123 la zona agrària i 231 el municipi.

En certa manera aquest ús recorda al de clau alternativa d'una relació en termes de la teoria de base de dades, o sigui la formada per aquells camps que determinen unívocament l'element a representar.

Com es lliga tot ?

Un cop tenim els objectes espacials digitalitzats, afegim la informació temàtica dels objectes on s'ha de posar per a cada camp identificatiu de l'objecte la següent informació: el nom primari i/o el nom secundari. Val a dir que aquesta identificació serà el nexe d'unió de les dues "bases".

identificador	propietari	superfície	valor
.....			
zona1	Miquel Guàrdia	350 m2	120.000€



La representació gràfica de la zona1, s'hauria fet mitjançant fitxers conservadors de la topologia, com per exemple estructures arc, on zona1 vindria delimitada per 6 arcs, les referències dels polígons que l'envolten, etc.

Recerca i recuperació de la informació geogràfica

L'objectiu és obtenir un mapa, taula de valors que continguin els objectes geogràfics amb un conjunt particular d'atributs espacials o temàtics (un lloc o una variable). Així és com funciona la gran majoria de SIG.

Cerca espacial

Aquest tipus de cerca contesta a: que hi ha en tal lloc ?

La cerca la podem fer especificant un domini espacial o especificant una condició geomètrica.

Via domini espacial

S'indica un punt, una línia o un polígon i s'extrauen tots els objectes que hi toquen o estiguin dintre el domini especificat. En un sig, aquestes indicacions es poden fer assenyalant alguns punts de la pantalla (mapa), assenyalant un punt i un radi (indicant per això un cercle), dibuixant una caixa rectangular o seleccionant una línia poligonal.

Especificant una condició geomètrica

El domini espacial s'estableix usant una condició geomètrica. Com per exemple indicant que les coordenades satisfacin certa condició o indicant una àrea circular. De la base de dades temàtica, s'extrauen tots aquells atributs els quals l'objecte geogràfic compleix l'esmentada condició geomètrica.

Cerca temàtica

Aquest tipus de cerca contesta: en quin lloc hi ha aquest atribut que compleix certa condició? Com en el cas anterior podem fer-ho de dues maneres: amb especificació simbòlica o condició simbòlica

Mitjançant especificació simbòlica

En aquest cas es s'indica la variable nom que es vol localitzar. Això provoca dos resultats, una taula de valors amb tots els atributs i un mapa. Aquest tipus de cerca traduït a sentències SQL seria del tipus:

SELECT camp1,.... FROM taula WHERE campn LIKE .....

Mitjançant condició simbòlica

En aquest cas s'indica una condició d'un o varis atributs temàtics. Llavors tots els objectes espacials que compleixin l'esmentada condició s'extrauen via una taula i es representen en el mapa. En sentències SQL les condicions s'afegirien usant els operadors numèrics <, >, etc i entre les condicions els operadors lògics AND, OR, etc

Cerca estratigràfica

Un dels aspectes més importants i que resol amb elegància són les cerques estratigràfiques, és a dir, entre dues o més capes. Aquest tipus de cerca usa per una banda eines geomètriques que combinades amb les cerques temàtiques donen resposta a situacions gens trivials. Un exemple ens ho aclarirà:

Volem construir un abocador de fems en una zona de la qual disposem les següents capes.

- recursos hídrics
- vegetació
- assentaments de població
- carreteres
- relleu
- tipus de sòl

Quina seria el lloc o llocs adients ?

En llenguatge col·loquial respondríem dient, lluny de terrenys permeables, lluny de llocs on hi hagi vegetació endèmica i en perill d'extinció, lluny de rius per evitar la contaminació, lluny d'assentaments de població per evitar les males olors. En canvi, el volem a prop de les carreteres per facilitar l'accés. Respecte el relleu, malgrat no ser vinculant caldria no estar ni en cims (per evitar les males olors) ni en valls (per evitar l'estancament de l'aigua).

Contestar directament a aquesta pregunta en un mapa on s'hagin fusionat totes aquests estrats d'informació pot arribar a ser molt difícil.

La resposta seria donar un conjunt possible (a priori no definitiu) de punts i anar provant les condicions abans descrites. Això fa que s'incorpori una nova capa ("abocadors") .Clarament hem de poder quantificar que vol dir "lluny" i "a prop" en el sig. Això es traduirà a una certa distància mínima o màxima. Per cada capa aquesta distància pot ser diferent.

Així del conjunt proposat inicialment volem que estigui a més de 50 km dels terrenys permeables. Primerament resoldrem en la capa "tipus de sòl" aquelles zones que siguin permeables. Entre la capa "abocadors" i la capa resultant anterior ens quedarem aquells abocadors que estiguin a més de 50. Un cop obtingut aquest conjunt de punts el "creuarem" amb la capa resultant de vegetació endèmica i ens quedarem amb els abocadors que estiguin a més de 100 km d'aquestes zones.
Clarament el conjunt de punts es fa cada cop més petit, fins i tot, pot arribar a no tenir cap element, la qual cosa faria proposar un nou conjunt d'abocadors.

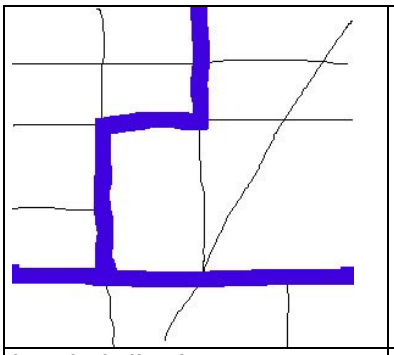
Aquest tipus de cerca com hem dit abans queda restringida a mètodes geomètrics dels elements de les capes. Així ens trobem amb les següents situacions

		Capa A		
		Punts	Línies	Polígons
Capa B	Punts	coincidència o distància mínima	punt en línia	punt en polígon
	Línies		intersecció de línies	línia en polígon
	Polígons			intersecció de polígons

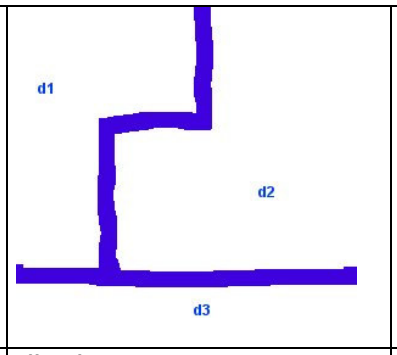
Amb aquesta visió estratigràfica i tenint en compte que entre les taules hi pot haver lligams per referències de claus foràmies en els GIS la introducció de dades temàtiques (atributs) es fa d'acord amb l'objecte geogràfic representat, per tant la vinculació o relació amb les altres taules es realitza per condicions geomètriques entre els objectes abans esmentades.

L'exemple següent ens ho aclareix.

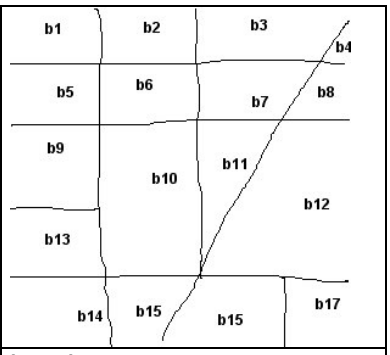
Imaginem-nos que tenim una distribució de barris i de districtes d'una població, tal com mostren les figures següents i volem saber quina és la taxa de desocupació d'un districte.



barris i districtes



districtes



barris

Si optéssim per declarar les referències externes mitjançant claus foràmies, tindríem a la taula de districtes.

id_districte	població desocupada			
d1	...	...	...	....
d2	...			
d3				
.....				

.. i la de barris

id_barri	id_districte	població activa		
b1	d1	...	...	....

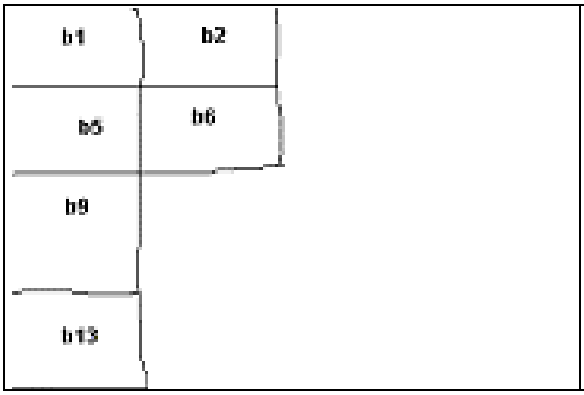
b2	d1			
b3	d1			
b4	d1			
...				

Observi's que per a conèixer la taxa de desocupació d'un districte cal saber quina ocupació hi ha en el districte.

Una instrucció SQL que la resoldria seria:

select id, desocupada **as** aturada, **sum**(poblacio_activa) **as** activa,(desocupada / (desocupada + sum(poblacio_activa))) **a**s taxa **from** districtes, barris **where** districtes.id=barris.id_districte **group by** id_districte ;

Una possible solució geomètrica i sense tenir en compte les relacions entre taules seria per cada districte obtenir el nivell de població activa, ja que coneixem la població desocupada. Això es faria intersectant la capa districtes amb la capa barris, però de manera que només sortissin els barris del districte1. Amb aquest nou conjunt de polígons podem saber amb quina població activa hi comptem, ja que només hem de sumar tots valors de l'atribut d'aquesta consulta.

	Resultat d'obtenir els barris del districte1.

Plantejament d'un projecte en concret: la xarxa de distribució de l'aigua.

En aquest apartat tracta del funcionament d'una xarxa de distribució d'aigua urbana,

Xarxa pública d'Aigua. Modelatge geogràfic dels elements que intervenen.

Quins punts de vista (capes) podem incloure ?

Hem de tenir clar quin tipus d'aigua es transporta, per això és necessari saber un poc quins són els estats en que es pot trobar l'aigua.

Segons aigues de barcelona, els estats de l'aigua dins el cicle urbà són:
dolça – potable – residual – depurada – reciclada – restituïda.

dolça : aigua que es troba en el medi natural, ja sigui formant llacs, rius o reserves subterrànies.

potable: aigua que ha passat per una ETAP (estació de tractament d'aigua potable) ,estació que mitjançant un tractament físic i químic . És apta pel consum humà i permet l'elaboració d'aliments.

Residual: aigua que ha estat usada i conté en diferents graus residus del consum humà, comerç, industrial, etc.

Depurada: aigua que ha estat sotmesa a un procés de depuració eliminant la seva toxicitat. Aquesta aigua no és apta pel consum humà.

Reciclada: aigua depurada que s'usa en tasques secundàries (reg de jardins, neteja de carrers, extinció d'incendis)

Restituïda: aigua depurada amb la qualitat suficient que es torna al medi sense perill.

Notem que l'aigua pura és aquella que es troba absenta de components químics, És usada en laboratoris. No es troba en el medi natural.

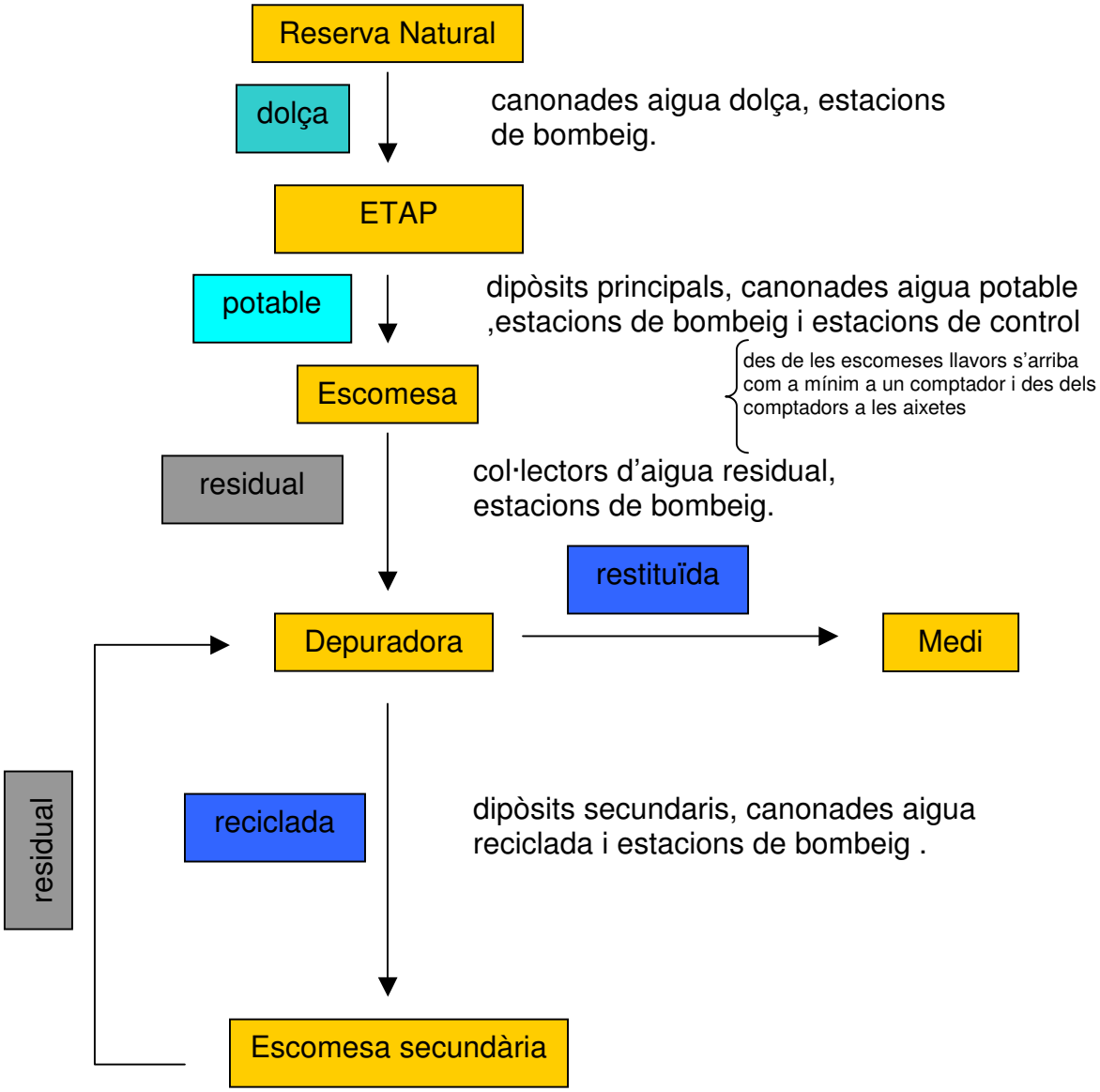
En funció del tipus d'aigua establim quin és el procés, o millor dit , el camí que recórrer l'aigua des de que surt des d'un riu, embassament, glacera o reserva subterrània fins que es tornada al medi (restituïda).

El camí de l'aigua urbana.

L'aigua **dolça** és transportada des dels llocs naturals fins a les plantes ETAPs. Podem trobar dipòsits intermitjos que controlen el seu flux. Un cop l'aigua és apta pel consum humà (**potable)** passa a les canonades de transport d'aigua potable. Aquestes canonades porten ja a les escomeses dels abonats al servei d'aigua. En aquestes canonades ens podem trobar amb diferents estacions de control de la qualitat, així com estacions de bombeig. Res no impedeix que les estacions facin les dues coses. Existeixen canonades de diferent diàmetre, segons la destinació de les mateixes. Hi ha canonades que desemboquen en d'altres per així repartir l'aigua allà on faci falta.

Ara l'aigua és usada pel consumidor, ingerida (beguda, cuinada,etc) com manipulada (rentaplats, rentadora, etc). En aquest moment l'aigua (ara és **residual** i és tòxica en diferents graus)es retorna per altres canonades a les estacions depuradores (col·lector) . Enmig també ens podem trobar amb estacions de bombeig d'aigües residuals. També existeixen diferents gruixos de diàmetre segons el cabdal a transportar. Aquesta aigua és traslladada fins les estacions depuradores. Des d'aquestes estacions l'aigua –**depurada**- es retornada per canonades a punts estratègics per usos secundaris (boques per reg de jardins, extinció d'incendis, neteja de carrers). Aquesta és l'aigua que se'n diu **reciclada** Aquestes aigües tornen a l'estació depuradora via els mateixos col·lectors de les aigües residuals. També les aigües depurades es porten per canonades especials al medi (mar) –**restituïdes** .

Un gràfic ens mostra els elements que intervenen en la seva distribució:


Reserva Natural
dolça
canonades aigua dolça, estacions de bombeig.
ETAP
potable
dipòsits principals, canonades aigua potable ,estacions de bombeig i estacions de control
Escomesa
des de les escomeses llavors s'arriba com a mínim a un comptador i des dels comptadors a les aixetes
residual
col·lectors d'aigua residual, estacions de bombeig.
restituïda
Depuradora
Medi
residual
reciclada
dipòsits secundaris, canonades aigua reciclada i estacions de bombeig .
Escomesa secundària

Identificació entitats geogràfiques i objectes geogràfics

Tal com es veu en el diagrama anterior s'observen les següents entitats que poden ser representades per objectes geogràfics que adaptats a l'escala apropiada poden ser:

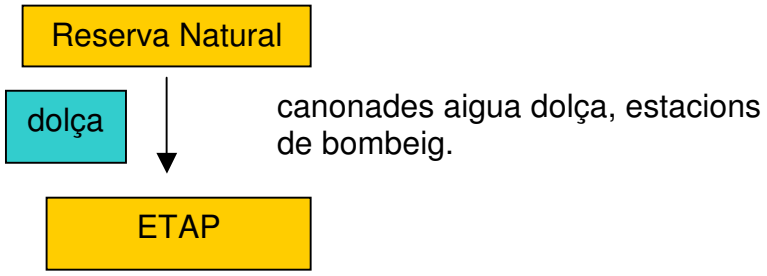
Polígons – reserves d'aigua.
Línies – canonades.
Punts – ETAPs, depuradores, dipòsits principals (aigua potable), dipòsits secundaris (aigua reciclada) ,estacions de control, estacions d'elevació, escomeses principals i secundàries, les boques de reg, els hidrants, les bies, les fonts públiques i finalment les claus de pas en les canonades (vàlvules).

Nota: Segons l'escala a que treballem alguns elements que he considerat representables per punts poden ser tractats com a polígons, sobretot aquells elements que es representen per construccions i que en proporció superen a les canonades.
Clarament la visió de canonada o col·lector per línies es clara independent de la secció, doncs aquesta pot ser especificada en les dades temàtiques de la base de dades. A més la idea de flux dóna un caràcter lineal.

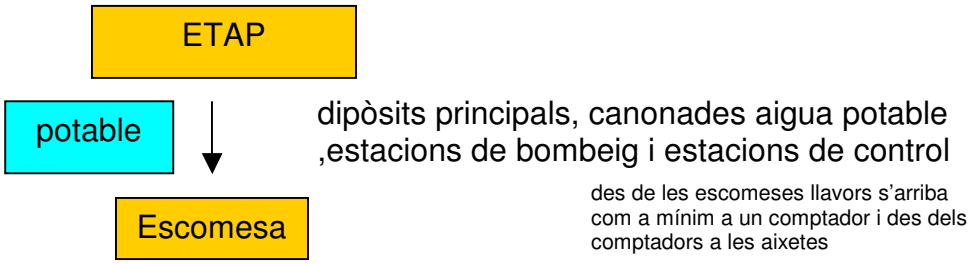
Model estratigràfic

Segons l'estat de l'aigua tractada tindrem la nostra informació geogràfica separada per capes. En aquest projecte només es té en compte la distribució d'aigua potable, és a dir, des de que surt de la etap fins que arriba a casa nostra. En aquest camí l'aigua també flueix cap a altres elements de la xarxa, com poden ser fonts, hidrants, boques de reg, etc.

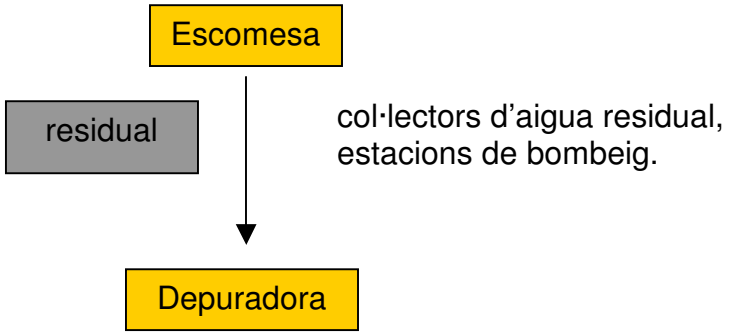
1. **captació de l'aigua . En aquesta cap intervenen les reserves i les etaps. Es correspondria a la part del diagrama següent:**



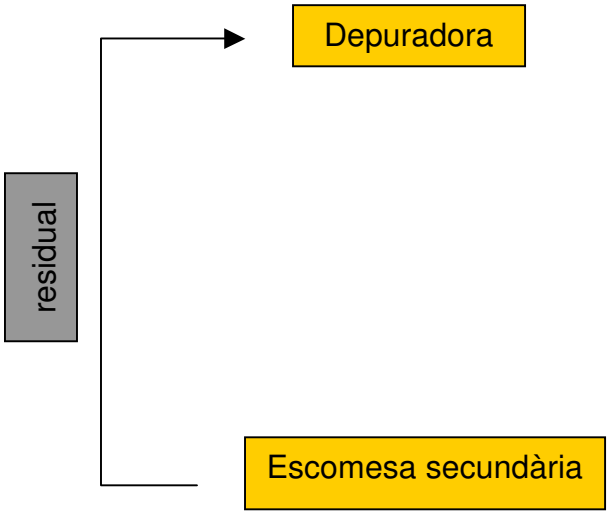
2. **distribució de l'aigua. Es correspondria amb el diagrama següent**:



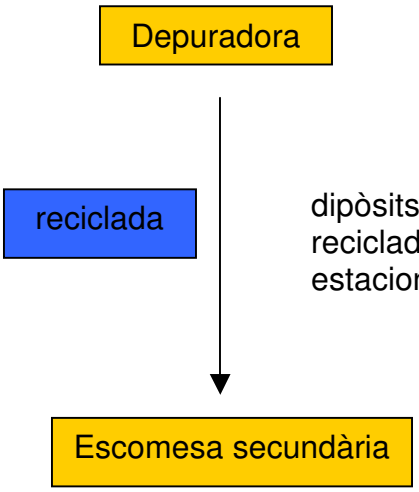
3. **recollida de l'aigua.** Es correspondria amb els dos diagrames següents:



juntament amb aquest altre diagrama. Ens hem de fixar que malgrat l'origen de l'aigua sigui diferent, aquesta es prescindeix passant a ser residual.



4. redistribució de l'aigua



Model estratigràfic en la distribució de l'aigua.

- etaps (punts)
- dipòsits (punts)
- estacions d'elevació (punts)
- canonades (línies). Els nodes dels quals serien llocs on conflueixen més de dues canonades o b
- vàlvules (punts)
- bies (punts)
- hidrants (punts)
- boques de reg (punts)
- escomeses (punts)
- comptadors (punts)
- fonts públiques (punts)
- orografia (polígons)
- illes de cases (polígons)
- fonts públiques (punts)

En aquesta visió i tenint en compte la naturalesa de la xarxa, aquesta és mallada, tots els elements puntuals es troben lligats a la xarxa (línies formant circuits), ja sigui com elements terminals o com elements intermitjos. S'ha de tenir en compte que hi ha elements de la xarxa pública que correspondrien a la part particular, és a dir, tant les bies, columnes seques o alguns hidrants formen part dels edificis.

Per la resta d'estats disposaríem de les capes adients al seu tractament.

Tant la distribució de l'aigua, la seva recollida i la seva distribució segueix un model de circuits o de xarxes. Les raons que sigui mallada es mostren en les especificacions expressades en el reial decret 140/2003, 7 de febrer.

REIAL DECRET 140/2003, de 7 de febrer, pel qual s'estableixen els criteris sanitaris de la qualitat de l'aigua de consum humà. («BOE» 45, de 21-2-2003.)

En el capítol 12, es comenta el següent:

Article 12. Distribució de l'aigua de consum humà.

1. Les xarxes de distribució pública o privada han de ser en la mesura que sigui possible de disseny mallat, per poder eliminar punts i situacions que facilitin la contaminació o el deteriorament de l'aigua distribuïda. Han de disposar de mecanismes adequats que en permetin el tancament per sectors, per tal de poder aïllar àrees davant situacions anòmales, i de sistemes que permetin les purgues per sectors per protegir la població de possibles riscos per a la salut.

Descripció dels atributs

En tots els objectes representats més avall existiria un identifiador únic (id_objecte).

etaps (estacions tractament aigua potable) entrada en servei, capacitat màxima de tractament (m^3/segon)	

estacions de bombeig (elevació) entrada en servei, cabal elevació (m^3/segon), alçada elevació (m), potència instal·lada	

dipòsits reguladors Entrada en servei, nombre de components, altura de l'aigua, superfície	

comptadors tipus de comptador (domèstic, industrial, agrícola, d'obres), pressió nominal màxima, cabal, any de construcció, any de posada en funcionament, marca i model	

Els **bies** (boques d'incendis equipades) són elements que es col·loquen en els edificis a prop de les sortides o portes. Hi pot haver uns quants, seguint la normativa (màxim 50 m i mínim de 25. En un gis bidimensional la localització ens indicarà l'edifici en concret (suposadament representat per un punt). Els camps més interessants serien la longitud de la màniga, data construcció, data col·locació, data última revisió i localització indirecta del bie. Això és important ja que malgrat els bies es trobin situats en una mateixa ubicació x,y, caldrà saber la component d'alçària. Per això en els gis bidimensionals, seria interessant saber una referència indirecta respecte l'edifici.	

hidrants
tipus (columna seca-humida/arqueta), diàmetre (80/100). Així tenim 4 possibilitats.

arqueta	80 mm	
arqueta	100 mm	
columna	80 mm	70-45-45
columna	100 mm	100-70-70

Apart de les característiques tècniques cal afegir altres camps d'interès com poden ser data posada en marxa, data última revisió, data últim us, marca, model, ubicacio (publica/privada).
Cal dir que en edificis amb protecció espefífica d'hidrants han de disposar de la senyalització prescriptiva.

escomeses

Pràcticament les escomeses són els punts finals que l'empresa gestora de l'aigua hi intervé, a partir d'aquí ja és responsabilitat dels usuaris.

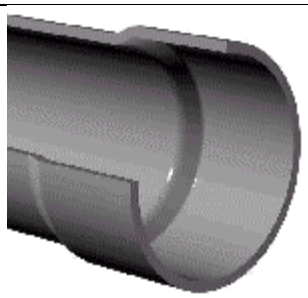
A la imatge i d'esquerra a dreta, escomesa, vàlvula empresa , comptador i vàlvula de l'usuari.

Realment de les escomeses ens interessa la data de col·locació, tipus de racor

canonades

el cabdal (litres/segon), el diàmetre, material, any de construcció, timbratge (pressió nominal)

Cal ressaltar que les canonades estan construïdes de diferent material segons els avanços tecnològics, per tant cal tenir present aquest fet. Per això en un tram de canonada que existeix un canvi de material convindria establir un node.

