CREACIÓ AUTOMÀTICA DE GRÀFICS ESTADÍSTICS A PARTIR DE DADES DE SOROLL D'EUROPA AMB TECNOLOGIES OPEN SOURCE EN EL VISOR NOISE MAP VIEWER DE L'ETC-LUSI

PROJECTE FINAL

MÀSTER EN TECNOLOGIES DE LA INFORMACIÓ GEOGRÀFICA

11ª EDICIÓ

Universitat Autònoma de Barcelona. Departament de Geografia

Autora: Montserrat Oller Garcia

Tutor LIGIT: Ignacio Ferrero Beato

Tutors ETC-LUSI: Walter Simonazzi

i César Martínez

Data: Febrer 2010

11 mtig 2009
Professionals per a la societat de la informació

UAB
Universitat Autònoma de Barcelona
Departament de Geografia

ETC LUSI
European Topic Centre
Land Use and Spatial Information

RESUM

Títol: Creació automàtica de gràfics estadístics a partir de dades de soroll d'Europa amb tecnologies Open Source en el visor Noise Map Viewer de l'ETC-LUSI.

Autora: Montserrat Oller Garcia

Data: Febrer de 2010

Paraules clau: SIG, *Open Source, Java*, *MapFish*, *JFreeChart*, *PostgreSQL*, ETC-LUSI.

La present memòria exposa el projecte final del Màster en Tecnologies de la Informació Geogràfica en la seva 11ª edició. Aquest està organitzat pel Departament de Geografia de la Universitat Autònoma de Barcelona, i s'imparteix al LIGIT (Laboratori d'Informació Geogràfica i de Teledetecció). El projecte neix del conveni entre la Universitat Autònoma de Barcelona i l'ETC-LUSI (*European Topic Centre on Land Use and Spatial Information*) com a entitat acollidora d'un estudiant del màster per a realitzar les seves pràctiques professionals.

El projecte *Noise Map Viewer* per Europa ve determinat per la Comissió Europea que té una Directiva sobre Soroll Ambiental, que sol·licita mapes de soroll segons la població exposada a aquest. Per aquest motiu la Directiva Europea, que pertany a l'Agència Europea de Medi Ambient, demana als països membres, de manera obligatòria, unes dades de soroll determinades. L'ETC-LUSI ha desenvolupat el visor *Noise Map Viewer* per a l'Agència Europea del Medi Ambient, amb l'objectiu de presentar les dades geogràfiques i estadístiques pertinents, i poder-les fer arribar al major nombre d'usuaris possible interessats en temes de soroll.

Per al projecte final de màster l'ETC-LUSI proposa que en el seu visor *Noise Map Viewer* es generin els gràfics estadístics de soroll d'Europa automàticament, ja que inicialment s'han generat de forma manual. Per això serà necessari crear un procés que s'integri dins de *MapFish* i pugui tornar els gràfics demanats per l'usuari.

La metodologia a seguir en el projecte s'ha basat en Mètrica versió 3 (Metodologia de Planificació, Desenvolupament i Manteniment de Sistemes d'Informació) promoguda pel Ministeri d'Administracions Públiques del Govern Espanyol, adaptant-la a les necessitats del present projecte. Així doncs s'han determinat les quatre fases de que consta el projecte: anàlisi, disseny del software, desenvolupament i implementació.

Cal recalcar que per a la realització de la tasca encomanada s'han fet servir només tecnologies *Open Source* (software distribuït i desenvolupat lliurement - Codi obert) les quals cada cop estan agafant més rellevància.

En la fase d'anàlisi es veurà la definició del sistema que inclou la identificació de l'entorn tecnològic utilitzat fins aleshores, l'establiment de requisits i la definició de les interfícies d'usuari. Finalment entrarem a les conclusions de l'anàlisi de les diferents tecnologies i llibreries existents per a poder desenvolupar la tasca encomanada. Per això s'ha de tenir en compte que el visor està construït amb *MapFish,* que és un entorn *JavaScript* basat en *OpenLayers* i utilitzant com a servidors *MapServer* amb l'*Apache* i l'IIS (*Internet Information Services*) que conté la pàgina web de *Noise*. Podem avançar que la llibreria escollida i en la qual s'ha basat el desenvolupament del projecte ha estat *JFreeChart* que genera gràfics de manera dinàmica. El llenguatge utilitzat per al desenvolupament dels gràfics ha estat *Java*.

En el disseny del software hi trobarem els casos d'ús tan necessaris per al desenvolupament del visor. Per això necessitem saber com un usuari interactua amb el sistema i quina és la resposta obtinguda.

Pel que fa el desenvolupament de l'aplicació que ha de generar els gràfics veurem quin és el programari utilitzat en cada una de les diferents fases de construcció del codi (*Java*) que ha de generar els gràfics (*servlet*), així com la base de dades utilitzada i els programes intermedis que ens ajuden a la seva construcció final. Per a la realització del *servlet* en codi *Java* que ens tornarà els gràfics s'ha utilitzat l'*Eclipse Galileo* (JEE) que és un Entorn de Desenvolupament Integrat (IDE: *Integrated Development Environment*). La base de dades utilitzada és *PostgreSQL* que conté una eina d'administració anomenada *pgAdminIII* on es pot veure la base de dades creada amb les taules corresponents. També serà important veure els passos intermedis que ens permeten passar la informació alfanumèrica a la base de dades definitiva.

Per últim la fase d'implementació que consta de dues parts: primer integrar el *servlet* dins del servidor corresponent (*Tomcat*) i segon integrar el *servlet* al *MapFish* modificant el codi *JavaScript* existent que ha de tornar la petició de l'usuari.

La memòria reflexa la possibilitat de crear aplicacions complexes utilitzant tecnologies *Open Source*, el que suposa un estalvi econòmic important per a l'empresa. Cal destacar la facilitat que suposa treballar amb aquest tipus de tecnologies degut a la gran quantitat d'informació i ajuda que es pot trobar a la xarxa.

El projecte s'ha complert en el període establert i la generació de gràfics de manera automàtica suposarà un estalvi d'hores invertides en la generació manual dels gràfics i, per tant, en l'actualització de la informació.

AGRAÏMENTS

Primer de tot agrair als meus tutors la seva ajuda incondicional, les hores dedicades, els seus consells i els seus ànims. Al Walter pels seus consells en la part més teòrica del projecte, al Nacho i al Cèsar per la seva ajuda i paciència en tantes i tantes hores de programació.

També vull donar les gràcies al Dr. Joan Nunes per donar-me la oportunitat de realitzar aquest màster. Al Nacho per la seva tasca com a Director acadèmic del projecte. Al Miquel Àngel per la seva paciència amb nosaltres. A tot l'equip del LIGIT per la seva dedicació. Al personal de l'ETC-LUSI per la seva amable acollida. Als meus companys de màster pel bon ambient creat en el grup.

A la meva família i amics, en especial als meus pares, per animar-me a embarcar-me en aquest projecte i pels ànims donats en tot moment.

Gràcies a tots!

ÍNDEX

1. Introducció

1.1. Presentació

La present memòria exposa el projecte final del Màster en Tecnologies de la Informació Geogràfica (MTIG - 11ª edició) i es desenvolupa en col·laboració entre el Centre Temàtic Europeu d'Usos del Sòl i Informació Espacial (*European Topic Centre on Land Use and Spatial* - ETC-LUSI) i el Departament de Geografia de la Universitat Autònoma de Barcelona (UAB). Es troba emmarcat dins el projecte *Noise Map Viewer* i la tasca principal és realitzar un mòdul dins el visor de *noise*, que llegeixi les dades de soroll d'Europa contingudes en taules, el qual ha de generar automàticament els gràfics estadístics sobre aquestes dades (inicialment la generació d'estadístiques es feia de forma manual amb imatges JPEG que estaven enllaçades amb els diferents mapes temàtics) i les presenti a l'usuari.

El projecte *Noise Map Viewer* per Europa és el visor desenvolupat per l'ETC-LUSI per a l'Agència Europea de Medi Ambient (AEMA; *European Environment Agency* – EEA) i neix de la necessitat de mostrar les dades de soroll d'Europa, de manera que puguin arribar al major nombre de persones interessades en aquests temes. El visor pretén mostrar de la manera més clara possible les principals dades geogràfiques i estadístiques reportades pels diferents països.

Aquest projecte ve determinat per la Comissió Europea que té una Directiva sobre Soroll Ambiental la qual sol·licita mapes de soroll segons la població exposada a aquest. Per aquest motiu la Directiva Europea que pertany a l'AEMA i la qual té com objectiu principal oferir informació sòlida sobre el medi ambient, demana als països membres, de manera obligatòria, unes dades de soroll determinades, les quals segueixen unes normes específiques a l'hora de ser reportades.

Els països membres són un total de trenta-dos, els vint-i-set membres de la Unió Europea més Islàndia, Liechtenstein, Noruega, Suïssa i Turquia (http://www.eea.europa.eu/es/about-us/countries-and-eionet).

Les dades vénen donades a nivell de país i se centra en les estadístiques de soroll a les que està exposada la població en les aglomeracions i fora d'elles. Les principals fonts de soroll estudiades són: ferrocarrils, aeroports, carreteres i indústria (aquesta última només en el cas de les aglomeracions). La informació es recull en diferents bandes de soroll, donades en decibels. D'aquesta manera es poden distingir dues franges horàries diferents: durant el dia (Lden: *day*, *evening*, *night*) i durant la nit (Lnight: *night*). El nombre de decibels a tenir en compte són de >55, >65 i >75 durant el dia, i de >50, >60 i >70 per la nit. El nombre de decibels varia ja que durant la nit es considera que el soroll afecta més la població.

Per a la visualització d'aquesta informació s'ha desenvolupat el visor web *Noise Map Viewer*, on es poden veure els diferents mapes temàtics amb les corresponents dades estadístiques i els seus gràfics.

L'ETC-LUSI s'encarrega de:

- La recol·lecció i processat de la informació reportada pels països membres segons la directiva 2002/49/EC de *Noise* (http://ec.europa.eu/environment/noise/directive.htm).
- Desenvolupament d'un visor web que permetrà visualitzar la informació reportada pels països membres, en resposta a les seves obligacions de *reporting*, a la Directiva sobre Soroll Ambiental d'Europa.

1.2. Marc institucional

L'ETC-LUSI té la seva seu a la Universitat Autònoma de Barcelona. Aquest centre dóna suport a l'Agència Europea de Medi Ambient (AEMA) en la vigilància de l'ús i dels canvis en la cobertura del sòl a Europa i analitza les conseqüències ambientals. Aquests canvis estan fortament connectats amb les decisions polítiques, com les directives europees i lleis nacionals; el centre s'encarrega d'avaluar l'eficàcia d'aquests instruments i donar recomanacions a l'Agència.

La seva experiència es basa en la gestió de bases de dades i avaluacions ambientals i regionals de les zones rurals, zones de muntanya, zones urbanes o costaneres.

L'ETC-LUSI forma part de la Xarxa Europea d'Informació i d'Observació Ambiental (*European Environmental Information and Observation Network* - EIONET), i està cooperant amb altres institucions europees com el Centre Comú d'Investigació (*Joint Research Centre* - JRC), l'Eurostat i les diferents direccions generals de la Comissió Europea.

Es dedica a recopilar, gestionar, analitzar i mostrar dades espacials relacionals d'ús del sòl i té una gran experiència en la gestió de dades i controls de qualitat. La tasca de l'ETC-LUSI es basa també en el desenvolupament de solucions de codi obert per seguir la Directiva *Inspire*. Aquesta estableix una infraestructura d'informació espacial a Europa per donar suport a les polítiques comunitàries de medi ambient i les polítiques i activitats que puguin tenir un impacte sobre ell. També segueix la política europea de Servei Compartit d'Informació Ambiental (*Shared Environmental Information Service* - SEIS) per incorporar sistemes autònoms i solucions de programari en una xarxa de recursos distribuïts i per simplificar l'accés a dades per part del públic.

2. Objectius del projecte

2.1. Finalitat del projecte

El projecte consistirà en la creació de gràfics estadístics de soroll d'Europa de forma automàtica amb tecnologies *Open Source* dins el visor, ja existent, *Noise Map Viewer* (http://noise.eionet.europa.eu/index.html) per Europa de l'ETC-LUSI. Per això s'haurà de fer una anàlisi en profunditat del sistema d'informació, com funciona i quines són les necessitats de l'ETC-LUSI a l'hora de mostrar les dades estadístiques amb el gràfic corresponent. A conseqüència d'això s'estudiaran les diferents tecnologies i llibreries existents a utilitzar durant tot el projecte, el que permetrà escollir les més adequades per al desenvolupament del procés, que un cop integrat en el visor generarà els gràfics estadístics automàticament. La integració d'aquest procés a la web s'haurà de fer de manera que no afecti el treball realitzat fins aleshores i serà el que substituirà la part que mostra les estadístiques de forma manual (mitjançant imatges JPEG).

L'objectiu principal del projecte és donar a l'estudiant del màster unes bases concretes per a realitzar qualsevol altre projecte d'aquestes característiques, tenint en compte que cal adaptar-les a les eventuals necessitats. També fer conèixer el gran ventall de tecnologies existents per tal de realitzar un sistema d'informació orientat a objectes o estructurat, i les diferents arquitectures que ens hi podem trobar. Així doncs, en el cas concret que ens ocupa, té com a objectiu mostrar les diferents llibreries existents per la generació de gràfics estadístics. Això ve acompanyat de l'aprenentatge en bases de dades, i passos intermedis d'automatització de processos que ens faciliten les tasques a realitzar. No ens hem d'oblidar de la importància d'aprendre els diferents llenguatges de programació que s'hauran d'utilitzar dins el projecte: *Java* i *JavaScript*.

2.2. Pla de treball

Per tal d'organitzar el projecte s'ha desenvolupat un diagrama de Gantt (*Gantt chart*) que permet mostrar el temps de dedicació previst per a les diferents tasques a realitzar durant el projecte i durant un temps determinat, en aquest cas s'ha dividit en setmanes al llarg dels tres mesos de durada del projecte. Per la posició de les barres en el diagrama, que simbolitzen les diferents tasques a desenvolupar, podrem veure quina és la relació o la independència entre elles. A continuació es pot veure una taula on s'especifica en quina setmana s'inicia cada una de les feines i la duració d'aquesta. Si mirem l'índex del projecte i el diagrama de Gantt podrem veure que les principals fases hi queden reflectides, tot i que en el diagrama s'hi pot veure d'una manera més resumida. El diagrama és una eina molt útil durant el temps de projecte, doncs és el que ens mostra en tot moment si el projecte està en un punt òptim o no.

Tasca	Inici	Duració (setmanes)
Anàlisi del sistema	0	3
1. Definició del sistema	0	2
2. Establiment de requisits	0	2
3. Definició interfícies d'usuari	0	2
4. Conclusions anàlisi tecnologies	2	1
Disseny del software	0	2
1. Casos d'ús	0	2
Desenvolupament	2	10
1. Preparació de l'entorn de treball	2	2
2. Base de dades	3	1
3. Llenguatge de programació	3	2
4. Desenvolupament del procés	3	9
Implementació	10	1
1. Implementació al Tomcat	10	1
2. Implementació a MapFish	10	2
Proves	4	9
Aprovació	13	1
Glossari	0	5

Figura 2.1. Taula resum per a l'elaboració del diagrama de Gantt.

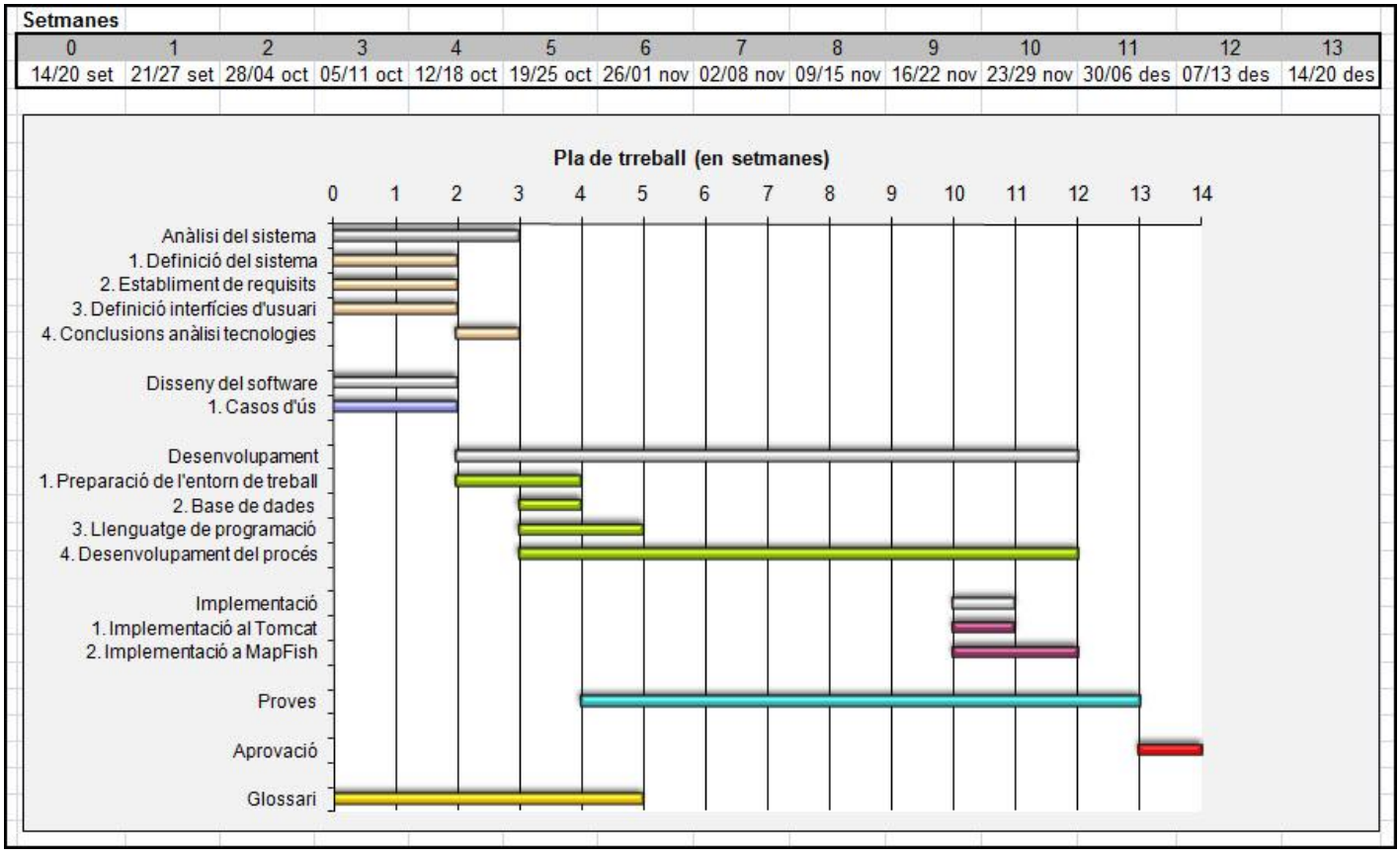


Figura 2.2. Diagrama de Gantt.

2.3. Metodologia

La metodologia a seguir en el projecte s'ha basat en Mètrica versió 3, que és una Metodologia de Planificació, Desenvolupament i Manteniment de Sistemes d'Informació, promoguda pel Ministeri d'Administracions Públiques del Govern Espanyol. Aquesta metodologia s'ha adaptat a les necessitats del present projecte, i dóna les pautes a seguir en tots aquells projectes de desenvolupament de sistemes d'informació orientats a objectes o estructurats. L'actual projecte està orientat a objectes i d'aquesta manera podem definir les quatre grans fases de que constarà:

1. Anàlisis
2. Disseny del software
3. Desenvolupament
4. Implementació

Es pot veure com aquestes grans fases són les que porten el fil conductor del projecte, i per tant, de la present memòria.

3. Anàlisis del sistema d'informació

3.1. Definició del sistema

La definició del sistema es basa en dos apartats principals:

a) Descripció del sistema: el sistema d'informació està composat per dues bases de dades, per una banda tenim la base de dades alfanumèrica (*Access*), i per l'altra en tenim una de geogràfica, ambdues unides en una *geodatabase*. Per mostrar aquestes dades es va realitzar un primer esborrany del disseny del visor, en el que s'hi poden veure: la taula de continguts[1], la llegenda[2], les eines de navegació[3], el mapa[4], una vista general del mapa[5], un apartat per les estadístiques[6] i un lloc de descàrrega de dades[7].

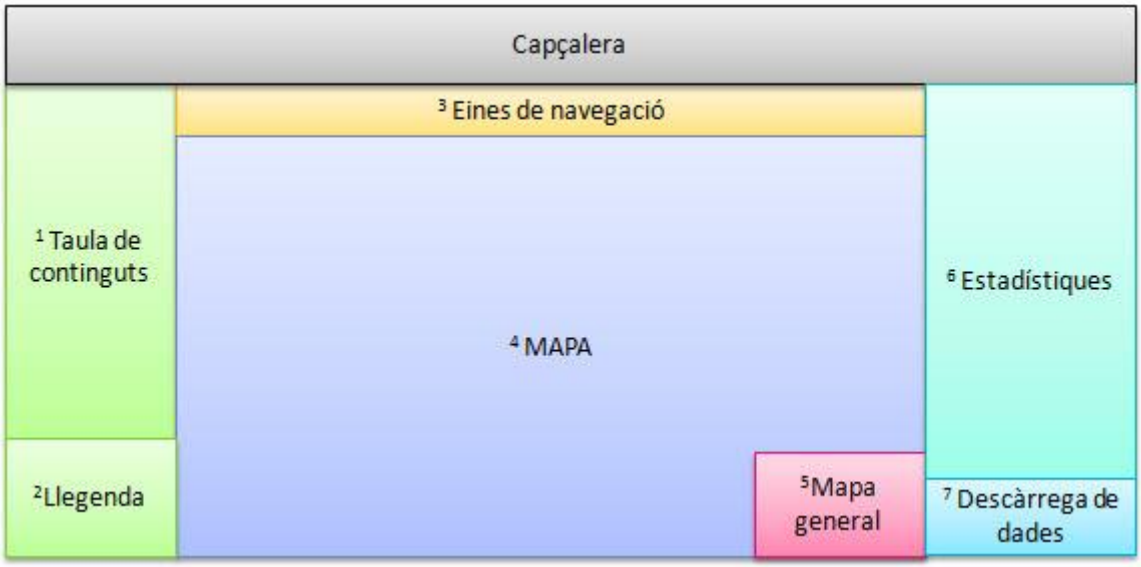


Figura 3.1. Visió general dels components. Noise Map Viewer.