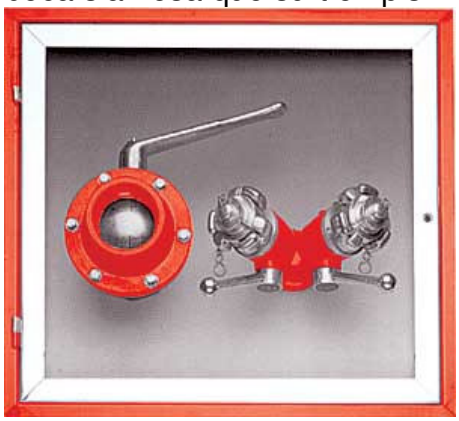


vàlvules tipus de vàlvula (papallona,guillotina, bola, etc), material, pressió, diàmetre, marca, model, data d'instal·lació	

b**oca de reg** data posada en servei, marca, model	

font aigua potable data posada en servei, model, disponibilitat	

Elements que no formen pròpiament part de la xarxa de distribució d'aigua, però que poden ser útils en algunes situacions.

columna seca. Malgrat no hi porti aigua normalment, permet la connexió d'extensions de mànigues que poden venir de camions cisterna, d'hidrants, etc. D'aquests elements convé saber data de col·locació,model, marca, data última revisió, data últim ús.	 boca siamesa que surt en pis  vàlvula reguladora de pressió+ boca siamesa.

punt d'alimentació d'aigua de façana

Taula resum

element	**atributs**
etap	identificador,data entrada en servei, capacitat màxima de tractament (m^3/segon)
diposit	identificador,data entrada en servei, nombre de components, altura de l'aigua, superfície
estació elevació	identificador ,data entrada en servei, cabal elevació (m^3/segon), alçada elevació (m), potència instal·lada
canonada	identificador ,el cabdal, el diàmetre, material, any de construcció, timbratge (pressió nominal), proveïdor
vàlvula	identificador ,tipus de vàlvula (papallona, guillotina, bola, etc), material, pressió, diàmetre, marca, model, data d'instal·lació
hidrant	identificador ,tipus (columna seca-humida/arqueta), diàmetre (80/100), data construcció, data col·locació, data revisió, data últim ús, marca, model, servei (públic/privat)
bie	identificador ,la longitud de la mànega, data construcció, data col·locació, data última revisió, data últim ús i localització indirecta, marca, model
boca de reg	identificador ,marca, model, data entrada en servei
comptador	identificador ,tipus de comptador, pressió nominal màxima, cabal, any de construcció, any de posada en funcionament, marca, model
font aigua potable	identificador ,data posada en servei, model, disponibilitat
escomesa	identificador ,data col·locació, tipus de ràcor

Interrelacions de les entitats.

Malgrat existeixin interrelacions en la base de dades, la vinculació dels elements es fa geomètricament, és a dir, a través de consultes espacials coneixem la relació entre punt-línia i punt-punt. Així, per exemple podem demanar quins llocs d'escomesa es trobem annexats a canonades de 30 mm i de cert material. Els punts d'escomesa i canonades es troben lligats per una relació geomètrica prou evident. En aquesta consulta obtenim un conjunt de punts, subconjunt de tots els punts de connexió de la xarxa d'aigua que pengen de les esmentades canonades.

El resultat s'obté primer de tot fent una cerca de les canonades de 30 mm i de cert material per llavors realitzar una cerca geomètrica dels punts (llocs d'escomesa) propers a aquestes canonades.

El resultat de la consulta es veu en la seqüència d'imatges.

1) Totes les escomeses
2) Xarxa de canonades
3) Canonades de 30 mm i de pvc
4) Escomeses demanades

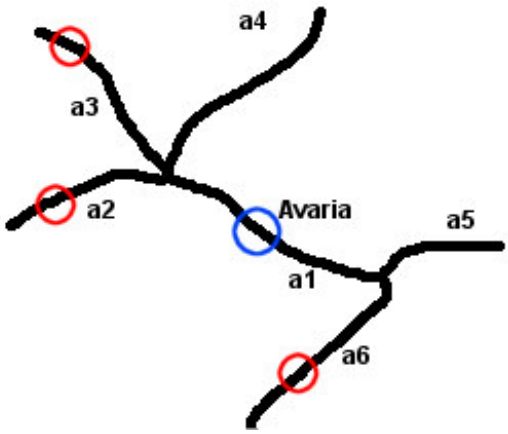
	totes les escomeses
	xarxa de canonades

	canonades de 30 mm i de pvc
	escomeses connectades a les esmentades canonades
	resultat obtingut

Situacions que es resoldrien

Control de fuites de la xarxa.

Saber quines vàlvules cal tancar en una zona que s'ha detectat una avaria (fuita d'aigua). Això es correspon a tancar el mínim nombre d'elles per a que el servei afecti al mínim nombre d'abonats i serveis oferts per la companyia.
Controlar això implica:
Si l'avaria es troba en una canonada cal saber el nombre de vàlvules d'aquesta canonada. Com l'avaria és referenciable (podem donar les coordenades), cal, llavors fer un recorregut per les canonades a fi de trobar les vàlvules que hi hagi, en cas de no trobar-ne o trobar-ne només una cal fer un recorregut per la canonada en profunditat i arribar als nusos i per a cada nus mirar si hi ha una vàlvula. Això ens portarà a un dibuix del tipus següent:



Seguir a cada costat de l'avaria i determinar els grafs en forma d'arbre de tal manera que el lloc de l'avaria seria el node arrel i les vàlvules les fulles dels dos arbres obtinguts. En la figura prèvia es veu que caldria tancar les vàlvules de les canonades a2,a3 i a6, però caldria continuar pels nodes d'a4 i a5 usant les eines pròpies del gis per determinar les vàlvules (fulles) restants. D'aquesta manera la resta de la xarxa continuaria tenint aigua, llevat de la part afectada.

Restitució alguns components de la xarxa.

En aquest apartat volem saber on es troben les canonades que per la seva naturalesa són propenses a avaries. L'empresa gestora de l'aigua vol un informe complert del cost de la seva restitució.

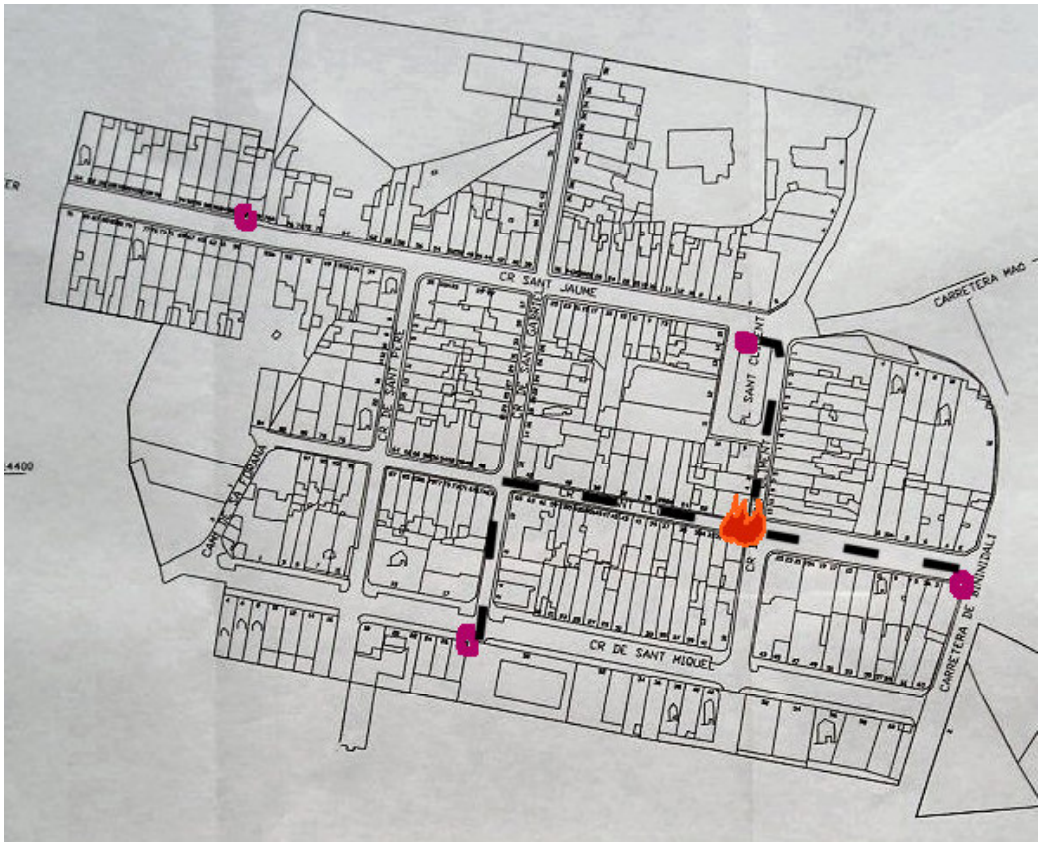
La fotografia anterior mostra les canonades de fibrociment de la població. En el gis s'ha fet una consulta temàtica per a localitzar-les.

De manera similar volem restituir els comptadors de certa marca que presenten algunes deficiències en els models construïts un determinat any.

Comptadors amb deficiències

Control d'incendis.

Trobar per exemple els hidrants disponibles en una àrea donada. S'ha declarat un foc en una zona. Volem saber on són els hidrants disponibles, i en concret aquells que per exemple es troben actualitzats .



A més i per seguretat voldríem saber també on es troben els bies més pròxims a la zona declarada

Com a complement a la recerca d'hidrants i de bies, es podria també cercar els punts d'alimentació de les boques seques.

Programari GIS-TFC-Didàctic

Introducció

Paral·lelament a l'estudi de l'aigua en una xarxa pública de distribució se'm va ocórrer (sobretot per a posar en pràctica els conceptes apresos) generar un petit SIG amb una vessant totalment educativa i que posés de manifest com he dit abans els conceptes estudiats . S'ha de dir que no s'ha treballat el tema des d'una òptica de desenvolupament de programari, ja que en principi la idea era mostrar un poc com es capturen els objectes geogràfics, s'enllacen amb dades temàtiques i es fan algunes petites consultes (per objecte i per tema). Com el tema de la captura de les dades geomètriques m'ha dut més temps del pensat, he decidit incloure pel TFC la captura de dades mitjançant a crides a la base de dades orcl (sgbd oracle) via jdbc. La captura de dades geomètriques es basa en l'entitat capa (o estrat). S'ha intentat seguir la política mostrada en els llibres presos com a base, separant les dades geogràfiques de les temàtiques, usant estructures de dades no persistents pel seu tractament en memòria i fitxers (streams) per a guardar-les. Només les dades temàtiques es desen mitjançant el sgbd. En un futur s'usaran les extensions que ofererixen els actuals sgbds per emmagatzemar els objectes geogràfics juntament amb la seva topologia.

Aquest petit GIS tracta la captura de dades geogràfiques a partir d'imatges escanejades (format gif i jpg) basant-se en un model vectorial. Com la captura es fa mitjançant coordenades físiques del dispositiu (pantalla), es prescindeix de les coordenades UTM, però no hi hauria cap problema en adjuntar les coordenades utm. El que sí és important es incloure l'escala en pixels i aquesta és possible si existeix l'escala gràfica del mapa. En cas de no trobar-ne l'escala gràfica s'han d'indicar en el dibuix alguns elements per a poder deduir-la. Mireu l'annex 2.

Les classes que s'implementen es mostren també en l'annex 2.

Funcionalitats

Funcionalitats actives

Captura de dades geogràfiques (els objects) per estrats. A partir d'imatges escanejades d'atles, plànols, etc i indicant l'escala a que es treballarà, capturarem els elements geogràfics que vulguem. (punts, línies o polígons).
Captura de dades temàtiques ().
Consulta de dades sobre el gis.
Directament sobre el mapa. A partir d'un llistat dels objectes geogràfics.

Funcionalitats pendents

Consulta de les situacions dels objectes a partir de consulta a la base de dades, és a dir, cerques espacials mitjançant condicions i especificacions geomètriques.

Les consultes intra-capes estan pendents de les relacions geomètriques de:

punt –punt (coincidència) punt –línia (pertany) punt-polígon (és dins)
linia –línia (posició relativa) línia –polígon(posició relativa)
polígon-polígon (posició relativa) i potser no es trobarà activada (pendent d'implementar)
L'anàlisi de les capes des del punt de vista geomètric.
L'anàlisi de les variables temàtiques usant tècniques estadístiques.

Especificació general

El tros de programari lliurat permet en principi la introducció de dades dels objectes geogràfics, que com he dit abans es fan a partir d'imatges escanejades per un servidor de plànols de poblacions obtinguts dels ajuntaments, o bé, fotografies de mapes dels atles. N'adjunt unes quantes fotografies.

Els objectes geogràfics a tractar són punts, línies i polígons. Tots tres son formats en la seva essència per Arcs.
Un arc és una successió de punts sense autointerseccions i que té un punt inicial i un punt final. A aquests punts els anomenen nodes sí, són extrems d'un arc aïllat, són interseccions de 3 o més arcs o bé si coincideixen, com pot ser el cas d'una línia tancada o polígon aïllat.

Amb aquesta idea subjacent tenim que els punts són "arcs" amb el mateix node inicial i final i sense punts intermitjos.
L'objecte línia format per un arc, el qual disposa dels nodes inicials i finals diferents amb 0 o més punts intermitjos o bé els mateixos nodes inicial i final, però amb punts intermitjos, en nombre superior a dos.
L'objecte polígon és format per una successió d'arcs format entre ells un circuit. Si el polígon és tancat és format per un únic arc amb node final i inicial iguals però amb 2 o més punts intermitjos.

Aquestes idees queden reflexades en les següents definicions de classe (java).

Altre cop us remeto a l'annex 2.

Petita ajuda de captura de dades geogràfiques.

Per a provar el programari, almenys en l'entrada de dades, afegir les dades temàtiques i realitzar algunes consultes seguiu els següents passos.

1. Escolliu Digitalitzar mapa i trieu una imatge amb format jpg o gif.
2. Decidir el tipus de capa que voleu treballar. El menú és sensible al tipus escollit.

3. Seleccioneu del mapa els objectes geogràfics que voleu registrar. Per a cada objecte se us demanarà un nom (ha de ser únic) abans de seleccionar les coordenades, en el cas de nodes i vèrtexs. En el cas de polígons i arcs se us demanarà els arcs i els nodes que els formen respectivament. Els arcs, podeu completar-los inserint els vèrtexs.
4. És responsabilitat de l'usuari escullir les dades (nodes i vèrtexs) correctament, en ordre seqüencial geometricament.
5. Si pel que fos ens cansem d'introduir dades podem desar-les en un fitxer seqüencial que el programa reconeix. Més envant el podem recuperar i continuar la càrrega de les dades.
6. Per veure els objectes dibuixats sobre la capa podem escollir l'opció de dibuixar capa.
7. Per introduir les dades temàtiques aneu a Dades temàtiques --> Assignar Taula...
8. Un cop omplertes les taules amb les dades temàtiques, es poden fer les consultes de cerca temàtica (faltaria dibuixar els objectes trobats).
9. En el mapa també es pot saber les dades temàtiques clicant sobre l'objecte amb certa tolerància.

Accions bàsiques

Introduir un node: Simplement indicar el nom del node i llavors apuntar a una zona del mapa.

Introduir un arc: Simplement seleccionar el node inicial i node final.

Un cop tinguem l'arc, el podem omplir de vèrtexs intermitjos. Per acabar amb el darrer vèrtex fem un doble clic.

Introduir un polígon: Simplement seleccionar els arcs que formaran part.

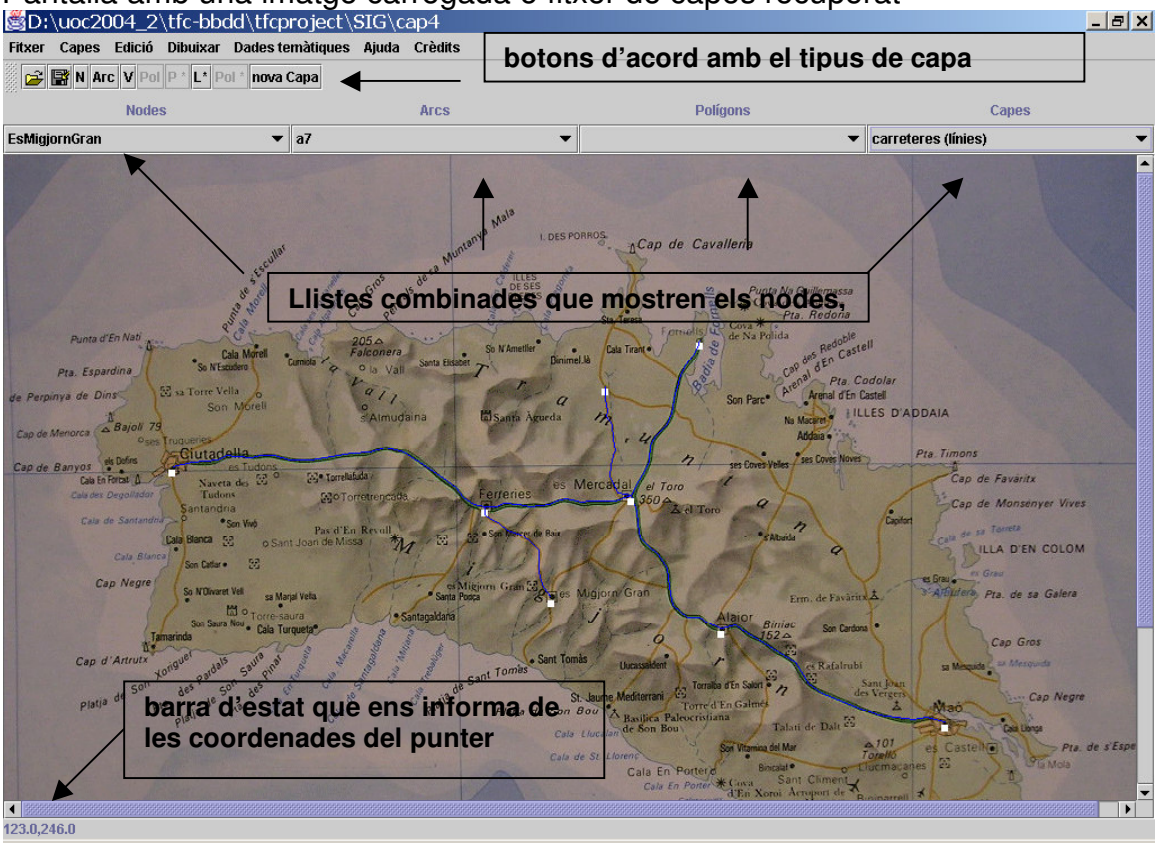
Introduir un punt "aïllat": Com un punt és un arc "atrofiat", se'ns demana el nom del node i a la vegada de l'arc. Llavors apuntem amb el ratolí en el mapa.

Introduir una línia aïllada: Donem nom a l'arc, llavors a continuació el nom del primer node, cliquem els punts intermitjos i finalment amb un doble clic apuntem al node final. Se'ns demanarà un nom d'aquest node.

Introduir un polígon aïllat: Donem un nom a l'arc i donem un nom al primer i últim node (són iguals). Anem fent clics per incorporar els punts intermitjos. Finalitzem amb un doble clic que identificarà l'últim punt abans del node final.

Algunes pantalles

Pantalla amb una imatge carregada o fitxer de capes recuperat

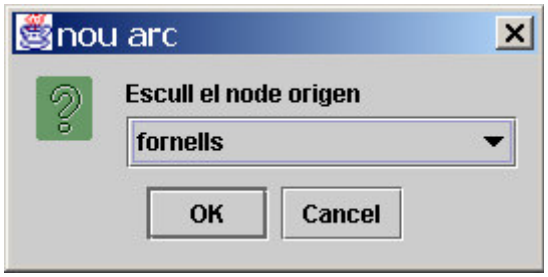


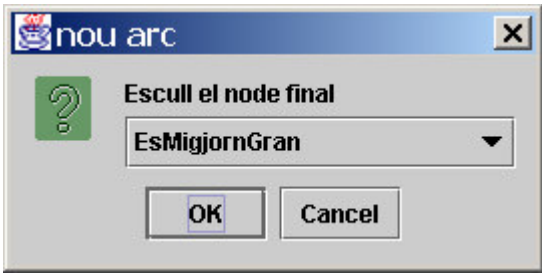
Nova Capa

Nou Node

Nou Arc

S'han d'incloure els nodes origen i final





Trobareu més pantalles i més explicacions en l'annex 2.

En un fitxer adjunt s'hi troben les classes implementades més els mapes usats en fer les proves.

Entorn d'execució necessari.

Cal que una instància de la base de dades estigui en execució i tots els processos auxiliars també. La instancia en el meu cas es diu "orcl". Cal canviar aquesta referència en una instància que tingueu disponible en el codi java GestorBaseDades. Mireu l'annex 2.
Per a la compilació i l'execució és necessari accedir als fitxers jar proporcionats per Oracle i que es troben en "directori_oracle"\lib. En el meu cas particular

D:\oracle\ora90\jdbc\lib i aquí dins hi ha entre altres el fitxer classes12.jar que especifica entre altres coses el drivermanager, i com s'accedeix al sgbd.

Execució del programa.

Per executar el programa

java – classpath .; directori_oracle\jdbc\lib\classes12.jar ApplicationSIG

Bibliogafia

Documentación. NTP 42: Bocas e hidrantes de incendio. Condiciones de instalación.
[http://www.mtas.es/insht/ntp/ntp_042.htm, diferents dates]

MySQL Reference Manual for version 4.0.18. Spatial Extensions in MySQL
[Document hipertext, diferents dates]

Canal de Isabel II. Instalaciones y explotación.
[http://www.cyii.es/www/publico/30/30.html, diferents dates]

Auxioc S.L . Hidrants
[http://auxifoc.com/catala/hidrants.htm, diferents dates]

Aigües de Barcelona
[http://www.agbar.es, diferents dates]

Steve Bobrowski. Oracle8i para Windows NT. Edición de aprendizaje.Editorial Osborne McGrawHill, 2000

Joaquin Bosque Sendra. Sistemas de Información Geogràfica. Editorial Rialp,1997 segona Edició,

Javier Gutiérrez Puebla, Michael Gould. SIG: Sistemas de Información Geográfica. Editorial Síntesis, primera edició, reimpressió 2000

Comas, D y Ruiz. Fundamentos de los sistemas de información geográfica. Ariel Geografia, primera edició 1993, Barcelona.

Norman L.Biggs. Matemática discreta. Vicens Vives, 1994.

John Zukowski.Programación en Java. Anaya Multimedia, 1999

Agustín Froufre. Java 2. Manual de usuario y tutorial. Editorial Rama, 3 edició

Oracle getting started – jdbc
[document hipertext, vàries dates]

Annex 1

Fitxer LlistaCoordenades.java

```
import java.io.*;
import java.util.*;
import javax.swing.*;

/**
 * <p>Títol: LlistaCoordenades</p>
 * <p>Descripció: Petita demostració de la captura de dades en format llista
de coordenades</p>
 * <p>Copyright: Joan Manuel González FebrerCopyright (c) 2004</p>
 * <p>Company:  UOC - tfc - bases de dades - Sistemes d'Informació
geogràfica/p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 */

public class LlistaCoordenades
{

  public static void main(String[] args)
  {
      Vector figures = new Vector();
      Poligonal poligonal =null;
      Punt punt= null;
      Vertex vertex = null;
      boolean ple=false;
      int comptats=0;
      boolean poligonalinici=true;

      for (int i=0;i<File.listRoots().length;i++)
        System.out.println(File.listRoots()[i].getName());
      File fitxer = new File("classes","llistadecoordenades.txt");

   try
   {
       FileReader freader = new FileReader(fitxer);
       BufferedReader bf = new BufferedReader(freader);
       String cadena;

       String identificador="";
       int vertexs=0;

       while(!(cadena=bf.readLine()).equals(""))
       {

         StringTokenizer token = new StringTokenizer(cadena);

         while (token.hasMoreElements())
         {
             // les variables segonacadena són els tokens,
             String segonacadena = token.nextToken();
             //
             //Separem les dades en identificado,nombre de punts o
```

```
abcises,ordenades.
             //

             if (identificaToken(segonacadena).equals("I")) //el token és un
identificador

             {
               identificador=(String) treuIdentificador(segonacadena)[0];
               try
               {
                   vertexs =
Integer.parseInt((String)treuIdentificador(segonacadena)[1]);
               }
               catch(Exception e)
               {
                   System.out.println("No és un fitxer vàlid");
               }
               comptats = 0;


               if (vertexs==1)
               {
                   punt = new Punt(identificador);
                   figures.add(punt);

               }
               else

               {
                   poligonal= new Poligonal(identificador);
               }


             }
        else//el token és un vertex.
        {

            vertex=treuVertex(segonacadena);
            if (vertexs==1)
            {

              punt.x=vertex.x;
              punt.y=vertex.y;


            }
            else
            {

              poligonal.afegirVertex(vertex);
              comptats=comptats+1;
              if(comptats==vertexs) figures.add(poligonal);

            }
        }
}

}
bf.close();
freader.close();
```

```
}
catch (FileNotFoundException ex)
{
  System.out.println("Fitxer no trobat");
}
catch(IOException ioex){
  System.out.println("s'ha produit una excepció entrada / sortida");
}


Iterator it =figures.iterator();

while (it.hasNext())
{
  Object ob =it.next();
  if ( ob instanceof Punt)
  {
      Punt p = (Punt) ob;
      System.out.println(p.nom);
      System.out.println("("+p.x+","+p.y+")");
  }
  if ( ob instanceof Poligonal)
  {
      Poligonal p = (Poligonal) ob;
      Vertex v1 = (Vertex) p.vertexs.firstElement();
      Vertex v2 = (Vertex) p.vertexs.lastElement();
      String tipus ="";
      if (v1.equals(v2)) tipus = "Polígon";
      else tipus ="Línia poligonal";
      System.out.println(p.nom+" " +tipus);
      Iterator it2 = p.vertexs.iterator();
      while(it2.hasNext())
      {
        Object ob2 = it2.next();
        Vertex v = (Vertex) ob2;
        System.out.print("("+v.x+","+v.y+")\t");
      }
System.out.println();
  }
}

  }

   static public String identificaToken(String cadena){

      StringTokenizer scadena = new StringTokenizer(cadena,",");
      String primertoken = scadena.nextToken();
      String segontoken = scadena.nextToken();



//la cadena és un nombre.
   try {
       Float.parseFloat(primertoken);
   }
   catch (Exception ex)
   {


       // és un identificador
```

```
        return "I";

    }

    //no és un identificador, són vèrtexs


    return "V";

  }
  static public Object[] treuIdentificador(String cadena){
      Object o[] = new Object[2];
      StringTokenizer scadena = new StringTokenizer(cadena,",");
      o[0]=scadena.nextToken();
      o[1]=scadena.nextToken();

      return o;
  }

  static public Vertex treuVertex(String cadena)
  {

      Vertex vertex = new Vertex();
      StringTokenizer scadena = new StringTokenizer(cadena,",");

      vertex.afegeix(Float.parseFloat(scadena.nextToken()),Float.parseFloat(sca
dena.nextToken()));
      return vertex;
  }

}
```

Fitxer Poligonal.java

```
import java.util.*;

/**
 * <p>Títol: Poligonal</p>
 * <p>Descripció: Classe que descriu una poligonal</p>
 * <p>Copyright: Joan Manuel González FebrerCopyright (c) 2004</p>
 * <p>Company:  UOC - tfc - bases de dades - Sistemes d'Informació
geogràfica/p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 */

public class Poligonal {
Vector vertexs;

static int quants=0;

String nom;
  public Poligonal(String nom)
  {
      vertexs= new Vector();
      this.nom=nom;
  }
public void afegirVertex(Vertex ver){
```

```
  vertexs.add(ver);
  quants = quants+1;

}
public int[] obtenirX(){
  int capacitat;

  if (esUnPoligon())
   capacitat = vertexs.size();
  else capacitat = vertexs.size();
  //System.out.println(capacitat);
  int x[] = new int[capacitat];
  for (int i=0;i<capacitat;i++){
      x[i]=(int)(((Vertex) vertexs.get(i)).x);
  }

  return x;

}
public int[] obtenirY(){
  int capacitat;
  if(esUnPoligon())
   capacitat = vertexs.size();
  else capacitat=vertexs.size();

  int y[] = new int[capacitat];
  for (int i=0;i<capacitat;i++){
      y[i]=(int)(((Vertex) vertexs.get(i)).y);
  }

  return y;
}
public boolean esUnPoligon()
{
      return ((Vertex) vertexs.firstElement()).equals((Vertex)
vertexs.lastElement());
}
public String[] llistVertexs()
{
String cadena[]= new String[vertexs.size()];

for (int i=0;i<cadena.length;i++)
  cadena[i]=Float.toString(((Vertex)
vertexs.get(i)).x)+","+Float.toString(((Vertex) vertexs.get(i)).y);
return cadena;
}

}
```

Fitxers Punt.java i Vertex.java

```
/**
 * <p>Títol: Punt</p>
 * <p>Descripció: Classe que descriu una punt (coordenades)</p>
 * <p>Copyright: Joan Manuel González FebrerCopyright (c) 2004</p>
 * <p>Company:  UOC - tfc - bases de dades - Sistemes d'Informació
```

```
geogràfica/p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 */

public class Punt extends Vertex
{
  String nom;

  public Punt(String nom)
  {
      this.nom=nom;
  }

}



/**
 * <p>Títol: Vèrtex</p>
 * <p>Descripció: Classe que descriu un vèrtex</p>
 * <p>Copyright: Joan Manuel González FebrerCopyright (c) 2004</p>
 * <p>Company:  UOC - tfc - bases de dades - Sistemes d'Informació
geogràfica/p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 */

public class Vertex
{
float x;
float y;


  public void afegeix(float x, float y)
  {
      this.x=x;
      this.y=y;

  }
  public boolean equals(Object o)
  {
      if(o instanceof Vertex)
      {
        Vertex v = (Vertex) o;
        if (this.x==v.x && this.y==v.y)
            return true;
      }
      return false;
  }


}
```

Annex 2

Classe Arc

```
import java.util.*;
import java.io.*;


/**
 * <p>Títol: Arc</p>
 * <p>Descripció: Classe que descriu un arc, des del punt de vista
dels Sistemes Geogràfics d'Informació.
 * Un arc no és més que una sequència de punts (segments), sense
autointerseccions.
 * Els extrems s'anomenen nodes. Així mateix un arc és orientat si no
és un arc puntual, és a
 * dir, la sequència formada pel node inicial, vèrtexs intermitjos i
node final, li donen un sentit.</p>
 * <p>TFC- 2003/04-2: </p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 *
 */

class Arc extends Poligonal implements Serializable{



/** constant que identifica l'arc linial*/
static String LINIA="Arc linial";
/** constant que identifica l'arc puntual, és a dir, format per un
únic punt*/
static String PUNT ="Arc puntual";
/** constant que identifica l'arc poligonal, és a dir, un polígon
tancat*/
static String POLIGON="Arc poligonal";
/** el node inicial */
Node primer;
/**el node final */
Node ultim;
/** el poligon a la dreta */
Poligon aDreta;
/** el poligon a l'esquerra */
Poligon aEsquerra;
/** el pes de l'arc. Això és útil quan volem realitzar optimitzacions
de camins,etc.
 * En principi val 1, però es pot anar canviant segons els pesos
adjudicats en les
 * variables temàtiques. En el cas que volem saber el camí més curt
indicant el mínim
 * nombre de nodes travessats usarem un pes de 1.0, però si l'aresta
té un cost de
 * recorregut adjudicat, l'haurem d'indicar
 */
float pes = 1.0f;
/**
```

```
 * Constructor d'arc amb un únic paràmetre.
 * @param nom - El nom de l'arc
 */

  public Arc(String nom)  {
      super(nom);
  }

  /**
   * Mètode que afegeix un vèrtex a l'arc. Els vèrtexs són punts
intermitjos entre els
   * dos nodes.
   * @param   v - Vertex, el vèrtex a afegir.
   */
  public void afegirVertex(Vertex v)
  {
      vertexs.add(v);
  }


/**
 * Retorna el node inicial d'aquest arc.
 * @return Node - el node inicial
 */
  public Node nodeInicial()
  {

      return primer;
  }
  /**
  * Retorna el node final d'aquest arc.
  * @return Node - el node final
 */
  public Node nodeFinal()
  {

      return ultim;
  }



  /**
   * mètode que torna l'arc en forma d'array de vèrtexs, incloent els
nodes final i
   * inicial.
   * @return Vertex[] - l'array de vèrtexs
   *
   */
  public Vertex[] arrayVertexs()
  {
      Vertex[] v= new Vertex[vertexs.size()+2];
      v[0]= primer;
      v[vertexs.size()+1]=ultim;
      for (int i=0;i<vertexs.size();i++)
        v[i+1] = (Vertex) vertexs.toArray()[i];

      return v;
  }
  /**
   * Accessor d'escriptura del node inicial.
   * @param unNode - el node inicial.
```

```
    */
   public void setNodeInicial(Node unNode)
   {
       primer=unNode;
   }
   /**
    * Accessor d'escriptura del node final.
    * @param unNode - el node final
    */
   public void setNodeFinal(Node unNode)
   {
       ultim=unNode;
   }

   /**
    * Mètode que diu si l'arc és un punt o no.
    * @return boolean - cert si l'arc és un punt.
    */
   public boolean esPunt()
   {
       if (vertexs.size()==0 && primer.equals((Node) ultim)) return
true;
       return false;
   }
   /**
    * Mètode que diu si l'arc és una línia o no.
    * @return boolean - cert si l'arc és una línia.
    */
   public boolean esLinia()
   {
       if (!nodeFinal().equals((Vertex) nodeInicial())) return true;
       return false;
   }
   /**
    * Mètode que diu si l'arc és un polígon.
    * @return boolean - cert si l'arc és un polígon.
    */
   public boolean esTancat()
   {
       if(esLinia() || esPunt()) return false;
       return true;
   }
   /**
    * Mètode toString de l'arc. Això mostrarà només el nom.
    * @return String
    */
   public String toString()


   {
        return this.nom;
   }
   /**
    * Mètode toString2 de l'arc. Això mostrarà el nom, el primer node,
l'últim, el nombre
    * de vèrtexs intermitjos, i la seva longitud.
    * @return String
    */
   public String toString2()
```

```
  {
      String cadena=null;
      cadena= nom+primer.toString();
      cadena = "NOM: "+nom+ "\nNODE INICIAL: "+primer.toString() +
"\nNODE FINAL: " + ultim.toString()+ "\nNOMBRE DE VERTEXS INTERMITJOS:
"+vertexs.size()+"\nLONGITUD: "+ Float.toString(longitud());
      return cadena;
  }
  /**
   * Mètode que torna el tipus en format cadena (String) de l'arc.
Aquest pot ser: PUNT, LINIA o POLIGON
   * @return String - el tipus d'Arc.
   */
  public String getTipus()
  {
      if (esLinia()) return this.LINIA;
      if (esPunt()) return this.PUNT;
      else
        return this.POLIGON;

  }


  static void main(String cad[])
  {
      Vertex a = new Vertex();
      Vertex b = new Vertex();
      Vertex c = new Vertex();
      Vertex d = new Vertex();

      a.afegeix(0f,0f);
      b.afegeix(1f,0f);
      c.afegeix(1f,1f);
      Arc arc = new Arc("prova");
      arc.afegirVertex(a);
      arc.afegirVertex(b);
      arc.afegirVertex(c);
      arc.afegirVertex(a);
      System.out.println(arc.toString());
      System.out.println(arc.getTipus());

  }

  /**
   * Mètode que torna la longitud (en pixels) de l'arc. Aquest mètode
té en compte
   * els nodes inicial i final que també formen part de l'arc,
obviament.
   * @return float - la longitud de l'arc.
   */
  public float longitud(){
      Poligonal p = new Poligonal(this.nom,this.arrayVertexs());
      return p.longitud();
      //float d1 = (new Segment(primer,(Vertex)
vertexs.firstElement())).distancia();
      //float d2 = (new Segment(ultim,(Vertex)
vertexs.lastElement())).distancia();
      //return super.longitud()+d1+d2;
  }
  /**
   * Mètode que torna l'àrea que hi ha sota l'arc. És una mesura
```

```
auxiliar en el
   * càlcul d'àrees de polígons formats per arcs.
   * @return float - l'àrea de l'arc.

   */
  public float areaSotaArc()
  {
      Vertex[] aV = arrayVertexs();
      Segment s = null;
      float area = 0f;
      for (int i=0;i<aV.length-1;i++)
      {
        s = new Segment(aV[i],aV[i+1]);
        area = area + s.areaSotaSegment();
      }
      return area;
  }


 /**
  * Mètode que torna si un node, passat com a paràmetre pertany a
l'arc. Això és
  * útil quan volem conèixer el nombre d'arestes d'un node.
  * @param d Node
  * @return boolean - cert si el node pertany a l'arc, fal en cas
contrari.
  */
  public boolean conteNode(Node d)
  {
      if(d.equals(nodeInicial()) || d.equals(nodeFinal())) return
true;
      return false;
  }

/**
 * Mètode que torna un array d'enters (les ordenades). Aquest mètode
és util quan es passen
 * a mètodes d'objectes gràfics, com per exemple drawPolyline d'un
objecte Graphics.
 * Si les coordenades de l'arc són: (x1,y1) ... (xn,yn), El mètode
torna les
 * (x1,...,xn).
 *
 * @return int[] - l'array (x1,....,xn)
 */

  public int[] obtenirX(){
      int total = arrayVertexs().length;
      int x[] = new int[total];
      for (int i=0;i<total;i++)
        x[i]=(int)arrayVertexs()[i].getX();


      return x;

  }

  /**
   * Mètode que torna un array d'enters (les ordenades). Aquest mètode
és util quan es passen
   * a mètodes d'objectes gràfics, com per exemple drawPolyline d'un
```

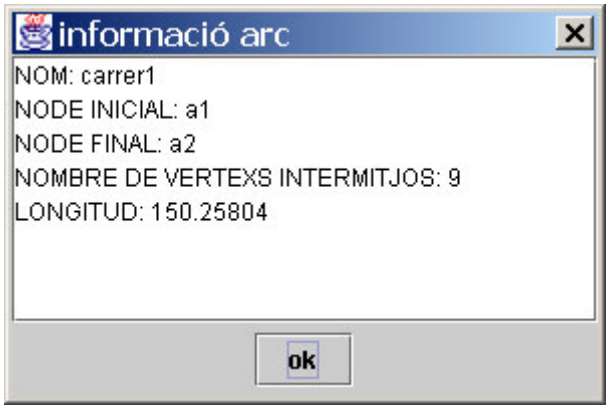
```
objecte Graphics.
   * Si les coordenades de l'arc són: (x1,y1) ... (xn,yn), El mètode
torna les
   * (y1,...,yn).
   *
   * @return int[] - l'array (y1,....,yn)
   */
  public int[] obtenirY(){
      int total = arrayVertexs().length;
      int y[] = new int[total];
      for (int i=0;i<total;i++)
        y[i]=(int)arrayVertexs()[i].getY();


      return y;

  }

  /**
   * Mètode que torna el nombre de talls en la semirrecta. Aquest
mètode s'usarà en
   * el càlcul de si un punt és interior o no a un polígon.
   * @param v - Vertex
   * @return int - el nombre de talls que la semirrecta dreta amb
origen el vèrtex
   * talla l'arc
   */

  public int nombreTallsSemirrecta(Vertex v){
      int total =0;
      float pendent=0f;
      float termeIndependent=0f;
      //agafem cada segment de l'arc i veiem si queda a l'esquerra o a
la dreta.
      for(int i=0;i<arrayVertexs().length-1;i++)
      {
        Segment s = new
Segment(arrayVertexs()[i],arrayVertexs()[i+1]);
        try
        {
         pendent = s.pendent();
        }
        catch (Exception ex)
        {
            //això vol dir que el pendent és infinit.
        }
        termeIndependent= s.v1.getY()-pendent*s.v1.getX();
        float y0 =pendent*v.getX()+termeIndependent;

        boolean condicio1 = (s.v1.getY()<v.getY()) &&
(v.getY()<s.v2.getY());
        boolean condicio2 = (s.v2.getY()<v.getY()) &&
(v.getY()<s.v1.getY());

        if ( condicio1 || condicio2)
        {
         if (pendent*(v.getY()-y0)>0)  total = total +1;
        }

      }
```

```
      return total;
  }
  /**
   * Mètode que torna la sinuositat d'un arc.La sinuositat es calcula
fent la divisió
   * entre la longitut real i la longitut que hi ha entre els seus
nodes.
   * @return float - la variables sinuositat.
   *
   */

  public float sinuositat()
  {

        Segment s = new Segment(nodeInicial(),nodeFinal());
        if (s.distancia()==0) return 0f;
        else
            return this.longitud()/s.distancia();


  }
/**
 * Mètode que posa el polígon passat com a paràmetre a l'esquerra o
dreta d'aquest arc.
 * Aquest mètode funciona bé si el centroide és dins el polígon, ja
que usa el producte
 * vectorial com a eina per a decidir l'orientació de l'arc.
 *
 * @param p Poligon
 */
  public void topologia(Poligon p)
  {
      //D'aquest arc consideram el segment
      Vector2D v2D = new Vector2D(p.centroide(),this.nodeInicial());
      Vector2D w2D= new Vector2D(this.nodeInicial(),this.nodeFinal());
      if (v2D.sentit(w2D)<0) aDreta=p; //arc sentit horari respecte el
polígon
      if (v2D.sentit(w2D)>0) aEsquerra=p; //arc sentit antihorari
respecte el polígon
  }

  /**
   * Mètode que torna un booleà decidint si el vèrtex passat com a
paràmetre
   * es dins l'arc.
   * @param v Vertex
   * @return boolean, cert si el conté, fals en cas contrari
   */
  public boolean conteVertex(Vertex v)
  {
      boolean trobat = false;
      boolean fi = false;
      int i =0;
      while (!fi && !trobat)
      {
        System.out.println("longitud de vèrtexs
"+arrayVertexs().length +" i la 'i' val = "+i);

        Segment s = new
Segment(arrayVertexs()[i],arrayVertexs()[i+1]);
        trobat = s.pertanyVertex(v);
```

```
        if(i==(arrayVertexs().length-2)) {fi=true;}
        System.out.println(fi);
        i=i+1;
      }
      return  trobat;
  }

  /**
   * Mètode accessor d'escriptura del pes.
   * @param unPes float
   */
  public void setPes(float unPes)
  {
      pes = unPes;
  }

  /**
   * Mètode accessor de lectura del pes.
   * @return float, el pes de l'arc.
   */
  public float getPes()
  {
      return pes;
  }
  }
```

Classe ArcDialog

Mostra informació dels nodes i arcs




```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;


/**
 * <p>Títol: ArcDialog</p>
 * <p>Descripció: Finestra que mostra la informació referent a un
arc</p>
 * <p>TFC- 2003/04-2: </p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 *
 */

public class ArcDialog extends JDialog
{
  /* diferents elements gràfics de la finestra*/

  private JPanel panellPrincipal = new JPanel();
  private BorderLayout borderLayout1 = new BorderLayout();
  private JPanel panellSecundari = new JPanel();
  private JButton okButton = new JButton();
  private JScrollPane panellScrolable = new JScrollPane();
  private JTextArea area = new JTextArea();

/** dades que es mostraran en el quadre de diàleg*/
  private String dades;
/** altres dades auxiliars que també es mostraran */
  private String altresDades ;

  /**
   * Constructor del quadre amb 5 paràmetres
   * @param frame Frame ,el marc pare.
   * @param title String el títol
   * @param modal boolean, si és modal o no.
   * @param dades String, les dades a mostrar.
   * @param altresDades String, les altres dades a mostrar.
   */
  public ArcDialog(Frame frame, String title, boolean modal,String
dades,String altresDades)
  {
```

```
        super(frame, title, modal);
        this.dades=dades;
        this.altresDades=altresDades;

        try {
          jbInit();
          pack();
        }
        catch(Exception ex)
        {
          ex.printStackTrace();
        }
    }


/**
 * Mètode que inicialitza els components gràfics
 * @throws Exception
 */
    private void jbInit() throws Exception
    {
        panellPrincipal.setLayout(borderLayout1);
        okButton.setText("ok");
        okButton.addActionListener(new java.awt.event.ActionListener()
        {
          public void actionPerformed(ActionEvent e)
          {
              okButton_actionPerformed(e);
          }
        }
        );
        area.setText(dades+"\n\n"+altresDades);
        getContentPane().add(panellPrincipal);
        panellPrincipal.add(panellSecundari, BorderLayout.SOUTH);
        panellSecundari.add(okButton, null);
        panellPrincipal.add(panellScrolable, BorderLayout.CENTER);
        panellScrolable.getViewport().add(area, null);
    }
    public JTextArea getArea()
    {
        return area;
    }


    void okButton_actionPerformed(ActionEvent e)
    {
        this.dispose();
    }
}
```

Classe Capa

```
import java.util.*;
import java.io.*;
/**
 * <p>Títol: Capa</p>
 * <p>Descripció: Classe que descriu una capa o estrat, des del punt
de vista dels Sistemes Geogràfics d'Informació.
 * Una capa descriu els elements que hi contindran (punts, línies,
polígons), així com
 * altre informació rellevant, com pot ser l'escala, unitats, nom del
fitxer de captura, nom
 * de la taula a la base de dades.
 * <p>TFC- 2003/04-2: </p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 *
 */

class Capa implements Serializable
{
  /** vector que contindrà els objectes geogràfics. Aquests poden ser
només o polígons o arcs.
   * Ambdós elements tenen un atribut que és el nom que serà el que
s'identifiqui amb el
   * nom de la base de dades
   */
  private  Vector objectesGeografics = new Vector();
  /** real que indica l'escala usada */
  private float escala;
  /**Cadena que identifica les unitats (mm,cm,pixels,etc)*/
  private String unitats;
  /**Cadena que identifica el nom de la capa*/
  private  String nom;
  /**Cadena que identifica el nom del fitxer imatge associat*/
  private String mapa;

  /** Cadena que conté el nom de la taula de la base de dades*/
  private String nomTaulaDades=null;
  /** enter que identifica el tipus de capa*/
  private  int tipus;
  /** constant entera que identifica una capa de punts*/
  static int PUNTS = 0;
  /** Constant entera que identifica una capa de línies*/
  static int LINIES = 1;
  /** Constant entera que identifica una capa de polígons*/
  static int POLIGONS = 2;

  /** Constructor amb 5 paràmetres.
   * @param  nom String
   * @param  tipus int
   * @param mapa String
   * @param escala float
   * @param unitats String
   */

  public Capa(String nom,int tipus,String mapa,float escala,String
unitats)
  {
```

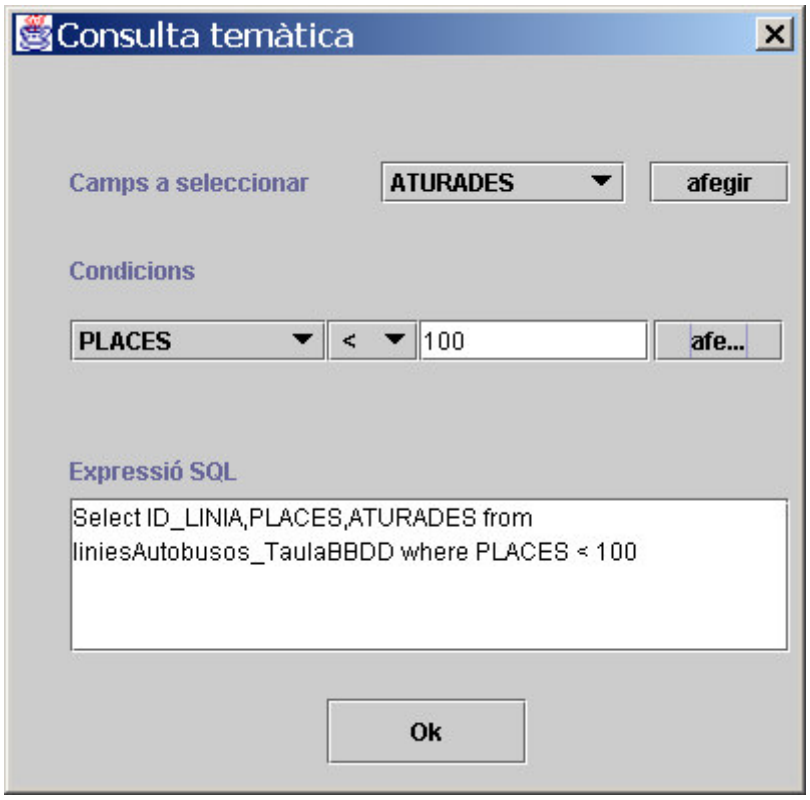
```
        this.nom=nom;
        this.tipus=tipus;
        this.mapa=mapa;

    }

    /**
     * Accessor de lectura del nom de la capa
     * @return String el nom de la capa
     */
    public String getNom()
    {
        return nom;
    }

    /**
     * Accessor de lectura del tipus de capa
     * @return String, la cadena "punts", "linies" o "polígons"
     */
    public String getTipus()
    {
        if(tipus==PUNTS) return "punts";
        if(tipus==LINIES) return "línies";
        return "polígons";
    }

    /**
     * Accessor de lectura del tipus de capa en format numèric.
     * @return int, el tipus de capa en aquest format.
     */
    public int getTipusInt()
    {
        return tipus;
    }

    /**
     * Accessor de lectura del nom del fitxer del mapa en fomrat jpg o
gif.
     * @return String, el nom del fitxer.
     */
    public String getMapa()
    {
        return mapa;
    }

    /**
     * Mètode que torna un objecte File associat a la cadena getMapa()
     * @return File, l'objecte file associat al mapa
     */
    public File getMapaFile()
    {
        return new File(getMapa());
    }

    /**
     * Mètode toString d'aquest objecte, consistent en el nom de la capa
i el tipus de capa.
     * @return String, la cadena que identifica aquest objecte.
     */
    public String toString()
    {
```

```
        return getNom()+" ("+getTipus()+")";
    }


    /**
     * Mètode que afegeix un objecte geogràfic a la capa. Si la capa es
de punts o línies els
     * objectes són arcs
     * @param  unObjecte Object
     */
    public void setObjecteGeografic(Object unObjecte)
    {
        objectesGeografics.add(unObjecte);
    }

    /**
     * Mètode que torna els objectes geogràfics en forma de vector
     * @return Vector, el vector d'objectes geogràfics.
     */
    public Vector getObjectesGeografics()
    {
        return this.objectesGeografics;
    }

    /**
     * Mètode que retorna una cadena amb la informació de la capa,
consistint en:
     * el nom, el tipus i el nombre d'objectes introduits.
     * @return String, l'esmentada informació
     */
    public String toString2()
    {
        return "NOM: "+ toString()+"\nMIDA: "+objectesGeografics.size();
    }

    /**
     * Mètode que torna la sinuositat en el cas de ser una capa de
línies.
     * La sinuositat es calcula com la mitja de les sinuositats de cada
arc.
     *
     * @return float, la sinuositat global de la capa..
     */
    public float sinuositat()
    {
        float sinuositat =0f;
        if (getTipusInt()==Capa.LINIES)
        {
          Iterator it = getObjectesGeografics().iterator();
          while (it.hasNext())
          {
              sinuositat = sinuositat + ((Arc) it.next()).sinuositat();
          }
          sinuositat = sinuositat/getObjectesGeografics().size();

        }
        return sinuositat;
    }
/**
 * Mètode que torna tots els nodes dels objectes geogràfics usats,
independent
```

```
 * del tipus d'objecte emmagatzemat.
 * @return Vector
 */
public Vector nodes()
{
  Vector unsNodes= new Vector();
  if (this.getTipusInt()==Capa.POLIGONS)
  {
      Iterator it = getObjectesGeografics().iterator();
      while(it.hasNext())
      {
        Poligon pol =(Poligon) it.next();
        Iterator itPol = pol.arcs.iterator();
        while(itPol.hasNext())
        {
            Arc arc = (Arc) itPol.next();
            if(!unsNodes.contains(arc.nodeInicial()))
unsNodes.add(arc.nodeInicial());
            if(!unsNodes.contains(arc.nodeFinal()))
unsNodes.add(arc.nodeFinal());
        }
      }
  }
  else
  {
      Iterator it2 = getObjectesGeografics().iterator();
      while (it2.hasNext())
      {
        Arc arc2 = (Arc) it2.next();
            if(!unsNodes.contains(arc2.nodeInicial()))
unsNodes.add(arc2.nodeInicial());
            if(!unsNodes.contains(arc2.nodeFinal()))
unsNodes.add(arc2.nodeFinal());
      }
  }


return unsNodes;
}

/**
 * Mètode accessor d'escriptura del nom de la taula de dades.
 * @param nom String, el nom de la taula de dades.
 */
public void setNomTaulaDades(String nom)
{
  nomTaulaDades=nom;
}

/**
 * Mètode accessor de lectura del nom de la taula de dades.
 * @return String, el nom de la taula de dades.
 */
public String getNomTaulaDades()
{
  return nomTaulaDades;
}

/**
 * Mètode accessor de lectura de l'escala de la capa.
 * @return float , l'escala.
```

```
 */
public float getEscala()
{
  return escala;
}

/**
 * Mètode accessor de lectura de l'unitat usada.
 * @return String , les unitats empleades.
 */
public String getUnitats()
{
  return unitats;
}

}
```