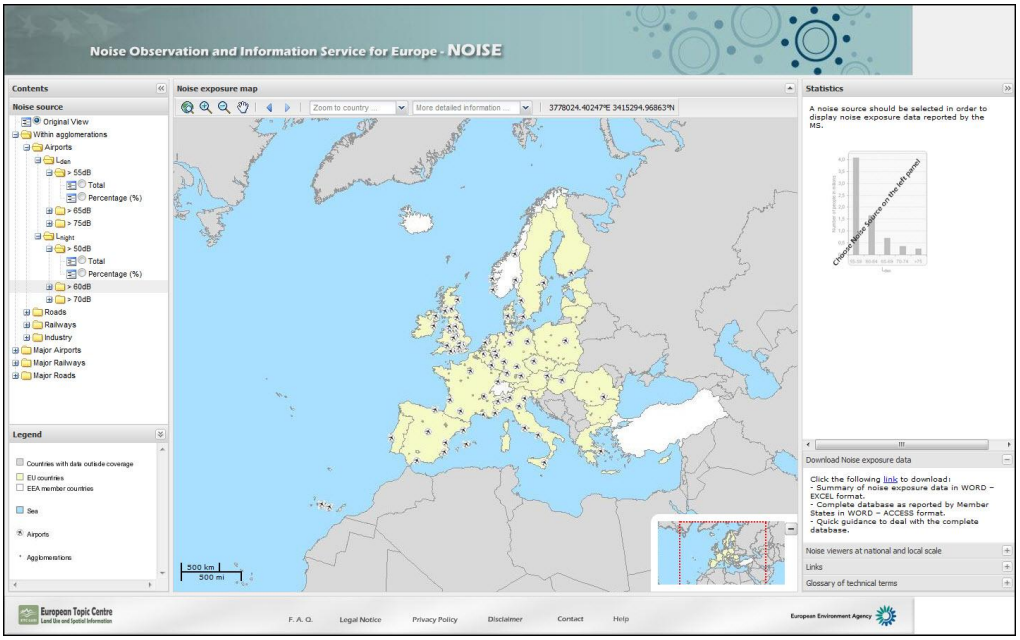


Figura 3.2. Noise Map Viewer.

b) Descripció del projecte: la informació recollida està pensada per a que arribi al major nombre d'usuaris possibles a nivell europeu interessats en temes de soroll. Per això s'ha desenvolupat el visor *Noise Map Viewer* on es mostren unes dades estadístiques concretes, dividides en diferents categories i subcategories, i en un entorn geogràfic concret (Europa). Les dades s'ofereixen a nivell de país, en unes bandes de soroll determinades per la Directiva Europea, i en dos divisions principals: durant el dia (Lden: >55, >65 i >75 dB) i durant la nit (Lnight: >50, >60 i >70 dB). Es diferencia la població exposada al soroll dins les aglomeracions (*airports*, *roads*, *railways* i *industry*) i fora d'elles (*major airports*, *major railways* i *major roads*). També en alguns casos es pot escollir si es volen veure les dades totals o relatives. Inicialment la generació d'estadístiques dins del visor, es realitzava de manera manual, les imatges es van exportar a un fitxer imatge (JPEG), les quals es van vincular a les dades per a la presentació final a l'usuari. Donat que aquesta tasca es pot automatitzar, el projecte consistirà en la generació de dades estadístiques de forma automàtica.

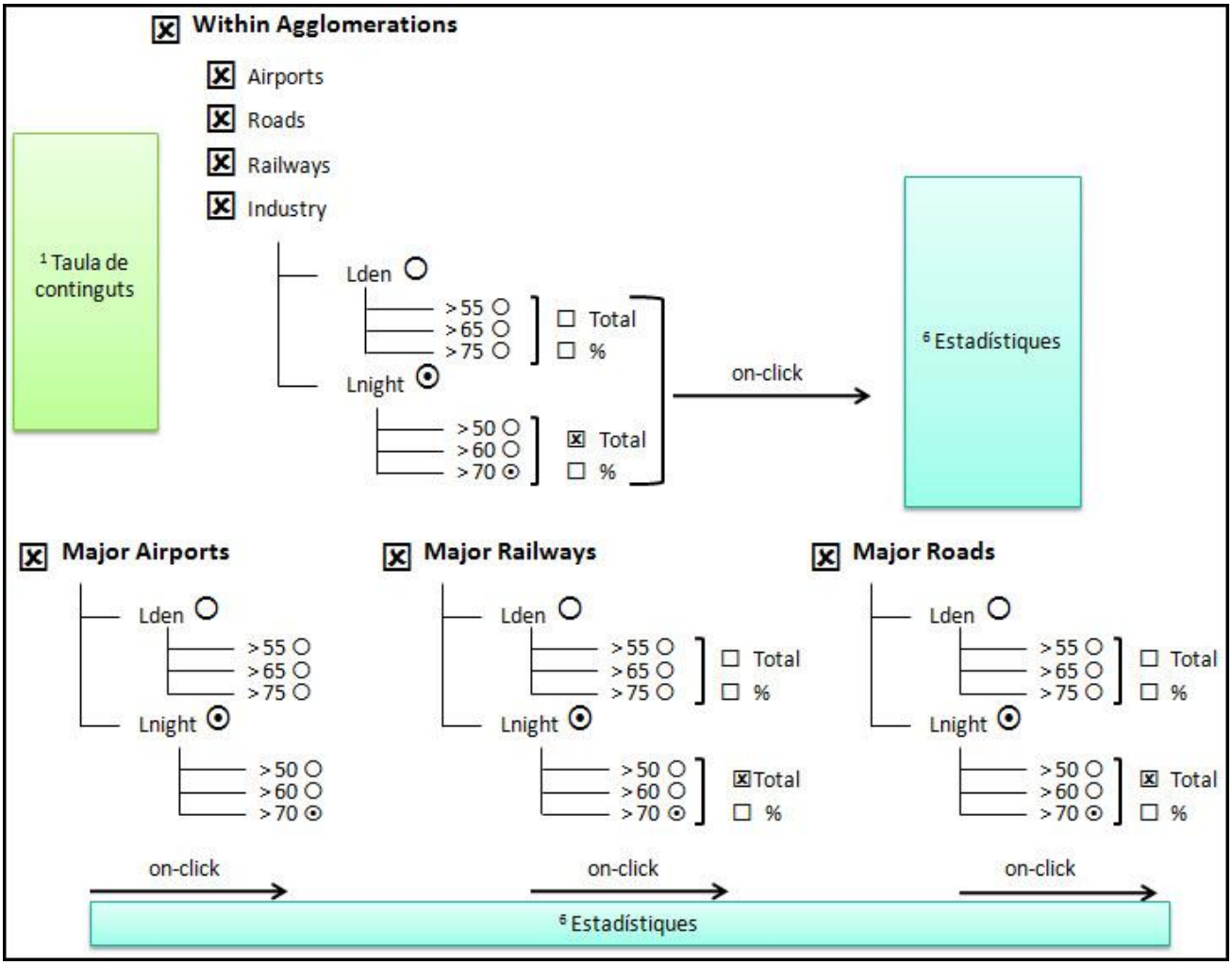


Figura 3.3. Taula de continguts. Noise Map Viewer.

3.1.1. Identificació de l'entorn tecnològic

Per tal de desenvolupar el projecte, una de les primeres tasques realitzades, ha estat la identificació de l'entorn tecnològic utilitzat fins el moment. Això serà la base a l'hora de decidir quina llibreria n'és compatible per generar els gràfics estadístics demanats.

A continuació es pot veure el llistat de les tecnologies utilitzades:

- Sistema Operatiu *Windows 2003 Server*
- *MapServer*: *Apache* (port 8080)
- *MapFish* } *JavaScript*
- *OpenLayers* }
- IIS (*Internet Information Services*): ***web_noise***, JPEG, fitxers HTML/CSS

Per dur a terme el projecte s'ha d'analitzar *MapFish* en profunditat, doncs és clau en la presa de decisions posteriors. El *MapFish* és un web 2.0 *mapping application framework*, compost per dues parts: *MapFish* client i *MapFish server*. El *MapFish* client és un entorn *JavaScript*, basat en *OpenLayers* per la part de la cartografia, i ExtJS i GeoExt per la part GUI (*Graphical User Interface*). En quant a servidor és tractat com a tal i es composa de diferents mòduls que poden utilitzar diferents llenguatges (*Java*, PHP...). Pot ser utilitzat amb *MapServer* o qualsevol servidor cartogràfic que sigui capaç de comunicar-se amb protocols oberts com WMS o WFS. Integra diversos components i el suport de la última tecnologia web 2.0.

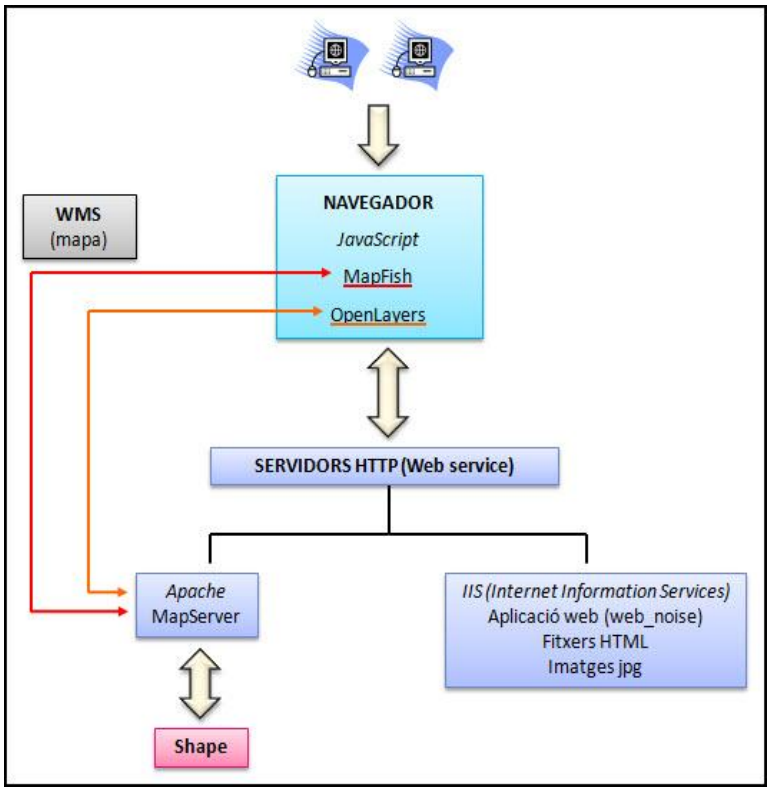


Figura 3.4. Arquitectura inicial del sistema.

3.1.2. Especificació d'estàndards i normes

Tots els sistemes d'informació geogràfica es regeixen per uns estàndards i normes universals que asseguren la interoperabilitat dels sistemes creats. La interoperabilitat és la capacitat dels sistemes d'informació, i procediments als que donen suport, de compartir dades i fer possible l'intercanvi d'informació i coneixements entre ells. Es parla de la interoperabilitat d'una web com una condició necessària per a que els usuaris tinguin un accés complet a la informació disponible.

Aquests estàndards i normes vénen definides per les següents organitzacions:

- W3C (*World Wide Web Consortium*; http://www.w3.org/ , http://www.w3c.es/): és un consorci internacional que treballa per a desenvolupar i promocionar estàndards Web. La seva missió és "Guiar la Web cap al seu màxim potencial a través del desenvolupament de protocols i pautes que assegurin el creixement futur de la Web".

- ISO (*International Organization for Standardization*; http://www.iso.org/iso/home.htm): l'Organització Internacional per a l'Estandardització és una organització no governamental que es compon per diferents representants d'organismes de normalització de cent seixanta-dos països. La seva funció és l'elaboració d'estàndards i normes internacionals per a tots els camps de la indústria i el comerç. Pel que fa la informació geogràfica, hi trobem l'ISO/TC 211 (http://www.isotc211.org/) que té per objecte establir un conjunt estructurat d'estàndards per a la informació dels objectes o fenòmens que estan directament o indirectament associats amb una ubicació relativa a la Terra. Aquests estàndards poden especificar, per a la informació geogràfica, mètodes, eines i serveis per a la gestió de dades (incloent definició i descripció), adquisició, processament, anàlisi, accés, presentació i transferència de dades en format digital-electrònic entre els diferents usuaris, sistemes i localitzacions. Ha d'enllaçar amb les normes apropiades per a la tecnologia de la informació i les dades quan sigui possible, i ha de proporcionar un marc per al desenvolupament del sector de les aplicacions específiques d'ús de dades geogràfiques.

- OGC (*Open Geospatial Consortium*, http://www.opengeospatial.org/): és un consorci internacional de tres-centes noranta companyies, agències governamentals i universitats que participen en un procés de consens per desenvolupar estàndards d'interfície a disposició del públic. La seva finalitat és la definició d'estàndards oberts i interoperables dins dels Sistemes d'Informació Geogràfica i de la *World Wide Web*. Persegueix acords entre les diferents empreses del sector que possibilitin la interoperació dels seus sistemes de geoprocessament i facilitin l'intercanvi de la informació geogràfica en benefici dels usuaris. Anteriorment va ser conegut com a *Open GIS Consortium*.

- CEN (*The European Committee for Standardization*, http://www.cen.eu/cenorm/homepage.htm): és una organització no lucrativa privada que té com a missió fomentar l'economia europea en el negoci global, el benestar dels ciutadans europeus i el medi ambient proporcionant una plataforma eficient per a l'elaboració, el manteniment i la distribució de normes europees i altres especificacions tècniques (aquests estàndards tècnics promouen entre d'altres la interoperabilitat de xarxes, protecció del medi ambient, la investigació i desenvolupament de programes, etc.).

3.2. Establiment de requisits

3.2.1. Anàlisis i obtenció de requisits funcionals

Per entendre aquest apartat, primer de tot s'ha de definir què volem dir amb requisits funcionals. Un requisit funcional defineix el comportament intern del software: càlculs, detalls tècnics, manipulació de dades i altres funcionalitats específiques que mostren com els casos d'ús (que veurem en l'apartat quatre d'aquesta memòria) seran portats a la pràctica. Així doncs, els requisits i els casos d'ús es complementen en un procés bidireccional, ja que la definició dels requisits funcionals es pot fer abans o després de definir els casos d'ús, depenent de les nostres necessitats.

D'aquesta manera, en aquest apartat, es definirà com funcionava el visor internament en la seva fase inicial, quan els gràfics apareixien com un JPEG, i com funciona finalment un cop els gràfics es generen de forma automàtica a través del *servlet* creat.

En el primer cas, quan l'usuari escull una de les opcions dins de la taula de continguts del visor, aquest va a buscar, per una banda, el WMS (*Web Map Service*) que li dóna el mapa que ha de mostrar, i per l'altra banda, la imatge JPEG que conté el gràfic escollit, per això es connecta amb el servidor que conté la web (IIS), on s'hi troben les imatges (carpeta *graphs*), i escull mitjançant un codi assignat el gràfic corresponent del país, la font de soroll, la franja horària i els decibels escollits per l'usuari.

En el segon cas, quan l'usuari escull una de les opcions dins de la taula de continguts del visor, segueix anant a buscar el WMS per tal de mostrar el mapa demanat. A continuació ja no busca la imatge JPEG, si no que es connecta al *Tomcat* que és el servidor que conté l'arxiu *noise.war*. Aquest arxiu és el que conté el *servlet* generat, que té els mateixos codis d'imatge que té el codi *JavaScript* de la web, es connecta amb el servidor que conté la web (en aquest cas s'ha posat dins de l'*Apache*), i mitjançant aquests codis genera la imatge de manera automàtica.

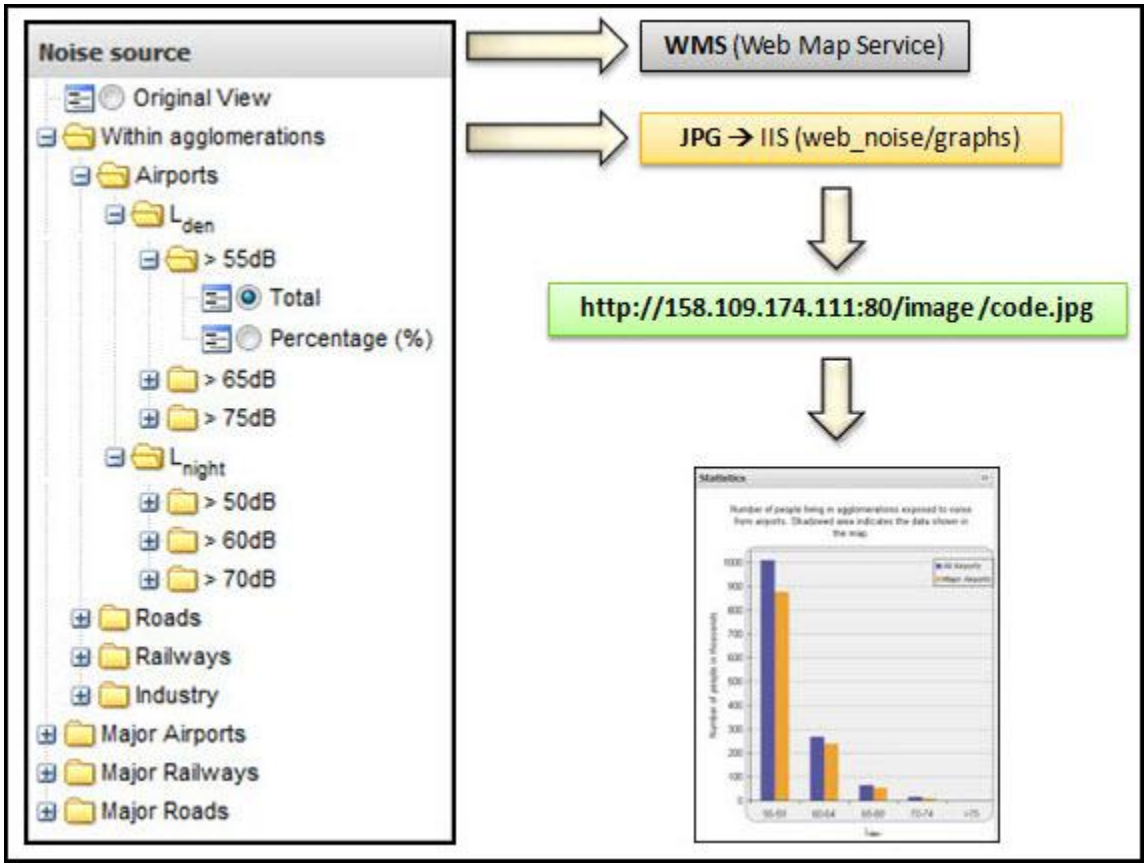


Figura 3.5. Requisits funcionals inicials. Noise Map Viewer.

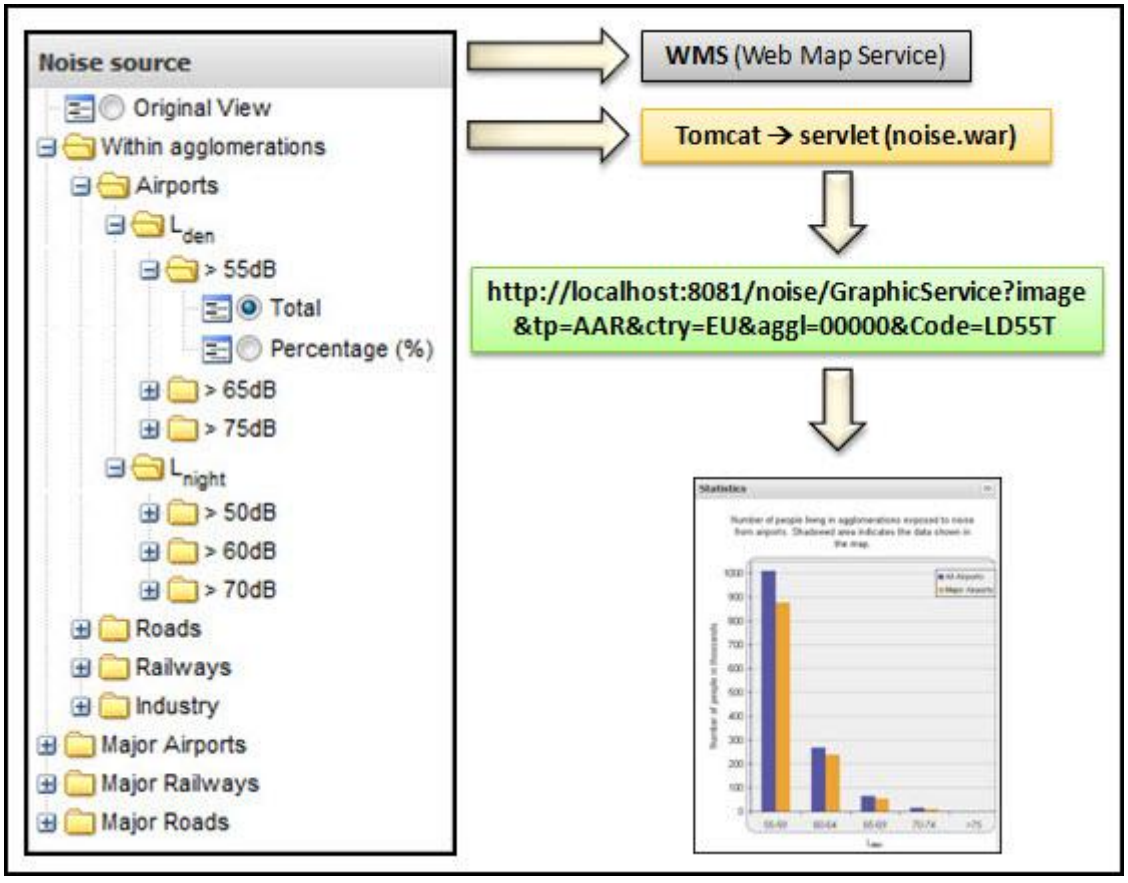


Figura 3.6. Requisits funcionals finals. Noise Map Viewer.

3.2.2. Anàlisis i obtenció de requisits tecnològics

Tenint en compte que el visor està desenvolupat amb *MapFish*, s'haurà de realitzar un estudi de quina llibreria és compatible amb aquest i quina és la millor solució per al desenvolupament dels gràfics estadístics de manera automàtica. Com a resultat d'aquest anàlisi obtenim la taula següent, on es pot observar algunes de les diferents llibreries i tecnologies existents:

Open Source	Open Source	Pagament
JFreeChart (llibreria Java)	JOpenChart (llibreria Java)	ElegantJ Charts (llibreria Java)
jCharts (aplicació Java, servlets, JSP)	Chart2D (llibreria Java)	Easy Charts (llibreria Java)
CeWolf (XML)	JMagallanes (Java i J2EE, es basa en JFreeChart)	Dotnetcharting (VB i C#)
Ext JS (llibreria JavaScript)	ZedGraph (C ++, VB)	ChartDirector (llibreria Java)
BIRT (basada en Eclipse, Java i J2EE)	SpiceCharts (llibreria Java)	Ext JS (llibreria JavaScript)
JCCKit (llibreria Java)	JChart2d (Java Swing widget)	

Figura 3.7. Taula comparativa de llibreries i tecnologies que generen gràfics estadístics.

Font llibreries Open Source: http://java-source.net/open-source/charting-and-reporting

No totes les llibreries de la taula són òptimes per al desenvolupament de la tasca encomanada. Primer de tot s'han de descartar les llibreries de pagament, ja que el nostre projecte es basa única i exclusivament en tecnologies *Open Source* (codi lliure). De les que són de codi lliure, no totes ens serveixen, ja que algunes no fan servir el tipus de llenguatge que a nosaltres ens interessa (*Java*), i no totes es connecten a bases de dades. Finalment la llibreria que més s'adapta a les nostres necessitats és *JFreeChart* (http://www.jfree.org/index.html).

JFreeChart és una llibreria *Java*, totalment lliure, que facilita als programadors la tasca de mostrar gràfics de qualitat professional en les seves aplicacions. Aquesta llibreria té un ampli conjunt de característiques que han fet que ens decidim a utilitzar-la les quals inclouen:

- Una consistent i ben documentada API (Interfície de Programació d'Aplicacions - *Application Programming Interface*), que suporta una àmplia gamma de gràfics.
- Un disseny flexible que és fàcil d'estendre, donant suport a la banda del servidor i a la del client.