Classe CercaLlocsDialog




```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
/**
 * <p>Títol: CercaLlocsDialog</p>
 * <p>Descripció: Finestra de captura d'una sentencia select,
permetent triar els camps i condicions</p>
 * <p>TFC- 2003/04-2: </p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 *
 */

public class CercaLlocsDialog extends JDialog

{

  /*
   * diferents elements gràfics
   */
  private JPanel panellPrincipal = new JPanel();
  private Capa cap;
  private JLabel etiquetaCamps = new JLabel();
  private JComboBox comboCamps = new JComboBox();
  private JButton btnAfegirCamps = new JButton();
  private JScrollPane panellScrolable = new JScrollPane();
  private JLabel etiquetaCondicions = new JLabel();
  private JComboBox comboCamps2 = new JComboBox();
  private JComboBox comboOperadors = new JComboBox();
  private JTextField textCondicio = new JTextField();
```

```
  private JButton btnAfegirCondicions = new JButton();
  private JLabel etiquetaSQL = new JLabel();
  private JTextArea textAreaSQL = new JTextArea();

  /**
   * el gestor que controlarà els resultats.
   */
  private GestorBaseDades gbd;
  private String[] condicions ={" = "," LIKE "," < "," > "};
  private JButton btnOk = new JButton();
  private int quantesCondicions=0;
  private int quantsCamps=0;
  private String cadena1=" ";
  private String cadena2=" ";


  /**
   * Constructor del quadre amb 5 paràmetres
   * @param frame Frame, el marc pare.
   * @param title String, el títol del marc
   * @param modal boolean, si és modal o no.
   * @param laCapa Capa, la capa a consultar
   * @param unGbd GestorBaseDades, el gestor
   */
  public CercaLlocsDialog(Frame frame, String title, boolean
modal,Capa laCapa, GestorBaseDades unGbd) {
      super(frame, title, modal);
      cap = laCapa;
      gbd=unGbd;

      try
      {
        capaAmbDades();
        ompleDades();
        jbInit();
        setBounds(200,200,400,375);

        setVisible(true);
        pack();

      }
      catch(Exception ex)
      {
        JOptionPane.showMessageDialog(null,ex.getMessage());
        ex.printStackTrace();
      }

  }

  /**
   * Mètode que inicialitza els components gràfics
   * @throws Exception
   */
  private void jbInit() throws Exception
  {
      etiquetaCamps.setText("Camps a seleccionar");
      etiquetaCamps.setBounds(new Rectangle(29, 54, 128, 17));
      panellPrincipal.setLayout(null);
      btnAfegirCamps.setBounds(new Rectangle(318, 52, 70, 21));
      btnAfegirCamps.setText("afegir");
      btnAfegirCamps.addActionListener(new
```

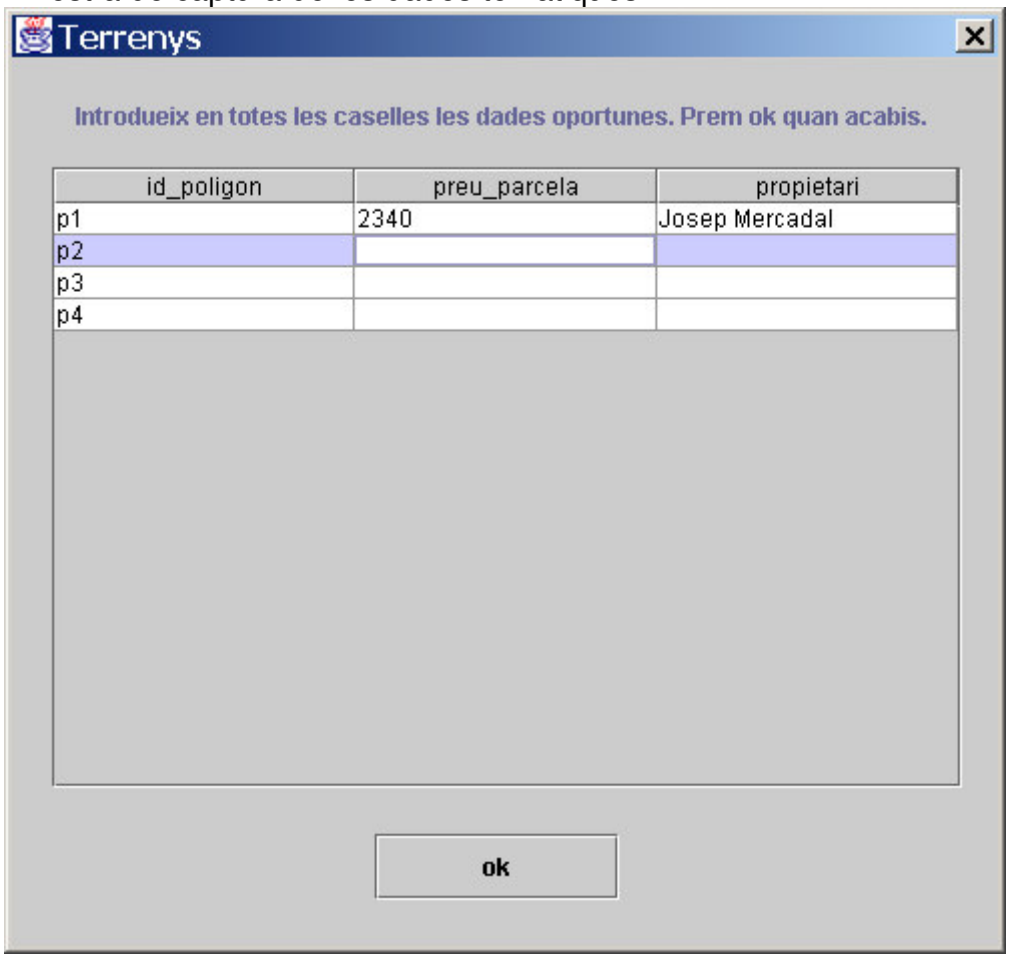
```
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            btnAfegirCamps_actionPerformed(e);
        }
      });
      etiquetaCondicions.setText("Condicions");
      etiquetaCondicions.setBounds(new Rectangle(29, 93, 100, 26));
      btnAfegirCondicions.setBounds(new Rectangle(320, 131, 65, 21));
      btnAfegirCondicions.setText("afegir");
      btnAfegirCondicions.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            btnAfegirCondicions_actionPerformed(e);
        }
      });
      etiquetaSQL.setText("Expressió SQL");
      etiquetaSQL.setBounds(new Rectangle(29, 195, 90, 23));
      textAreaSQL.setEditable(false);
      textAreaSQL.setLineWrap(true);
      textAreaSQL.setWrapStyleWord(true);
      btnOk.setBounds(new Rectangle(157, 320, 100, 33));
      btnOk.setText("Ok");
      btnOk.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            btnOk_actionPerformed(e);
        }
      });
      panellScrolable.setBounds(new Rectangle(29, 220, 359, 77));
      textCondicio.setBounds(new Rectangle(203, 131, 115, 21));
      comboOperadors.setBounds(new Rectangle(157, 131, 46, 21));
      comboCamps2.setBounds(new Rectangle(29, 131, 128, 21));
      comboCamps.setBounds(new Rectangle(184, 52, 122, 21));
      panellPrincipal.add(etiquetaCondicions, null);
      panellPrincipal.add(etiquetaCamps, null);
      panellPrincipal.add(btnAfegirCamps, null);
      panellPrincipal.add(comboCamps2, null);
      panellPrincipal.add(textCondicio, null);
      panellPrincipal.add(btnAfegirCondicions, null);
      panellPrincipal.add(comboOperadors, null);
      panellPrincipal.add(comboCamps, null);
      panellPrincipal.add(etiquetaSQL, null);
      panellPrincipal.add(btnOk, null);
      panellPrincipal.add(panellScrolable, null);
      panellScrolable.getViewport().add(textAreaSQL, null);
      this.getContentPane().add(panellPrincipal, BorderLayout.CENTER);
  }

  /**
   * Mètode que comprova que la capa tingi assignada una taula de la
base de dades.
   * @throws Exception
   */
  private void capaAmbDades() throws Exception
  {
      if (cap.getNomTaulaDades()==null) throw new Exception("Sense
taula assignada");
  }

  /**
   * Mètode que omple les dades a mostrar al quadre de diàleg. Són
dades a mostrar,
```

```
   * els camps de la taula i  els operadors
   */
  private void ompleDades()
  {
      //JOptionPane.showMessageDialog(null,cap.getNomTaulaDades());
      //if(gbd!=null) System.out.println("gbd no és nul");

      int tope = gbd.campsTaula(cap.getNomTaulaDades()).length;


      for (int i=0;i<tope;i++)
      {
        comboCamps.addItem(gbd.campsTaula(cap.getNomTaulaDades())[i]);

comboCamps2.addItem(gbd.campsTaula(cap.getNomTaulaDades())[i]);
      }
      for (int k=0;k<condicions.length;k++)
      {
        comboOperadors.addItem(condicions[k]);
      }
  }


  /**
   * Mètode que permet triar un operador de la consulta select. Els
operadors són AND i OR
   * @return String, una de les dues cadenes "AND" o "OR".
   */
  private String triaOperador()
  {
      String input = null;
      while (input == null)
      {
        String[] opcions ={" AND "," OR "};
        input = (String) JOptionPane.showInputDialog(this,"Operador
?",null,JOptionPane.QUESTION_MESSAGE,null,opcions,opcions[0]);
      }
      return input;

      }

      /**
       * Mètode que respón al botó btnAfegirCondicions
       * @param e ActionEvent, la polsació del botó
       */
  void btnAfegirCondicions_actionPerformed(ActionEvent e)
  {

      if (quantesCondicions==0) cadena2 =" where "+
(String)comboCamps2.getSelectedItem()+comboOperadors.getSelectedItem()
+textCondicio.getText();
   if(quantesCondicions>0)
   {

cadena2=cadena2+triaOperador()+(String)comboCamps2.getSelectedItem()+c
omboOperadors.getSelectedItem()+textCondicio.getText();
   }
   quantesCondicions=quantesCondicions+1;
   textAreaSQL.setText("Select "+cadena1 + " from
"+cap.getNomTaulaDades()+ cadena2);
  }
```

```
  /**
   * Mètode que respón al botó btnAfegirCamps
   * @param e ActionEvent, la polsació del botó
   */
  void btnAfegirCamps_actionPerformed(ActionEvent e)
  {
      if(quantsCamps==0) cadena1
=(String)comboCamps.getSelectedItem();
      if(quantsCamps>0) cadena1 =
cadena1+","+(String)comboCamps.getSelectedItem();
      quantsCamps=quantsCamps+1;
      textAreaSQL.setText("Select "+cadena1 + " from
"+cap.getNomTaulaDades()+ cadena2);
  }

  /**
   * Mètode que respón al botó btnOk
   * @param e ActionEvent, la polsació del botó.
   */
  void btnOk_actionPerformed(ActionEvent e)
  {
      gbd.dadesTaula(textAreaSQL.getText());
   dispose();
  }
}
```

Classe DadesTematiquesDialog

Finestra de captura de les dades temàtiques.




```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

/**
 * <p>Títol: DadesTematiquesDialog</p>
 * <p>Descripció: Finestra que permet introduir la informació temàtica d'una
capa</p>
 * <p>TFC- 2003/04-2: </p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 *
 */
public class DadesTematiquesDialog extends JDialog
{
  /** diferents elements gràfics de la finestra*/
  private JPanel panellPrincipal = new JPanel();
  private JScrollPane panellScrolable = new JScrollPane();
  private JTable taula = new JTable();
  private JButton btnOk = new JButton();
  private JLabel etiquetaInformacio = new JLabel();

  public DadesTematiquesDialog(Frame frame, String title, boolean modal,
JTable taula) {
```

```
      super(frame, title, modal);
      this.taula=taula;
      try {
        jbInit();
        pack();
      }
      catch(Exception ex) {
        ex.printStackTrace();
      }
  }


  private void jbInit() throws Exception {
      panellPrincipal.setLayout(null);
      panellScrolable.setBounds(new Rectangle(20, 54, 454, 310));
      btnOk.setBounds(new Rectangle(181, 387, 122, 33));
      btnOk.setText("ok");
      btnOk.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            btnOk_actionPerformed(e);
        }
      });
      etiquetaInformacio.setText("Introdueix en totes les caselles les dades
oportunes. Prem ok quan " +
      "acabis.");
      etiquetaInformacio.setBounds(new Rectangle(32, 15, 430, 27));
      getContentPane().add(panellPrincipal);
      panellPrincipal.add(panellScrolable, null);
      panellPrincipal.add(btnOk, null);
      panellPrincipal.add(etiquetaInformacio, null);
      panellScrolable.getViewport().add(taula, null);



  }


  /**
   * Mètode que respón al botó btnOk
   * @param e ActionEvent
   */
  void btnOk_actionPerformed(ActionEvent e)
  {
      dispose();

  }

  /**
   * Mètode accessor de lectura de la taula gràfica.
   * @return JTable, la taula
   */
  public JTable obteTaula()
  {
      return taula;
  }

  /**
   * Permet fer proves.
   * @param args String[]
   */
  static void main(String args[])
```

```
  {
      String[] cols = {"id_poligon","preu_parcela","propietari"};
      String[] tipus={"varchar2(10)","int","varchar2(20)"};
      Object[][]
dades={{"p1",null,null},{"p2",null,null},{"p3",null,null},{"p4",null,null}};
      TaulaModelSIG tms = new TaulaModelSIG(cols,tipus,dades);

      DadesTematiquesDialog dtd = new
DadesTematiquesDialog(null,"Terrenys",true,new JTable(tms));
      dtd.setBounds(new Rectangle(0,0,500,600));
   dtd.setVisible(true);
  }
}
```

Classe EscalaDialog

Aquesta classe permet introduir l'escala amb la que es treballarà. Tot mapa digitalitzat ha de contenir l'escala a que es troba dibuixat. Ara bé, això no determina realment l'escala dins el sig, ja que pot passar que la fotografia es trobi ampliada o reduïda. Per tant, per saber realment l'escala, hem de fer una conversió. Per fer això haurem d'usar dos mètodes en funció de si existeix o no l'escala gràfica del mapa o no.

Escala real amb l'existència de l'escala gràfica del mapa analògic.

Si el mapa analògic disposa d'escala gràfica el procés és molt senzill, ja que els píxels que ocupin en pantalla seran els equivalents en la unitat corresponent de la realitat i tant se val que la imatge estigui ampliada o reduïda.

Escala real amb escala numèrica del mapa analògic.

En aquest cas cal introduir d'alguna manera una "escala gràfica". La manera més fàcil és dibuixar una escala gràfica a partir d'un segment simple. Una altra és incorporar un objecte el qual sabem les seves mides reals.

La interfície ha de mostrar o incorporar uns procediments de captura de les dades rellevants, com l'escala del mapa i la distància de l'escala gràfica en el mapa digitalitzat. Aquesta última ha de ser en píxels.

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;


/**
 * <p>Title: EscalaDialog</p>
```

```
 * <p>Description: Finestra que permet introduir l'escala de totes les
capes</p>
 * <p>TFC- 2003/04-2: </p>
 * <p>Company: </p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 */

public class EscalaDialog extends JDialog
{

  /**
   * elements gràfics
   */
  private JPanel panellPrincipal = new JPanel();
  private JTextField textInput = new JTextField();
  private String[] unitats= {"mm","cm","m","dm","hm","km","pixels"};
  private JComboBox comboUnitats = new JComboBox(unitats);
  private JButton botoOk = new JButton();
  private JLabel etiqueta1 = new JLabel();
  private JLabel etiqueta2 = new JLabel();
  private JTextField textPixels = new JTextField();
  private JLabel etiqueta3 = new JLabel();
  private BorderLayout borderLayout1 = new BorderLayout();
  private GridBagLayout gridBagLayout1 = new GridBagLayout();

  /**
   * Constructor amb tres paràmetres.
   * @param frame, el marc pare.
   * @param title, el títol del marc.
   * @param modal, el fet de si és modal o no.
   */
  public EscalaDialog(Frame frame, String title, boolean modal)
  {
      super(frame, title, modal);
      try {
        jbInit();
        setBounds(200,200,400,300);

        setVisible(true);

      }
      catch(Exception ex)
      {
        ex.printStackTrace();
      }
  }

  /**
   * Constructor sense paràmetres
   */
  public EscalaDialog()
  {
      this(null, "", false);
  }

  /**
   * Mètode que inicialitza els components gràfics
   * @throws Exception
   */
  private void jbInit() throws Exception
```

```
  {
      panellPrincipal.setLayout(gridBagLayout1);
      botoOk.setText("Ok");
      botoOk.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            botoOk_actionPerformed(e);
        }
      });
      etiqueta1.setText("escala gràfica del mapa");
      etiqueta2.setText("equival a ");
      etiqueta3.setText("píxels");
      this.getContentPane().setLayout(borderLayout1);
      getContentPane().add(panellPrincipal, BorderLayout.CENTER);
      panellPrincipal.add(etiqueta2,  new GridBagConstraints(0, 2, 1, 1, 0.0,
0.0
            ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(21,
72, 0, 0), 12, 0));
      panellPrincipal.add(textInput,  new GridBagConstraints(0, 1, 2, 1, 1.0,
0.0
            ,GridBagConstraints.WEST, GridBagConstraints.HORIZONTAL, new
Insets(20, 72, 0, 0), 88, 1));
      panellPrincipal.add(textPixels,  new GridBagConstraints(1, 2, 2, 1, 1.0,
0.0
            ,GridBagConstraints.WEST, GridBagConstraints.HORIZONTAL, new
Insets(17, 0, 0, 0), 88, 1));
      panellPrincipal.add(etiqueta3,  new GridBagConstraints(3, 2, 1, 1, 0.0,
0.0
            ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(18,
12, 0, 70), 11, 3));
      panellPrincipal.add(botoOk,  new GridBagConstraints(1, 3, 2, 1, 0.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new
Insets(26, 0, 19, 10), 28, -1));
      panellPrincipal.add(etiqueta1,  new GridBagConstraints(0, 0, 3, 1, 0.0,
0.0
            ,GridBagConstraints.WEST, GridBagConstraints.NONE, new Insets(33,
72, 0, 8), 13, 0));
      panellPrincipal.add(comboUnitats,  new GridBagConstraints(2, 1, 2, 1,
1.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.HORIZONTAL, new
Insets(20, 36, 0, 70), 21, 1));
      this.setResizable(false);

  }

  /**
   * Mètode que torna en un array Object l'escala. En el primer element tornem
les unitats
   * usades i en el segon el valor numèric.
   * @return Object[] les unitats i l'escala.
   */
  public Object[] obteEscala()
  {
      float resultat ;
      Object cadena[]= new Object[2];
      cadena[0] = (String) this.comboUnitats.getSelectedItem();
      try
      {
        resultat
=Float.parseFloat(textInput.getText())/Float.parseFloat(textPixels.getText());
        cadena[1]=new Float(resultat);
      }
```

```
        catch( NumberFormatException ne)
        {
        //Aquí s'ha de donar un feedback a l'usuari per a que ho introdueixi
correctament.
        }
        return cadena;
    }


    /**
     * Mètode que respon al botó ok.
     * @param e ActionEvent
     */
    void botoOk_actionPerformed(ActionEvent e)
    {
        dispose();

    }

    /**
     * Permet fer proves
     * @param args String[]
     */
    static void main(String[] args)
    {
        new EscalaDialog(null,"Nova digitalització",true);
    }
}
```

Classe GestorBaseDades

Aquesta classe és la que en enginyeria de programari i en concret en els diagrames de col·laboració se'n diuen classes de control.

La càrrega del driver es fa usant la cadena : ***oracle.jdbc.OracleDriver***, tal com es pot veure en la següent part de codi:

```
public void obreDriver(){
  try
  {
    //càrrega del driver
    Class.forName("oracle.jdbc.OracleDriver")  ;

  }
  catch (Exception e)
  {
  {
     JOptionPane.showMessageDialog(null,e.getMessage());}

     System.out.println("ERROR: No puc carregar el driver JDBC.");
     return;
  }
}
```

Connexió a Oracle

Per a connectar amb oracle usem la cadena ***jdbc:oracle:thin:@portatil:1521:orcl*** . El codi següent ens ho mostra.

```
public void connecta()
{
  String url="jdbc:oracle:thin:@portatil:1521:orcl";

  // les connexions es fan usant el usuari scott/tiger
  try{
  conn = DriverManager.getConnection(url,"scott","tiger");}
  catch(SQLException sqle){
  JOptionPane.showMessageDialog(null,sqle.getMessage());}

}
```

Val a dir que els paràmetres ajustables en instal·lacions pròpies són:

La part ***jdbc:oracle:thin*** s'ha de deixar tal com està . Especifica el tipus de driver que subministra Oracle. Existeix un altre anomenat OCI, que si bé és més ràpid no és tant flexible. El driver THIN, malgrat perdi rapidesa permet la connexió via applets, no utilitza programari addicional en la part client, és independent de la

plataforma. El fet que estigui fet 100% en Java permet totes aquestes característiques.

portatil és el nom de l'ordinador que executa el sgbd Oracle. Altres opcions són: indicar "localhost", el nom de l'ordinador en qüestió o bé usar la seva adreça ip.
1521. Aquest nombre és el port que escolta Oracle. S'ha de deixar.
orcl. És el nom de la instància de la base de dades. Es pot canviar per una altra existent.

Les dades de l'usuari són les que hi ha per defecte. El nom és **scott** i la clau de pas és **tiger**.

```
import java.util.*;

import java.sql.*;
import javax.swing.*;

import java.io.*;

class GestorBaseDades
{
//
  Connection conn;
  Statement sta;
  ResultSet rs;


  /**
   * Constructor sense paràmetres. Per a realitzar correctament les
operacions
   * de càrrega, de connexió i de tancament s'usen els corresponents
mètodes.
   */
public GestorBaseDades()
{



// en la següent cadena cal substituir portatil per el nom del host i
orcl pel
// nom de la base de dades a que s'accedeix.




}

/**
 * Mètode que carrega el driver per a Oracle.
 */
public void obreDriver(){
  try
  {
      //càrrega del driver
      Class.forName("oracle.jdbc.OracleDriver")  ;

  }
```

```
  catch (Exception e)
  {
  {
        JOptionPane.showMessageDialog(null,e.getMessage());}

        System.out.println("ERROR: No puc carregar el driver JDBC.");
        return;
  }
}

/**
 * Mètode que fa la connexió usant la cadena
 */

public void connecta()
{
  String url="jdbc:oracle:thin:@portatil:1521:orcl";

  // les connexions es fan usant el usuari scott/tiger
  try{
      conn = DriverManager.getConnection(url,"scott","tiger");
  }
  catch(SQLException sqle)
  {
      JOptionPane.showMessageDialog(null,sqle.getMessage());
  }

}

/**
 * Mètode que crea una taula en la base de dades. L'array
bidimensional format
 * per 2 files i n columnes. La primera fila identifica els noms dels
camps i la segona
 * els tipus de les dades que s'emmagatzeran.
 * @param nom  el nom de la taula a crear.
 * @param camps  l'array bidimensional format per 2 files i n columnes
 * @param n la longitud de les columnes
 */
public void crearTaula(String nom,String[][] camps,int n)
{
  String cadena = "CREATE TABLE " +nom+"(";
for (int i=0; i<n;i++)
  if(i!=n-1)
  cadena = cadena + camps[i][0] + " "+ camps[i][1]+",";
else
  cadena = cadena + camps[i][0] + " "+ camps[i][1]+")";
System.out.println(cadena);
  try {
      sta = conn.createStatement();
      sta.executeUpdate(cadena);
      JOptionPane.showMessageDialog(null,"OK, taula creada
correctament");
  }
  catch (Exception ex)
  {

      JOptionPane.showMessageDialog(null,ex.getMessage());

  }
}
```

```
/**
 * Mètode que insereix dades en la taula. Les dades es passen com un
array.
 * @param nom  el nom de la taula de dades
 * @param dades  l'array de dades a passar.
 */
public void inserirDades(String nom,Object[] dades)
{
String cadena = "INSERT INTO " + nom + " values(";
  for (int i=0;i<dades.length;i++)
  {
      if(i!=dades.length-1)
      {
      if(dades[i] instanceof String || dades[i] instanceof Boolean)
dades[i]="'"+dades[i]+"'";
        cadena = cadena + dades[i]+",";
      }
      else
      {
        if(dades[i] instanceof String || dades[i] instanceof Boolean)
dades[i]="'"+dades[i]+"'";
        cadena = cadena + dades[i]+")";
      }
  }


  try
  {
      sta=conn.createStatement();
      sta.executeUpdate(cadena);
      sta.close();
  }
  catch (SQLException sqle)
  {
      JOptionPane.showMessageDialog(null,sqle.getMessage());
  }

}

/**
 * Mètode auxiliar usant en fase de proves que mostra la descripció
d'una taula.
 * @param taula el nom de la taula que volem descriure.
 */

public void descriuTaula(String taula)
{
 try
 {
   sta=conn.createStatement();
   ResultSet rs = sta.executeQuery("Select * from "+ taula);
   ResultSetMetaData meta = rs.getMetaData();
   for (int i=1; i<meta.getColumnCount()+1;i++)
   {
       System.out.println("nom:--> "+meta.getColumnName(i)+ "  tipus:-
-> " + meta.getColumnTypeName(i));
   }
   rs.close();sta.close();

 }
```

```
 catch (Exception ex)
 {
   JOptionPane.showMessageDialog(null,ex.getMessage());
 }



}

/**
 * Mètode que mostra una taula gràfica d'una taula que conté totes les
files que cumpleixen
 * certes condicions.
 * @param taula elnom de la taula
 * @param campsC els camps que volem comparar
 * @param condicions les condicions que volem que es cumpleixin
 */
public void dadesTaula(String taula,String[] campsC,String[]
condicions)
{
  try
  {
      String cadena ="";
      sta=conn.createStatement();
       ResultSet rs;
       for (int k1=0;k1<campsC.length;k1++)
                   if(k1!=campsC.length-1)
                     cadena = cadena +
campsC[k1]+"='"+condicions[k1]+"',";
                else
                    cadena = cadena +
campsC[k1]+"='"+condicions[k1]+"'";
                // System.out.println(cadena);
        rs = sta.executeQuery("select * from " + taula +" where "+
cadena);
        mostraTaula(rs);
        rs.close();
        sta.close();
  }
  catch (SQLException ex)

  {

  }

}

/**
 * Mètode que mostra una taula gràfica d'alguns camps que conté  totes
les files que cumpleixen
 * certes condicions.
 * @param taula el nom de la taula
 * @param camps els camps que volem mostrar
 * @param campsC els camps que volem comparar
 * @param condicions les condicions que volem es cumpleixin
 */
public void dadesTaula(String taula,String[] camps,String[]
campsC,String[] condicions)
{

  try
```

```
  {
      //if(camps.length==0)
      String cadena ="";
      sta=conn.createStatement();
       ResultSet rs;
      if(camps==null)
      {
      rs = sta.executeQuery("Select * from "+taula);
      }
      else if(condicions==null && campsC==null)
      {
        for (int k=0;k<camps.length;k++)
            if(k!=camps.length-1)
              cadena = cadena + camps[k]+",";
        else
            cadena = cadena + camps[k]+" ";
        rs = sta.executeQuery("select "+cadena+" from "+taula);
      }
      else
      {
        for (int k=0;k<camps.length;k++)
            if(k!=camps.length-1)
              cadena = cadena + camps[k]+",";
        else
            cadena = cadena + camps[k]+" from "+taula+ " where ";
        for (int k1=0;k1<campsC.length;k1++)
            if(k1!=campsC.length-1)
              cadena = cadena + campsC[k1]+"='"+condicions[k1]+"',";
        else
            cadena = cadena + campsC[k1]+"='"+condicions[k1]+"'";
         System.out.println(cadena);
        rs = sta.executeQuery("select "+cadena);
      }

      ResultSetMetaData metaRs = rs.getMetaData();

      String[] nomsColumnes = new String[metaRs.getColumnCount()];
      Vector[] cells = new Vector[nomsColumnes.length];
      Object[][] dades;

      for( int col = 0; col < nomsColumnes.length; col++) {
         nomsColumnes[col] = metaRs.getColumnName(col + 1);
         cells[col] = new Vector();
      }