- Suport a molts tipus de producció, inclosos els components *Swing* (llibreria gràfica per *Java* que inclou *widgets* per a la intefície gràfica de l'usuari com caixes de text, botons, desplegables, i taules), arxius d'imatge (incloent PNG i JPEG), gràfics vectorials i diferents formats d'arxiu (incloent PDF, EPS i SVG).
- Es distribueix sota els termes *GNU Lesser General Public Licence* (LGPL), que permet el seu ús en aplicacions pròpies.

De la gran gamma de gràfics que ens ofereix aquesta llibreria, ens haurem de fixar ens els gràfics d'una o més barres, tant verticals com horitzontals. En els casos d'ús podrem veure quines opcions pot escollir l'usuari i quin gràfic se li mostra en cada un dels casos. A continuació es pot veure una imatge dels principals tipus de gràfics que es generaran per al visor *Noise Map Viewer*.

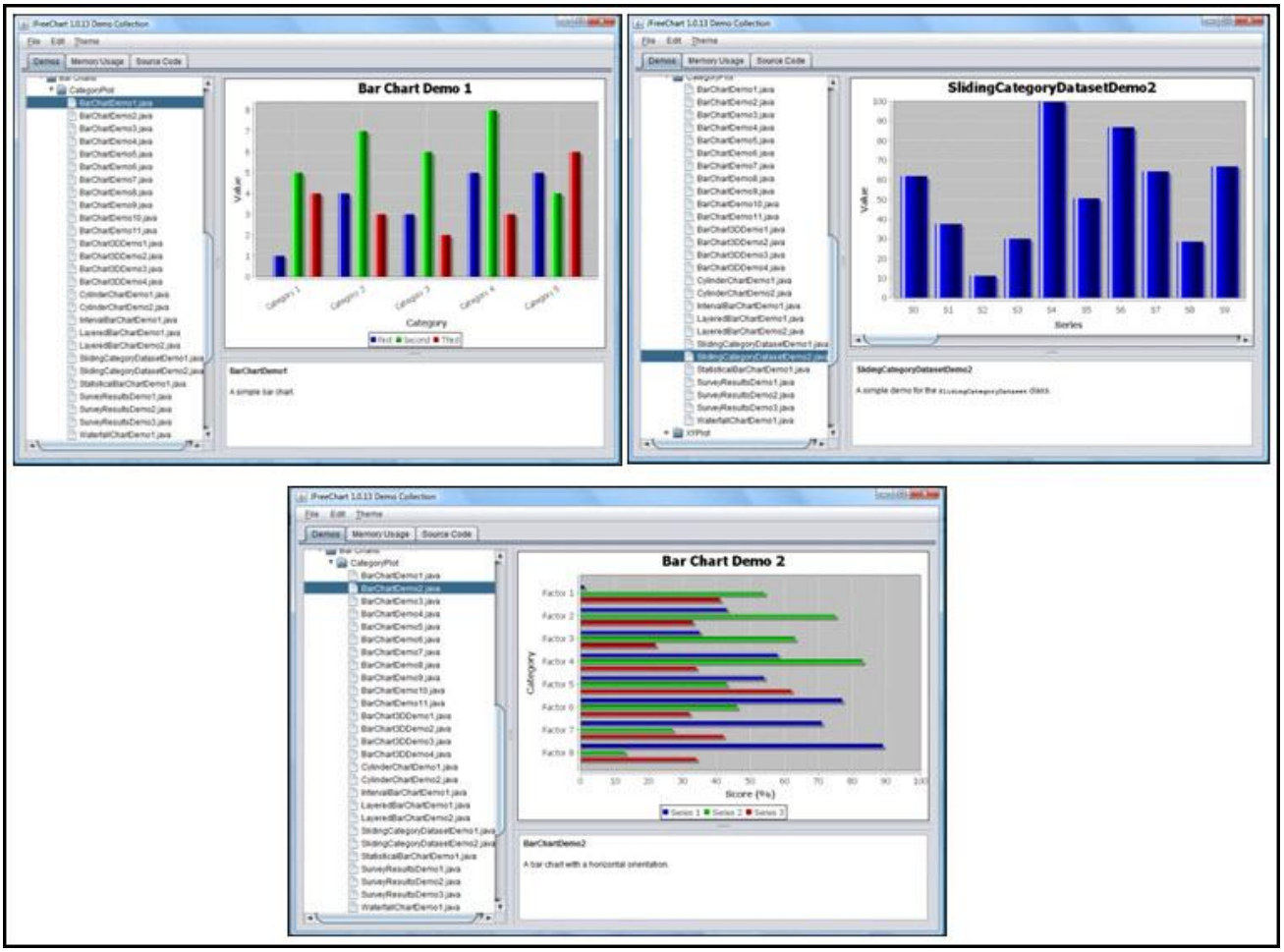


Figura 3.8. Exemple d'alguns gràfics que es poden generar amb la llibreria JFreeChart.

Font: http://www.jfree.org/jfreechart/samples.html

Tenint tot això en compte, i sabent com es comportarà cada un dels elements que formaran la nova arquitectura del sistema, podem fer el següent esquema:

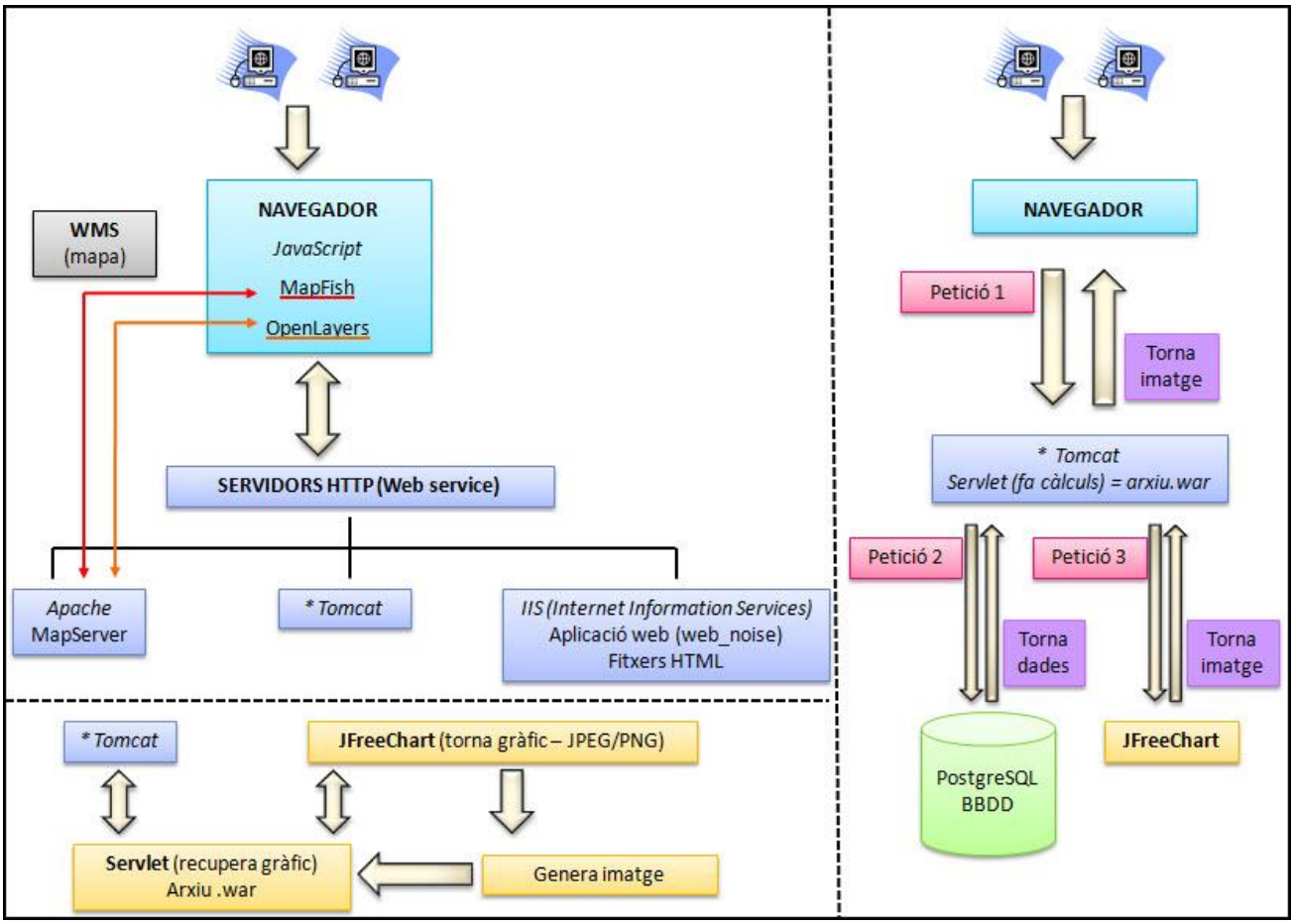


Figura 3.9. Arquitectura final del sistema.

3.3. Definició de les interfícies d'usuari

3.3.1. Especificació del comportament dinàmic de la interfície

Una interfície d'usuari és el mitjà amb que l'usuari pot comunicar-se amb un ordinador, i comprèn tots els punts de contacte entre l'usuari i l'equip. El principal objectiu d'una interfície d'usuari, és que aquest es pugui comunicar a través d'ella amb algun tipus de dispositiu (per exemple un PC). Aconseguida aquesta comunicació apareix un segon objectiu: que aquesta comunicació es pugui desenvolupar de la manera més fàcil i còmoda possible per a l'usuari.

Nosaltres ens fixarem en les Interfícies Gràfiques d'Usuari (GUI) que utilitzen un conjunt d'imatges i objectes gràfics per representar la informació i accions disponibles a la interfície. Aquests elements gràfics, permeten interactuar de forma molt més intuïtiva amb un sistema

informàtic, que no pas el clàssic sistema per línies de comandes. Generalment les interfícies gràfiques utilitzen controls virtuals com: botons, separadors de pestanyes, indicadors gràfics, barres de desplaçament, indicadors de progrés, i barres d'ajustament.

Per això, l'objectiu d'aquest punt, és fer una anàlisi dels processos del sistema d'informació que requereixen una interacció de l'usuari, amb la finalitat de crear una interfície que satisfaci tots els requisits establerts, tenint en compte els diferents tipus d'usuari a qui pot anar dirigit. Es tracta de crear una interfície flexible, coherent, eficient i fàcil d'utilitzar. D'aquesta manera es defineix el format i contingut de les interfícies de pantalla especificant el seu comportament dinàmic.

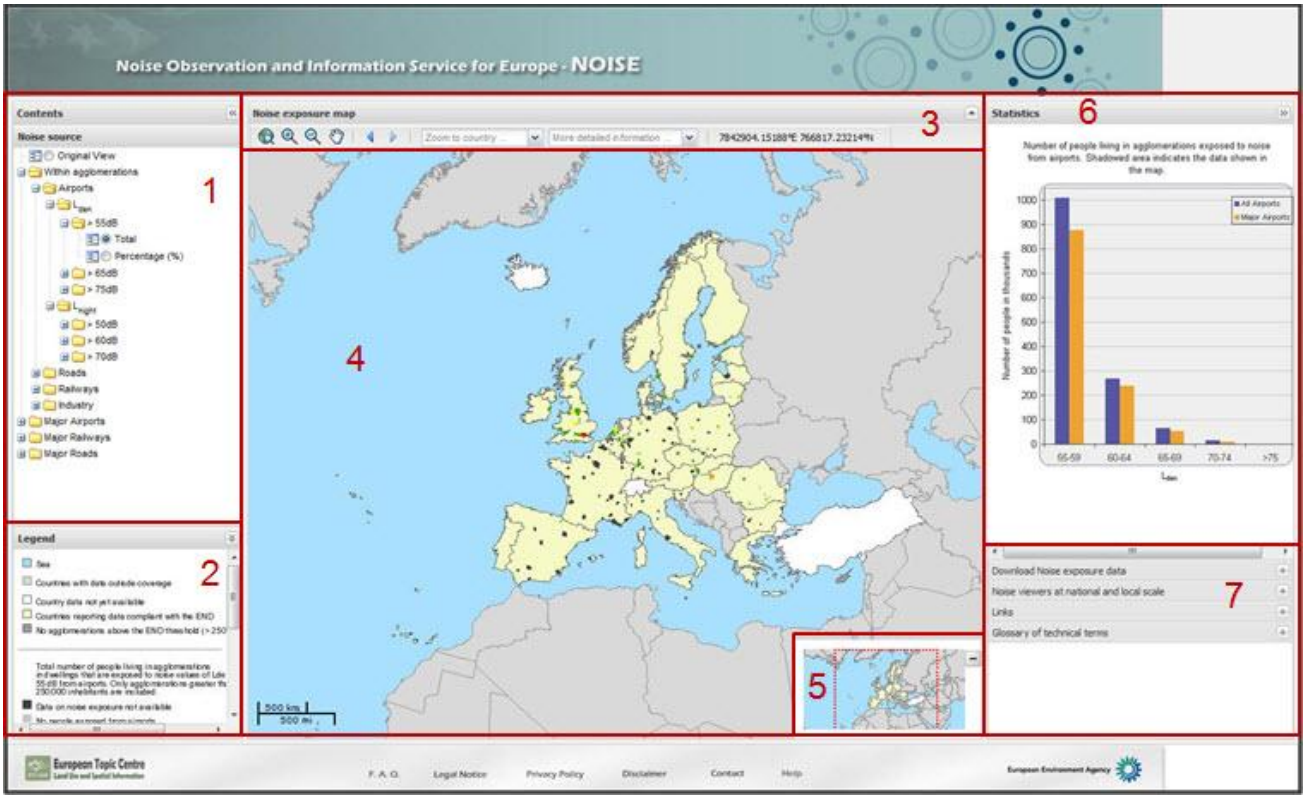


Figura 3.10. Interfície d'usuari. Noise Map Viewer.

Quan obrim el visor trobem els diferents components que l'integren. En algun dels components l'usuari ha de realitzar una acció per obtenir la informació desitjada, mentre que d'altres components són fixos o canvien automàticament per una acció realitzada en alguna altra part de la interfície.

En la taula de continguts (1) l'usuari ha d'escollir entre les opcions disponibles, la opció desitjada. Pot escollir si vol veure la informació dins o fora de les aglomeracions i una font de soroll determinada, si vol veure les dades del dia o de la nit, i pot escollir també una banda de soroll determinada. Un cop fet això la llegenda (2) s'actualitza sense que l'usuari hagi de fer res més, a

l'igual que el gràfic estadístic (6) i el mapa (4), que mostra el mapa temàtic corresponent. Si l'usuari en les eines de navegació (3), on hi trobem els *combos* de país i aglomeració entre d'altres, no escull res, per defecte ens mostra les dades d'Europa. Si no pot escollir per quin país vol veure les dades i també per quina de les seves aglomeracions, això comporta una actualització del mapa (4) que ens mostra amb un zoom la zona escollida, i una actualització del mapa general (5). Pel que fa les eines de navegació (3) també hi trobem uns botons de zoom, i les coordenades del punt en el qual tenim situat el punter del ratolí. Per últim tenim una zona de descàrrega de dades (7), en la qual l'usuari a través d'un *link* pot baixar les dades de soroll, amb les quals s'està generant tota la informació del visor. És aquí on l'usuari també pot trobar un apartat de *links* i un glossari de diferents termes.

La única diferència que existeix en la interfície d'usuari, és la manera en que es mostra el gràfic estadístic en la interfície inicial i la final (un cop generat el procés que automatitza la creació dels gràfics), de totes maneres, això és imperceptible per a l'usuari que no nota cap canvi en el funcionament del visor, ja que el canvi només afecta la part interna del sistema. En la interfície inicial la imatge es crea mitjançant un *link* a una imatge JPEG, i en la interfície final el gràfic es genera de manera automàtica, a través del *servlet* instal·lat dins del *Tomcat*, que a partir d'ara serà el que generarà la imatge del gràfic demanat.

3.4. Conclusions de l'anàlisi de tecnologies

Un cop hem definit el sistema, hem establert els requisits funcionals i tecnològics (on hem escollit la llibreria existent que més s'adapta a les necessitats del present projecte), i hem definit les interfícies d'usuari, podem veure la importància de seguir un projecte ben estructurat, sabent què s'ha de fer i què volem aconseguir en cada un dels punts anteriors. És en aquest punt que s'ha d'estructurar tota la informació recollida fins aleshores i acabar de prendre decisions (cada una d'elles ha quedat reflectida en el punt anterior corresponent).

Pel que fa l'anàlisi de les tecnologies i llibreries existents per a realitzar els gràfics estadístics, és important destacar la gran quantitat que en podem trobar, les múltiples opcions que ens donen i la gran quantitat de llenguatges en que es pot programar (el llenguatge més utilitzat, però, com es pot veure a la figura 3.7 és *Java*). També és important saber les característiques del nou component a integrar en un entorn ja creat, veure si són compatibles o no, ja que això suposa un estalvi de temps important. Cal destacar també la gran quantitat de tecnologies *Open Source* que es poden trobar per contra de les de pagament, i la gran ajuda i informació que podem obtenir per la xarxa a l'hora de programar.

4. Disseny del software

4.1. Casos d'ús

L'especificació dels casos d'ús és obligatòria en aquells projectes que estan orientats a objectes, com a recolzament a l'obtenció de requisits.

Els casos d'ús ens proporcionen un o més escenaris, que ens indiquen com hauria d'interactuar el sistema amb l'usuari per obtenir un objectiu específic. Un cas d'ús és una seqüència d'interaccions, que es desenvolupen entre un sistema i l'usuari, respecte un esdeveniment que aquest inicia sobre el propi sistema. Cada un d'ells descriu com arribar a una única resolució i descriu una característica del sistema, per això, sovint s'han d'especificar múltiples casos d'ús que ens acabin de definir el sistema. Els casos d'ús no descriuen funcionalitats internes del sistema, només mostren quins passos ha de seguir l'usuari per realitzar una tasca i obtenir la informació desitjada.

En el projecte s'han hagut d'especificar quines opcions pot escollir l'usuari: font de soroll, franja horària i banda de soroll (dB) a nivell europeu, de país o d'aglomeració. Com a resultat de la seva petició obté un tipus de gràfic estadístic o un altre.

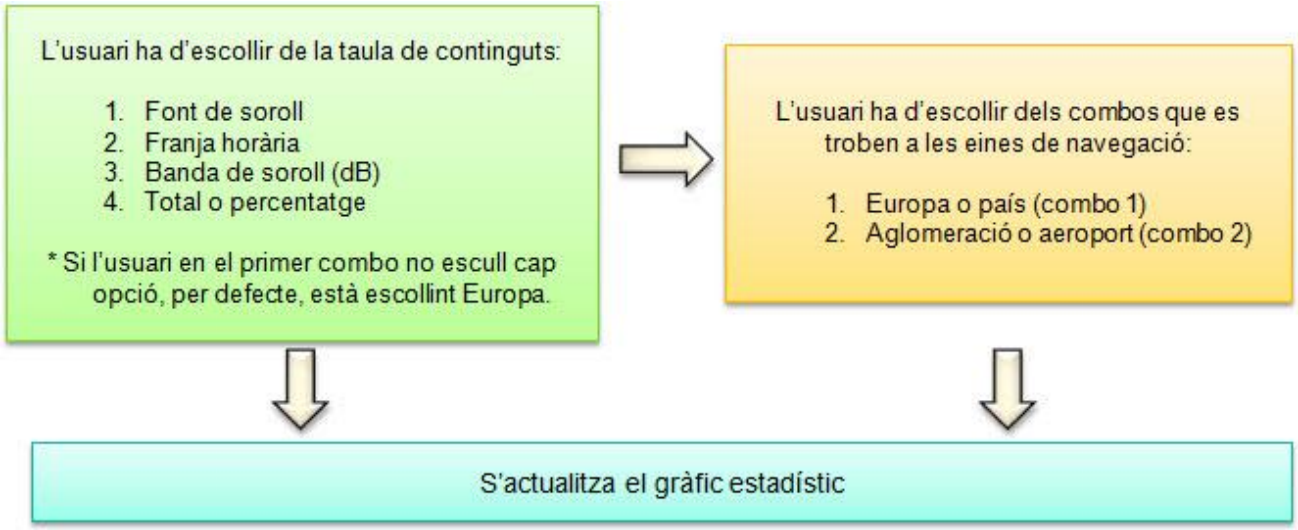


Figura 4.1. Esquema dels passos a seguir en el visor per obtenir les dades de soroll i el gràfic estadístic corresponent.

Pel que fa a la font de soroll, es pot escollir la informació dins (*within agglomerations*) i fora de les aglomeracions (*major*). Per cada una d'elles es pot escollir les dades per aeroports, carreteres i trens amb l'excepció de la indústria que només la podem obtenir dins les aglomeracions. Cada font de soroll es pot escollir a nivell d'Europa i de país. Pel que fa les aglomeracions només poden ser

escollides quan la font de soroll es troba situada dins de les aglomeracions. En el cas dels *major airports*, trobem que podem escollir les dades per aeroport, dada que es situa al mateix nivell de les aglomeracions. Cada una d'aquestes fonts es pot veure pels seus totals o en percentatge, excepte en el cas dels *major airports* que només podem obtenir el total.

Font de soroll	Europa	País	Aglomeració	Aeroport	Total	Percentatge	Tràfic
Within agglomerations Airports	X	X	X	-	X	X	-
Within agglomerations Roads	X	X	X	-	X	X	-
Within agglomerations Railways	X	X	X	-	X	X	-
Within agglomerations Industry	X	X	X	-	X	X	-
Major Airports	X	X	-	X	X	-	X
Major Railways	X	X	-	-	X	X (Relatiu: Hab/km)	-
Major Roads	X	X	-	-	X	X (Relatiu: Hab/km)	-

Figura 4.2. Resum de les dades a escollir dins la taula de continguts i combos del visor.

Un cop escollida la font de soroll que l'usuari vol, ha d'escollir la franja horària de la qual vol obtenir les dades: de dia (Lden: *day*, *evening*, *night*) o bé de nit (Lnight: *night*). Cada una d'aquestes franges, està dividida en tres bandes de soroll diferents, de dia són: >55, >65 i >75 i de nit són: >50, >60 i >70. Aquesta informació es troba reflectida en el gràfic en cinc categories diferents, que també canvien depenent de la franja horària escollida. De dia les categories són: 55-59, 60-64, 65-69, 70-74 i >75 i de nit són: 50-54, 55-59, 60-64, 65-69 i >70.

L'usuari pot veure de forma visual, sense recórrer a la taula de continguts, quina és la informació que està visualitzant en cada moment, ja que damunt del gràfic s'ha afegit una ombra que marca la banda de soroll seleccionada, excepte en el cas de país dins les aglomeracions, on s'agafa la banda de soroll escollida fent el sumatori de les categories superiors (p.e. >55 = 55-59 + 60-64 + 65-69 + 70-74 + >75; >65 = 65-69 + 70-74 + >75). Si no hi ha dades per la opció que l'usuari ha marcat, apareix el missatge de "*No data available*", i si aquell país no ha de donar les dades, per un o altre motiu, apareix el missatge "*The noise source does not apply to this country*" o "*The noise source does not apply to this agglomeration*", segons si volem les dades per país o aglomeració.

Franja horària i banda de soroll (dB)			
Lden	**Categories**	**Lnight**	**Categories**
>55	55-59 60-64	>50	50-54 55-59
>65	65-69 70-74	>60	60-64 65-69
>75	> 75	>70	>70

Figura 4.3. Franja horària i bandes de soroll (dB) amb les corresponents categories del gràfic.