// Inserim totes les dades de la consulta en el corresponent  vector
ceslls[i]

      while(rs.next())
      {

         for(int col = 0; col < nomsColumnes.length; col++)
         {
              Object casella = rs.getObject(nomsColumnes[col]);
              cells[col].add(casella);

         }

      }
      int files=cells[0].size();
      int columnes = nomsColumnes.length;
```

```
      dades = new Object[files][columnes];

      for (int i=0;i<files;i++)
        for (int j=0;j<columnes;j++)
            dades[i][j]=cells[j].elementAt(i);


      // imprimim les capçaleres
      for(int col = 0; col < nomsColumnes.length; col++)
         System.out.print(nomsColumnes[col].toUpperCase() + "\t");
      System.out.println();

      // imprimim les dades
      while(!cells[0].isEmpty())
      {
        for(int col = 0; col < nomsColumnes.length; col++)
            //System.out.print(cells[col].toString()+ "\t");
            //a mesura que els anem imprimint, els anem esborrant
            System.out.print(cells[col].remove(0).toString()+ "\t");
        System.out.println();

      }

  TaulaModelSIGOut tmso = new TaulaModelSIGOut(nomsColumnes,dades);
  JTable tau = new JTable(tmso);
  JFrame marc = new JFrame();

  JScrollPane jsp = new JScrollPane();

  jsp.getViewport().add(tau);
  marc.getContentPane().add(jsp);
   marc.setBounds(0,0,300,400);
  marc.setVisible(true);



  rs.close();
  sta.close();
  }
  catch (SQLException ex)
  {
      JOptionPane.showMessageDialog(null,ex.getMessage());
  }



}



/*
public void consulta(String cadena)
{
  try
  {

      sta = conn.createStatement();

      ResultSet rs = sta.executeQuery("select * from ciutatsMenorca
where nom ='"+cadena+"'");
```

```
      while(rs.next())
      {
        JOptionPane.showMessageDialog(null,rs.getInt(1)+"
"+rs.getString(2)+" "+rs.getInt(3));
      }

      rs.close();
      sta.close();

  }
  catch(SQLException sqle)
  {
      JOptionPane.showMessageDialog(null,sqle.getMessage());
  }

}

*/

/**
 * Mètode que ens desconnecta de la base de dades.
 */
public void desconnecta()
{
  try {
        conn.close();
  }
  catch (Exception ex) {

  }


}

/**
 * Mètode que accepta una transacció
 */
public void acceptaTransaccio()
{
  try
  {
  sta=conn.createStatement()  ;
  sta.executeUpdate("commit");
  }
  catch (SQLException sqlex) {
      JOptionPane.showMessageDialog(null,sqlex.getMessage());
  }

}

/**
 * Mètode que esborra una taula de la base de dades. La taula ha
d'existir.
 * @param taula el nom de la taula a esborrar
 */
public void esborraTaula(String taula)
{
  try
  {
      sta = conn.createStatement();
      sta.executeUpdate("drop table "+taula);
```

```
  }
  catch (SQLException sqlex)
  {
       JOptionPane.showMessageDialog(null,sqlex.getMessage());
  }

}

//per fer algunes proves.
static void main(String[] args){
  GestorBaseDades gbd = new GestorBaseDades();

  gbd.obreDriver();
  gbd.connecta();
  String[] camps1 = {"table_name"};
  String[] camps2 ={"owner"};
  String[] condicions = {"SCOTT"};
 //gbd.esborraTaula("pous_taulabbdd");
  gbd.esborraTaula("carrers_taulaBBDD");
  gbd.acceptaTransaccio();
  gbd.dadesTaula("carrers_TaulaBBDD",null,null,null);
  //gbd.descriuTaula("ciutatsMenorca");
  gbd.desconnecta();
}

/**
 * Mètode que mostra una finestra gràfica a partir d'un objecte
ResultSet que inclou
 * tota la informació, és a dir, els camps a mostrar, els noms dels
camps, els tipus dels
 * camps, el contingut de totes les files, etc.
 * @param rs ResultSet
 */
public void mostraTaula(ResultSet rs)
{
  try
  {
       ResultSetMetaData metaRs = rs.getMetaData();

        String[] nomsColumnes = new String[metaRs.getColumnCount()];
        Vector[] cells = new Vector[nomsColumnes.length];
        Object[][] dades;

        for( int col = 0; col < nomsColumnes.length; col++) {
             nomsColumnes[col] = metaRs.getColumnName(col + 1);
             cells[col] = new Vector();
        }

// Inserim totes les dades de la consulta en el corresponent  vector
ceslls[i]

        while(rs.next())
        {

             for(int col = 0; col < nomsColumnes.length; col++)
             {
                Object casella = rs.getObject(nomsColumnes[col]);
                cells[col].add(casella);

             }
```

```
        }
        int files=cells[0].size();
        int columnes = nomsColumnes.length;
        dades = new Object[files][columnes];

        for (int i=0;i<files;i++)
          for (int j=0;j<columnes;j++)
              dades[i][j]=cells[j].elementAt(i);
        TaulaModelSIGOut tmso = new
TaulaModelSIGOut(nomsColumnes,dades);
        JTable tau = new JTable(tmso);
        JFrame marc = new JFrame();

        JScrollPane jsp = new JScrollPane();

        jsp.getViewport().add(tau);
        marc.getContentPane().add(jsp);
        marc.setBounds(0,0,300,400);
        marc.setVisible(true);



   }
   catch (SQLException sqlex)
   {

   }


}

/**
 * Mètode que torna un array que conté els noms dels camps d'una
taula.
 * @param cadena el nom de la taula
 * @return String[] l'array que conté el noms del camp
 */
public String[] campsTaula(String cadena)
{
   String[] camps;
   ResultSetMetaData rsmtd;
   try
   {
       sta = conn.createStatement();
       ResultSet rs = sta.executeQuery("select * from "+cadena);
       rsmtd = rs.getMetaData();
       camps = new String[rsmtd.getColumnCount()];
       for (int i=0;i<rsmtd.getColumnCount();i++)
       camps[i]=rsmtd.getColumnName(i+1);

   }
   catch (SQLException ex)
   {

       ex.printStackTrace();
       camps=null;
   }

   return camps;
}
/**
```

```
 * Mètode que executa una sentència SQL passada com a paràmetre.
Aquesta funció captura la
 * variable cadena a partir d'un quadre de diàleg que la construeix.
És requisit que la
 * cadena passada com a paràmetre sigui ben formada. És
responsabilitat de la funció que crida
 * que això sigui així.
 * @param cadena el nom de la taula
 */

  public void dadesTaula(String cadena)
  {
      try
      {

      sta = conn.createStatement();
      rs = sta.executeQuery(cadena);
      mostraTaula(rs);
      }
      catch (SQLException ex)
      {
        ex.getMessage();
      }



  }
}
```

Interfície Graf

```
import java.util.*;
/**
 * Interfíci agafada i adaptada de la implementació feta per la
professora
 * Reyes Pavón Rial (pavon@uvigo.es)
 * de l'escola superior d'informàtica de la universitat de Vigo
(campus d'Ourense).
 * S'han suprimit els mètodes d'inserció d'arcs i de nodes així com
els de suprimir.
 */

public interface Graf
{
/**
 * Mètode que torna tots els nodes del graf.
 */
  public Enumeration nodes();
/**
 * Mètode que torna tots els arcs del graf.
 */
  public Enumeration arcs();
/**
 * Mètode que torna tots els arcs adjacenta al node v
 */
  public Enumeration adjacents(Node v);
}
```

Classe GrafSIG

Aquesta classe servirà de col·laboració per altres classes que executin algorismes de camins òptims.

```
import java.util.*;

/**
 *
 * <p>Títol: GrafSIG</p>
 * <p>Descripció: Classe que crea un graf implementant els mètodes de
la interfície
 * que declara</p>
 * <p>TFC- 2003/04-2: </p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 */
class GrafSIG implements Graf
{
  Enumeration arcs;
  Enumeration nodes;

  /**
   * Constructor del graf a partir d'un vector d'arcs passats com a
paràmetre.
   * @param unsArcs l'objecte Vector que conté tots els arcs.
   */
  public GrafSIG(Vector unsArcs)
  {
      Vector unsNodes = new Vector();
      arcs=unsArcs.elements();
      Iterator it = unsArcs.iterator();
      while (it.hasNext())
      {
        Arc a = ((Arc) it.next());
        if (!unsNodes.contains(a.nodeInicial()))
unsNodes.add(a.nodeInicial());
        if (!unsNodes.contains(a.nodeFinal()))
unsNodes.add(a.nodeFinal());
      }
      nodes=unsNodes.elements();
  }
  public Enumeration nodes()
  {
  return nodes;
  }
/**
 * Mètode que torna tots els arcs del graf.
 */
  public Enumeration arcs()
  {
  return arcs;
  }
/**
 * Mètode que torna tots els arcs adjacents al node v
 */
  public Enumeration adjacents(Node v)
  {
      return v.adjacents(arcs());
```

```
    }

}
```

Classe MarcSIG

Aquesta classe és la més important doncs és la que controla mostra la interfície gràfica i permet interacctuar amb l'usuari. No es troben implementades totes les funcionalitats i de les que hi ha s'haurien de retocar algunes.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import java.io.*;
import java.awt.geom.*;

/**
 * <p>Title: MarcSIG</p>
 * <p>Description: Aquesta classe és la peça central de la interfície
gràfica. Es troba
 * en fase inicial (sense finalitzar). Mostra tots els elements del
sig com poden ser
 * menús, quadres de llista, panells i respon als events de
ratolí.</p>
 * <p>TFC- 2003/04-2: </p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 */

public class MarcSIG extends JFrame
{
  /** nom del fitxer de capes*/
  private File nomFitxerCapes;
  /** nom del fitxer imatge*/
  private File nomFitxerImatge;
  /** valor lògic que indica si la capa té taula i per tant nom o no*/
  private boolean hihaNomTaulaDades=false;
  /** el gestor que connecta amb el sgbd*/
  private GestorBaseDades gbd = new GestorBaseDades();
  /** imatge (mapa) que es veu*/
  private ImageIcon iconaActual;
  /** la capa d'informació actual*/
  private Capa capaActiva ;
  /** aquí guardem les capes*/
  private Vector capes=new Vector();
  /** valor lògic que indica si el mapa és visible*/
  private boolean hihaMapa=false;
  /** vector on es guarden temporalment els arcs que es van dibuixant
per cada capa*/
  private Vector arcs=new Vector();
  private Vector nodes =new Vector();
  private boolean hihaArc = false;
  private boolean esPuntAillat = false;
  private boolean esLiniaAillada = false;
  private boolean esPoligonAillat = false;
  private int alçada;
  private float escala;
  private boolean hihaEscala=false;
  private String unitats= null;
  /** variable que es diu que es dibuixa un node*/
  private boolean hihaNode=false;
```

```
/** variable que es diu que es dibuixa una linia tancada*/
private boolean esLiniaTancada = false;

/** nom del node temporal */
private String nomNode = null;


/** variable que diu que es dibuixa un vèrtex*/
private boolean hihaVertex=false;



private Arc arc ;

//elements gràfics
private JPanel contentPane;
private JMenuBar barraMenus = new JMenuBar();
private JMenu menuFitxer = new JMenu();
private JMenu menuAjuda = new JMenu();
private JMenuItem menuAjudaSobre = new JMenuItem();
private JToolBar jToolBar = new JToolBar();
private JButton botoObrirMapa = new JButton();
private JButton botoTancaMapa = new JButton();
//imatges
private ImageIcon image1;
private ImageIcon image2;
private ImageIcon image3;
private ImageIcon nodeIcon;
private ImageIcon arcIcon;
private JLabel statusBar = new JLabel();
private BorderLayout borderLayout1 = new BorderLayout();
private JMenuItem menuFitxerDigitalitza = new JMenuItem();
private JMenuItem menuFitxerSurt = new JMenuItem();
private Image img;
private JScrollPane jScrollPane1 = new JScrollPane();
private JLabel imatgeEtiqueta;
private JMenu menuEdicio = new JMenu();
private JMenuItem menuEdicioNodes = new JMenuItem();
private JMenuItem menuEdicioVertexs = new JMenuItem();
private JMenuItem menuFitxerTancaMapa = new JMenuItem();
private JPanel panellCentral = new JPanel();
private BorderLayout borderLayout2= new BorderLayout();
private BorderLayout borderLayout3 = new BorderLayout();
private JComboBox comboArcs = new JComboBox();
private JMenuItem menuEdicioEscala = new JMenuItem();
private JComboBox comboNodes = new JComboBox();
private JMenuItem menuFitxerDesar = new JMenuItem();
private JMenuItem menuFitxerRecuperar = new JMenuItem();
private JMenu menuDibuixar = new JMenu();
private JMenu menuDades = new JMenu();
private JMenuItem menuDibuixiarVeureCapaActual = new JMenuItem();
private JMenu menuDibuixarVeureCapa = new JMenu();
private JMenu menuCapes = new JMenu();
private JMenuItem menuCapesModificar = new JMenuItem();
private JMenuItem menuCapesSeleccionar = new JMenuItem();
private JMenu menuCredits = new JMenu();
private JButton botoNode = new JButton();
private JButton botoArc = new JButton();
private JButton botoVertex = new JButton();
private JMenuItem menuEdicioArc = new JMenuItem();
```

```
  private JMenuItem menuEdicioPoligonAillat = new JMenuItem();
  private JMenuItem menuEdicioLiniaAillada = new JMenuItem();
  private JMenuItem menuEdicioPoligon = new JMenuItem();
  private JMenuItem menuEdicioPuntAillat = new JMenuItem();
  private JPanel panellCombos = new JPanel();
  private JButton botoPoligon = new JButton();
  private GridLayout gridLayout1 = new GridLayout(2,3);
  private JLabel nodesEtiqueta = new JLabel();
  private JLabel arcsEtiqueta = new JLabel();
  private JButton botoNovaCapa = new JButton();
  private JButton botoPuntAillat = new JButton();
  private JButton botoLiniaAillada = new JButton();
  private JButton botoPoligonAillat = new JButton();
  private JLabel capesEtiqueta = new JLabel();
  private JComboBox comboCapes = new JComboBox();
  private JMenuItem menuDibuixarTornaEnrera = new JMenuItem();
  private JMenuItem menuCapesNova = new JMenuItem();
  private JLabel poligonsEtiqueta = new JLabel();
  private JComboBox comboPoligons = new JComboBox();
  private JMenuItem menuDadesAssignar = new JMenuItem();
  private JMenuItem menuDadesAnalisi = new JMenuItem();
  private JMenu menuDadesConsulta = new JMenu();
  private JMenuItem menuDadesConsultaDades = new JMenuItem();
  private JMenuItem menuDadesConsultaLloc = new JMenuItem();
  private JMenuItem menuDadesEntreCapes = new JMenuItem();
//fi elements gràfics


/**
 * Constructor sense paràmetres
 */
  public MarcSIG()
  {
      //enableEvents(AWTEvent.WINDOW_EVENT_MASK);
      try
      {
        jbInit();
        gbd.obreDriver();
        gbd.connecta();

      }
      catch(Exception e)
      {
        e.printStackTrace();
      }
  }

  /**
   * Mètode inialitzador del marc. Incorpora tots els elements gràfics
presents i els
   * hi afegeix els escoltadors pertinents
   * @throws Exception
   */
  private void jbInit() throws Exception
  {
      image1 = new
ImageIcon(MarcSIG.class.getResource("openFile.gif"));
      image2 = new
ImageIcon(MarcSIG.class.getResource("closeFile.gif"));
      image3 = new ImageIcon(MarcSIG.class.getResource("help.gif"));
```

```
      contentPane = (JPanel) this.getContentPane();
      //li apliquem un borderLayout
      contentPane.setLayout(borderLayout1);

//      this.setSize(new Dimension(564, 350));
      this.setSize(this.getMaximumSize());
      this.setTitle("SIG -tfc 2004");
      this.addWindowListener(new java.awt.event.WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            this_windowClosing(e);
        }

        public void windowOpened(WindowEvent e) {
            this_windowOpened(e);
        }
      });
      statusBar.setText(" ");
      menuFitxer.setText("Fitxer");
      menuAjuda.setText("Ajuda");
      menuAjudaSobre.setText("Sobre ...");

      menuAjudaSobre.addActionListener(new ActionListener()  {
        public void actionPerformed(ActionEvent e) {
            menuAjudaSobre_actionPerformed(e);
        }
      });
      botoObrirMapa.setIcon(image1);
      botoObrirMapa.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            try{
            botoObrirMapa_actionPerformed(e);
            }
            catch(Exception ex){

            }


        }
      });
      botoObrirMapa.setToolTipText("Digitalitza Mapa");
      botoTancaMapa.setIcon(image2);
      botoTancaMapa.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            botoTancaMapa_actionPerformed(e);
        }
      });
      botoTancaMapa.setToolTipText("Tanca Mapa");
      menuFitxerDigitalitza.setText("Digitalitza mapa jpg");
      menuFitxerDigitalitza.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            menuFitxerDigitalitza_actionPerformed(e);
        }
      });
      menuFitxerSurt.setText("Sortir");
      menuFitxerSurt.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            menuFitxerSurt_actionPerformed(e);
```

```
        }
      });



      imatgeEtiqueta = new JLabel();//(new ImageIcon(img));
      //imatgeEtiqueta =new JLabelEtiqueta();
      imatgeEtiqueta.addMouseListener(new java.awt.event.MouseAdapter()
{
        public void mouseClicked(MouseEvent e) {
            imatgeEtiqueta_mouseClicked(e);
        }
      });
      imatgeEtiqueta.addMouseMotionListener(new
java.awt.event.MouseMotionAdapter() {
        public void mouseMoved(MouseEvent e)
        {
            try
            {
              imatgeEtiqueta_mouseMoved(e);
            }
            catch (Exception ex)
            {
              ex.printStackTrace();
              JOptionPane.showMessageDialog(null,ex.getMessage());
            }


        }
      });
      menuEdicio.setText("Edició");
      menuEdicioNodes.setText("Nodes ...");
      menuEdicioNodes.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            menuEdicioNodes_actionPerformed(e);
        }
      });
      menuEdicioVertexs.setText("Vertexs");
      menuEdicioVertexs.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e)
        {
            try
            {
              menuEdicioVertexs_actionPerformed(e);
            }
            catch(Exception exc)
            {

JOptionPane.showMessageDialog(null,"quin"+exc.getMessage());
            }


        }
      });
      menuFitxerTancaMapa.setText("Tanca mapa jpg");
      menuFitxerTancaMapa.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            menuFitxerTancaMapa_actionPerformed(e);
        }
```

```
      });
      panellCentral.setLayout(borderLayout3);


      comboArcs.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            comboArcs_actionPerformed(e);
        }
      });
      menuEdicioEscala.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            menuEdicioEscala_actionPerformed(e);
        }
      });
      menuFitxerDesar.setText("Desar capes");
      menuFitxerDesar.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e)
        {
            try
            {
              menuFitxerDesar_actionPerformed(e);
            }
            catch (Exception ex)
            {
              JOptionPane.showMessageDialog(null,ex.getMessage());

            }




        }
      });
      menuFitxerRecuperar.setText("Recuperar capes");
      menuFitxerRecuperar.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            menuFitxerRecuperar_actionPerformed(e);
        }
      });
      menuDibuixar.setText("Dibuixar");
      menuDades.setText("Dades temàtiques");
      menuDibuixiarVeureCapaActual.setText("Veure capa actual");
      menuDibuixiarVeureCapaActual.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e)
        {
        try {
            menuDibuixarVeureCapaActual_actionPerformed(e);
        }
        catch (Exception ex)
        {
        JOptionPane.showMessageDialog(null,ex.getMessage());
        }


        }
      });
      menuDibuixarVeureCapa.setText("Veure capa ...");
```

```
      menuCapes.setText("Capes");
      menuCapesModificar.setText("Modificar capa");
      menuCapesModificar.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            menuCapesModificar_actionPerformed(e);
        }
      });
      menuCapesSeleccionar.setText("Seleccionar capa ...");
      menuCredits.setText("Crèdits");

      botoNode.setPreferredSize(new Dimension(51, 27));
      botoNode.setToolTipText("Node");
      botoNode.setText("N");
      botoNode.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            botoNode_actionPerformed(e);
        }
      });
      botoArc.setPreferredSize(new Dimension(51, 27));
      botoArc.setToolTipText("Arc");
      botoArc.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            botoArc_actionPerformed(e);
        }
      });
      botoArc.setText("Arc");
      /** comboNodes.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            comboNodes_actionPerformed(e);
        }
      });
      */
      comboNodes.addActionListener(new java.awt.event.ActionListener()
{
        public void actionPerformed(ActionEvent e) {
            comboNodes_actionPerformed(e);
        }
      });
      botoVertex.setText("V");
      botoVertex.addActionListener(new java.awt.event.ActionListener()
{
        public void actionPerformed(ActionEvent e) {
            botoVertex_actionPerformed(e);
        }
      });
      botoVertex.setToolTipText("Vèrtex");
      botoVertex.setPreferredSize(new Dimension(51, 27));
      menuEdicioArc.setText("Arc ...");

      menuEdicioArc.addActionListener(new
java.awt.event.ActionListener()
      {
        public void actionPerformed(ActionEvent e)
        {
            try
            {
              menuEdicioArc_actionPerformed( e);
            }
            catch (Exception ex)
```

```
            {
              JOptionPane.showMessageDialog(null,ex.getMessage());
            }
        }
      });

      menuEdicioPoligonAillat.setText("Polígon aïllat ...");
      menuEdicioPoligonAillat.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            menuEdicioPoligonAillat_actionPerformed(e);
        }
      });
      menuEdicioLiniaAillada.setText("Línia aïllada ... ");
      menuEdicioLiniaAillada.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            menuEdicioLiniaAillada_actionPerformed(e);
        }
      });
      menuEdicioPoligon.setText("Poligon ...");
      menuEdicioPoligon.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e)
        {
            try
            {
              menuEdicioPoligon_actionPerformed(e);
            }
            catch (Exception ex)
            {
              ex.printStackTrace();
              JOptionPane.showMessageDialog(null,ex.getMessage());
            }


        }
      });
      menuEdicioPuntAillat.setText("Punt aïllat ...");
      menuEdicioPuntAillat.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            menuEdicioPuntAillat_actionPerformed(e);
        }
      });
      menuEdicioEscala.setText("Escala ...");
      botoPoligon.setToolTipText("Polígon");
      botoPoligon.setText("Pol");
      botoPoligon.addActionListener(new java.awt.event.ActionListener()
{
        public void actionPerformed(ActionEvent e)
        {
            try
            {
              botoPoligon_actionPerformed(e);
            }
            catch (Exception ex)
            {
              JOptionPane.showMessageDialog(null,ex.getMessage());
              ex.printStackTrace();
            }
```

```
        }
      });
      panellCombos.setLayout(gridLayout1);
      nodesEtiqueta.setHorizontalAlignment(SwingConstants.CENTER);
      nodesEtiqueta.setText("Nodes");
      arcsEtiqueta.setHorizontalAlignment(SwingConstants.CENTER);
      arcsEtiqueta.setText("Arcs");
      botoNovaCapa.setToolTipText("Capa de punts");
      botoNovaCapa.setText("nova Capa");
      botoNovaCapa.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            botoNovaCapa_actionPerformed(e);
        }
      });
      gridLayout1.setColumns(4);
      botoPuntAillat.setText("P *");
      botoPuntAillat.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            botoPuntAillat_actionPerformed(e);
        }
      });
      botoLiniaAillada.setText("L*");
      botoLiniaAillada.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            botoLiniaAillada_actionPerformed(e);
        }
      });
      botoPoligonAillat.setText("Pol *");
      botoPoligonAillat.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            botoPoligonAillat_actionPerformed(e);
        }
      });
      capesEtiqueta.setHorizontalAlignment(SwingConstants.CENTER);
      capesEtiqueta.setText("Capes");
      menuDibuixarTornaEnrera.setText("Torna enrera");
      menuDibuixarTornaEnrera.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            menuDibuixarTornaEnrera_actionPerformed(e);
        }
      });
      menuCapesNova.setText("Nova ...");
      menuCapesNova.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            menuCapesNova_actionPerformed(e);
        }
      });
      comboCapes.addActionListener(new java.awt.event.ActionListener()
{
        public void actionPerformed(ActionEvent e) {
            comboCapes_actionPerformed(e);
        }
      });
```

```
      poligonsEtiqueta.setHorizontalAlignment(SwingConstants.CENTER);
      poligonsEtiqueta.setText("Polígons");
      menuDadesAssignar.setText("Assignar taula dades a capa");
      menuDadesAssignar.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e)
        {
            try
            {
              menuDadesAssignar_actionPerformed(e);
            }
            catch (Exception ex)
            {
              JOptionPane.showMessageDialog(null,ex.getMessage());
            }


        }
      });
      menuDadesAnalisi.setText("Anàlisi ...");
      menuDadesConsulta.setText("Consultes");
      menuDadesConsultaDades.setText("Dades");
      menuDadesConsultaDades.addActionListener(new
java.awt.event.ActionListener() {;
        public void actionPerformed(ActionEvent e) {
            menuDadesConsultaDades_actionPerformed(e);
        }
      });
      menuDadesConsultaLloc.setText("Llocs");
      menuDadesEntreCapes.setText("Entre capes");
      jToolBar.add(botoObrirMapa);
      jToolBar.add(botoTancaMapa);
      jToolBar.add(botoNode, null);
      jToolBar.add(botoArc, null);
      jToolBar.add(botoVertex, null);
      menuFitxer.add(menuFitxerDigitalitza);
      menuFitxer.add(menuFitxerTancaMapa);
      menuFitxer.addSeparator();
      menuFitxer.add(menuFitxerDesar);
      menuFitxer.add(menuFitxerRecuperar);
      menuFitxer.addSeparator();
      menuFitxer.add(menuFitxerSurt);
      barraMenus.add(menuFitxer);
      barraMenus.add(menuCapes);
      barraMenus.add(menuEdicio);
      barraMenus.add(menuDibuixar);
      barraMenus.add(menuDades);
      barraMenus.add(menuAjuda);
      barraMenus.add(menuCredits);
      this.setJMenuBar(barraMenus);
      // aquí modificacions



//Els primers elements


      jScrollPane1.setAutoscrolls(true);
      panellCentral.add(panellCombos, BorderLayout.NORTH);
```

```
      panellCentral.add(jScrollPane1,  BorderLayout.CENTER);
      jScrollPane1.getViewport().add(imatgeEtiqueta,
BorderLayout.CENTER);
      panellCombos.add(nodesEtiqueta, null);
      panellCombos.add(arcsEtiqueta, null);
      panellCombos.add(poligonsEtiqueta, null);
      panellCombos.add(capesEtiqueta, null);
      panellCombos.add(comboNodes, null);
      panellCombos.add(comboArcs, null);
      panellCombos.add(comboPoligons, null);
      panellCombos.add(comboCapes, null);

      contentPane.add(jToolBar, BorderLayout.NORTH);
      contentPane.add(statusBar, BorderLayout.SOUTH);
      contentPane.add(panellCentral,BorderLayout.CENTER);

//      contentPane.add(jScrollPane1, BorderLayout.CENTER);

      menuEdicio.add(menuEdicioNodes);
      menuEdicio.add(menuEdicioArc);
      menuEdicio.add(menuEdicioVertexs);
      menuEdicio.add(menuEdicioPoligon);
      menuEdicio.addSeparator();
      menuEdicio.add(menuEdicioPuntAillat);
      menuEdicio.add(menuEdicioLiniaAillada);
      menuEdicio.add(menuEdicioPoligonAillat);
      menuEdicio.addSeparator();
      menuEdicio.add(menuEdicioEscala);
      menuDibuixar.add(menuDibuixiarVeureCapaActual);
      menuDibuixar.add(menuDibuixarVeureCapa);
      menuDibuixar.add(menuDibuixarTornaEnrera);
      menuDades.add(menuDadesAssignar);
      menuDades.add(menuDadesConsulta);
      menuDades.add(menuDadesAnalisi);
      menuCapes.add(menuCapesNova);
      menuCapes.add(menuCapesModificar);
      menuCapes.add(menuCapesSeleccionar);
      jToolBar.add(botoPoligon, null);
      jToolBar.add(botoPuntAillat, null);
      jToolBar.add(botoLiniaAillada, null);
      jToolBar.add(botoPoligonAillat, null);
      jToolBar.add(botoNovaCapa, null);
      menuDadesConsulta.add(menuDadesConsultaDades);
      menuDadesConsulta.add(menuDadesConsultaLloc);
      menuDadesConsulta.add(menuDadesEntreCapes);
      menuCredits.add(menuAjudaSobre);
      //Els elements primers


  }

  //Acció fitxer sortir
  public void jMenuFileExit_actionPerformed(ActionEvent e) {
      System.exit(0);
  }
  //Acció ajuda sobre...
  public void menuAjudaSobre_actionPerformed(ActionEvent e) {
      ajudaMapaBoto_actionPerformed(e);
  }
  //Mètode sobrecarregat que permet tancar l'aplicació quan es tanca
la finestra.
```

```
  protected void processWindowEvent(WindowEvent e) {
      super.processWindowEvent(e);
      if (e.getID() == WindowEvent.WINDOW_CLOSING) {
        jMenuFileExit_actionPerformed(null);
      }
  }


  // mètode que segons l'acció que es tracti, registra nodes, vertexs,
consulta polígons, etc. Diriem
  // que és el més important. Aquest n'és dels mètodes més importants
doncs registra tota
  // l'activitat realitzada amb el ratolí en la zona de dibuix.
  void imatgeEtiqueta_mouseClicked(MouseEvent e)
  {

      Vertex vertex = new Vertex();

      float fx =(float) e.getPoint().getX();
      float fy =(float) e.getPoint().getY();


      if (hihaMapa && hihaEscala)
      {
        fy = alçada -fy;
        vertex.afegeix(fx,fy);
        if (hihaNomTaulaDades )
        {

            Iterator it =
capaActiva.getObjectesGeografics().iterator();

            if(capaActiva.getTipusInt()==Capa.PUNTS)
            {
              while (it.hasNext())
              {
                  Arc auxiliar = ((Arc) it.next());

                  float fx1 = auxiliar.nodeFinal().getX();
                  float fy1 = auxiliar.nodeFinal().getY();

                  if(Math.abs(fx1-fx)<3 && Math.abs(fy1-fy)<3)
                  {

              String[] camp ={"id_punt"};
              String[] condicio= new String[1];
              condicio[0]=auxiliar.nom;

gbd.dadesTaula(capaActiva.getNomTaulaDades(),camp,condicio);


                  }
              }
            }
            if(capaActiva.getTipusInt()==Capa.LINIES)
            {

              Vertex v = new Vertex();
              v.afegeix(fx,fy);
              while (it.hasNext())
              {
```

```
                    Arc arcLinia = (Arc) it.next();
                    if(arcLinia.conteVertex(v))
                    {
                      String[] camp={"id_linia"};
                      String[] condicio= new String[1];
                      condicio[0]=arcLinia.nom;


gbd.dadesTaula(capaActiva.getNomTaulaDades(),camp,condicio);
                    }
                  }
                }
                if (capaActiva.getTipusInt()==Capa.POLIGONS)
                {
                  Vertex v = new Vertex();
                  v.afegeix(fx,fy);
                  while (it.hasNext())
                  {
                      Poligon pol = (Poligon) it.next();
                      if(pol.conteVertex(v))
                      {
                        String[] camp={"id_poligon"};
                        String[] condicio= new String[1];
                        condicio[0]=pol.nom;


gbd.dadesTaula(capaActiva.getNomTaulaDades(),camp,condicio);
                      }
                  }

                }

          }

        }
        String valorX = String.valueOf(fx);
        String valorY= String.valueOf(fy);
        int polsacions = e.getClickCount();


        if( hihaNode)
        {

          // com hi ha node el guardem.
          Node n=null;
          try
          {
              n = new Node();
          }
          catch (Exception ex)
          {

          }


          n.afegeix(fx,fy);
          n.setNom(nomNode);
          // l'afegim al combo.
          comboNodes.setRequestFocusEnabled(false);
          comboNodes.addItem(n);
          hihaNode = false;
```

```
        this.setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
      }

      if (hihaVertex)
      {

        //Els vèrtexs s'introdueixen un darrera l'altre, és obligació
de l'operador
        //inserir-los en el correcte ordre.

        arc.afegirVertex(vertex);
        if (polsacions ==2)
        {
            hihaVertex=false;
            this.setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
        }

      }
      if (esPuntAillat)
      {

        //amb tota una operació afegim el node inicial, el final i
l'arc.
        Node n = vertex.toNode();
        n.setNom(arc.nom);
        comboNodes.addItem(n);
        arc.setNodeFinal(n);
        arc.setNodeInicial(n);
        arcs.add(arc);
        comboArcs.addItem(arc);
        capaActiva.setObjecteGeografic(arc);
        //menuDibuixar.setEnabled(true);
        esPuntAillat=false;
        this.setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
      }

      if(esLiniaAillada)
      {

        if(polsacions==1)
        {
            if(arc.nodeInicial()==null)
            {
              Node d = vertex.toNode();
              String nom = JOptionPane.showInputDialog(this,"Nom
d'aquest node");
              d.setNom(nom);
              arc.setNodeInicial(d);
              comboNodes.addItem(d);
            }

            arc.afegirVertex(vertex);
        }
        else  //s'han fet 2 o més polsacions
        {
            Node d1 = vertex.toNode();
            String nom1 = JOptionPane.showInputDialog(this,"Nom
d'aquest node");
            d1.setNom(nom1);
            arc.setNodeFinal(d1);
            comboNodes.addItem(d1);
```

```
            //ara convertim el primer i el darrer a Node.
            // i els substituim a l'arc.
            arcs.add(arc);
            comboArcs.addItem(arc);
            capaActiva.setObjecteGeografic(arc);

            esLiniaAillada=false;
        }

        this.setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
      }
      if(esPoligonAillat)
      {

        if (polsacions==1)
        {
            if(arc.nodeInicial()==null)
            {
              Node d = vertex.toNode();
              String nom = JOptionPane.showInputDialog(this,"Nom
d'aquest node");
              d.setNom(nom);
              arc.setNodeInicial(d);
              arc.setNodeFinal(d);
              comboNodes.addItem(d);
            }

            arc.afegirVertex(vertex);

        }
        else //s'han fet 2 o més polsacions
        {
            arc.afegirVertex(vertex);
            arcs.add(arc);
            comboArcs.addItem(arc);
            Poligon p = new Poligon(arc.nom);
            p.afegirArc(arc);
            comboPoligons.addItem(p);
            capaActiva.setObjecteGeografic(p);
            esPoligonAillat=false;
        }

        this.setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
      }

  }

  //acció que s'executa quan es pren sobre digitalitza mapa

  void botoObrirMapa_actionPerformed(ActionEvent e) throws Exception
  {
      comboArcs.removeAllItems();
      JFileChooser jfc = new JFileChooser("D:\\uoc2004_2\\tfc-
bbdd\\tfcproject\\SIG\\src\\mapes2");
      int k=jfc.showOpenDialog(this);
      if (k==1) throw new Exception();

      ImageIcon nova = new
ImageIcon(jfc.getSelectedFile().getAbsolutePath());
      nomFitxerImatge = jfc.getSelectedFile().getAbsoluteFile();
      iconaActual= nova;
```

```
      imatgeEtiqueta.setIcon(nova);
      alçada = nova.getIconHeight();
      System.out.println("alçada "+nova.getIconHeight()+" i
ample"+nova.getIconWidth());
      menuEdicio.setEnabled(true);

      hihaMapa=true;

      String a1 = "pixels";
      float a2 = 1f;


      System.out.println(a2+ " "+a1);
      escala = a2;
      unitats=a1;
      hihaEscala=true;
      novaCapa();


  }

  void menuFitxerDigitalitza_actionPerformed(ActionEvent e)
  {


      try
      {
        botoObrirMapa_actionPerformed(e);
      }
      catch(Exception ex)
      {

      }
  }

// acció que s'executa quan sortim de l'aplicatiu

  void menuFitxerSurt_actionPerformed(ActionEvent e)
  {
      surt();
  }

  /**
   * mètode surt, compartit per vàris objectes (la pròpia finestra i
el menú).
   */

  void surt()
  {
      int k = JOptionPane.showConfirmDialog(this,"De debó que vols
sortir ?","Sortir de SigTFC-2004-1",JOptionPane.OK_CANCEL_OPTION);
      if (k==0)
      {
        gbd.desconnecta();
        JOptionPane.showMessageDialog(this,"Fins la propera","Sortir
SigTFC-2004-1",JOptionPane.INFORMATION_MESSAGE);
        System.exit(0);
      }
  }

  //mètode que recull els moviments del ratolí i els mostra en la
```

```
barra d'estat per indicar la
  //posició

  void imatgeEtiqueta_mouseMoved(MouseEvent e) throws Exception
  {
      if (hihaMapa)
      {
        String valorX = String.valueOf(e.getPoint().getX());
        String valorY= String.valueOf(alçada-e.getPoint().getY());
        statusBar.setText(valorX+","+valorY);
        float fx =(float) e.getPoint().getX();
        float fy =(float) e.getPoint().getY();
        fy =alçada -fy;
      }

  }

  //en tancar-se la finestra
  void this_windowClosing(WindowEvent e)
  {
      surt();
  }

  //en carregar-se la finestra activem aquells elements del menú que
ens interessen
  void this_windowOpened(WindowEvent e)

  {
      // com no hi ha mapa deshabilitem les accions d'edició.
      menuEdicio.setEnabled(false);
      menuDibuixar.setEnabled(true);
      menuCapes.setEnabled(false);
      menuDades.setEnabled(true);
      menuFitxerTancaMapa.setEnabled(false);
      menuFitxerDesar.setEnabled(true);
      botoArc.setEnabled(false);
      botoVertex.setEnabled(false);
      botoNode.setEnabled(false);
      botoPoligon.setEnabled(false);
      botoPuntAillat.setEnabled(false);
      botoLiniaAillada.setEnabled(false);
      botoPoligonAillat.setEnabled(false);
  }

  // mes que tancar mapa, desassignem la imatge del panell central.
  void menuFitxerTancaMapa_actionPerformed(ActionEvent e)
  {
      imatgeEtiqueta.setIcon(null);

  }

  // com el mètode d'abans


  void botoTancaMapa_actionPerformed(ActionEvent e)
  {

      menuFitxerTancaMapa_actionPerformed(e);
  }

//en seleccionar un element del quadre combinat d'arcs fem que mostri
```

```
la informació de l'arc
  void comboArcs_actionPerformed(ActionEvent e)
  {
      //en agafar un ítem del combo arcs, cridem a un ArcDialog que ens
mostra tota la informació de
      //que disposem.

      if (comboArcs.getItemCount()!=0)
      {
        int i = comboArcs.getSelectedIndex();

        ArcDialog ad = new ArcDialog(this,"informació arc",true,((Arc)
arcs.get(i)).toString2(),"");
        //si hi ha dades temàtiques, es mostren.
        if(capaActiva.getNomTaulaDades()!=null &&
capaActiva.getTipusInt()!=Capa.POLIGONS)
        {
            String[] camp = new String[1];
            if (capaActiva.getTipusInt()==Capa.PUNTS)
            {
              camp[0] ="id_punt";
            }
            else
              camp[0] ="id_linia";

            String[] condicio= new String[1];
            condicio[0]=((Arc) comboArcs.getSelectedItem()).nom;

      gbd.dadesTaula(capaActiva.getNomTaulaDades(),camp,condicio);
        }

        ad.setBounds(0,0,300,200);

      ad.setVisible(true);

      }
  }

  //en seleccionar un element del quadre combinat d'arcs fem que
mostri la informació de l'arc

  void comboNodes_actionPerformed(ActionEvent e)
  {
      //Dels nodes introduïts, mostrarem les seves característiques i
els arcs a que pertanyen.

      if (comboNodes.getItemCount()!=0)
      {

        int i = comboNodes.getSelectedIndex();
        String cadena1="";
        if (comboArcs.getItemCount()!=0)
        {
            Arc[]  arcsArray;
            arcsArray =  ((Node)
comboNodes.getSelectedItem()).arcsNode(arcs.elements());

            for(int k=0;k<arcsArray.length;k++)
              cadena1 = cadena1+"\n"+arcsArray[k].nom;

        }
```

```
        ArcDialog ad = new
ArcDialog(this,comboNodes.getName(),true,((Node)
comboNodes.getItemAt(i)).toString2(),cadena1);
        ad.setBounds(0,0,300,200);
        ad.setVisible(true);
      }
  }

  //simplement mostra la informació de l'autor.

  void ajudaMapaBoto_actionPerformed(ActionEvent e)
  {
      JFrame sobre = new JFrame("TFC-SIG");
      JLabel etiqueta = new JLabel("Joan Manuel González Febrer uoc
2004");
      sobre.getContentPane().add(etiqueta);
      sobre.setSize(300,50);
      Dimension screenSize =
Toolkit.getDefaultToolkit().getScreenSize();
      Dimension frameSize = sobre.getSize();
      if (frameSize.height > screenSize.height) {
        frameSize.height = screenSize.height;
      }
      if (frameSize.width > screenSize.width) {
        frameSize.width = screenSize.width;
      }
      sobre.setLocation((screenSize.width - frameSize.width) / 2,
(screenSize.height - frameSize.height) / 2);
      sobre.setVisible(true);


  }

  //acció que permet canviar l'escala
  void menuEdicioEscala_actionPerformed(ActionEvent e)
  {
      if(JOptionPane.showConfirmDialog(this,"L'escala actual és 1 pixel
: "+(int) escala +" "+ unitats)==0)
      {
        EscalaDialog a =    new EscalaDialog(this,"Escala",true);
        String a1=    (String) a.obteEscala()[0];
        float a2 = ((Float) a.obteEscala()[1]).floatValue();
        System.out.println(a2+ " "+a1);
        escala = a2;
        unitats=a1;
      }


  }

//acció que permet introduir un node
  void menuEdicioNodes_actionPerformed(ActionEvent e)
  {
      hihaNode = true;
      // ara tots els clics del ratolí són considerats com a nodes//

      nomNode = JOptionPane.showInputDialog(this,"Dóna un nom al
node");
      this.setCursor(new Cursor(Cursor.CROSSHAIR_CURSOR));
```

```
  }

  //acció que permet desar totes les capes
  void menuFitxerDesar_actionPerformed(ActionEvent e) throws Exception

  {


      if(capes.size()==0) throw new Exception("No hi ha capes a
desar");
      JFileChooser jfc = new JFileChooser("D:\\uoc2004_2\\tfc-
bbdd\\tfcproject\\SIG");
      jfc.showSaveDialog(this);
      File fitxer= jfc.getSelectedFile();
      try
      {
        FileOutputStream fos = new FileOutputStream(fitxer);
        ObjectOutputStream ous = new ObjectOutputStream(fos);
        ous.writeObject(capes);
        //escrivim ara dos elements importants: l'escala usada, és a
dir, a que
        //equival cada pixel i les unitats usades.
        //escrivim l'escala.
        ous.writeFloat(escala);
        ous.writeObject(unitats);
        ous.writeObject(nomFitxerImatge);
        ous.flush();
        ous.close();
        nomFitxerCapes=fitxer;
        this.setTitle(this.getTitle()+"
"+nomFitxerCapes.getCanonicalPath());

      }
      catch (Exception ex)
      {
      }
  }

//acció que permet recuperar les capes

  void menuFitxerRecuperar_actionPerformed(ActionEvent e)
  {
      JFileChooser jfc = new JFileChooser("D:\\uoc2004_2\\tfc-
bbdd\\tfcproject\\SIG");
      int k=    jfc.showOpenDialog(this);

      File f = jfc.getSelectedFile();
      System.out.println(f.getAbsolutePath());
      try
      {
        FileInputStream fis = new FileInputStream(f);
        ObjectInputStream ois = new ObjectInputStream(fis);

        capes = (Vector) ois.readObject();

        escala = ois.readFloat();
        hihaEscala=true;
        unitats = (String) ois.readObject();
        nomFitxerImatge = (File) ois.readObject();
        iconaActual = new ImageIcon(nomFitxerImatge.getAbsolutePath());
        imatgeEtiqueta.setIcon(iconaActual);
```

```
        hihaMapa=true;
        alçada=iconaActual.getIconHeight();

        //posem els valors i mostrem l'escala.
        JOptionPane.showMessageDialog(this,Float.toString(escala)
+"unitats en " + unitats);

        System.out.println(capes.size());

        buidaTot();
        //Carreguem les capes a comboCapes.
        capaActiva = (Capa) capes.firstElement();
        transformaMarcSig(capaActiva.getTipusInt());
        Iterator it = capes.iterator();
        while (it.hasNext())
        {
            comboCapes.addItem((Capa) it.next());
        }
        ompleCombos(capaActiva);


        ois.close();
        nomFitxerCapes=f;
        this.setTitle(nomFitxerCapes.getCanonicalPath());
        menuDibuixar.setEnabled(true);
      }
      catch (Exception ex)
      {
        ex.printStackTrace();

      }


  }

  //dibuixa els elements gràfics introduits sobre el llenç de color
gris.
  void menuDibuixarVeureCapaActual_actionPerformed(ActionEvent e)
throws Exception
  {
      //en veure capa actual fem el següent
      //1, treiem la imatge del mapa
      // obtenim el context gràfic del panel.
      if(capaActiva.getNomTaulaDades()!=null)
hihaNomTaulaDades=true;

      Color colorArc = Color.blue;
      Color colorNode = Color.white;

      Graphics g = imatgeEtiqueta.getGraphics();
      if (capaActiva==null ) throw new Exception("No hi ha capes");
      if(capaActiva.getObjectesGeografics().isEmpty()) throw new
Exception("La capa activa no té objectes");
      if(capaActiva.getTipusInt()==Capa.PUNTS &&
comboArcs.getItemCount()!=0)
      {
        Iterator it = capaActiva.getObjectesGeografics().iterator();
        while (it.hasNext()){
            Arc a = (Arc) it.next();
            //com l'arc és d'un únic punt, llavors.
            Vertex v = a.nodeFinal();
```

```
        g.setColor(colorNode);
        g.fillRect((int)(v.getX()-4),alçada-(int)(v.getY()-4),8,8);}


      }
      //ara són polígons
      if (capaActiva.getTipusInt()==Capa.POLIGONS &&
!capaActiva.getObjectesGeografics().isEmpty())
      {
        Iterator it = capaActiva.getObjectesGeografics().iterator();
        while (it.hasNext())
        {
            Poligon p = (Poligon) it.next();
            Iterator it2 = p.arcs.iterator();
            while (it2.hasNext())
            {
              Arc ar= (Arc) it2.next();
              int ys[] = ar.obtenirY();
              for (int j =0;j<ys.length;j++) ys[j]=alçada-ys[j];
              g.setColor(colorNode);
              g.fillRect((int)(ar.nodeFinal().getX()-3),alçada-
(int)(ar.nodeFinal().getY()-3),6,6);

              g.fillRect((int)(ar.nodeInicial().getX()-3),alçada-
(int)(ar.nodeInicial().getY()-3),6,6);
              g.setColor(colorArc);
              g.drawPolyline(ar.obtenirX(),ys,ar.obtenirX().length);
            }
        }
      }
      //ara són línies
      if (capaActiva.getTipusInt()==Capa.LINIES &&
!capaActiva.getObjectesGeografics().isEmpty())
      {

        Iterator it = capaActiva.getObjectesGeografics().iterator();

        while (it.hasNext())
        {

            Arc arc1 = (Arc) it.next();
            int ys[] = arc1.obtenirY();
            for (int j =0;j<ys.length;j++) ys[j]=alçada-ys[j];
            g.setColor(colorNode);
            g.fillRect((int)(arc1.nodeFinal().getX()-3),alçada-
(int)(arc1.nodeFinal().getY()-3),6,6);

            g.fillRect((int)(arc1.nodeInicial().getX()-3),alçada-
(int)(arc1.nodeInicial().getY()-3),6,6);
            g.setColor(colorArc);
            g.drawPolyline(arc1.obtenirX(),ys,arc1.obtenirX().length);
        }
      }

  }



  void botoNode_actionPerformed(ActionEvent e)
  {
      menuEdicioNodes_actionPerformed( e) ;
```

```
  }

  void botoArc_actionPerformed(ActionEvent e)
  {
      try
      {
        menuEdicioArc_actionPerformed( e);
      }
      catch (Exception ex)
      {
        JOptionPane.showMessageDialog(this,"nnn"+ex.getMessage());

      }


  }

  //acció que permet introduir els vèrtexs usant el ratolí

  void menuEdicioVertexs_actionPerformed(ActionEvent e) throws
Exception
  {

      if(comboArcs.getItemCount()==0) throw new Exception("no hi ha
arcs");
      //informem a l'usuari de com a de incloure els vèrtexs.
      String cadena = "Els vèrtexs els has d'anar introduïnt\n, clicant
sobre el mapa, per això triaràs\n"+
                              "un arc existent i amb el ratolí
clicaràs\n de forma que formin una polilínia.\n Per finalitzar,"+
                              "clica dos cops seguits en el\n vèrtex
anterior al node final";

      //indiquem que ara els clics són per crear vèrtexs.
      hihaVertex=true;
      Arc[] llista=new Arc[comboArcs.getItemCount()];
      for (int i=0;i<comboArcs.getItemCount();i++)
        llista[i]=(Arc) comboArcs.getItemAt(i);
      arc = (Arc) JOptionPane.showInputDialog(this,"Tria l'arc al qual
inserirem els vèrtexs","Vèrtex
nou",JOptionPane.QUESTION_MESSAGE,null,llista,llista[0]);

      JOptionPane.showMessageDialog(this,cadena,"Introducció de
vèrtexs",JOptionPane.INFORMATION_MESSAGE);
      this.setCursor(new Cursor(Cursor.CROSSHAIR_CURSOR));
  }

//acció que permet introduir els vèrtexs usant el botó apropiat
  void botoVertex_actionPerformed(ActionEvent e)
  {
      try
      {
        menuEdicioVertexs_actionPerformed(e);
      }
      catch(Exception exc)
      {
        JOptionPane.showMessageDialog(this,exc.getMessage());
      }
  }

  //acció que permet registrar un arc. Dóna una excepció si no hi ha
```

```
nodes disponibles.
  //És suficient que n'hi hagi un ja que pot ser un punt.

  void menuEdicioArc_actionPerformed(ActionEvent e)  throws Exception
  {
      arc =new Arc(JOptionPane.showInputDialog(this,"nou arc","dóna un
nom a l'arc",JOptionPane.QUESTION_MESSAGE));
      if (comboNodes.getItemCount()==0) throw new Exception("No hi ha
nodes per a fer un arc");

      Node[] llista=new Node[comboNodes.getItemCount()];
      for (int i=0;i<comboNodes.getItemCount();i++)
        llista[i]=(Node) comboNodes.getItemAt(i);
      Node nodeInicial = (Node)
JOptionPane.showInputDialog(this,"Escull el node origen",
                                                  "nou
arc",JOptionPane.QUESTION_MESSAGE,null,llista,llista[0]);
      Node nodeFinal = (Node) JOptionPane.showInputDialog(this,"Escull
el node final",
                                                  "nou
arc",JOptionPane.QUESTION_MESSAGE,null,llista,llista[0]);

      nodeInicial.augmenta();
      nodeFinal.augmenta();
      arc.setNodeInicial(nodeInicial);
      arc.setNodeFinal(nodeFinal);

      if (capaActiva.getTipusInt()==Capa.LINIES ||
capaActiva.getTipusInt()==Capa.PUNTS)
      {

        capaActiva.setObjecteGeografic(arc);

      }

      arcs.add(arc);

      comboArcs.addItem(arc);


  }


//acció que introdueix una línia aillada
  void menuEdicioLiniaAillada_actionPerformed(ActionEvent e)
  {
        botoLiniaAillada_actionPerformed(e);
  }

  void menuEdicioPoligonAillat_actionPerformed(ActionEvent e)
  {
      esPoligonAillat = true;
      arc = new Arc(JOptionPane.showInputDialog(this,"Dóna un nom"));
      this.setCursor(new Cursor(Cursor.CROSSHAIR_CURSOR));
  }


  // Mètode que canvia els estats dels ítems del menú segons sigu el
tipus de capa.
  // Això és útil per a evitar que l'usuari s'equivoqui.
```

```
public void transformaMarcSig(int tipus)
{
    if (tipus==Capa.PUNTS)
    {
      menuCapes.setEnabled(true);
      //accions menu que es modifiquen
      menuEdicio.setEnabled(true);
      menuEdicioNodes.setEnabled(false);
      menuEdicioArc.setEnabled(false);
      menuEdicioVertexs.setEnabled(false);
      menuEdicioPoligon.setEnabled(false);
      menuEdicioPuntAillat.setEnabled(true);
      menuEdicioLiniaAillada.setEnabled(false);
      menuEdicioPoligonAillat.setEnabled(false);
      menuEdicioEscala.setEnabled(true);

      //botons que es modifiquen
      botoNode.setEnabled(false);
      botoArc.setEnabled(false);
      botoVertex.setEnabled(false);
      botoPoligon.setEnabled(false);
      botoPuntAillat.setEnabled(true);
      botoLiniaAillada.setEnabled(false);
      botoPoligonAillat.setEnabled(false);
    }
    if (tipus==Capa.LINIES)
    {
      menuCapes.setEnabled(true);
      //accions menu
      menuEdicio.setEnabled(true);
      menuEdicioNodes.setEnabled(true);
      menuEdicioArc.setEnabled(true);
      menuEdicioVertexs.setEnabled(true);
      menuEdicioPoligon.setEnabled(false);
      menuEdicioPuntAillat.setEnabled(false);
      menuEdicioLiniaAillada.setEnabled(true);
      menuEdicioPoligonAillat.setEnabled(false);
      menuEdicioEscala.setEnabled(true);

      //botons
      botoNode.setEnabled(true);
      botoArc.setEnabled(true);
      botoVertex.setEnabled(true);
      botoPoligon.setEnabled(false);
      botoPuntAillat.setEnabled(false);
      botoLiniaAillada.setEnabled(true);
      botoPoligonAillat.setEnabled(false);


    }
    if (tipus==Capa.POLIGONS)
    {
      menuCapes.setEnabled(true);
      //accions menu
      menuEdicio.setEnabled(true);
      menuEdicioNodes.setEnabled(true);
      menuEdicioArc.setEnabled(true);
      menuEdicioVertexs.setEnabled(true);
      menuEdicioPoligon.setEnabled(true);
      menuEdicioPuntAillat.setEnabled(false);
      menuEdicioLiniaAillada.setEnabled(false);
```

```
        menuEdicioPoligonAillat.setEnabled(true);
        menuEdicioEscala.setEnabled(true);

        //botons
        botoNode.setEnabled(true);
        botoArc.setEnabled(true);
        botoVertex.setEnabled(true);
        botoPoligon.setEnabled(true);
        botoPuntAillat.setEnabled(false);
        botoLiniaAillada.setEnabled(false);
        botoPoligonAillat.setEnabled(true);


      }

  }


  //quan seleccionem menu punt aillat.
  void menuEdicioPuntAillat_actionPerformed(ActionEvent e)
  {

      //deleguem a botoPuntAillat_actionPerformed
      botoPuntAillat_actionPerformed(e);
  }

  //accio que permet introduir un punt aillat
  void botoPuntAillat_actionPerformed(ActionEvent e)
  {
      esPuntAillat=true;

      arc = new Arc(JOptionPane.showInputDialog(this,"Dóna un nom a
l'arc"));
      this.setCursor(new Cursor(Cursor.CROSSHAIR_CURSOR));
  }

  //acció que perrmet introduir una línia aillada
  void botoLiniaAillada_actionPerformed(ActionEvent e)
  {
      esLiniaAillada = true;

      arc = new Arc(JOptionPane.showInputDialog(this,"Dóna un nom"));
      this.setCursor(new Cursor(Cursor.CROSSHAIR_CURSOR));

  }

  //acció que crea un polígon nou. Llença una excepció si no hi ha
arcs
  void botoPoligon_actionPerformed(ActionEvent e) throws Exception
  {

      if (arcs.size()==0) throw new Exception("No hi ha arcs");
      //obrim una caixa de diàleg per escollir els arcs que formaran el
polígon

      PoligonDialog polD = new PoligonDialog(this,"nou
poligon",true,arcs);

      //fem la caixa visible
      //polD.setVisible(true);
      Poligon pAux  = polD.getPoligon();
```

```
      //dotem de la topologia al polígon.
      pAux.topologia();
      //afegim el poligon a la capa.
      capaActiva.setObjecteGeografic(pAux);
      comboPoligons.addItem(pAux);
      // el poligon on és ?
      menuDibuixar.setEnabled(true);

  }

  //acció que mostra la imatge sobre la qual definim els objectes
geogràfics
  void menuDibuixarTornaEnrera_actionPerformed(ActionEvent e)
  {
      imatgeEtiqueta.setIcon(iconaActual);
  }
  // acció que permet crear polígons per delegació

  void menuEdicioPoligon_actionPerformed(ActionEvent e) throws
Exception
  {
      //deleguem a botoPoligon_actionPerformed
      botoPoligon_actionPerformed(e);
  }

//mètode que crea una nova capa
  public void novaCapa()
  {
      Capa cap;
      String[] tipus = {"punts","línies","polígons"};
      String tipusCapa = (String)
JOptionPane.showInputDialog(null,"indica tipus de capa","nova
capa",JOptionPane.INFORMATION_MESSAGE,null, tipus, tipus[0]);
      String nomCapa= JOptionPane.showInputDialog(this,"nom de la
capa","nova capa",JOptionPane.INFORMATION_MESSAGE);
      if (tipusCapa.equals(tipus[0]))
      {
        cap = new Capa(nomCapa,0,"null",escala,unitats);
      }
      else
        if (tipusCapa.equals(tipus[1]))
        {
            cap=new Capa(nomCapa,1,"null",escala,unitats);
        }
        else
        {
            cap = new Capa(nomCapa,2,"null",escala,unitats);
        }

        //com que tenim capa ara ja podem indicar
        capes.add(cap);
        capaActiva = cap;
        comboCapes.addItem(cap);

        comboCapes.setSelectedItem(cap);

        //segons el tipus de capa, cal modificar les accions del menú.
        transformaMarcSig(cap.getTipusInt());
  }

  //acció de menu que crea una nova capa
```

```
  void menuCapesNova_actionPerformed(ActionEvent e)
  {
      //En nova capa hem de buidar els combos i els vectors que tinguin
arcs i nodes.
      buidaTot();
      novaCapa();
  }

  //mètode que en canviar de capa, carrega als combos  i els vectors
d'arcs i nodes
  //amb els elements oportuns


  public void buidaTot()
  {
      comboArcs.removeAllItems();
      comboNodes.removeAllItems();
      comboPoligons.removeAllItems();

      arcs.removeAllElements();
      nodes.removeAllElements();
  }

  //acció que en canviar de capa actualitza els vectors i combos via
delegació d'altres
  //mètodes, o sigui, treu la capa antiga i carrega la seleccionada.

  void comboCapes_actionPerformed(ActionEvent e)
  {
      if (comboCapes.getItemCount()!=0){
        //En seleccionar una capa hem d'omplir tot el que tenen.
        Capa capaAux = (Capa) comboCapes.getSelectedItem();
        //  System.out.println(capaActiva.getNom());
        System.out.println(capaAux.getNom());
        if(!capaActiva.getNom().equals(capaAux.getNom()))
        {
            transformaMarcSig(capaAux.getTipusInt());
            //JOptionPane.showMessageDialog(null,"són diferents");
            buidaTot();
            capaActiva = capaAux;
            //
            if (capaActiva.getTipusInt()==Capa.PUNTS)
            {
              Iterator it
=capaActiva.getObjectesGeografics().iterator();
              while (it.hasNext())
              {
                  Arc ar= (Arc)it.next();


                  arcs.add(ar);
                  nodes.add(ar.nodeFinal());
                  comboArcs.addItem(ar);
                  comboNodes.addItem(ar.nodeFinal());


              }
            }

            if( capaActiva.getTipusInt()==Capa.LINIES)
```

```
            {
              Iterator it
=capaActiva.getObjectesGeografics().iterator();
              while (it.hasNext())
              {
                  Arc ar= (Arc)it.next();

                  arcs.add(ar);

                  //aquí tenim el problema del nodes repetits.
                  Node un = ar.nodeInicial();
                  Node dos = ar.nodeFinal();
                  if (!nodes.contains(un))
                  {
                    nodes.add(un);
                    comboNodes.addItem(un);
                  }
                  if (!nodes.contains(dos))
                  {
                    nodes.add(dos);
                    comboNodes.addItem(dos);
                  }
                  comboArcs.addItem(ar);
              }

            }

            if (capaActiva.getTipusInt()==Capa.POLIGONS)
            {
              Iterator it
=capaActiva.getObjectesGeografics().iterator();
              while (it.hasNext())
              {

                  Poligon po= (Poligon)it.next();

                  //aquí tenim el problema dels arcs compartits i
nodes, pero procedirem com abans
                  Iterator itpo = po.arcs.iterator();
                  while (itpo.hasNext())
                  {
                    Arc au = (Arc) itpo.next();
                    if(!arcs.contains(au))
                    {
                        arcs.add(au);
                        comboArcs.addItem(au);
                        //comprovem ara els nodes
                        Node un = au.nodeInicial();
                        Node dos = au.nodeFinal();
                        if (!nodes.contains(un))
                        {
                          nodes.add(un);
                          comboNodes.addItem(un);
                        }
                        if (!nodes.contains(dos))
                        {
                          nodes.add(dos);
                          comboNodes.addItem(dos);
                        }
```

```
                    }
                }
                comboPoligons.addItem(po);

            }

        }
        //
      }
      //JOptionPane.showMessageDialog(null,"són iguals");
    }
  }//fi comboCapes

  //acció de botó que crea una nova capa
  void botoNovaCapa_actionPerformed(ActionEvent e)
  {
      buidaTot();
      novaCapa();
  }


  //acció que crea un polígon aillat per delegació a l'acció de menú.
  void botoPoligonAillat_actionPerformed(ActionEvent e)
  {
      menuEdicioPoligonAillat_actionPerformed(e);
  }

  //acció que omple el quadre combinat de capes en passar-li una capa
com a parèmetre
  public void ompleCombos(Capa cap)
  {
      if (cap.getTipusInt()==Capa.PUNTS)
      {
        Iterator it =cap.getObjectesGeografics().iterator();
        while (it.hasNext())
        {
            Arc ar= (Arc)it.next();
            arcs.add(ar);
            nodes.add(ar.nodeFinal());
            comboArcs.addItem(ar);
            comboNodes.addItem(ar.nodeFinal());


        }
      }

      if( cap.getTipusInt()==Capa.LINIES)

      {
        Iterator it =cap.getObjectesGeografics().iterator();
        while (it.hasNext())
        {
            Arc ar= (Arc)it.next();

            arcs.add(ar);

            //aquí tenim el problema del nodes repetits.
            Node un = ar.nodeInicial();
            Node dos = ar.nodeFinal();
            if (!nodes.contains(un))
            {
```

```
              nodes.add(un);
              comboNodes.addItem(un);
            }
            if (!nodes.contains(dos))
            {
              nodes.add(dos);
              comboNodes.addItem(dos);
            }
            comboArcs.addItem(ar);
        }

      }

      if (cap.getTipusInt()==Capa.POLIGONS)
      {
        Iterator it =cap.getObjectesGeografics().iterator();
        while (it.hasNext())
        {
            Poligon po= (Poligon)it.next();

            //aquí tenim el problema dels arcs compartits i nodes, pero
procedirem com abans
            Iterator itpo = po.arcs.iterator();
            while (itpo.hasNext())
            {
              Arc au = (Arc) itpo.next();
              if(!arcs.contains(au))
              {
                  arcs.add(au);
                  comboArcs.addItem(au);
                  //comprovem ara els nodes
                  Node un = au.nodeInicial();
                  Node dos = au.nodeFinal();
                  if (!nodes.contains(un))
                  {
                    nodes.add(un);
                    comboNodes.addItem(un);
                  }
                  if (!nodes.contains(dos))
                  {
                    nodes.add(dos);
                    comboNodes.addItem(dos);
                  }


              }
            }
            comboPoligons.addItem(po);

        }

      }
  }

  //acció que assigna les dades temàtiques als objectes geogràfics,
demanant els camps que
  //intervendran i finalment introduint les dades via una taula

  void menuDadesAssignar_actionPerformed(ActionEvent e) throws
Exception
  {
```

```
      Capa cap = capaActiva;
      if (cap.getObjectesGeografics().isEmpty()) throw new
Exception("No hi ha objectes geografics a la capa");
      String nomTaula = cap.getNomTaulaDades();
      if (nomTaula!=null) throw new Exception("La capa ja té taula
assignada "+nomTaula);
      Vector camps = new Vector();
      Vector tipusCamps = new Vector();

      String[] columnes;
      Object[][] dades;
      String[] tipusColumnes;
      String
opcions[]={"varchar2(20)","varchar(30)","varchar2(50)","int","number(5
)","number(8,2)","number(10,2)","date","boolean"};

      if (cap.getTipusInt()==Capa.POLIGONS)
      {
        camps.add("id_poligon");
        tipusCamps.add("varchar2(20)");
        boolean nofi=true;
        while (nofi)
        {
            String camp = JOptionPane.showInputDialog(null,"Nom del
camp ?\n (per acabar ok sense cap nom)");


            if (!camp.equals(""))
            {
              String tipus= (String)
JOptionPane.showInputDialog(null,"tipus del
camp","tipus",JOptionPane.QUESTION_MESSAGE,null,opcions,opcions[0]);
              camps.add(camp);
              tipusCamps.add(tipus);
            }
            else nofi=false;
        }
        //cream una taula amb les dades
        columnes = new String[camps.toArray().length];
        tipusColumnes = new String[tipusCamps.toArray().length];
        for (int k =0;k<camps.toArray().length;k++)
        {
            columnes[k] = (String) camps.toArray()[k];
            tipusColumnes[k] = (String) tipusCamps.toArray()[k];
        }
        int n = cap.getObjectesGeografics().size();
        int m = columnes.length;
        System.out.println(n+ "--"+m);
        dades = new Object[n][m];
        for (int i=0;i<n;i++) //posem a la primera columna el nom de
l'element geografic.
            dades[i][0]=
((Poligon)cap.getObjectesGeografics().elementAt(i)).getNom();


      }
      else if (cap.getTipusInt()==Capa.LINIES)
      {
        camps.add("id_linia");
        tipusCamps.add("varchar2(20)");
        boolean nofi=true;
```

```
        while (nofi)
        {
            String camp = JOptionPane.showInputDialog(null,"Nom del
camp ?\n (per acabar ok sense cap nom)");
            if (!camp.equals(""))
            {
              String tipus= (String)
JOptionPane.showInputDialog(null,"tipus del
camp","tipus",JOptionPane.QUESTION_MESSAGE,null,opcions,opcions[0]);
              camps.add(camp);
              tipusCamps.add(tipus);
            }
            else nofi=false;
        }
        columnes = new String[camps.toArray().length];
        tipusColumnes = new String[tipusCamps.toArray().length];
        for (int k =0;k<camps.toArray().length;k++)
        {
            columnes[k] = (String) camps.toArray()[k];
            tipusColumnes[k] = (String) tipusCamps.toArray()[k];
        }
        int n = cap.getObjectesGeografics().size();
        int m = columnes.length;
        System.out.println(n+ "--"+m);
        dades = new Object[n][m];
        for (int i=0;i<n;i++)
            dades[i][0]=  ((Arc)
cap.getObjectesGeografics().elementAt(i)).nom;


      }
      else //(cap.getTipusInt()==Capa.PUNTS)
      {

        camps.add("id_punt");
        tipusCamps.add("varchar2(20)");
        boolean nofi=true;
        while (nofi)
        {
            String camp = JOptionPane.showInputDialog(null,"Nom del
camp ?\n (per acabar ok sense cap nom)");
            if (!camp.equals(""))
            {
              String tipus= (String)
JOptionPane.showInputDialog(null,"tipus del
camp","tipus",JOptionPane.QUESTION_MESSAGE,null,opcions,opcions[0]);
              camps.add(camp);
              tipusCamps.add(tipus);
            }
            else nofi=false;
        }
        //cream una taula amb les dades

        columnes = new String[camps.toArray().length];
        tipusColumnes = new String[tipusCamps.toArray().length];
        System.out.println(tipusColumnes.length);
        System.out.println(columnes.length);
        for (int k =0;k<camps.toArray().length;k++)
        {
            columnes[k] = (String) camps.toArray()[k];
            tipusColumnes[k] = (String) tipusCamps.toArray()[k];
```

```
        }

        int n = cap.getObjectesGeografics().size();
        int m = columnes.length;

        dades = new Object[n][m];
        for (int i=0;i<n;i++)
            dades[i][0]=
((Arc)cap.getObjectesGeografics().elementAt(i)).nom;


      }
      TaulaModelSIG tms = new
TaulaModelSIG(columnes,tipusColumnes,dades);

      JTable taula = new JTable(tms);


      DadesTematiquesDialog dtd = new DadesTematiquesDialog(null,"dades
a entrar",true,taula);

      dtd.setBounds(new Rectangle(0,0,500,600));
      dtd.setVisible(true);

      String[][] dades1 = new String[columnes.length][2];
      for (int l=0;l<columnes.length;l++)
      {
        dades1[l][0]=columnes[l];
        dades1[l][1]=tipusColumnes[l];
      }
      String nomtaula = cap.getNom()+"_TaulaBBDD";
      gbd.crearTaula(nomtaula,dades1,columnes.length);

      int i1;
      for ( i1=0; i1<taula.getRowCount();i1++)
        //preparem la taula d'objectes a inserir.
      {

        Object[] dades2 = new Object[taula.getColumnCount()];
        for (int j1=0;j1<taula.getColumnCount();j1++)
        {

            dades2[j1]=taula.getValueAt(i1,j1);



        }
        gbd.inserirDades(nomtaula,dades2);
      }
      gbd.acceptaTransaccio();
      //un cop creada, ja podem posar-la a la capa.
      cap.setNomTaulaDades(nomtaula);
      //caldria guardar els canvis fets.

      //el següent tros de codi desa el fitxer amb nom
nomFitxerImatge.getAbsolutePat()
      //amb la taula assignada a la capa activa.
      try
      {
        FileOutputStream fos = new FileOutputStream(nomFitxerCapes);
        ObjectOutputStream ous = new ObjectOutputStream(fos);
```

```
        ous.writeObject(capes);
        //escrivim ara dos elements importants: l'escala usada, és a
dir, a que
        //equival cada pixel i les unitats usades.
        //escrivim l'escala.
        ous.writeFloat(escala);
        ous.writeObject(unitats);
        ous.writeObject(nomFitxerImatge);
        ous.flush();
        ous.close();


      }
      catch (Exception ex)
      {

      }




  }

  //acció que desvincula la base de dades de la capa actual,
esborrant-la de la base
  //de dades i traient el nom d'aquesta a la capa.

  void menuCapesModificar_actionPerformed(ActionEvent e)
  {
      //esborrem la taula de la base de dades i

      gbd.esborraTaula(capaActiva.getNomTaulaDades());

      //traiem de la capa activa la taula assignada.
      capaActiva.setNomTaulaDades(null);
  }

  //accio que permet fer una consulta
  void menuDadesConsultaDades_actionPerformed(ActionEvent e)
  {
      CercaLlocsDialog c= new CercaLlocsDialog(this,"Consulta
temàtica",true,capaActiva,gbd);

  }

}//fi marcSIG
```