Els gràfics varien en el seu format segons la opció escollida. Trobem gràfics amb dues barres, que comparen la mateixa font de soroll dins i fora de les aglomeracions, i d'una sola barra, que fan referència només a la font que s'està escollint sense cap altra dada més afegida. El gràfic pot ser de barres verticals o horitzontals (aquests últims només en el cas de país dins les aglomeracions). En d'altres casos el gràfic no és possible al nivell desitjat, i apareix un missatge que alerta l'usuari conforme ha d'escollir un altre tipus d'informació.

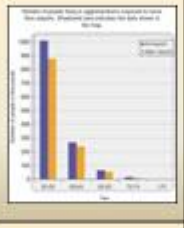
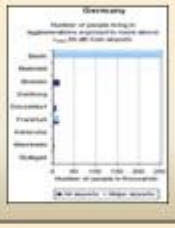
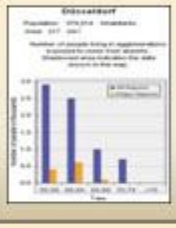
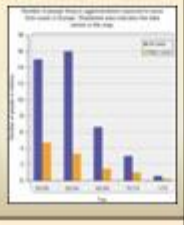
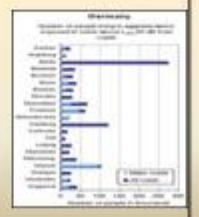
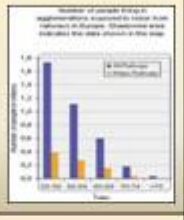
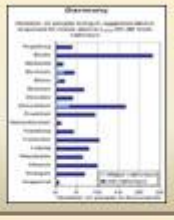
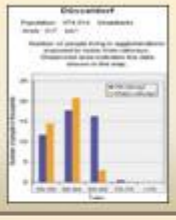
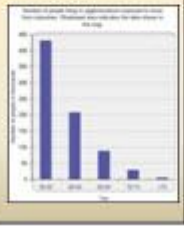
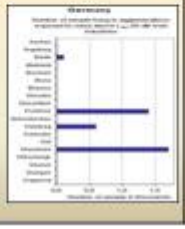
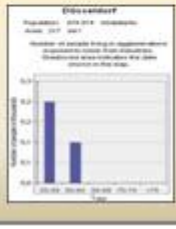
Tipus de gràfic estadístic			
Font de soroll	**Europa**	**País**	**Aglomeració**
Within agglomerations Airports			
Within agglomerations Roads			
Within agglomerations Railways			
Within agglomerations Industry			

Figura 4.4. Tipus de gràfic estadístic dins les aglomeracions segons la font de soroll.

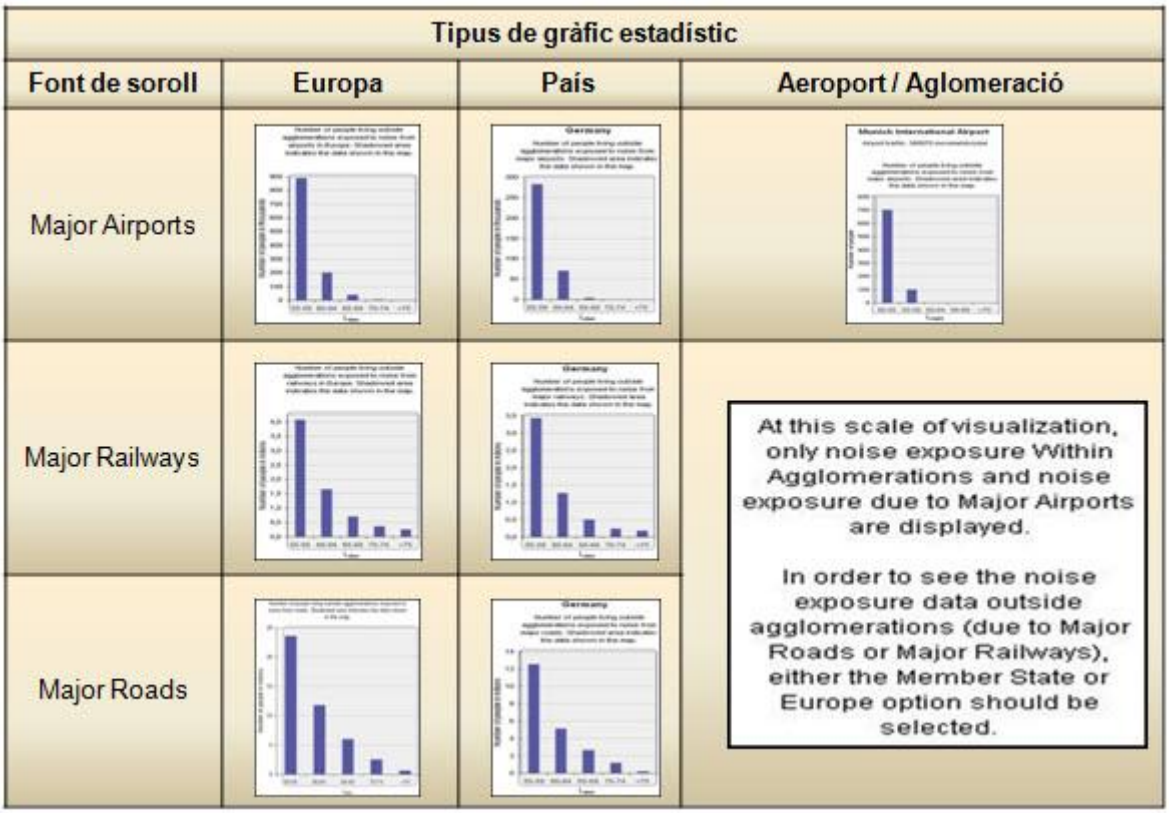


Figura 4.5. Tipus de gràfic estadístic fora de les aglomeracions segons la font de soroll.

Els casos d'ús ens seran molt útils a l'hora de començar a programar, doncs ens crea un esquema dels processos i/o mètodes que hem de crear, si poden ser generals per una o vàries peticions de l'usuari, o si cada cas requereix un procés únic i diferenciat. Tot això servirà per fer la tasca més fàcil a l'hora de crear consultes SQL (Structured Query Language), per fer la petició de les dades requerides a la base de dades, així com aquelles coses comunes a l'hora de crear els diferents gràfics estadístics que es poden obtenir.

5. Desenvolupament de l'aplicació

5.1. Preparació de l'entorn de treball: instal·lació de programari

Per realitzar aquest projecte s'han hagut d'instal·lar diversos programes o components. Cada un d'ells serveix en una part diferent del projecte, tot i que en alguns casos s'han utilitzat de forma paral·lela.

Els programes utilitzats són:

- *Microsoft Office* ***Access***: sistema de gestió de base de dades relacional.

- *Microsoft Office* ***Excel***: aplicació per a utilitzar fulls de càlcul.

- *Mozilla* ***FireFox*** *3.5.3*: navegador web lliure multi-plataforma.

- ***Eclipse***: és un entorn de desenvolupament integrat (IDE) de codi obert multi-plataforma, per *Java EE*. Concretament, es farà servir *l'Eclipse Galileo 3.5*.

- *Apache* ***Tomcat 6.x*** : és un servidor web, contenidor de *servlets*.

- ***MapServer*** - *MS4W*: el *MapServer* és una plataforma *Open Source* per publicar dades espacials i mapes interactius en aplicacions web. El paquet *MS4W* inclou, entre d'altres, l'***Apache*** que és un servidor web HTTP de codi obert.


- ***JFreeChart 1.0.13***: llibreria *Java* que genera gràfics. Inclou el *JFreeChart 1.0.13 javadocs*. Per tal que aquest funcioni s'ha d'instal·lar també la llibreria ***JCommon 1.0.16***.

- *Pentaho Kettle 3.2.0*: eina que integra metadades i facilita la seva extracció, transformació i càrrega. Concretament farem servir el component ***Spoon***.

- ***PostgreSQL 8.4***: sistema de gestió de base de dades relacional orientada a objectes de software lliure. També s'ha hagut d'instal·lar la interfície gràfica ***pgAdminIII 1.10*** del *PostgreSQL*, que serveix com a eina d'administració de les bases de dades incloses dins d'aquest. Permet, entre d'altres, fer consultes SQL per al desenvolupament de bases de dades complexes.

- ***Macromedia Dreamweaver 8***: programa que s'enfoca a la construcció i edició d'aplicacions web.

En els següents apartats podrem anar veient cada un d'aquests programes o components amb més detall.

5.1.1. Servidors

Per a realitzar aquest projecte es necessiten dos servidors. D'una banda necessitem un servidor web HTTP, i de l'altra un servidor web que faci de contenidor de *servlets* (el procés que s'ha de generar per crear el gràfics estadístics de forma automàtica serà un *servlet*).

Pel que fa el servidor web HTTP s'ha fet servir l'*Apache*. Per instal·lar-lo s'ha baixat el paquet *MapServer – MS4W*, que és l'entorn que s'ha utilitzat per construir el visor *Noise Map Viewer*, que instal·la un entorn de servidor web preconfigurat que inclou els següents components:

- *Apache* HTTP *Server* versió 2.2.11
- PHP versió 5.3.0
- *MapServer* CGI 5.4.2
- MapScript 5.4.2 (CSharp, *Java*, PHP, Python)
- Inclou suport per Oracle 11g, i dades SDE
- MrSID suport integrat
- GDAL/OGR 1.6.1 i *Utilities*
- *MapServer Utilities*
- PROJ *Utilities*
- Shapelib *Utilities*
- Shp2tile *Utility*
- Shpdiff *Utility*
- AVCE00 *Utilities*
- OGR/PHP *Extension* 1.0.0
- OWTChart 1.2.0

Aquest paquet permet als usuaris de *MapServer* instal·lar un entorn de treball per al desenvolupament de *MapServer* dins de Windows, també serveix per empaquetar i distribuir aplicacions de *MapServer*, com és el cas del visor *Noise Map Viewer*.

Dins d'aquest servidor s'instal·la la carpeta ***web_noise***, que és la que conté la web, concretament dins la carpeta *htdocs*.

Com a servidor contenidor de *servlets* s'ha instal·lat *Tomcat 6.0.20* (*Apache Tomcat*) dins el nostre projecte (Proj_jfree).

El *Tomcat* és un servidor web amb suport de *servlets* i JSP (*JavaServer Pages*) que proporciona un entorn per al codi *Java*, i funciona en qualsevol sistema operatiu que disposi de màquina virtual *Java* (JVM - *Java Virtual Machine*), encara que no és un servidor d'aplicacions, i per això es presenta sovint en combinació amb el servidor web *Apache*. Al principi l'ús de *Tomcat* de forma autònoma era només recomanable per a entorns de desenvolupament i entorns amb requisits mínims, tant de velocitat com de gestió de transaccions. Actualment *Tomcat* pot funcionar per si mateix com a servidor web autònom en entorns d'alt nivell de tràfic i alta disponibilitat, de fet es planteja en algun moment instal·lar la carpeta ***web_noise*** dins del *Tomcat* i no dins de l'*Apache*.

Per defecte a l'instal·lar *Tomcat* es crea la jerarquia de directoris següent:

- bin: arrencada, aturada, i altres *scripts* i executables.
- conf: arxius per la configuració de *Tomcat*.
- lib: conté classes i recursos en arxius JAR.
- logs: registre i arxius de sortida.
- temp: directori que utilitza la JVM per arxius temporals.
- webapps: directori que conté les aplicacions web.
- work: fitxers temporals, pàgines JSP precompilades, i altres fitxers intermedis de les aplicacions web.

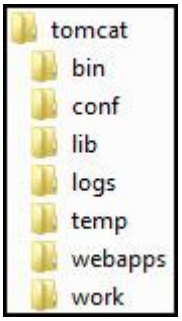


Figura 5.1. Jerarquia de directoris del Tomcat.

Pel que respecta a la versió de *Tomcat 6.0.20* trobem les següents característiques:

- Implementació del *Servlet 2.5* i *JSP 2.1*.
- Suport per *Unified Expression Language 2.1*.
- Dissenyat per funcionar amb *Java SE 5.0* i posteriors.

5.1.2. Eclipse

L'*Eclipse* és un entorn de desenvolupament integrat de codi obert, multi-plataforma per desenvolupar Aplicacions de Client Enriquit (RCP - *Rich Client Platform*) que està format pels següents components:

- Plataforma principal: inici d'*Eclipse* i execució dels connectors.
- OSGi: plataforma per a *bundling* estàndard.
- SWT: un *widget toolkit* portable.
- JFace: per treballar amb texts.
- Workbench: vistes, editors, perspectives i assistents.

L'*Eclipse* s'utilitza per desenvolupar entorns de desenvolupament integrats i utilitza mòduls (*plug-in*) per proporcionar la seva funcionalitat, proveeix al programador amb *frameworks* molt rics per al desenvolupament d'aplicacions gràfiques, definició i manipulació de models de programari, aplicacions web, etc. Inclou les eines de desenvolupament de *Java*, oferint un IDE amb un compilador de *Java* intern i un model complet dels arxius font de *Java*. Fa ús d'un espai de treball (*workspace*) permetent modificacions externes dels arxius mentre s'actualitzi l'espai de treball corresponent.

Per al nostre projecte s'ha baixat l'*Eclipse IDE* per *Java EE Developers*. Aquest paquet dóna les eines de desenvolupament de *Java*, suportant el desenvolupament de qualsevol aplicació *Java*, incloent-hi els *plug-ins* d'*Eclipse*. Inclou l'*Eclipse Galileo 3.5*. disponible des del 24 de juny de 2009.

Quan obrim l'*Eclipse*, primer de tot hem de definir on volem instal·lar el *workspace*:

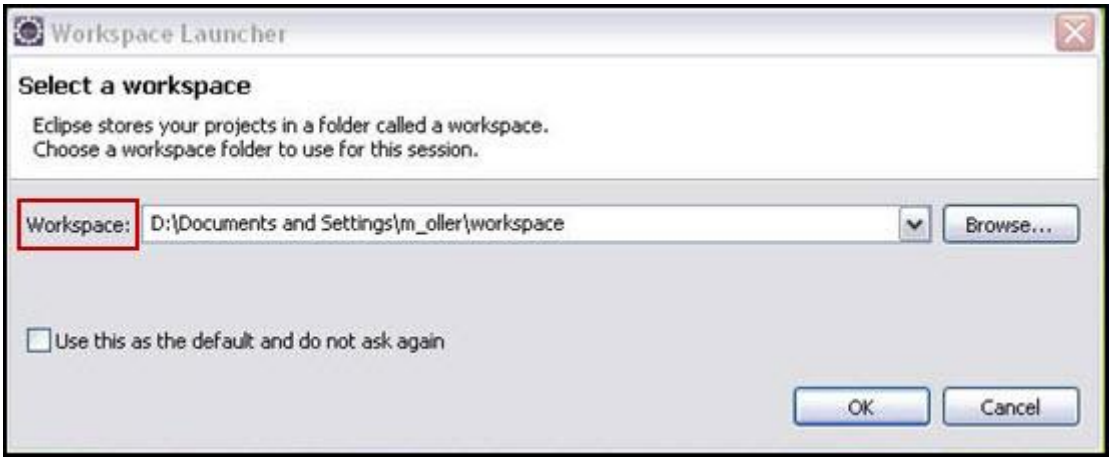


Figura 5.2. Workspace de l'Eclipse.

Tot seguit hem de crear el projecte (***noise***) on generarem el *servlet*:

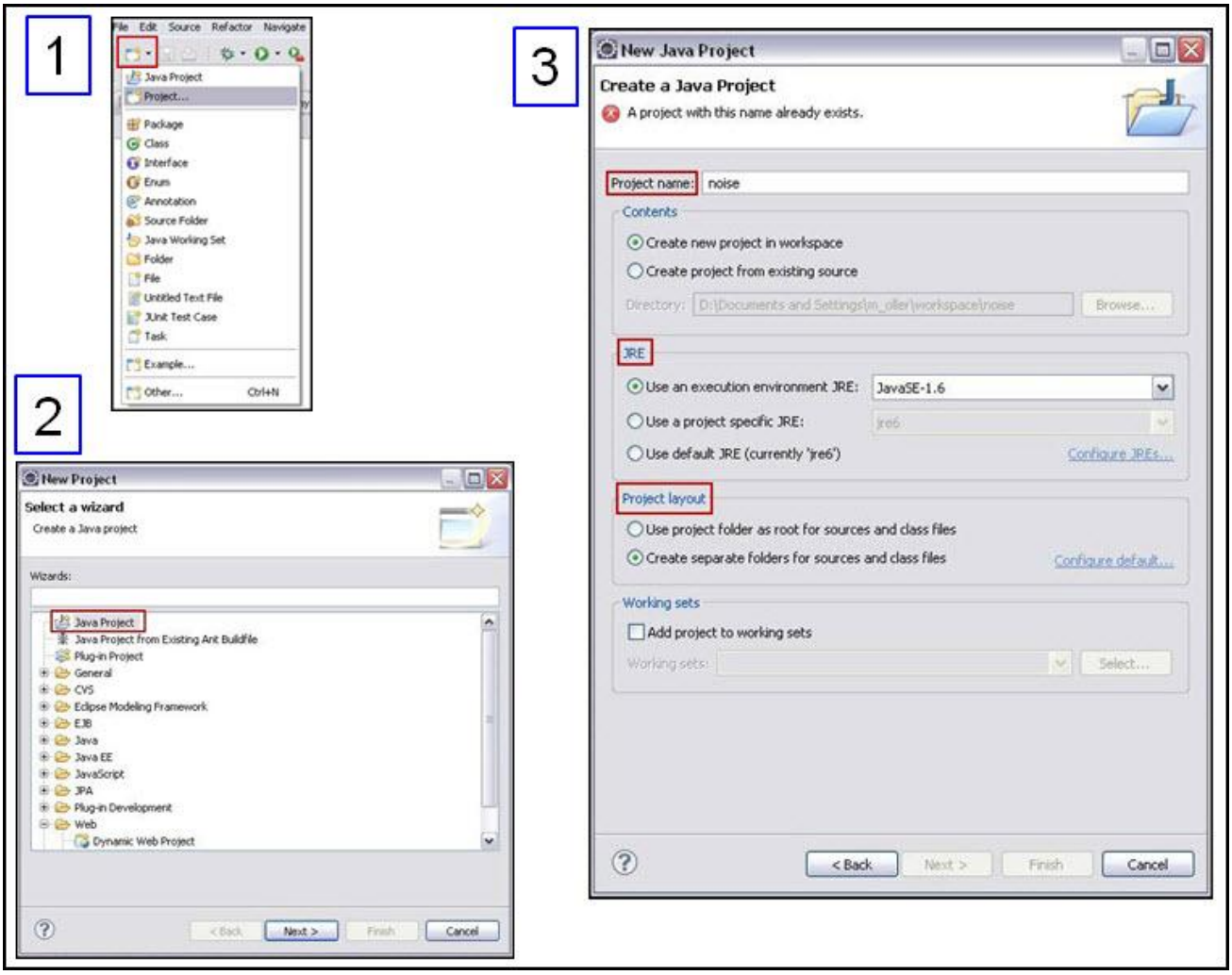


Figura 5.3. Com crear un nou projecte amb l'Eclipse.

Per últim definim el servidor que utilitzarem:

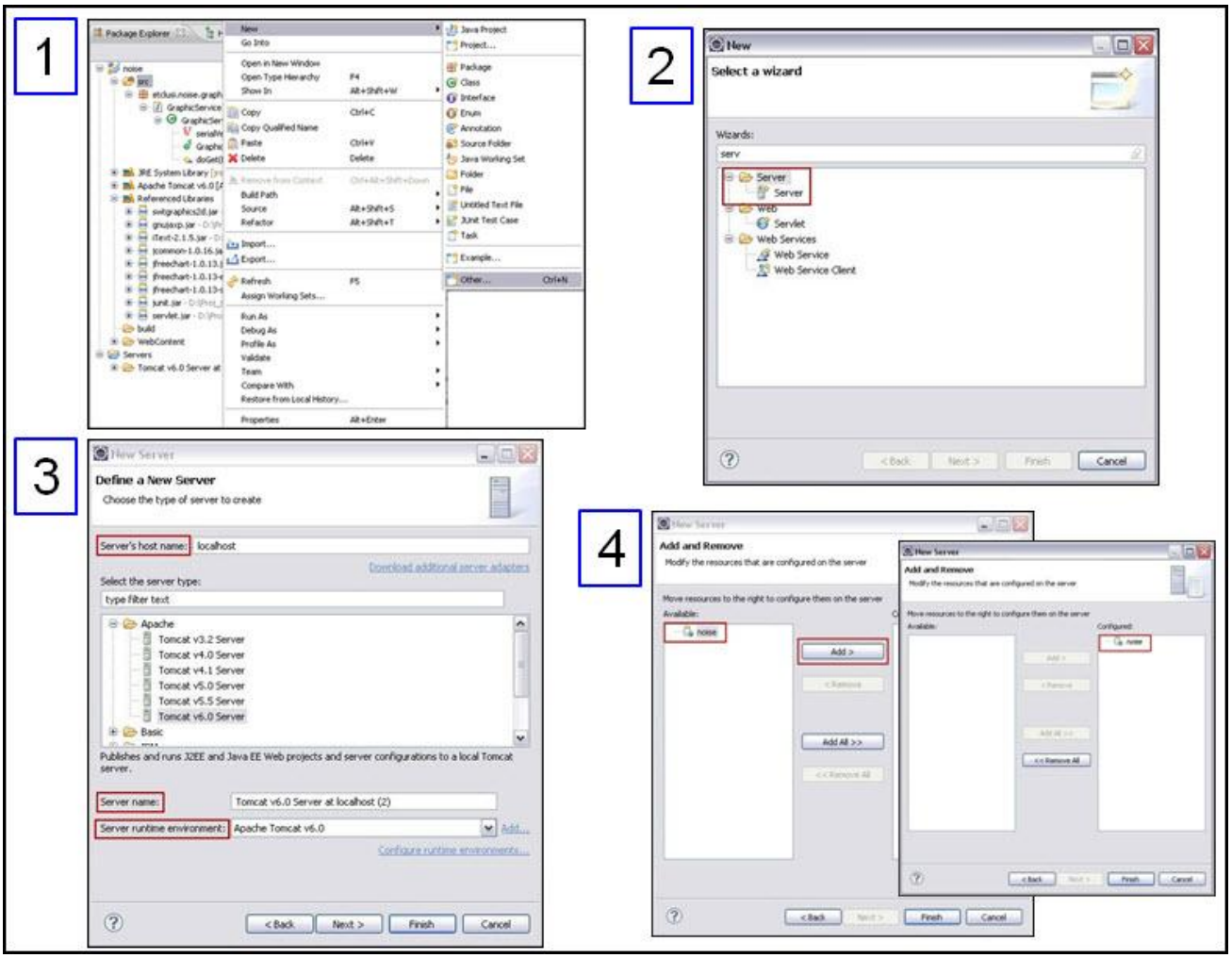


Figura 5.4. Com afegir el nou servidor al projecte noise amb l'Eclipse.

5.1.3. JFreeChart

JFreeChart és una llibreria *Java* que ens permet crear múltiples tipus de gràfics, suporta gràfics circulars (2D i 3D), gràfics de barres (horitzontal i vertical, regular i apilat), gràfics de línies, gràfics de dispersió, gràfics de sèries de temps, diagrames de Gantt, termòmetres i rellotges entre d'altres. Aquesta llibreria pot ser utilitzada en aplicacions, *applets*, *servlets* i JSP (*JavaServer Pages*).