Classe Poligon

```
import java.util.*;
import java.io.*;


/**
 * <p>Títol: Poligon</p>
 * <p>Descripció: Descriu un poligon segons la visió dels Sistemes
Geogràfics d'Informació.
 * Un polígon es format per una  sequència consecutiva d'arcs, de
manera que el
 * node inicial del següent coincideix amb el node inicial o final
 * de l'arc anterior. El polígon entés així té area, i disposa
d'alguns mètodes com
 * centre de gravetat, intersecció amb altres polígons, conversió a
punts, etc.</p>
 * <p>TFC- 2003/04-2: </p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 *
 */

public class Poligon  implements Serializable
{

  /**
   * El nom del polígon.
   */
  String nom;
  /**
   * Els arcs que formen el polígon.
   */
  Vector arcs;
  /**
   * Constructor de la classe Poligon.
   * @param nom  - el nom del polígon
   */

  public Poligon(String nom)
  {
      this.nom=nom;
      arcs = new Vector();
  }

  /**
   * Mètode que afegeix un arc al polígon
   * @param unArc -arc a afegir
   */

  public void afegirArc(Arc unArc)
  {

      arcs.add(unArc);


  }
```

```
  /**
   * Troba l'àrea d'un polígon. De fet és la suma de les àrees dels
arcs (àres sota arc)
   * @return float - l'area d'aquest polígon
   */
  public float area()
  {
      float area = 0;
      //Calculem per a cada arc que forma el polígon l'àrea total,
semblant a la longitud

      Iterator it;
      it = arcs.iterator();
      while (it.hasNext())
      {

        area = area + ((Arc) it.next()).areaSotaArc();
      }
      return area;
  }

  /**
   * Mètode que torna l'arc a un arc de l'esmentat polígon.
   * @param unArc - un arc del polígon.
   * @return - Arc, el següent arc.
   */
  public Arc seguentArc(Arc unArc)
  {

      //En un polígon, els nodes són compartits per dos arcs.

      return unArc.nodeFinal().arcsNode(arcs.elements())[0];


  }

  /**
   * Mètode que comprova si un punt és interior a un polígon. Per això
mirarem
   * el nombre de talls que la semirecta positiva d'aquest punt talla
a cada arc del
   * polígon. Un nombre parell indica que és fora. Un nombre imparell
indica que és dins.
   * @param v - El punt a comprovar.
   * @return boolean - cert, si el vèrtex pertany al polígon.
   */
  public boolean conteVertex(Vertex v)
  {
      //sumem el nombre de talls de la semirrecta que conté el vèrtex
amb cadascun dels
      //arcs.

      int total =0;
      Iterator it = arcs.iterator();
      while (it.hasNext())
      {
        total = total +((Arc) it.next()).nombreTallsSemirrecta(v);
      }
      if (total % 2 ==0) return false;
      return true;
  }
```

```
  // per implementar, aquí hem d'escriure el codi que doni el polígon
o polígons, resultat
  // d'intersectar el polígon passat com a paràmetre amb la instància
del polígon.
  public Poligon intersectaPoligon(Poligon p)
  {

      return this;
  }

  /**
   * Mètode que torna el centroide del polígon. El centroide que he
considerat usa
   * la mitja dels punts mitjos. Val a dir que pot passar que el
centroide pot estar
   * fora del polígon.
   * @return Vertex, el vèrtex centroide del polígon
   */
  public Vertex centroide()
  {
      float x=0f;
      float y=0f;

      for(int i=0;i<vertexsPoligon().length;i++)
      {
        x=(vertexsPoligon()[i]).getX() + x;
        y=(vertexsPoligon()[i]).getY()+y;
      }
      x=x/vertexsPoligon().length;
      y=y/vertexsPoligon().length;
      Vertex v = new Vertex();
      v.afegeix(x,y);
      return v;
  }

  /**
   * Mètode accesor de lectura del nom del polígon
   * @return String, el nom del polígon
   */
  public String getNom()
  {
      return nom;
  }
  /**
   * Mètode toString2 del Poligon. Dóna més informació que no toString
   * @return String, la cadena més complerta d'informació del polígon.
   */
  public String toString2()
  {
      String cadena = this.nom +" - "+ arcs.size()+" : ";

      Iterator it = this.arcs.iterator();
      while(it.hasNext())
      {

        cadena = cadena +((Arc) it.next()).nom+ ",";
      }
      cadena = cadena +"\b.";
      return cadena;
  }
```

```
  /**
   * Mètode toString del Polígon. Dóna només el nom.
   * @return el nom de polígon
   */
  public String toString()
  {

      return getNom();
  }

  /**
   * Mètode que torna tots els vèrtexs del polígon en array
   * @return, l'array de vèrtexs del polígon
   */
  public Vertex[] vertexsPoligon()
  {
      Vector v = new Vector();
      Iterator it = arcs.iterator();
      while (it.hasNext())
      {
        Vertex[] a = ((Arc) it.next()).arrayVertexs();
        for (int i=0;i<a.length;i++)
            if (!v.contains(a[i])) v.add(a[i]);

      }
      Vertex[] torna = new Vertex[v.size()];
      for(int i=0;i<v.size();i++)
        torna[i]=(Vertex) v.toArray()[i];
      return torna;
  }

  /**
   * Mètode que indica a tots els arcs que formen el polígon, com
aquest -el polígon-
   * es troba situat, si a esquerra o a dreta.
   */
  public void topologia()
  {
      Iterator it = arcs.iterator();
      while (it.hasNext())
      {
        ((Arc) it.next()).topologia(this);
      }
  }

  /**
   * Mètode que torna la circularitat. L'índex utilitzat és l'arrel
quadrada de
   * la relació ente l'àrea d'aquest polígon i l'àrea del cercle
d'igual perímetre
   * que el polígon.(Bosque Sendra pàgina 232). Un valor prop a 1
indica que el polígon
   * és prou regular (s'acosta a un cercle). Un valor prop a 0 indica
una forma molt irregular
   * @return la circularitat del polígon.
   * @throws Exception
   */
  public float circularitat() throws Exception
  {
      float circularitat=0f;
```

```
      float r = perimetre()/(2*(float)Math.PI);
      float areaCercle=((float) Math.PI)*r*r;
      if (perimetre()==0f) throw new Exception("perímetre polígon 0");
      circularitat = (float) Math.sqrt((float) area()/areaCercle);
      return circularitat;
  }

  /**
   * Mètode que torna el perímetre del polígon.
   * @return, el perímetre del polígon.
   */
  public float perimetre()
  {
      float perimetre=0f;
      Enumeration en = arcs.elements();
      while (en.hasMoreElements())
      {
        perimetre = perimetre + ((Arc) en.nextElement()).longitud();
      }
      return perimetre;
  }

  /**
   * Mètode que torna la contiguitat del polígon. La contiguitat és
defineix com el
   * nombre de polígons que l'envolten. Una manera molt senzilla és
comptar el nombre
   * d'arcs del polígon que tenen atribuïda la topologia, és a dir,
coneixen el polígon
   * a dreta i a esquerra.
   * @return, el nombre de polígons que l'envolten.
   */
  public int continguitat()
  {
      int arcsCompartits=0;
      Enumeration en = arcs.elements();
      Arc a;
      while(en.hasMoreElements())
      {
        a= (Arc) en.nextElement();
        if(a.aDreta!=null && a.aEsquerra!=null) arcsCompartits =
arcsCompartits+1;
      }
      return arcsCompartits;
  }
}
```

Classe Poligonal

```
import java.util.*;
import java.io.*;

/**
 * <p>Títol: Poligonal</p>
 * <p>Descripció: Descriu una línia poligonal. Aquesta figura no
presenta autointerseccions.
 * I és la base de l'entitat Arc.</p>
 * <p>TFC- 2003/04-2: </p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 *
 */

public class Poligonal implements Serializable
{
  protected Vector vertexs;
  String nom;

  public Poligonal(String nom)
  {
      vertexs= new Vector();
      this.nom=nom;
  }

  /**
   * Constructor de la poligonal a partir de vèrtexs
   * @param nom . Nom de la línia poligonal
   * @param v . Array de vèrtexs.
   */

  public Poligonal(String nom,Vertex[] v)
  {
      this(nom);
      for (int i=0;i<v.length;i++)
        afegirVertex(v[i]);
  }

  /**
   * Mètode que afegeix un vèrtex a la línia poligonal
   * @param ver el Vèrtex que s'afegeix.
   */
  public void afegirVertex(Vertex ver)
  {
      vertexs.add(ver);
  }

  /**
   * Mètode que obté un array d'enters corresponents a les abcises
dels punts.
   * Aques mètode és útil alhora de dibuixar.
   * @return l'array d'abcises.
   */
  public int[] obtenirX()
  {
      int capacitat;
```

```
      if (esUnPoligon())
        capacitat = vertexs.size();
      else capacitat = vertexs.size();
      int x[] = new int[capacitat];
      for (int i=0;i<capacitat;i++)
      {
        x[i]=(int)(((Vertex) vertexs.get(i)).getX());
      }

      return x;

  }

  /**
 * Mètode que obté un array d'enters corresponents a les ordenades
dels punts.
 * Aques mètode és útil alhora de dibuixar.
 * @return l'array d'ordenades.
   */
  public int[] obtenirY()
  {
      int capacitat;
      if(esUnPoligon())
        capacitat = vertexs.size();
      else capacitat=vertexs.size();

      int y[] = new int[capacitat];
      for (int i=0;i<capacitat;i++)
      {
        y[i]=(int)(((Vertex) vertexs.get(i)).getY());
      }

      return y;
  }

  /**
   * Mètode que comprova si la figura és o no un polígon.
   * @return cert si és un polígon, fals en cas contrari.
   */
  public boolean esUnPoligon()
  {
      return ((Vertex) vertexs.firstElement()).equals((Vertex)
vertexs.lastElement());
  }

  /**
   * Mètode que torna la llista de vèrtexs en format cadena separats
per una coma.
   * Per exemple 4,5 7.5,8.25
   * @return la llista de vèrtexs.
   */
  public String[] llistVertexs()
  {
      String cadena[]= new String[vertexs.size()];

      for (int i=0;i<cadena.length;i++)
        cadena[i]=Float.toString(((Vertex)
vertexs.get(i)).getX())+","+Float.toString(((Vertex)
vertexs.get(i)).getY());
      return cadena;
  }
```

```
  /**
   * Mètode que torna la longitud de la línia poligonal
   * @return el valor de la longitud.
   */
  public float longitud()
  {

      if(vertexs.size()<=1) return 0f;
      float longitud =0f;
      Iterator it;
      Vertex primer;
      Vertex segon;
      it = vertexs.iterator();
      if (it.hasNext())
        primer = (Vertex) it.next();
      else primer = null;
      while (it.hasNext()){

        segon = (Vertex) it.next();
        Segment s = new Segment(primer,segon);
        longitud = longitud+ s.distancia();
        Vertex aux = segon;
        primer = aux;
      }
      return longitud;
  }


}
```