JFreeChart utilitza, a més a més, una altra llibreria que també ha de ser instal·lada. Aquesta és *JCommon*, que és una llibreria de classes *Java*, que conté una sèrie de classes que donen suport a:

- Configuració i dependència en la gestió del codi.
- Marc general de registre.
- Utilitats de text.
- Classes de la interfície d'usuari per veure informació sobre les aplicacions.
- Disseny personalitzat.
- Panell selector de dades.
- Utilitats de serialització (procés de codificació d'un objecte en un arxiu per transmetre'l a través d'una connexió de xarxa, com una sèrie de bytes o un format més llegible com p.e. XML).

Quan creem el projecte s'han d'instal·lar aquestes dues llibreries i també els *javadocs* (documentació API en format HTML a partir del codi font *Java*) de *JFreeChart*:

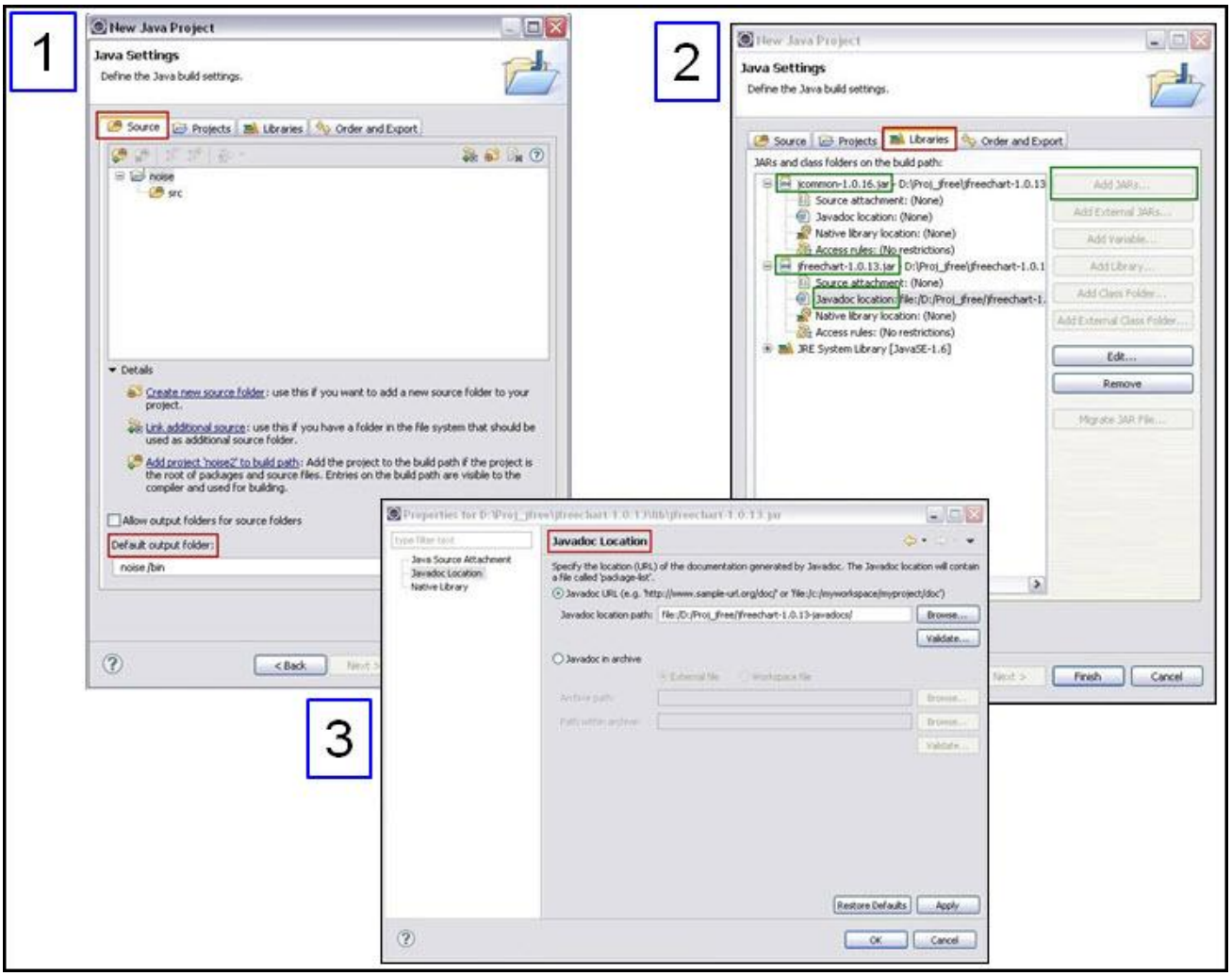


Figura 5.5. Instal·lació de les llibreries JFreeChart i JCommon al projecte noise amb l'Eclipse.

5.2. Base de dades

Per tal de construir una base de dades que pugui ser inclosa dins el servidor, que també conté la pàgina web, hem de decidir utilitzar una base de dades potent, que pugui incloure un gran nombre de dades i que a més a més sigui ràpida.

Hi ha diferents bases de dades que poden ser utilitzades per la nostra tasca com p.e. Microsoft SQL Server. En el nostre cas s'ha decidit utilitzar *PostgreSQL*.

PostgreSQL és un sistema de gestió de base de dades relacional orientada a objectes, de software lliure (*Open Source*) i de classe empresarial. Les seves principals característiques són:

- Té una arquitectura consolidada que dóna confiança, integritat de dades i correcció.
- Funciona en tots els sistemes operatius principals.
- És compatible amb ACID (en una base de dades, conjunt de característiques necessàries per a que una sèrie d'instruccions pugui ser considerada com una transacció).
- Té suport complet per subconsultes, les característiques d'integritat de dades inclou les claus principals, les claus foranies, restriccions, actualització, herència de taules, unions, procediments emmagatzemats, vistes, etc.
- Inclou la majoria de tipus de dades SQL, incloent *integer*, *numeric*, *boolean*, *char*, *varchar*, *date*, *interval*, i *timestamp* (seqüència de caràcters que indiquen data i hora en que ha passat un esdeveniment; *log*).
- Permet definir i crear un tipus de dades propi, amb funcions de recolzament i operadors que defineixen el seu comportament.
- És compatible amb l'emmagatzematge d'objectes binaris com imatges, vídeos o sons.
- Té interfícies de programació per *C / C* + +, *Java*, *.NET*, *Perl*, *Python*, *Ruby*, *Tcl* i ODBC entre d'altres.
- El llenguatge de programació que utilitza és molt similar al d'*Oracle*.
- Compta amb el sistema MVCC (Versió Multi-Control de Concurrència).
- És compatible amb conjunts de caràcters internacionals, *Unicode* i té en compte la configuració regional.
- Pot manipular una gran quantitat de dades i el nombre d'usuaris en un mateix instant pot ser elevadíssim.

Els límits generals de *PostgreSQL* són:

Limit	Value
Maximum Database Size	Unlimited
Maximum Table Size	32 TB
Maximum Row Size	1.6 TB
Maximum Field Size	1 GB
Maximum Rows per Table	Unlimited
Maximum Columns per Table	250 - 1600 depending on column types
Maximum Indexes per Table	Unlimited

Figura 5.6. Taula dels límits de PostgreSQL.

Font: http://www.postgresql.org/about/

Com a complement de *PostgreSQL* s'ha utilitzat *pgAdminIII* que és la seva plataforma de desenvolupament i administració, també de codi obert, dissenyada per respondre a les necessitats de tots els usuaris per escriure consultes SQL simples (inclou un editor de sintaxi SQL) per tal de desenvolupar bases de dades complexes.

A l'hora d'instal·lar *PostgreSQL*, a través de l'*Stack Builder 2.1.0* podem instal·lar altres components com *pgAdminIII*, *PostGIS* (mòdul que afegeix suport d'objectes geogràfics, converteix al *PostgreSQL* en una base de dades espacial per a la seva utilització en Sistemes d'Informació Geogràfica), etc. Nosaltres només instal·larem *pgAdminIII*.

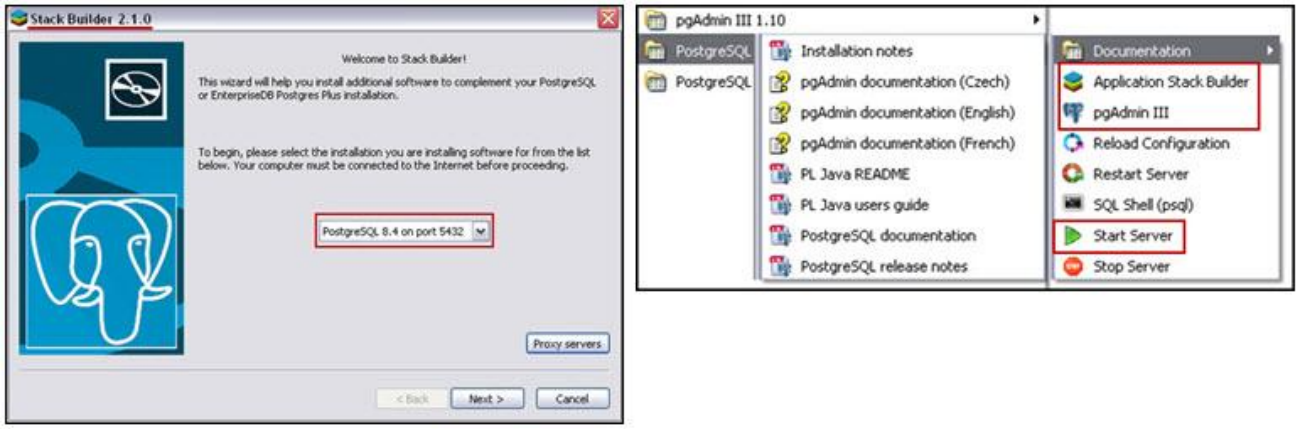


Figura 5.7. Instal·lació de PostgreSQL i components instal·lats.

Amb el *pgAdminIII* creem la nova base dades de *PostgreSQL* que anomenem ***postgres***.

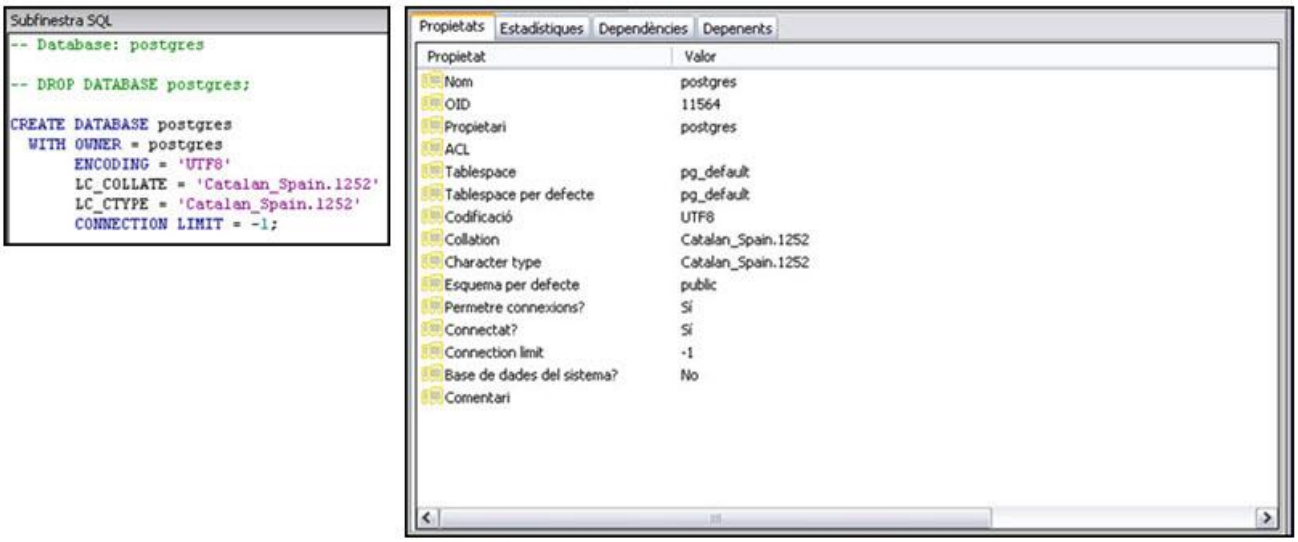


Figura 5.8. Creació de postgres i les seves propietats.

Un cop creada la nova base de dades de *PostgreSQL* (***postgres***) ja podem començar a passar-hi les dades de soroll que es troben dins la base de dades d'*Access* (***DataViewer***) i les taules que s'han construït amb l'*Excel* (***title***), que són taules per als títols variables dels gràfics estadístics.

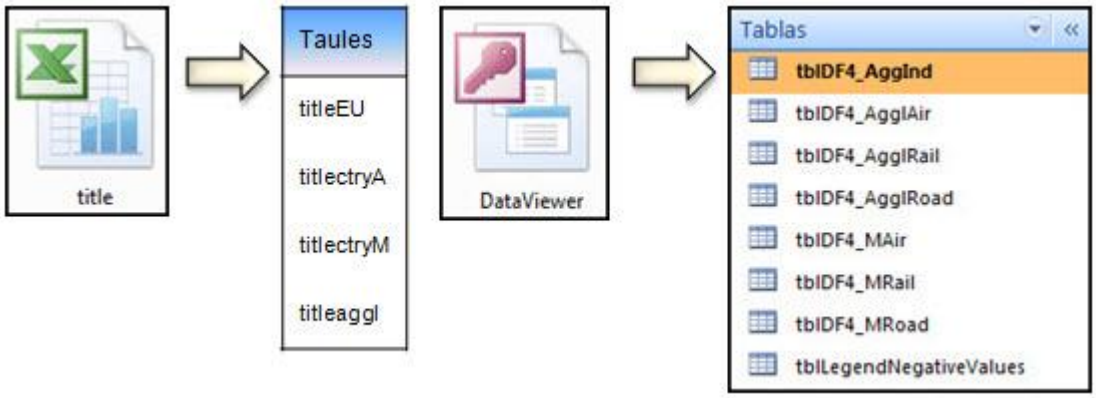


Figura 5.9. Taules de l'arxiu title i de la base de dades DataViewer.

Per a realitzar aquesta tasca s'ha utilitzat l'*Spoon* que ens ha servit per automatitzar els passos intermedis de bolcatge de dades d'una base de dades a l'altra. L'*Spoon* és una interfície gràfica d'usuari que permet dissenyar les transformacions i feines que es poden executar amb les eines de *Kettle* (programa d'integració de dades que ajuda a l'extracció, transformació, transport i càrrega de dades).

Icon	Description
	Create a new job or transformation
	Open transformation/job from file if you're not connected to a repository or from the repository if you are connected to one.
	Save the transformation/job to a file or to the repository.
	Save the transformation/job under a different name or filename.
	Open the print dialog.
	Run transformation/job: runs the current transformation from XML file or repository.
	Preview transformation: runs the current transformation from memory. You can preview the rows that are produced by selected steps.
	Run the transformation in debug mode allowing you to troubleshoot execution errors.
	Replay the processing of a transformation for a certain date and time. This will cause certain steps (Text File Input and Excel Input) to only process rows that failed to be interpreted correctly during the run on that particular date and time.
	Verify transformation: Spoon runs a number of checks for every step to see if everything is going to run as it should.
	Run an impact analysis: what impact does the transformation have on the used databases.
	Generate the SQL that is needed to run the loaded transformation.
	Launches the database explorer allowing you to preview data, run SQL queries, generate DDL and more.

Figura 5.10. Barra d'eines de l'Spoon.

Font: http://wiki.pentaho.com/display/EAI/.01+Introduction+to+Spoon

Per poder connectar ***DataViewer*** amb l'*Spoon* i poder començar les transformacions d'un tipus de taula a una altra, hem de crear una connexió ODBC (*Open Database Connectivity*) que farà possible accedir a qualsevol de les seves dades des de qualsevol aplicació.

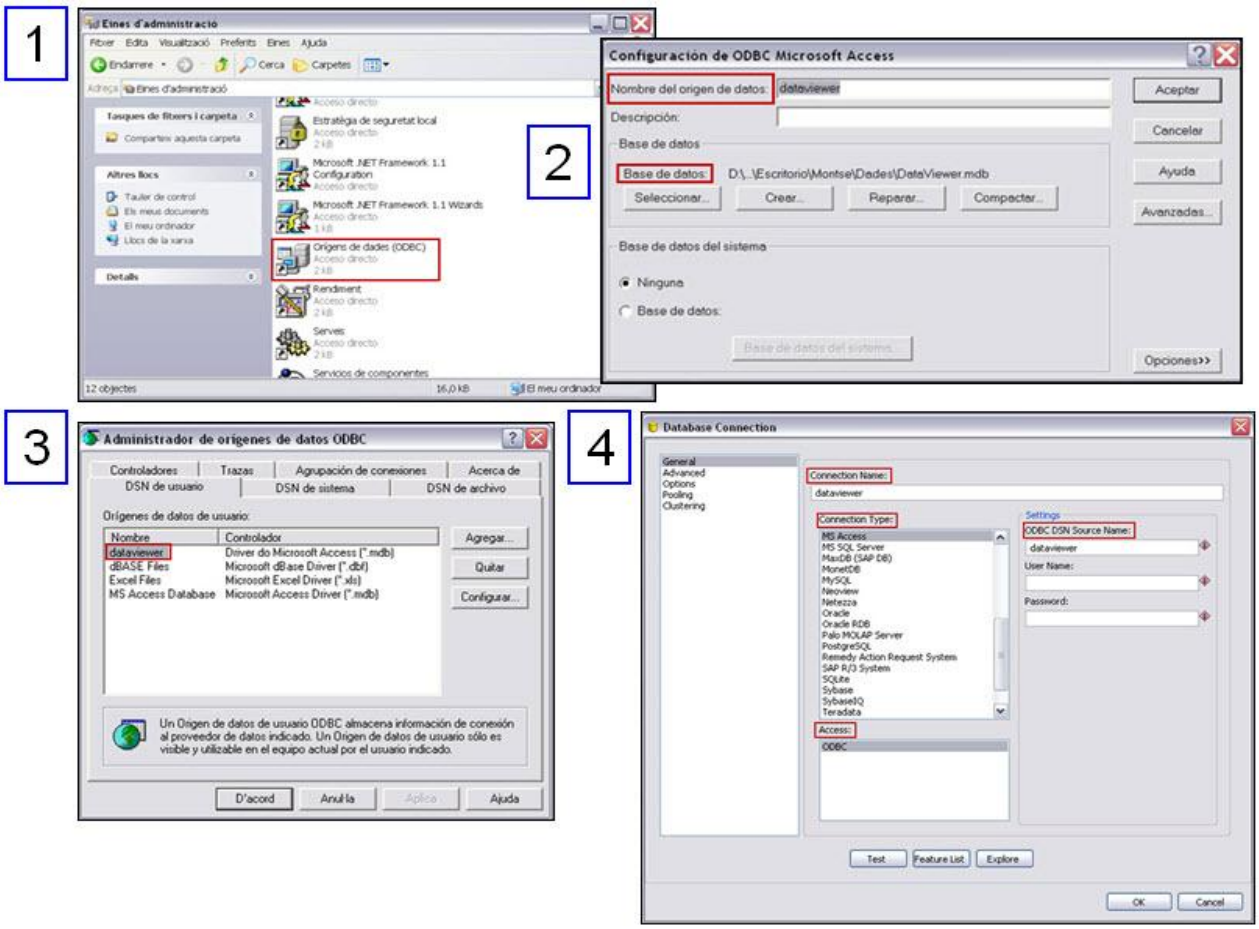


Figura 5.11. Connexió ODBC. DataViewer.

Un cop feta aquesta connexió hem d'escollir una taula d'entrada (***DataViewer***) i una de sortida (***postgres***, base de dades creada prèviament). D'aquesta manera escollim les dades d'entrada i creem les noves taules dins de ***postgres*** on introduïm les dades de soroll necessàries.

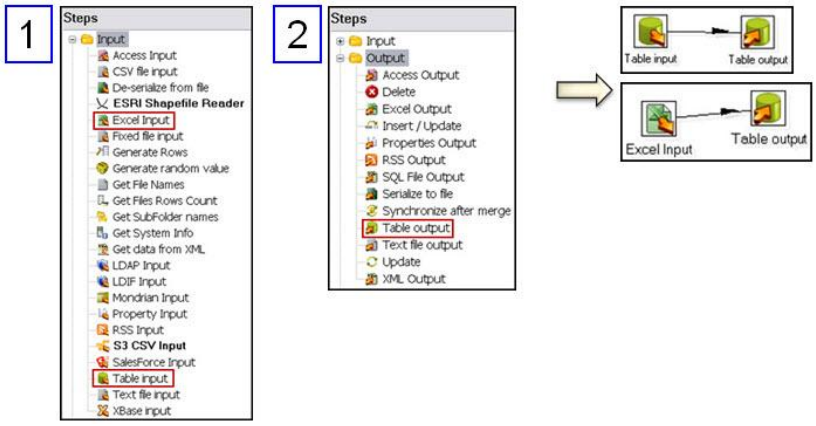


Figura 5.12. Transformacions de l'Spoon. Taules d'entrada i de sortida.

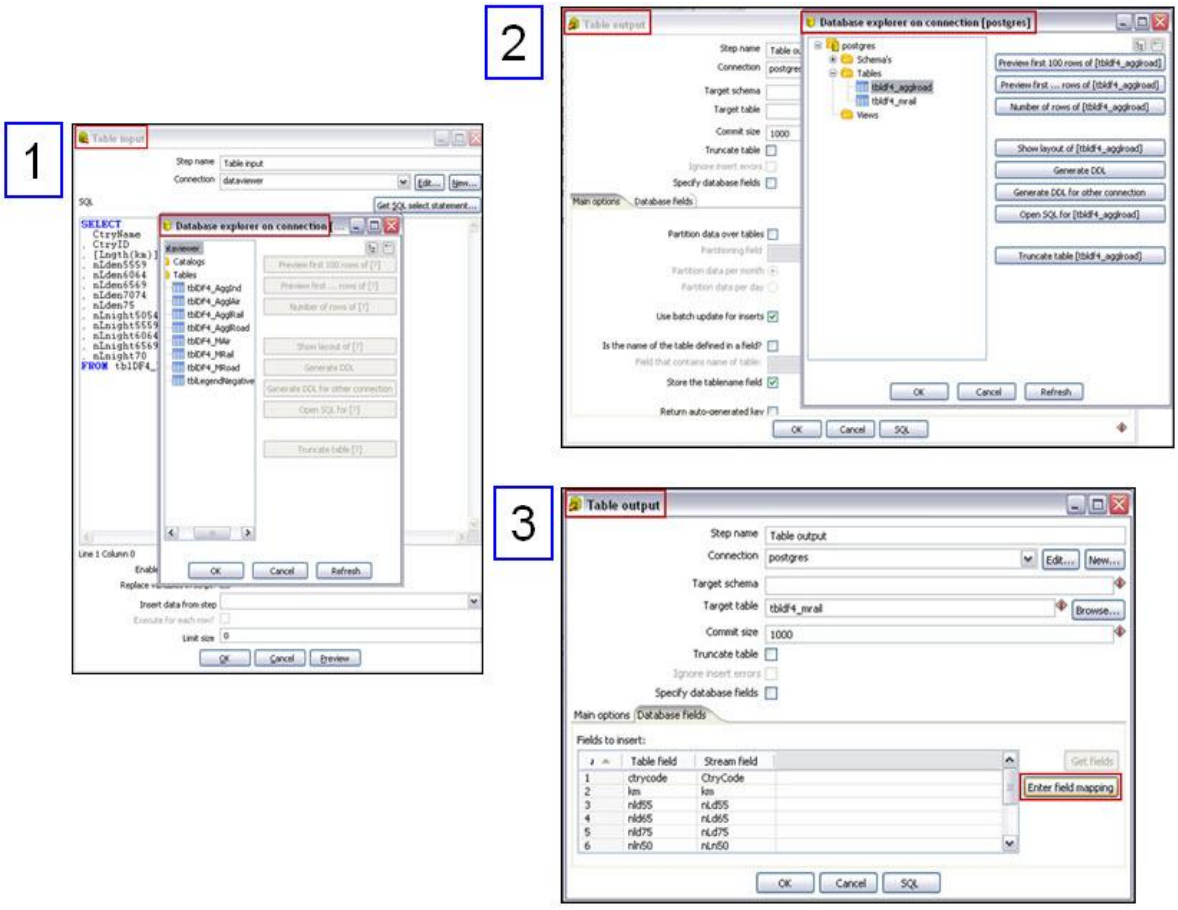


Figura 5.13. Selecció de la taula d'entrada i la de sortida amb l'Spoon.