Classe PoligonDialog

```
import java.awt.*;
import javax.swing.*;
import java.util.*;
import java.awt.event.*;

/**
 * <p>Títol: PoligonDialog</p>
 * <p>Descripció: Finestra que crea un polígon a a partir els arcs.</p>
 * <p>TFC- 2003/04-2: </p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 *
 */

public class PoligonDialog extends JDialog
{

  /*
  * diferents elements gràfics.
  */
  private JPanel panel1 = new JPanel();
  private JTextField txtPoligon = new JTextField();
  private JLabel etiquetaNom = new JLabel();
  private JLabel jLabel2 = new JLabel();
  private JButton btnOk = new JButton();
  private JTextArea jTextArea1 = new JTextArea();
  private JScrollPane panellScrolable = new JScrollPane();
  private Frame marc;

  /**
   * el polígon que es crearà.
   */
  private Poligon poligon;
  private JList llistaPoligons = new JList();

  /**
   * Constructor amb 4 paràmetres. El més important de tots ells és el
vector
   * d'arcs, dels quals es triarà un subconjunt. És responsabilitat del
usuari comprovar
   * que es troben tots ells enllaçats
   * @param frame el marc pare
   * @param title el títol
   * @param modal si és modal o no
   * @param arcs el vector d'arcs.
   */
  public PoligonDialog(Frame frame, String title, boolean modal,Vector
arcs)
  {
      super(frame, title, modal);
      try {
        marc=frame;
        llistaPoligons = new JList(arcs);
        jbInit();
        setBounds(200,200,400,300);

        setVisible(true);
```

```
        pack();
      }
      catch(Exception ex)
      {
        ex.printStackTrace();
      }
  }

  /**
   *  Constructor sense paràmetres.
   *
   */
  public PoligonDialog()
  {
      this(null, "", false,null);
  }

  /**
   * Mètode iniciador de components gràfiques
   * @throws Exception
   */
  private void jbInit() throws Exception
  {
      panel1.setLayout(null);
      txtPoligon.setBounds(new Rectangle(160, 205, 183, 23));

      etiquetaNom.setFont(new java.awt.Font("Dialog", 1, 12));
      etiquetaNom.setText("nom del poligon");
      etiquetaNom.setBounds(new Rectangle(64, 203, 94, 22));
      jLabel2.setFont(new java.awt.Font("Dialog", 1, 12));
      jLabel2.setText("arcs disponibles");
      jLabel2.setBounds(new Rectangle(64, 87, 97, 17));
      btnOk.setBounds(new Rectangle(161, 252, 79, 27));
      btnOk.setEnabled(true);
      btnOk.setText("ok");
      btnOk.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(ActionEvent e) {
            btnOk_actionPerformed(e);
        }
      });
      jTextArea1.setBackground(new Color(212, 208, 200));
      jTextArea1.setFont(new java.awt.Font("Dialog", 0, 12));
      jTextArea1.setEditable(false);
      jTextArea1.setText("Per seleccionar arcs no consecutius, prem alhora
la tecla Ctrl " +
                         "mentres selecciones amb el ratolí els
diferents arcs");
      jTextArea1.setLineWrap(true);
      jTextArea1.setBounds(new Rectangle(64, 26, 279, 78));
      panellScrolable.setBounds(new Rectangle(64, 102, 279, 81));
      llistaPoligons.setBorder(BorderFactory.createLineBorder(Color.black));
      getContentPane().add(panel1);
      panel1.add(jTextArea1, null);
      panel1.add(jLabel2, null);
      panel1.add(etiquetaNom, null);
      panel1.add(txtPoligon, null);
      panel1.add(btnOk, null);
      panel1.add(panellScrolable, null);
      panellScrolable.getViewport().add(llistaPoligons, null);
  }
```

```
//acció en prémer sobre el botó Ok
  void btnOk_actionPerformed(ActionEvent e)
  {
      poligon = new Poligon(txtPoligon.getText());
      for (int i=0;i<llistaPoligons.getSelectedValues().length;i++)

        poligon.afegirArc((Arc)llistaPoligons.getSelectedValues()[i]);

      dispose();

  }


  /**
   * Mètode que retorna el polígon obtingut mitjançant aquest quadre de
diàleg.
   * @return el polígon
   */
  public Poligon getPoligon()
  {
      return poligon;
  }


}
```

Classe Segment

```
/**
 * <p>Títol: Classe Segment.</p>
 * <p>Descripció: Descriu un segment des del punt de vista
euclidià.</p>
 * <p>TFC- 2003/04-2: </p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 */

public class Segment
{

  /**
   * un vèrtex del segment
   */
  Vertex v1;

  /**
   * l'altre vèrtex del segment
   */
  Vertex v2;

  /**
   * Constructor amb dos paràmetres
   * @param v1
   * @param v2
   */
  public Segment(Vertex v1,Vertex v2)
  {
      this.v1=v1;
      this.v2=v2;
  }

  /**
   * Constructor amb quatre paràmetes. Les coordenades.
   * @param x1 abcisa primer punt
   * @param y1 ordenada primer punt
   * @param x2 abcisa segon punt
   * @param y2 ordenada segon punt
   */
  public Segment(float x1, float y1, float x2, float y2)
  {
      Vertex v1 = new Vertex();
      Vertex v2 = new Vertex();
      v1.afegeix(x1,y1);
      v2.afegeix(x2,y2);
      this.v1=v1;
      this.v2=v2;
  }

  /**
   * Mètode que obté la distància euclidiana del segment.
   * @return
   */
```

```
  public float distancia()
  {
      float a = (v2.getX()-v1.getX())*(v2.getX()-v1.getX());
      float b=  (v2.getY()-v1.getY())*(v2.getY()-v1.getY());

      return (float) Math.sqrt(a+b);
  }

  /**
   * Mètode que torna l'area sota el segment en el sistema cartesià.
   * Correspon a l'area del trapezoide delimitat pel segment i les
projeccions
   * dels extrems dels segment sobre les abcises i la projecció del
segment sobre
   * les abcises
   * @return l'area sota segment
   */

  public float areaSotaSegment()
  {
      return (v2.getX()-v1.getX())*(v2.getY()+v1.getY())/2;
  }

  /**
   * Mètode que obté el pendent del segment
   * @return el pendent
   * @throws Exception en cas que el segment sigui vertical
   */
  public float pendent() throws Exception
  {
      if (v1.getX()==v2.getY()) throw new  Exception("pendent
infinita");
      else return ((v1.getY()-v2.getY())/(v1.getX()-v2.getX()));
  }

  /**
   * Mètode que comprova si un punt (Vèrtex) passat com a paràmetre
pertany
   * al segment.
   * @param v El punt
   * @return cert o fals segons correspongui.
   */
  public boolean pertanyVertex(Vertex v)
  {
      float y;
      float x;
      try
      {
        y=v.getY()-v1.getY();
        x=pendent()*(v.getX()-v1.getX());
        if(x==y) return true;
        else return false;
      }
      catch (Exception ex)
      {
        if ((((v1.getY()-v.getY())< v2.getY()) || (v2.getY()-
v.getY())<v1.getY()) && v1.getX()==v.getX())

        return true;
        else return false;
      }
```

```
    }
}
```

Classe TaulaModelSIG

```
import javax.swing.table.AbstractTableModel;
import javax.swing.*;

/**
 *
 * <p>Title: </p>
 * <p>Description: Model de taula per a la captura de les dades. Útil
en la presentació
 * de les dades.</p>
 * <p>TFC- 2003/04-2: </p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 */

class TaulaModelSIG extends AbstractTableModel
{
  /** l'array de noms de les columnes*/
  final String columnNames[];

  /**
   * les dades a mostrar*/

  private Object[][] data;

  /**
   * els tipus de les columnes
   */
  private String[] tips;

  /**
   * Constructor amb tres paràmetres.
   * @param cols , els noms dels camps.
   * @param tipus , els tipus dels camps.
   * @param dades , les dades
   */
  public TaulaModelSIG(String[] cols,String[] tipus, Object[][] dades)
  {

      this.columnNames=cols;
      this.tips=tipus;
      this.data=dades;


  }

  /**
   * Mètode accessor de lectura del nom del camp en la columna i.
   * @param i, la columna
   * @return el nom de la columna
   */
  public String getColumnName(int i)
  {
      return columnNames[i];
  }

  /**
   * Mètode que torna el nombre de files
```

```
   * @return el nombre de files
   */
  public int getRowCount()
  {

      return data.length;
  }


  /**
   * Mètode que torna el nombre de columnes
   * @return el nombre de columnes
   */
  public int getColumnCount()
  {
      return columnNames.length;
  }


  /**
   * Mètode que torna en la fila row i columna col l'objecte. S'ha de
fer un càsting
   * @param row la fila
   * @param col la columna
   * @return l'objecte
   */
  public Object getValueAt(int row, int col)
  {
      return data[row][col];
  }

  /**
   * Mètode que obté el tipus  de columna
   * @param i l'índex de columna
   * @return el nom
   */

  public String getTipusColumnaBBDD(int i)
  {
      return tips[i];
  }
  /**
   * En aquest programari, la primera casella identifica l'objecte
geogràfic i per tant
   * no és editable.
   */

  public boolean isCellEditable(int row, int col)
  {

      if (col < 1)
      {
        return false;
      } else {
        return true;
      }
  }

  /**
   * Mètode que asigna un valor a la  taula en la columna row i
columna col
```

```
    * @param value l'objecte a asignar
    * @param row la fila
    * @param col la columna
    */
   public void setValueAt(Object value, int row, int col)
   {
       String cadena = tips[col];
       if(cadena.startsWith("var"))
       {
         try
         {
             data[row][col] = new String(value.toString());
             fireTableCellUpdated(row, col);
         }
         catch (NumberFormatException e)
         {
             JOptionPane.showMessageDialog(null,"La columna: " +
getColumnName(col) + " només accepta cadenes.");
         }
       }


       if(cadena.startsWith("int"))
       {
         try
         {
             data[row][col] = new Integer(value.toString());
             fireTableCellUpdated(row, col);
         }
         catch (NumberFormatException e)
         {
             JOptionPane.showMessageDialog(null,"La columna: " +
getColumnName(col) + " només accepta enters.");
         }
       }
       if(cadena.startsWith("number"))
       {
         try
         {
             data[row][col] = new Float(value.toString());
             fireTableCellUpdated(row, col);
         }
         catch (NumberFormatException e)
         {
             JOptionPane.showMessageDialog(null,"La columna: " +
getColumnName(col) + "només accepta reals");
         }
       }
       if(cadena.startsWith("boolean"))
       {
         try
         {
             String dada = value.toString();
             if(!((dada.equals("si")) || dada.equals("no"))) throw new
Exception("no és un valor lògic");
             if(dada.equals("si"))    data[row][col]= new Boolean(true);
             else data[row][col] = new Boolean(false);
             fireTableCellUpdated(row, col);
         }
         catch ( Exception e)
         {
```

```
            JOptionPane.showMessageDialog(null,"La columna: " +
getColumnName(col) + " només accepta cert o fals");
          }
        }



   }

}
```

Classe UtilitatsSIG

```
import javax.swing.*;
import java.util.*;
/**
 *
 * <p>Títol: UtilitatsSIG</p>
 * <p>Description: Classe que permet realitzar algunes utilitats. És la
responsable d'obtenir
 * els objectes comuns entre capes. El resultat serà sempre un conjunt d'objectes
de dimensió igual
 * al conjunt de dimensió més petita. Es troba en fase inicial i no presenta totes
les
 * funcionalitats previstes </p>
 * <p>TFC- 2003/04-2: </p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 */

 class UtilitatsSIG
{

/**
 * Mètode que torna una nova capa a partir de dues capes. Ambdues capes contenen
in-
 * formació que es fusionarà segons la naturalesa de cada capa. Així tenim:
 * punts-punts (coincidència o distància de separació màxima)
 * punts-línies (inclusió o distància de separació màxima)
 * punts-polígon (inclusió o distància de separació màxima)
 * línies-línies (intersecció)
 * línies-polígon (inclusió total o parcial)
 * polígons-polígons (intersecció)
 * A fi de modelitzar qualsevol d'aquestes situacions en la implementació,
 * partim del valor del tipus de capa.
 * Això només dóna 3 valors possibles 0,1 i 2. A fi de decidir el tipus de relació
usarem
 * el producte d'aquests valors, augmentats en 1, donant el seguent domini:
1,2,3,4,6,9
 * La seguent relació dóna l'equivalència:
 * punts-punts 1
 * punts-línies 2
 * punts-polígon 3
 * línies-línies 4
 * línies-polígon 6
 * polígons-polígons 9.
 *
 * @param a
 * @param b
 * @return
 * @throws Exception
 */
public Capa modela(Capa a, Capa b) throws Exception

{
if (a.getNomTaulaDades()==null || b.getNomTaulaDades()==null) throw new
Exception("Les capes han de tenir dades temàtiques");
int valor =(a.getTipusInt()+1)*(b.getTipusInt()+1);
Capa capaATornar;
switch (valor)
```

```
{
  case 1:capaATornar=puntPunt(a,b); break;
  case 2:capaATornar=puntLinia(a,b); break;
  case 3:capaATornar=puntPoligon(a,b);break;
  case 4:capaATornar=liniaLinia(a,b); break;
  /*case 6:capaATornar=liniaPoligon(a,b); break;
  default :capaATornar=poligonPoligon(a,b); break;*/
}
capaATornar=null;
return capaATornar;
}

/**
 * Mètode que torna una Capa a partir de dues capes de punts. La capa resultant
 * s'obté dels punts que es troben a una distància menor o igual que un llindar.
 * @param a
 * @param b
 * @return
 */
public Capa puntPunt(Capa a,Capa b)
{
  //a i b  contenen arcs que són nodes.
  float epsilon = tolerancia();

  Capa aTornar = new Capa(JOptionPane.showInputDialog(null,"nom de la
capa"),a.getTipusInt(),a.getMapa(),a.getEscala(),a.getUnitats());

  Enumeration ena = a.getObjectesGeografics().elements();
  Enumeration enb = b.getObjectesGeografics().elements();
  while (ena.hasMoreElements())
  {
      Arc arca= ((Arc) ena.nextElement());
      Node noda= arca.nodeInicial();
      while (enb.hasMoreElements())
      {
        Arc arcb=((Arc) enb.nextElement());
        Node nodb = arcb.nodeInicial();
        Segment s = new Segment(noda,nodb);
        if(s.distancia() < epsilon) aTornar.setObjecteGeografic(arca);

      }
  }

  if (aTornar.getObjectesGeografics().isEmpty()) return null;
  return aTornar;
}

/**
 * Mètode que torna una Capa a partir d'una capa de punts i l'altra d'arcs
(línies). La capa resultant
 * s'obté dels punts que es troben a una distància menor o igual que un llindar.
 * @param a
 * @param b
 * @return
 */
public Capa puntLinia(Capa a,Capa b)
{
  float epsilon = tolerancia();
  Capa capaPunts;
  Capa capaLinies;
  if(a.getTipusInt()==Capa.PUNTS)
```

```
      {
      capaPunts=a;
      capaLinies=b;
      }
  else
  {
      capaPunts=b;
      capaLinies=a;
  }
  Capa aTornar = new Capa(JOptionPane.showInputDialog(null,"nom de la
capa"),a.getTipusInt(),a.getMapa(),a.getEscala(),a.getUnitats());

  Enumeration ena = capaPunts.getObjectesGeografics().elements();
  Enumeration enb = capaLinies.getObjectesGeografics().elements();
  while (ena.hasMoreElements())
  {
      Arc arca= ((Arc) ena.nextElement());
      Node noda= arca.nodeInicial();
      while (enb.hasMoreElements())
      {
        Arc arcb=((Arc) enb.nextElement());
        int i=0;
        boolean trobat=false;
        while(i<arcb.arrayVertexs().length || !trobat)
        {
            Segment s = new Segment(noda,arcb.arrayVertexs()[i]);
            if(s.distancia() < epsilon)
            {
              aTornar.setObjecteGeografic(arca);
              trobat=true;
            }
        }



      }
  }

  if (aTornar.getObjectesGeografics().isEmpty()) return null;
  return aTornar;
}
public Capa puntPoligon(Capa a,Capa b)
{
  float epsilon = tolerancia();
  Capa capaPunts;
  Capa capaPoligons;
  if(a.getTipusInt()==Capa.PUNTS)
  {
      capaPunts=a;
      capaPoligons=b;
  }
  else
  {
      capaPunts=b;
      capaPoligons=a;
  }

  Capa aTornar = new Capa(JOptionPane.showInputDialog(null,"nom de la
capa"),a.getTipusInt(),a.getMapa(),a.getEscala(),a.getUnitats());
```

```
  Enumeration ena = capaPunts.getObjectesGeografics().elements();
  Enumeration enb = capaPoligons.getObjectesGeografics().elements();
  while (ena.hasMoreElements())
  {
      Arc arca= ((Arc) ena.nextElement());
      Node noda= arca.nodeInicial();
      while (enb.hasMoreElements())
      {
        Poligon pol=((Poligon) enb.nextElement());


        if(pol.conteVertex(noda)) aTornar.setObjecteGeografic(arca);

      }
  }

  if (aTornar.getObjectesGeografics().isEmpty()) return null;
  return aTornar;
}
public Capa liniaLinia(Capa a,Capa b)
{
//Prenem els rectangles que contenen els arcs (això funciona només en el cas que
siguin
//arcs amb poca sinuositat i que no "s'encargolin".
  Capa aTornar = new Capa(JOptionPane.showInputDialog(null,"nom de la
capa"),a.getTipusInt(),a.getMapa(),a.getEscala(),a.getUnitats());
  Enumeration ena = a.getObjectesGeografics().elements();
  while(ena.hasMoreElements())
  {
      Arc arca = (Arc) ena.nextElement();

  }
return aTornar;
}

/**
public Capa liniaPoligon(Capa a,Capa b)
{

}
public Capa poligonPoligon(Capa a,Capa b)
{

}*/
public float tolerancia()
{
  float valor=0f;
  String tolerancia;
  tolerancia = (String) JOptionPane.showInputDialog(null,"Escriu un valor");
  try
  {
  valor = Float.parseFloat(tolerancia);
  }
  catch (NumberFormatException ex)
  {
      JOptionPane.showMessageDialog(null,"La dada introduïda no és un real");
  }


  return valor;
```

```
}

/**
 * Mètode que donat un arc "prou regular", retorna els 4 vèrtexs del rectangle
 * envolvent. No és el mínim, però s'hi acosta.
 * @param a l'arc
 * @return l'array de vèrtexs del rectangle envolvent.
 */
/**
 *
public Vertex[] rectangle(Arc a)

{


  Vector2D v1 = new Vector2D(a.nodeInicial(),a.nodeFinal());
  Segment s = new Segment(a.nodeInicial(),a.nodeFinal());

  float max1=0f;
  float max2=0f;
  float distancia;
  int i1=0,i2=0;
      //calculem els punts que es troben a distància màxima  de la recta per cada
costat.
      for (int i=0; i<a.arrayVertexs().length;i++)
      {
        Vector2D v2 = new Vector2D(a.nodeInicial(),a.arrayVertexs()[i]);
        distancia =Math.abs(v1.sentit(v2))/v1.norma();

        try
        {
                    float valor = (a.arrayVertexs()[i].getX()-
a.nodeInicial().getX())*s.pendent()+a.nodeInicial().getX()+a.nodeInicial().getY();
                    if(a.arrayVertexs()[i].getY() > valor)
                    {
                      if(distancia > max1)
                      {
                          max1=distancia;
                          i1=i;
                      }
                    }
                    if (a.arrayVertexs()[i].getY()<valor)
                    {
                      if(distancia > max2)
                      {
                          max2=distancia;
                          i2=i;
                      }
                    }
        }
        catch (Exception ex)
        {
            System.out.println(ex.getMessage());

        }


      }

      //arribats a aquest punt tenim els 4 punts pertanyen a les 4 rectes que
l'envolten
```

```
        //l'arc.
        a.nodeFinal();
        a.nodeInicial();
        a.arrayVertexs()[i1].getX();
        a.arrayVertexs()[i2].getX();









}
*/
}
```

Classe Vector2D

```
/**
 *
 * <p>Title: Vector2D</p>
 * <p>Description: Vector des del punt de vista euclidià. En el pla
considerem els vectors
 * com aquells objectes provistos d'un sentit i una direcció. Els
vectors aquí usats es troben
 * referenciats segons la base canònica u1 = (1,0) i u2 = (0,1)</p>
 * <p>Copyright: Copyright (c) 2004</p>
 * <p>Company: </p>
 * @author unascribed
 * @version 1.0
 */

class Vector2D
{

  /**
   * component x del vector
   */
 private float x;

 /**
  * component y del vector
  */
 private float y;

 /**
  * Constructor amb dos paràmetres. Les coordenades del vector.
  * @param x1 abcisa
  * @param y1 ordenada
  */
 public Vector2D(float x1,float y1)
 {
   x=x1;
   y=y1;
 }

 /**
  * Constructor amb dos paràmetres. El vèrtex inicial i el final
  * @param vOrigen l'origen
  * @param vFinal el final
  */
 public Vector2D(Vertex vOrigen, Vertex vFinal)
 {
   x=vFinal.getX()-vOrigen.getX();
   y=vFinal.getY()-vOrigen.getY();
 }

 /**
  * accessor de lectura de l'abcisa.
  * @return
  */
 public float getX()
 {
   return x;
 }
```

```
 /**
  * accessor de lectura de l'ordenada.
  * @return
  */
 public float getY()
 {
   return y;
 }
/**
 * Mètode que torna el sentit del producte vectorial d'aquest vector
amb el que es
 * passa com a paràmetre. El sentit és cap a dins o cap a fora del pla
que delimiten
 * els dos vectors. Això és interessant per a conèixer la orientació
del arc d'un
 * polígon, si sentit és positiu l'arc és "counterclockwise"
(antihorari) respecte al polígon,
 * en cas contrari és "clockwise" (sentit horari).
 * @param u, Vector.
 */

public float sentit(Vector2D u)
{
  float sentit=0f;
  sentit =getX()*u.getY()-getY()*u.getY();
  return sentit;
}

/**
 * Mètode que obté la norma del vector.
 * @return la norma
 */
public float norma()
{
  return (float)Math.sqrt((float)(getX()*getX()+getY()*getY()));
}
}
```

Classe Vertex

```
/**
 * <p>Títol:Vertex </p>
 * <p>Descripció: Classe que descriu un vèrtex per ser usat en els
Sistemes d'Informació
 * Geogràfica</p>
 * <p>TFC- 2003/04-2: </p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 */
public class Vertex implements java.io.Serializable
{
  /** El nom del vèrtex*/
  private String nom;
  /** L'abcisa del vèrtex*/
  private float x;

  /**
   * L'ordenada del vèrtex
   */
  private float y;

  /**
   * Constructor sense paràmetres
   */
  public Vertex()
  {

  }

  /**
   * Mètode que assigna les components al vèrtex.
   * @param x l'abcisa
   * @param y l'ordenada
   */
  public void afegeix(float x, float y)
  {
      this.x=x;
      this.y=y;

  }

  /**
   * Mètode sobrecarregat d'Object que compara si dos vèrtexs són
iguals o no
   * @param o un vèrtex
   * @return fals o cert si tenen les mateixes coordenades
   */
  public boolean equals(Object o)
  {
      if(o instanceof Vertex)
      {
        Vertex v = (Vertex) o;
        if (this.x==v.x && this.y==v.y)
            return true;
      }
      return false;
```

```
}

/**
 * Mètode toString que retornna el nom del vèrtex.
 * @return el nom
 */
public String toString()
{

    return nom +" ("+Float.toString(x)+","+Float.toString(y)+") ";
}


/**
 * Accessor d'escriptura del nom
 * @param nom Nom del vèrtex
 */
public void setNom(String nom)
{
    this.nom=nom;
}

/**
 * Mètode que transforma el vèrtex en node.
 * @return Node
 */
public Node toNode()
{
    Node d=null;
    try
    {
      d= new Node();
      d.afegeix(this.x,this.y);
    }
    catch (Exception ex)
    {

    }

    return d;
}

/**
 * Mètode accessor de lectura de l'abcisa
 * @return l'abcisa
 */
public float getX()
{
    return x;
}

/**
 * Mètode accessor de lectura de l'ordenada
 * @return l'ordenada
 */
public float getY()
{
    return y;
}

/**
```

```
   * Mètode accessor de lectura del nom
   * @return el nom
   */
  public String getNom()
  {
      return nom;
  }



}
```

Classe ApplicationSIG

```
import javax.swing.UIManager;
import java.awt.*;
import javax.swing.*;

/**
 * <p>Title: ApplicationSIG</p>
 * <p>Description: És el punt d'entrada del programa</p>
 * <p>TFC- 2003/04-2: </p>
 * @author Joan Manuel González Febrer
 * @version 1.0
 */

public class ApplicationSIG
{
  private boolean packFrame = false;

  //Construct the application
  public ApplicationSIG()
  {
      MarcSIG frame = new MarcSIG();
      //Validate frames that have preset sizes
      //Pack frames that have useful preferred size info, e.g. from
their layout
      if (packFrame) {
        frame.pack();
      }
      else {
        frame.validate();
      }
      //Center the window
      Dimension screenSize =
Toolkit.getDefaultToolkit().getScreenSize();
      Dimension frameSize = frame.getSize();
      if (frameSize.height > screenSize.height) {
        frameSize.height = screenSize.height;
      }
      if (frameSize.width > screenSize.width) {
        frameSize.width = screenSize.width;
      }
      frame.setLocation((screenSize.width - frameSize.width) / 2,
(screenSize.height - frameSize.height) / 2);
      //maximitzem la finestra a la mida de la resolució
```

```
  frame.setSize(screenSize);

      //la posem

      frame.setLocation(0,0);

      frame.setVisible(true);
  }
  //Main method
  public static void main(String[] args) {
      try {
       //
UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
      }
      catch(Exception e) {
        e.printStackTrace();
      }
      JOptionPane.showMessageDialog(null,"Maximitzeu la finestra per
veure la barra d'estat");
      new ApplicationSIG();
  }
}
```