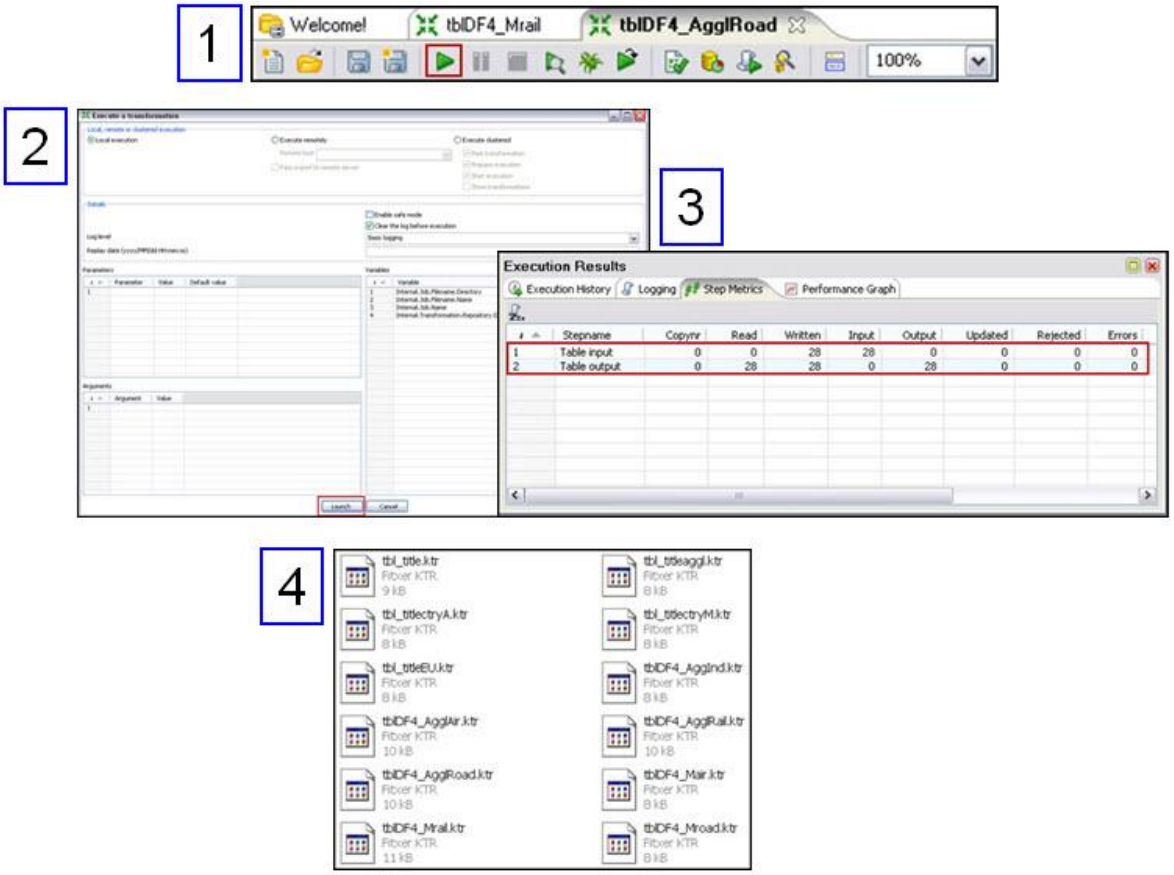


Figura 5.14. Execució de la transformació amb l'Spoon i taules finals.

Un cop tenim les taules dins de ***postgres*** ja podem començar a donar les claus primàries de cada una de les taules.

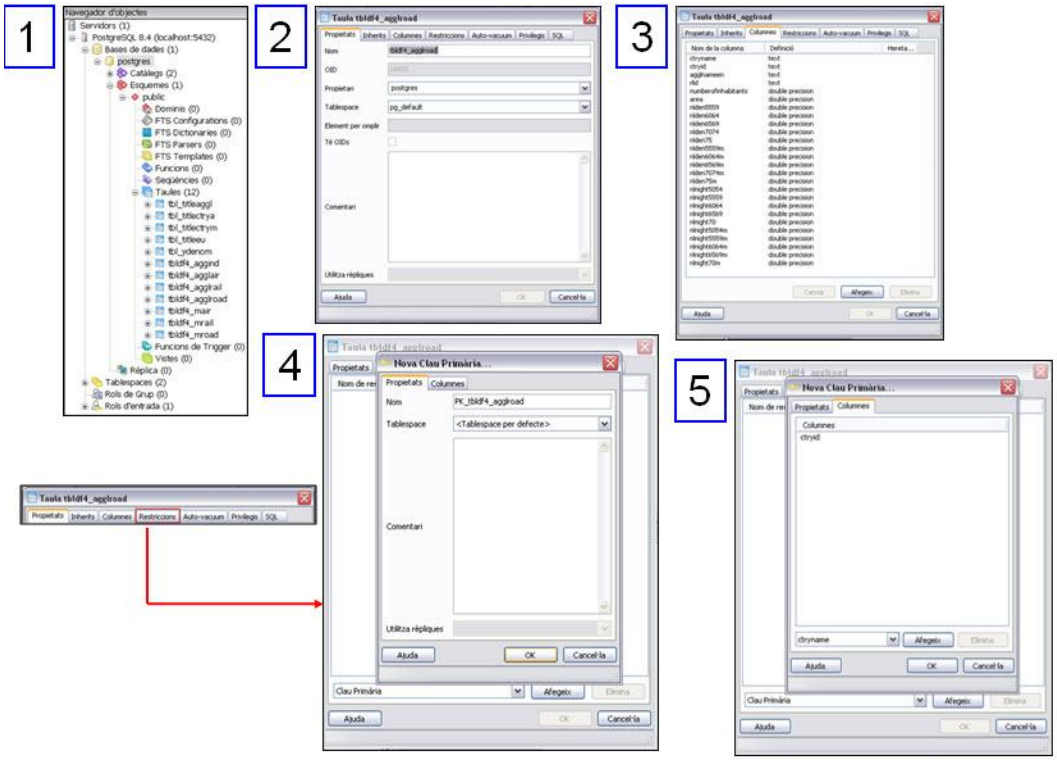


Figura 5.15. Taules de postgres i assignació de claus primàries.

Per últim farem una còpia de seguretat de la base de dades ***postgres***, la qual servirà per instal·lar-la en el servidor i d'altres equips, juntament amb el *servlet* i la web. Marquem la opció "amb OIDs" ja que és l'identificador de cada un dels diferents objectes de la base de dades.

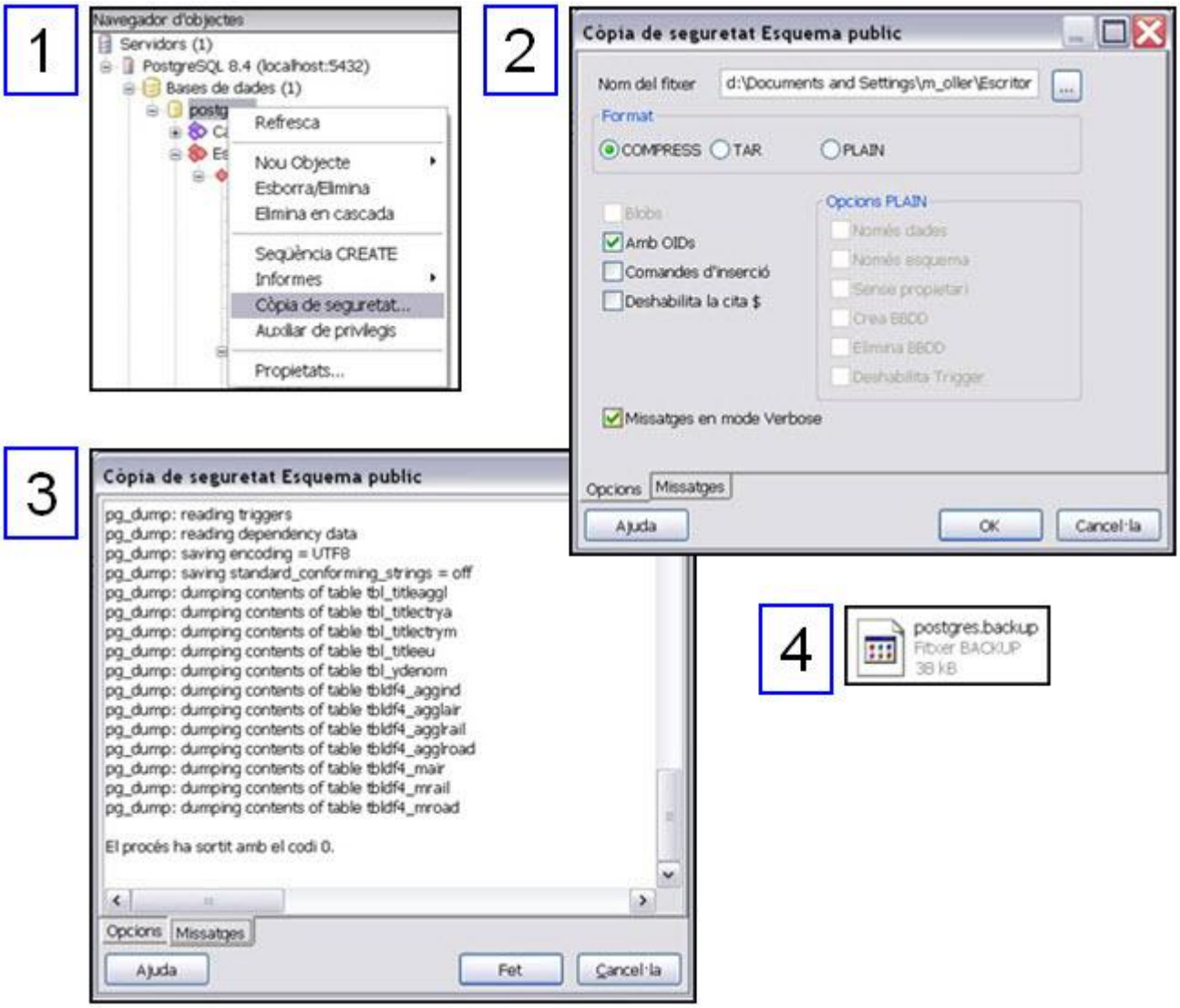


Figura 5.16. Còpia de seguretat de postgres.

5.3. Llenguatge de programació

Per tal de desenvolupar el *servlet* que generarà els gràfics s'ha fet servir el llenguatge de programació *Java*.

Java és un llenguatge de programació orientat a objectes desenvolupat per *Sun Microsystems* a principis dels anys 90. El llenguatge pren molta sintaxi dels llenguatges *C* i *C* ++ tot i que té un model d'objectes més simple. És un llenguatge multi-plataforma, flexible i potent per la facilitat amb

que es programa i pels resultats que ofereix. És possible trobar moltes llibreries de *Java* de codi lliure.

Entre les diferents implementacions de *Java* trobem *J2EE* (*JEE*, *Java Platform Enterprise Edition*) que serveix per aplicacions distribuïdes. Recordem que l'entorn de desenvolupament utilitzat és l'*Eclipse JEE*.

Les principals característiques de *Java* són:

- S'ha creat per ser un llenguatge senzill.
- Se centra en la manipulació, creació i construcció d'objectes (programació orientada a objectes).
- Permet la construcció d'aplicacions distribuïdes per mitjà d'una col·lecció especifica de classes.
- És un llenguatge robust i fiable, s'ha escrit pensant en poder verificar errors.
- Té pocs problemes de seguretat, característica molt important en les aplicacions distribuïdes per Internet.
- És independent de la plataforma final on s'executarà el programa.
- Els compiladors *Java* han anat millorant les seves prestacions.
- Permet l'execució de múltiples fils d'execució o vàries tasques simultàniament.
- En temps d'execució, l'entorn *Java* es pot estendre mitjançant enllaços a classes que poden estar localitzades en servidors remots o en una xarxa.

5.4. Desenvolupament del procés

5.4.1. Generació del *servlet*

Un *servlet* és un objecte *Java* que és executat per un servidor o contenidor d'aplicacions (*Tomcat*) i que respon a invocacions HTTP, servint pàgines dinàmiques. El contingut generat pot ser un fitxer de qualsevol tipus, la majoria de vegades HTML.

Per crear el nostre *servlet*, des de l'*Eclipse*, li hem assignat el projecte que l'haurà d'incloure (***noise***), hem creat la classe que crearà el gràfics (***GraphicService***), li hem donat un nom al paquet *Java* (***etclusi.noise.graphicsgenerator***), li hem assignat una superclasse (`javax.servlet.http.HttpServlet`) i hem creat el mètode `doGet` i `doPost`.

Una classe és un model o prototip amb els quals es creen objectes (paquet de software) i un paquet és un espai de noms per a organitzar classes i intefícies d'una manera lògica.

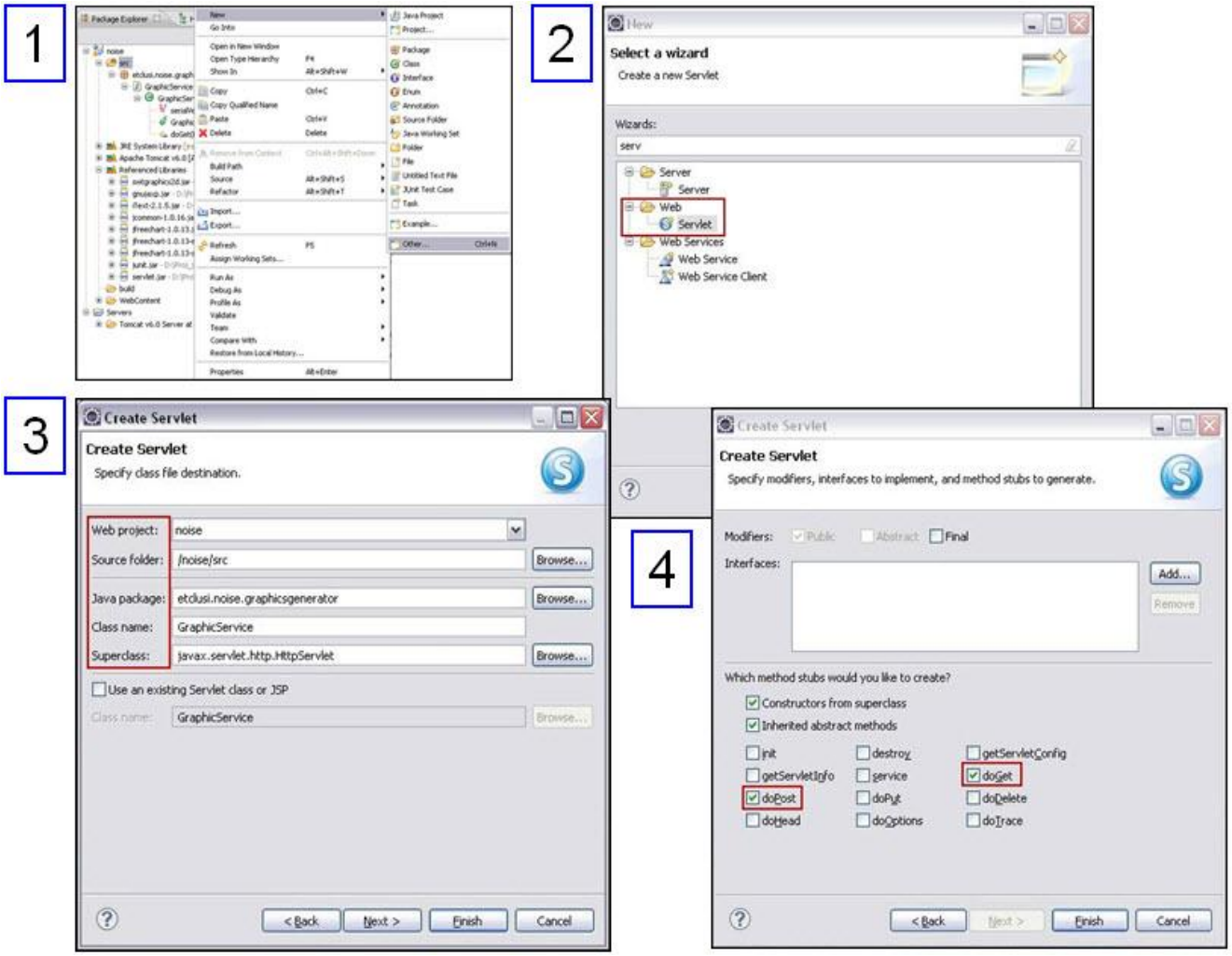


Figura 5.17. Creació del servlet amb l'Eclipse.

El paquet `javax.servlet.http` defineix subclasses específiques HTTP, com ara objectes de gestió, que lliguen les múltiples peticions dels usuaris i les respostes del servidor d'aplicacions. Un *servlet* implementa la interfície `javax.servlet.Servlet` o hereta les classes més convenients per a un protocol específic (p.e. `javax.servlet.HttpServlet`). A l'implementar aquesta interfície, el *servlet* és capaç d'interpretar els objectes de tipus `HttpServletRequest` (petició - `doGet`) i `HttpServletResponse` (resposta - `doPost`) que contenen la informació de la pàgina que ha invocat el *servlet*.

Per al nostre *servlet* hem importat les següents classes:

- `import javax.servlet.ServletException;`
- `import javax.servlet.http.HttpServlet;`
- `import javax.servlet.http.HttpServletRequest;`
- `import javax.servlet.http.HttpServletResponse;`

Els *servlets* poden estar empaquetats dintre d'un fitxer en format WAR (Arxiu d'Aplicació Web - *Web Application Archive*) que es posaran dins del contenidor. És el que hem fet nosaltres un cop hem acabat de programar el *servlet* que generarà els gràfics estadístics (***noise.war***).

El *servlet* funciona de la següent manera:

1. L'usuari fa una petició al servidor via URL.
2. El servidor rep la petició.
3. El *servlet* processa la petició i torna el resultat a l'usuari. Pot processar múltiples peticions de múltiples usuaris a la vegada.

5.4.2. Connexió a la base de dades

Per fer la connexió a la base de dades (***postgres***) ho farem via JDBC (*Java DataBase Connectivity*), que permet l'execució d'operacions sobre bases de dades des de *Java*, independentment del sistema operatiu que s'utilitzi o de la base de dades a la qual es vulgui accedir, utilitzant el llenguatge SQL (Llenguatge de Consulta Estructurat - *Structured Query Language*) del model de la base de dades que s'utilitzi.

Per funcionar necessita un *driver*, que són els programes de control que permeten la comunicació entre aplicacions *Java* i les bases de dades. En el nostre cas s'ha utilitzat: `Class.forName("org.postgresql.Driver");`.

JDBC ofereix el paquet java.sql, el qual ens dóna classes que ens permeten treballar amb bases de dades. Tenim:

- DriverManeger: per carregar un *driver*.
- Connection: per establir connexions amb les bases de dades.
- Statement: per crear consultes SQL que seran enviades a la base de dades.
- ResultSet: per emmagatzemar el resultat de la consulta.

El nostre *servlet* té les següents classes importades:

- `import java.sql.Connection;`
- `import java.sql.DriverManager;`
- `import java.sql.ResultSet;`
- `import java.sql.SQLException;`
- `import java.sql.Statement;`

Per fer la connexió a la nostra base de dades, creem *c* (nom de la connexió), en el qual s'agafa el *driver* que hem declarat abans, indiquem el tipus de connexió (JDBC) i la base de dades (*PostgreSQL*), la URL (en aquest cas *localhost*), i a continuació el nom de la base de dades (*username*: ***postgres***) i el *password* (etclusi):

```
c = DriverManager.getConnection("jdbc:postgresql://localhost/postgres",
"postgres", "etclusi");
```

5.4.3. Definició de consultes

A l'hora de fer les consultes, s'ha hagut de mirar amb detall quines són les peticions que pot fer l'usuari i quin tipus de gràfic obté com a resposta. Això ens permet agrupar el codi el màxim possible. D'aquesta manera dins del codi programat trobem consultes força complexes que engloben diverses consultes.

Per definir les consultes s'ha seguit el següent codi que serà el que ens indicarà el codi de consulta de l'usuari: `private JFreeChart barChart(String tp, String ctry, String aggl, String Code)`. Tenim que `tp` és el tipus de consulta que vol fer (p.e. aeroports dins les aglomeracions), `ctry` el país pel qual vol fer la consulta (s'han de posar les dues lletres que identifiquen el país, p.e. "DE" = Alemanya), `aggl` que fa referència a l'aglomeració o aeroport que es vol consultar (si és "00000" fem referència a Europa, si ens referim a un codi d'aglomeració concret hem de posar "0000+núm. d'aglomeració", p.e. "00004"), i per últim el `Code` on es marca si vol veure la informació de dia o de nit, la banda de decibels i si vol el total o el percentatge. Aquest codi ens marcarà posteriorment els canvis en el codi *JavaScript*.

Codis tp			
Codi	**Consulta**	**Codi**	**Consulta**
AAR	Aeroports dins les aglomeracions	MAR	Aeroports principals fora de les aglomeracions
ARD	Carreteres dins les aglomeracions	MRL	Ferrocarrils principals fora de les aglomeracions
ARL	Ferrocarrils dins les aglomeracions	MRD	Carreteres principals fora de les aglomeracions
AIN	Indústria dins les aglomeracions		

Figura 5.18. Taula dels codis tp.

Codis Code			
Codi	**Consulta**	**Codi**	**Consulta**
LD	Dia	55 o 65 o 75	Bandes de soroll de dia
LN	Nit	50 o 60 o 70	Bandes de soroll de nit
T	Total		
R	Percentatge		

Figura 5.19. Taula dels codis Code.

Per fer les consultes s'ha fet servir molt la sentència condicional, que és una o vàries instruccions que s'executen o no en funció del valor de la condició. Les més utilitzades són: *if/else* i *case* o *switch*. A continuació veiem alguns exemples:

- ```
  if  ((ctry.compareTo("EU") != 0) && (aggl.compareTo("00000")==0)
  && (tp.compareTo("AAR") == 0 || tp.compareTo("ARD") == 0
  ||tp.compareTo("ARL") == 0 ||tp.compareTo("AIN") == 0)) {
  horizontal = true;
      }
      else {
            horizontal = false;
      }
  ```
  ```

```
if (tp.compareTo("AAR") == 0){
table = "tbldf4_agglair";
major=true;
columnR = "numberofinhabitants";
}
else if (tp.compareTo("ARD") == 0){
table = "tbldf4_agglroad";
major=true;
columnR = "numberofinhabitants";
}
else if (tp.compareTo("ARL") == 0){
table = "tbldf4_agglrail";
major=true;
columnR = "numberofinhabitants";
}
else if (tp.compareTo("AIN") == 0 ){
table = "tbldf4_aggind";
major=false;
columnR = "numberofinhabitants";
}
```

```
switch (initValue){
   case 50:
   case 60:
   case 70:
         interval= "5054,5559,6064,6569,70";
         categories="50-54,55-59,60-64,65-69,>70";
         break;
   case 55:
   case 65:
   case 75:
         interval= "5559,6064,6569,7074,75";
```

```
            categories="55-59,60-64,65-69,70-74,>75";
            break;
        }
```

Per fer el codi també s'ha de tenir en compte com s'escriuen els diferents operadors en *Java*, a continuació un exemple dels que més s'han utilitzat:

Operadors Java			
Símbol	**Operació**	**Símbol**	**Operació**
>	Major	==	Igual
<	Menor	!=	diferent
>=	Major o igual	&&	And (lògic)
<=	Menor o igual	||	Or (lògic)

Figura 5.20. Taula d'operadors en Java.

5.4.4. Mètodes

En la programació orientada a objectes un mètode és una subrutina associada a una classe (mètodes de classe) o a un objecte (mètodes d'instància). En el nostre cas, els mètodes fan referència a la classe creada `GraphicService`. Un mètode doncs, consisteix en un sèrie de sentències per dur a terme una acció, amb un paràmetre d'entrada i un valor de sortida; és el que fa la petició a la classe per a que realitzi una tasca determinada.

El nostre codi s'ha dividit en diversos mètodes:

Figura 5.21. Mètodes de la classe GraphicService.

Com es pot veure a la imatge tenim una classe principal que és `GraphicService`. Aquesta conté diversos mètodes:

- Per crear el gràfic horitzontal amb les seves corresponents categories (`createCategoryDatasetHorizontal`).
- Per crear el gràfic vertical amb les seves corresponents categories (`createCategoryDatasetVertical`).
- Per donar tots els elements necessaris per construir el gràfic com el tipus de consulta, el país, l'aglomeració i el codi (`barChart`).
- Per arrodonir els eixos que tenen valors a la centena més propera (`getRoundInterval`).
- Per tornar els paràmetres adequats de la resposta demanada (`getParamEntero`).
- Per fer la petició i tornar-la (`processRequest`).
- Per últim tenim el mètode que torna el resultat (`doGet`).

Tots els mètodes creats són `private`, és a dir, només es pot accedir a ells des de la mateixa classe. Només l'últim és `protected`, al que podem accedir des del mateix paquet, i també des d'una subclasse de la classe en un altre paquet, aquests tipus de mètodes també sobreescriuen la classe superior.

5.4.5. Generació de gràfics

A l'hora de definir els gràfics s'ha hagut de diferenciar entre els gràfics horitzontals i els verticals, depenent de la SQL sol·licitada. Per crear-los utilitzem `JFreeChart chart; chart = ChartFactory.createBarChart`.

Per generar els gràfics amb *JFreeChart* s'han hagut d'importar les següents classes:

- `import java.awt.BasicStroke;`
- `import java.awt.Color;`
- `import java.awt.Font;`
- `import java.awt.GradientPaint;`
- `import org.jfree.chart.ChartFactory;`
- `import org.jfree.chart.ChartUtilities;`
- `import org.jfree.chart.JFreeChart;`
- `import org.jfree.chart.axis.AxisLocation;`
- `import org.jfree.chart.axis.CategoryAxis;`

- `import org.jfree.chart.axis.NumberAxis;`
- `import org.jfree.chart.axis.NumberTickUnit;`
- `import org.jfree.chart.plot.CategoryMarker;`
- `import org.jfree.chart.plot.CategoryPlot;`
- `import org.jfree.chart.plot.PlotOrientation;`
- `import org.jfree.chart.renderer.category.BarRenderer;`
- `import org.jfree.chart.title.TextTitle;`
- `import org.jfree.data.category.CategoryDataset;`
- `import org.jfree.data.category.DefaultCategoryDataset;`
- `import org.jfree.ui.Layer;`

Pel que fa als gràfics verticals s'ha hagut de definir les diferents categories de cada un d'ells (`CategoryDataset`), depenent de si els gràfics són el resultat de la petició de les dades de soroll de dia o de nit. Per marcar quin interval s'està visualitzant s'ha sobreposat una ombra en gris (`CategoryMarker, plot.addDomainMarker`). Aquests gràfics es poden obtenir per les dades totals o en percentatges. També s'ha hagut de diferenciar quins gràfics tenen una barra o dues per poder fer les comparatives corresponents.

Pel que respecta als gràfics horitzontals, aquests treuen el nom de l'aglomeració en un dels eixos i els valors (totals o en percentatge) en l'altre. En aquest cas han desaparegut les categories i l'ombra superposada ja que les dades que es visualitzen són per la banda de decibels demanada, fent la suma de les categories superiors.

També s'ha hagut de contemplar fer *querys* per fer els títols, subtítols, l'eix x i l'eix y variables, doncs, cada un d'aquests elements varien segons les dades demanades.

A l'hora de crear els gràfics es pot definir el color de les barres (`GradientPaint`), el contorn de les barres (`BarRenderer`), l'àrea del gràfic (`CategoryPlot`), el color de fons del gràfic (`chart.setBackgroundPaint`), la font (`Font`) de títols i subtítols, la llegenda visible o no visible (`true`/`false`) i la seva posició, els eixos (`NumberAxis`), el tipus d'imatge que volem tornar (`response.setContentType("image/png");`), etc.

Finalment per tornar el gràfic segons la petició de l'usuari utilitzem: `return chart;`.

6. Implementació

6.1. Implementació del *servlet* al *Tomcat*

Un cop s'ha acabat de programar el *servlet* s'ha d'exportar en un arxiu WAR (Arxiu d'Aplicació Web) per poder ficar-lo dins del *Tomcat* (es pot donar directament la ruta on volem que ho guardi en *Destination*: en el nostre cas el guardem dins de la carpeta *webapps* de *Tomcat*).

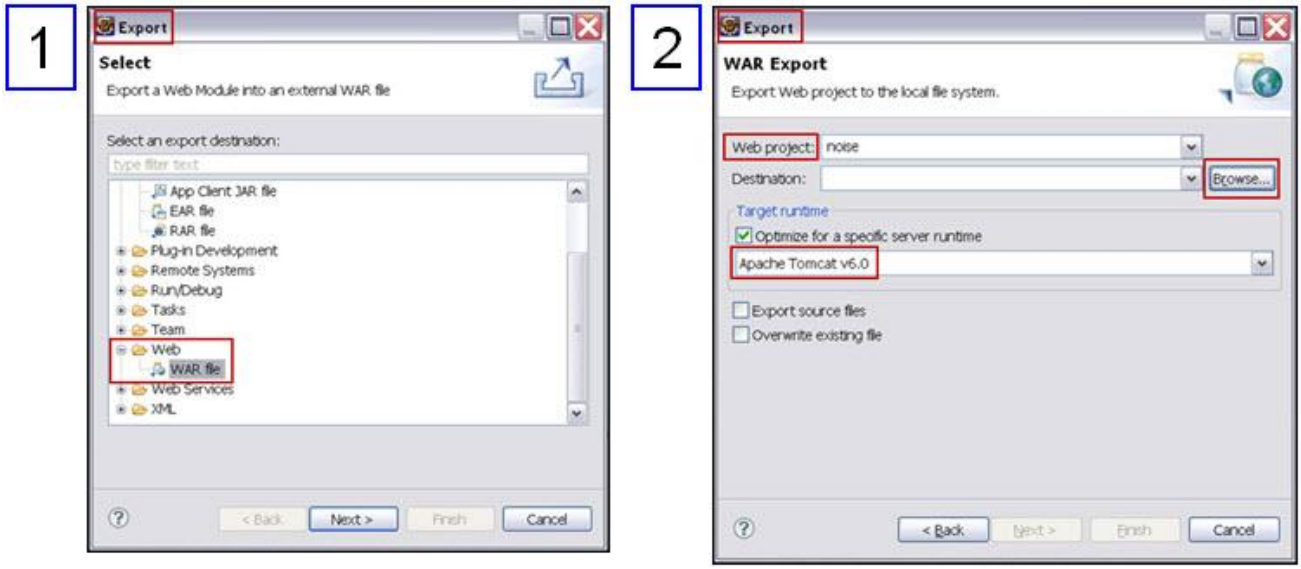


Figura 6.1. Exportació de noise a arxiu WAR.

Tot seguit llegim l'arxiu *Running.txt* que es troba dins de *Tomcat* i veiem que hem d'establir una variable d'entorn anomenada JRE_HOME.

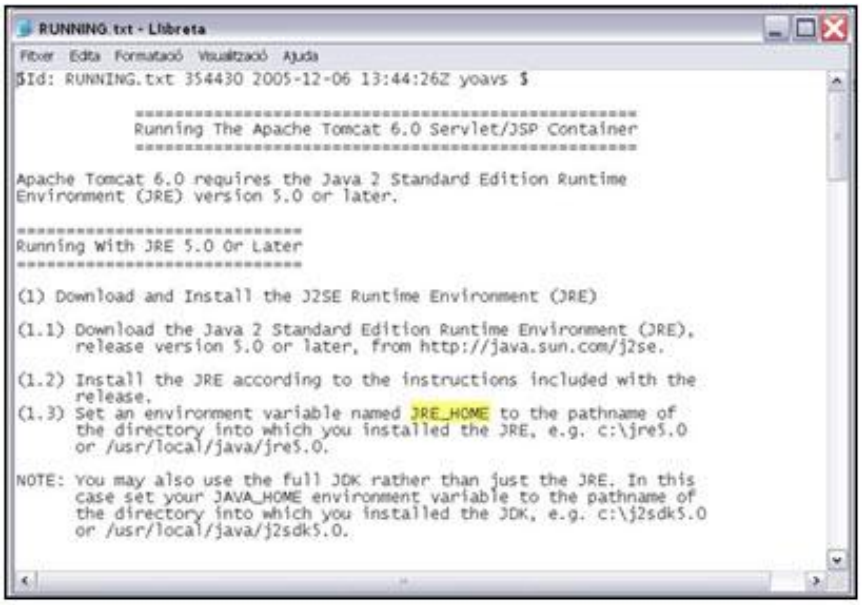


Figura 6.2. Running.txt de Tomcat.

Per fer això hem d'anar a les propietats del sistema, variables d'entorn i crear o modificar la variable donant-li la ruta del directori corresponent on tenim instal·lat el JRE (Java/jre6).

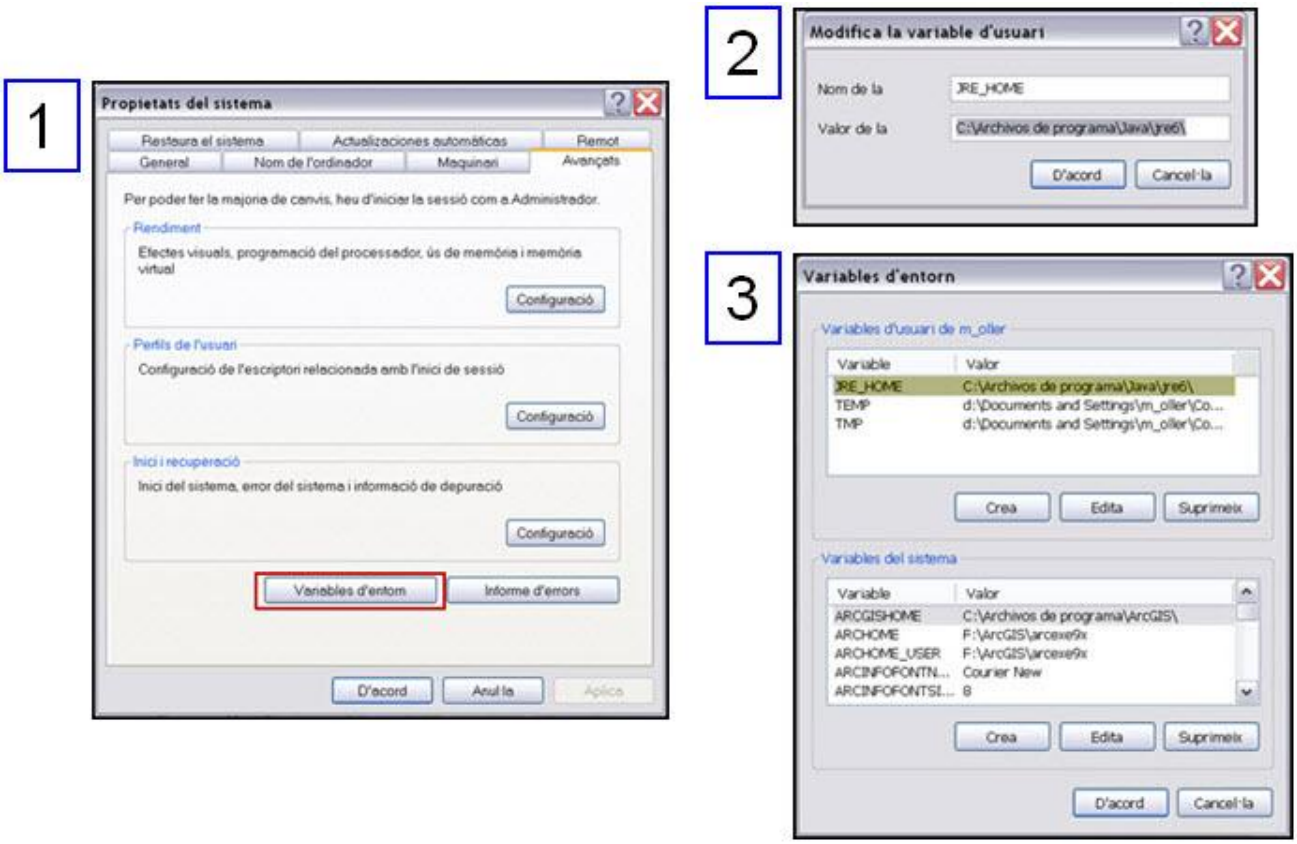


Figura 6.3. Aplicar variable d'entorn JRE_HOME.

També hem de mirar la configuració de *Tomcat* que es troba dins de l'arxiu *server.xml*, i allà hem de canviar el port a 8081 ja que per defecte ve amb el port 8080, però aquest és el que està utilitzant l'*Apache*.

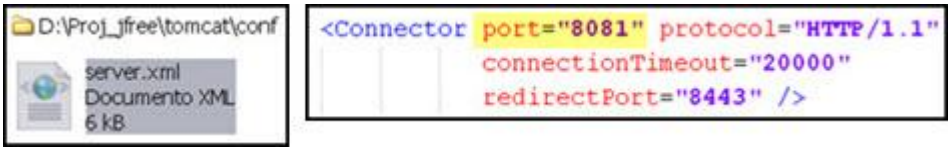


Figura 6.4. Modificació del port de Tomcat dins de server.xml.

Després de fer tot això ja podem iniciar *Tomcat* amb *startup.bat*, aquest detecta l'arxiu ***noise.war*** i el descomprimeix, tot seguit veiem la carpeta *noise* ja descomprimida que conté les carpetes META-INF i WEB-INF (dins d'aquesta trobem el ***graphicsgenerator***) que també veiem a l'*Eclipse* i és on es va guardant automàticament la classe que generarà els gràfics estadístics. Ara el *servlet* ja funciona correctament.

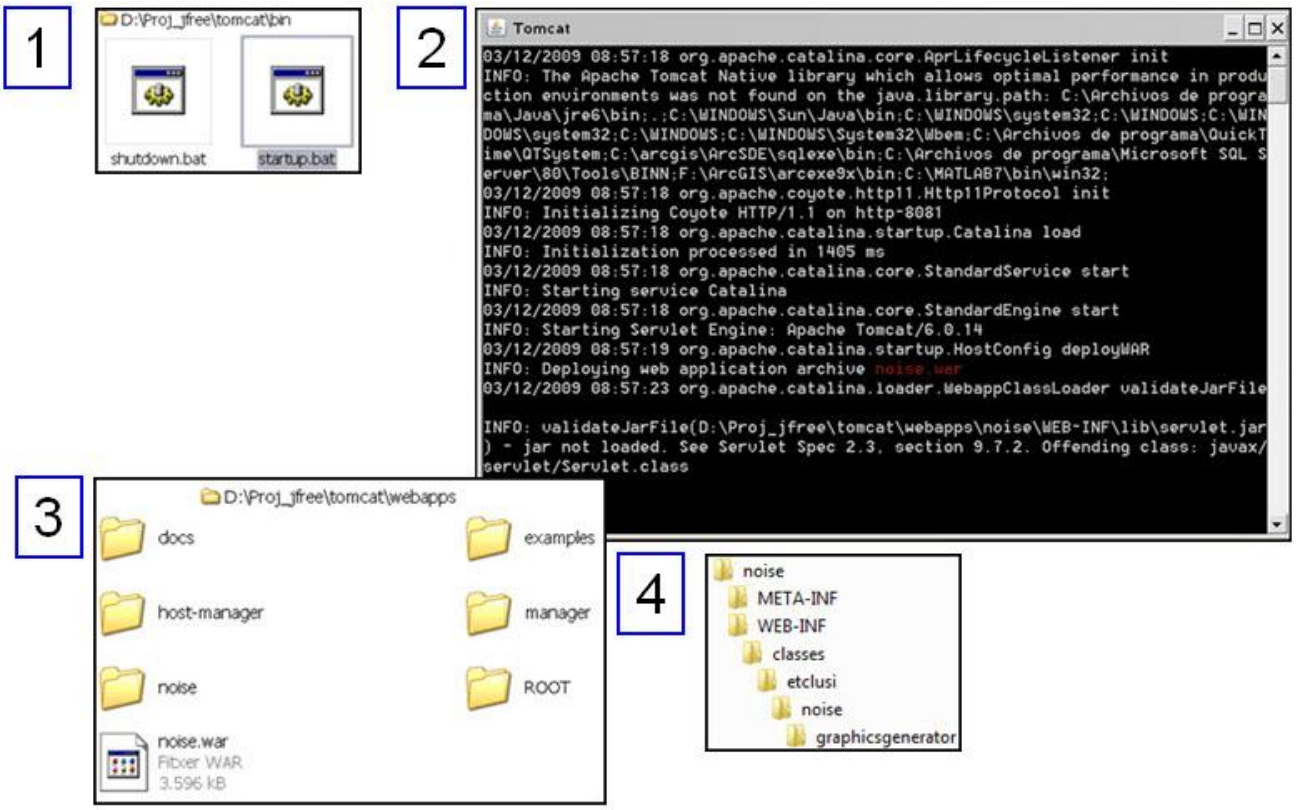


Figura 6.5. Tomcat i noise.war.

6.2. Implementació del *servlet* al *Mapfish*: *JavaScript*

JavaScript és un llenguatge *script* (que controla aplicacions) que s'utilitza per accedir a objectes en aplicacions. Principalment s'utilitza integrat en navegadors web permetent el desenvolupament d'interfícies d'usuari millorades i pàgines web dinàmiques. *JavaScript* ha rebut la influència de molts llenguatges i té una sintaxi similar a la de *Java* però molt més senzilla, cal especificar que tot i tenir un nom similar no deriva del llenguatge de programació *Java*.

Per últim, només queda canviar el codi *JavaScript* que està generant les imatges amb el programa *Dreamweaver*. També es podrà esborrar la carpeta que es troba dins de ***web_noise***

(*graphs*), que és la que conté les onze mil imatges generades manualment, ja que ara és el *servlet* qui les genera.

El codi que hem de canviar es troba en els arxius compilats de ***web_noise*** (que es troba dins la carpeta *htdocs* de l'*Apache*) concretament hem de canviar el codi de l'arxiu ***noisescript_comp.js*** que és el que controla els *combos* de país i d'aglomeració, i el ***LayerTree_comp.js*** que és el que controla els canvis que es fan a la taula de continguts.

Figura 6.6. Arxius compilats de web_noise.

El codi antic per obtenir les imatges dels gràfics era:

```
var graphsrc='<img src="http://158.109.174.111:8080/web_noise/graphs/';
```

En el codi nou ho fem canviant la variable *graphsrc* per la nova variable ***graphservlet*** i, a més a més, afegint les variables que formaran el codi de les imatges que s'han de generar:

```
var graphservlet='<img src="http://158.109.49.37:8081/noise/GraphicService?image';

var tp= "&tp="'; var ctry= "&ctry="'; var agglcd= "&aggl="'; var code2= "&Code="';
```

En aquest cas es posa la IP de l'ordinador corresponent i no l*ocalhost,* per poder fer proves des d'una altra màquina, i el port des del qual treballa el *Tomcat* (8081).

D'aquesta manera podem canviar la part de codi que ens interessa, tenint en compte les variables que controlen els *combos* de país i aglomeració i els seus valors, i tenint en compte les noves variables. Tenim per exemple el següent codi:

```
document.getElementById("graphs").innerHTML=graphservlet+tp+b[0]+ctry+ct_value+agglcd+aggl+code2+b[1]+'"
width="100%".
```

Variables codi JavaScript	
Variable	**Combo**
combo_ct	ctry
combo	aggl
ct_value	ctry
ct_val	aggl

Figura 6.7. Variables del codi JavaScript, combo i valor.

Primer ens connectem amb el nostre ***graphicservlet*** i després comencem a donar la resta de variables segons la petició feta. En aquest cas, l'exemple correspon a qualsevol petició que no sigui MRD o MRL (carreteres o ferrocarrils principals fora de les aglomeracions) i on el *combo* de país (*combo_ct.getvalue*) i d'aglomeració (*combo.getvalue*) estan activats (*true*). La "b" correspon a una variable creada per obtenir el *layer* corresponent.

Cada tipus de petició tindrà unes variables determinades. Aquestes peticions també es creen amb la sentència *else/if* que ja hem vist anteriorment.

7. Proves

Les proves del *servlet* s'han anat realitzant durant tot el procés per veure si els canvis que s'anaven fent en el codi (*Java*, *Eclipse*) ens donaven els resultats que volíem obtenir. Per això es canviava la ruta del ***GraphicService*** des del *FireFox* i podíem anar veient els gràfics resultants de la nostra petició.

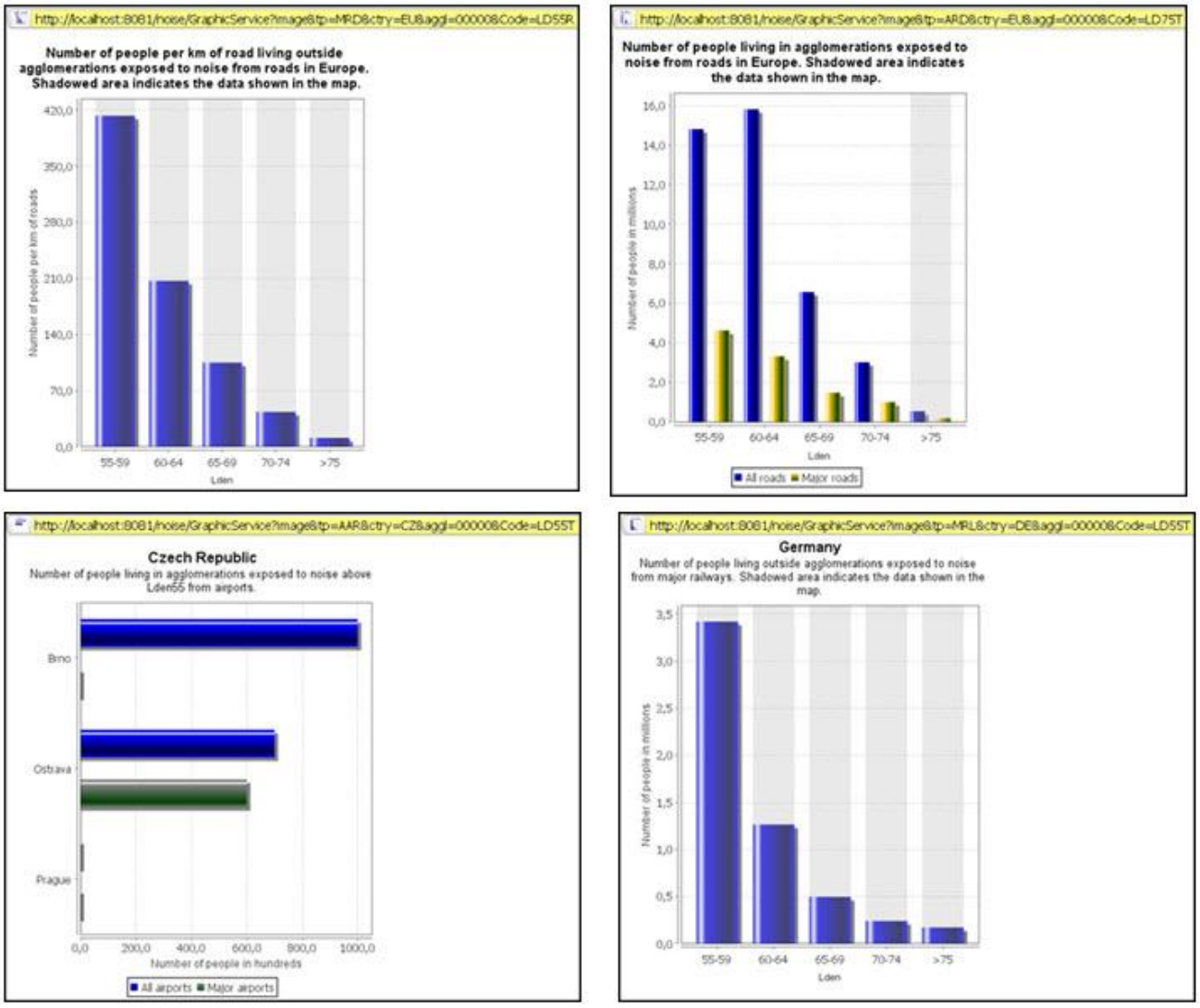


Figura 7.1. Exemples dels gràfics estadístics durant les proves. FireFox.

Per veure el codi amb més detall, i sobretot per veure els errors, s'entra en el mode *debug* des de l'*Eclipse*, que ens dóna tots els detalls de cada un dels processos que es realitzen en tot moment. Des d'aquí podem establir i veure els *breakpoint* (allà on volem que s'aturi el procés per veure si funciona correctament o no), les diferents *variables* i les *expressions*.

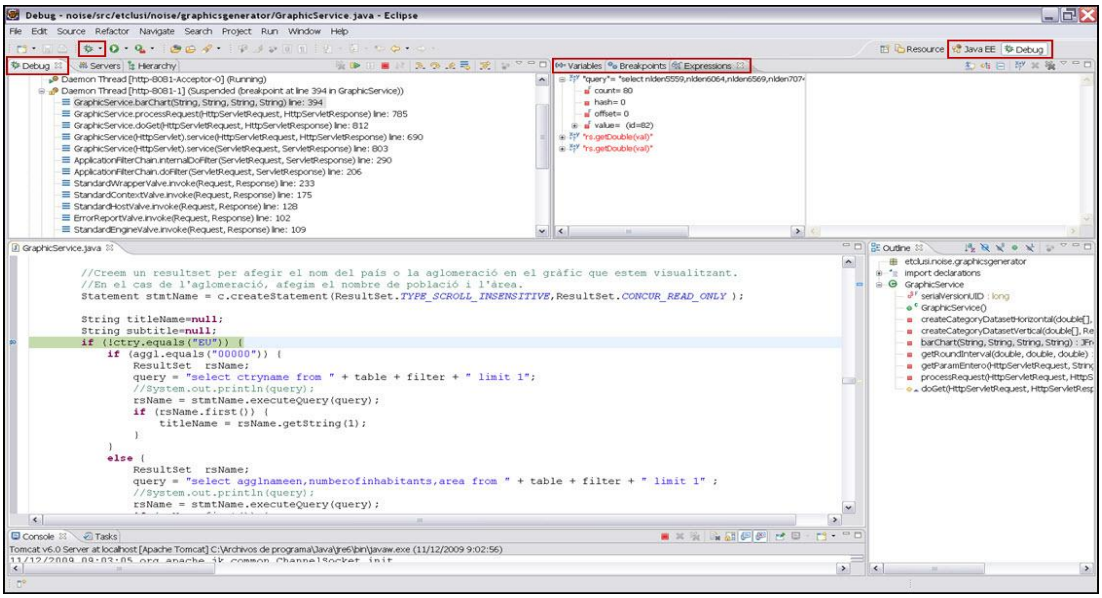


Figura 7.2. Mode debug de l'Eclipse.

Per fer les proves del codi *JavaScript* s'obre el navegador *FireFox* i s'accedeix a ***web_noise*** des del *localhost* (nom reservat per totes les computadores per referir-se a si mateixa) o des de la IP del nostre ordinador (en aquest cas és necessari per poder accedir-hi des d'una altra màquina: http://158.109.49.37/web_noise/index.html). Com estem en un medi de desenvolupament i no de producció, les proves s'han realitzat contra la pròpia màquina i des d'una altra màquina de l'ETC-LUSI (per fer això s'ha hagut de desactivar el *Firewall*).

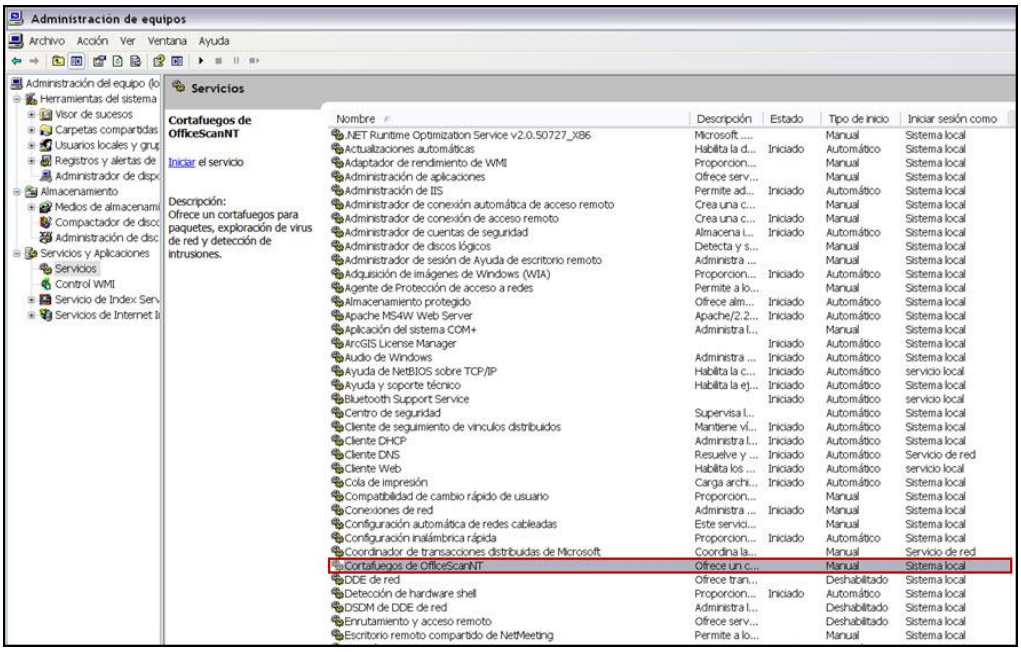
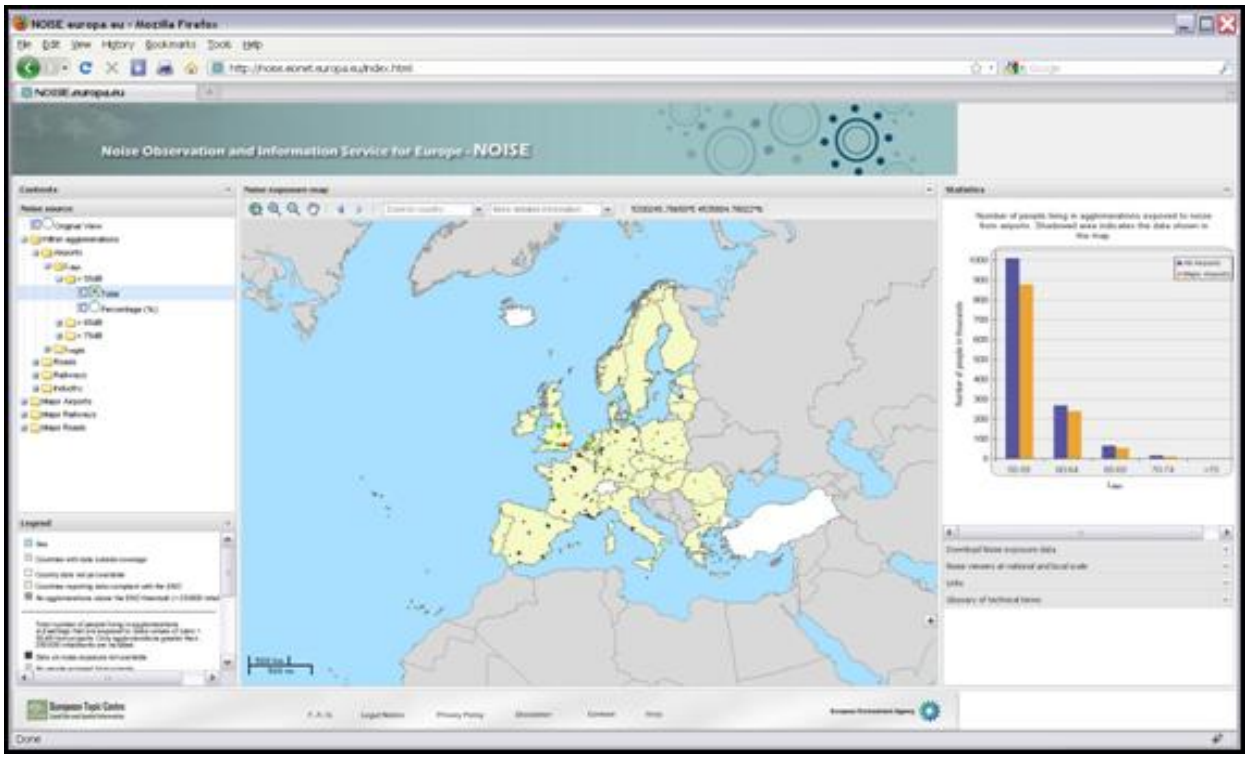


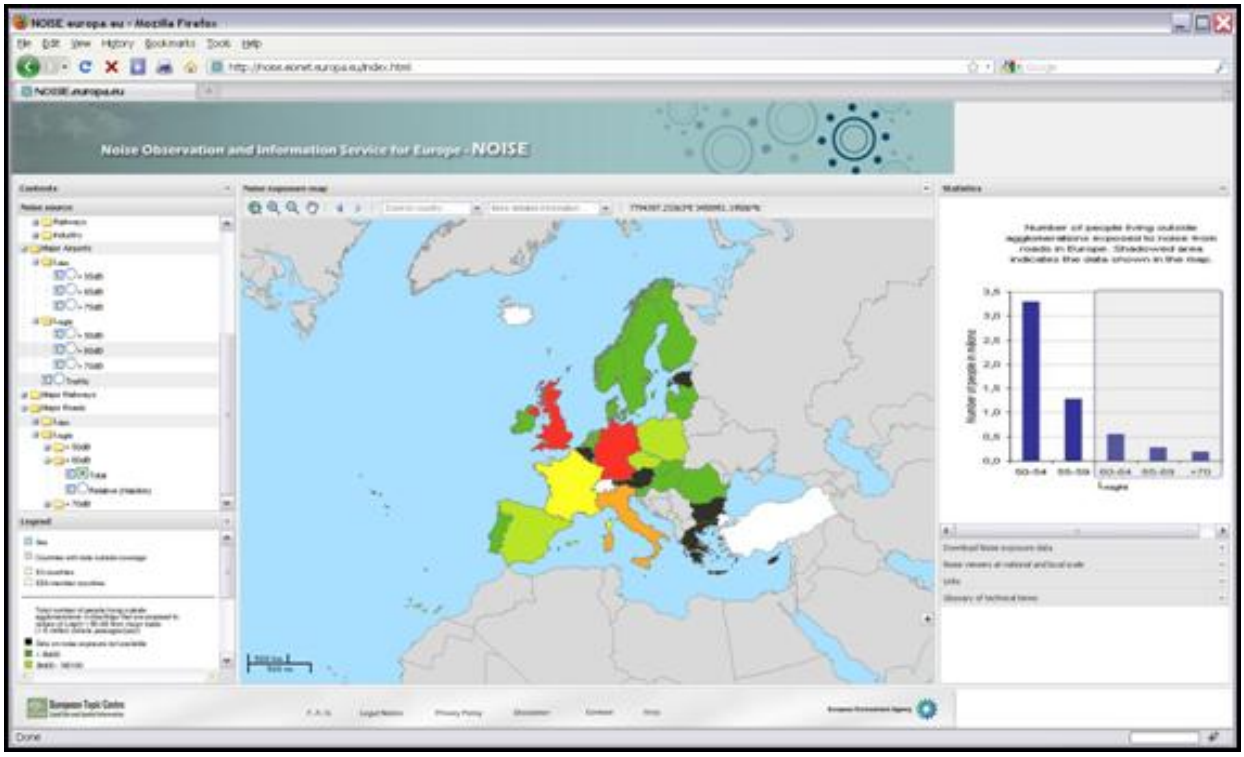
Figura 7.3. Serveis. Firewall.

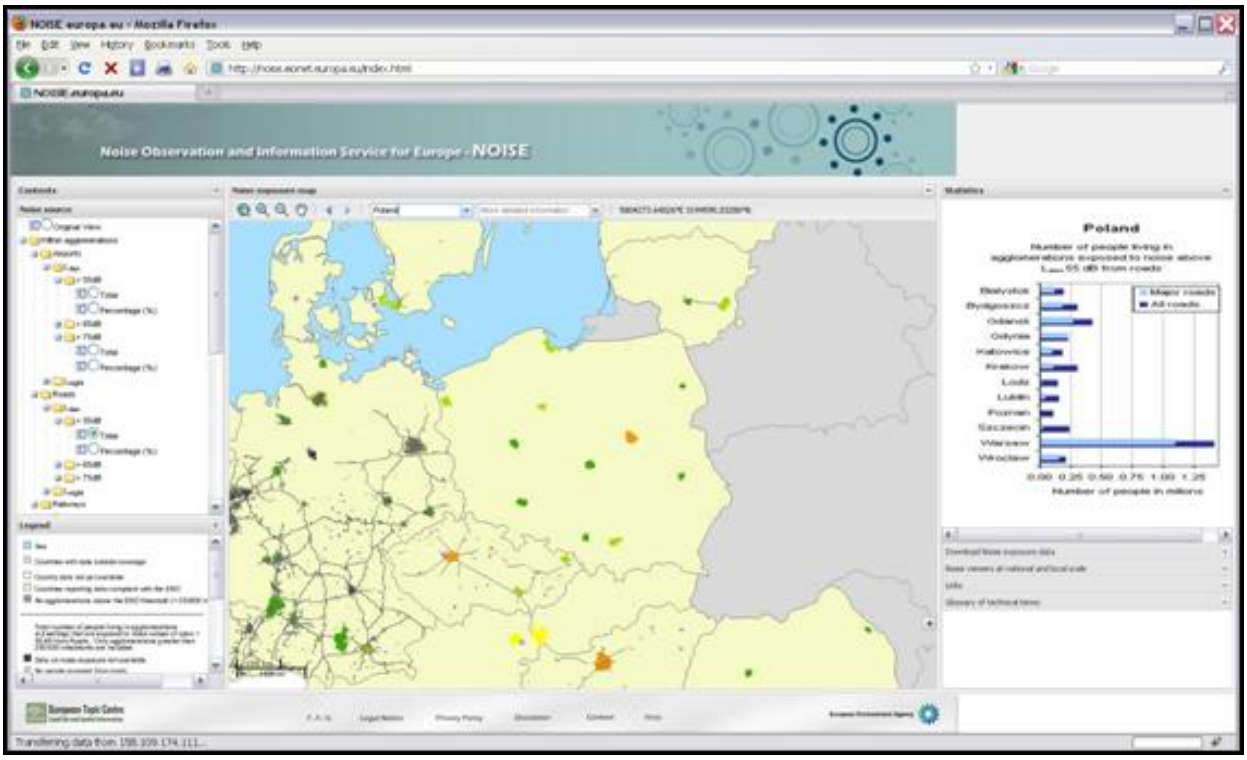
Amb tot això s'ha comprovat que el *servlet* funciona correctament des de la pàgina web i dóna els resultats esperats.

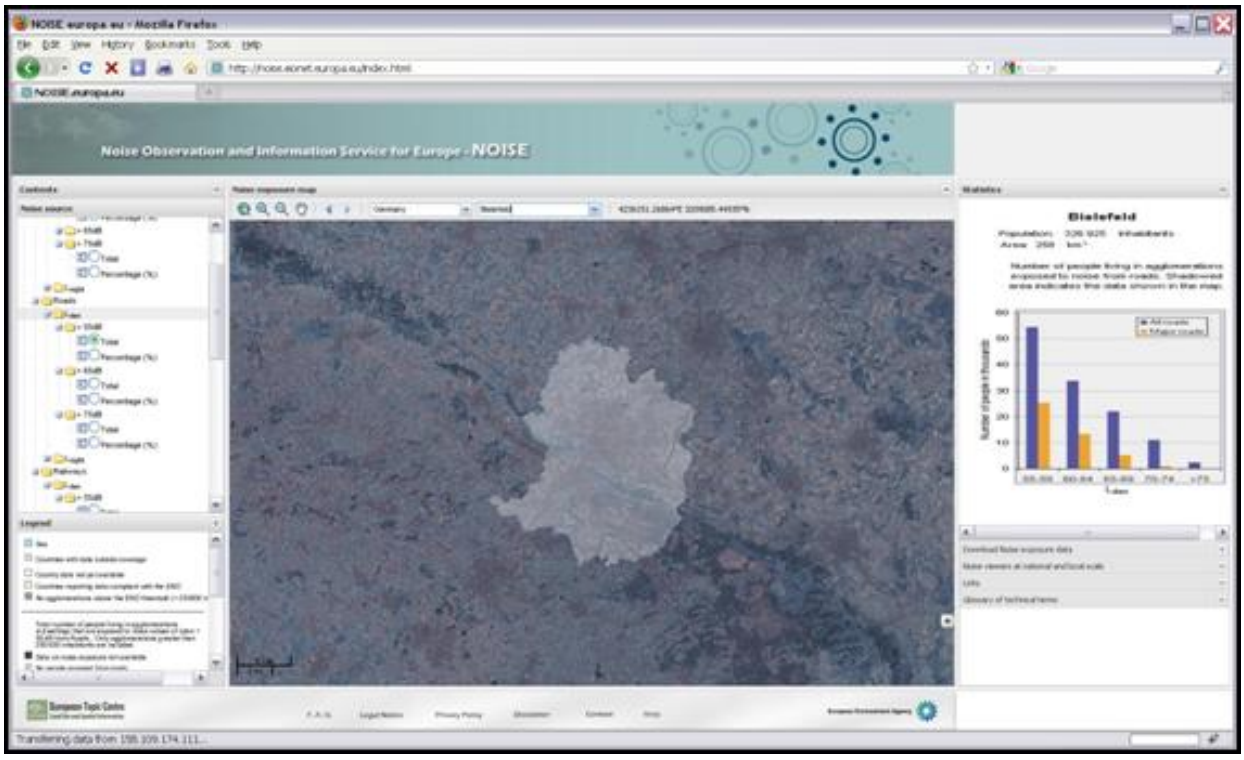
8. Resultats

Quan els gràfics apareixien en format JPEG en el visor *Noise Map Viewer* (http://noise.eionet.europa.eu/index.html) tenien el següent aspecte:

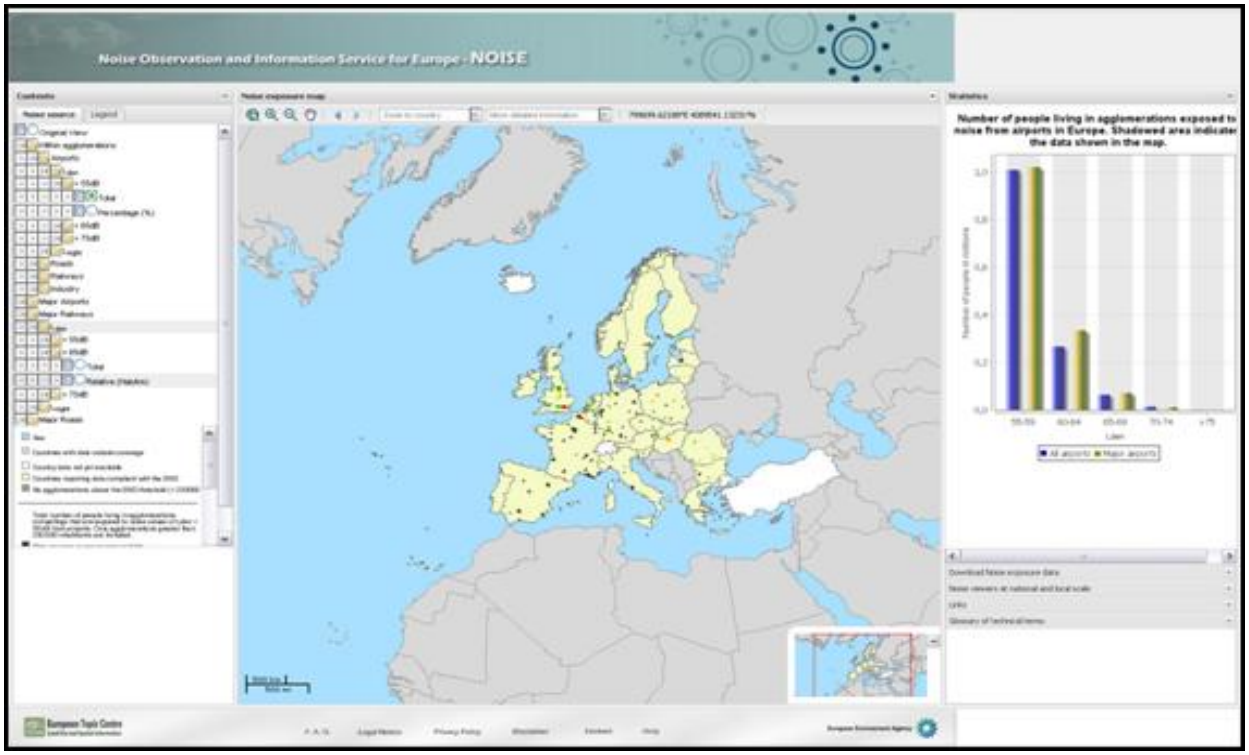


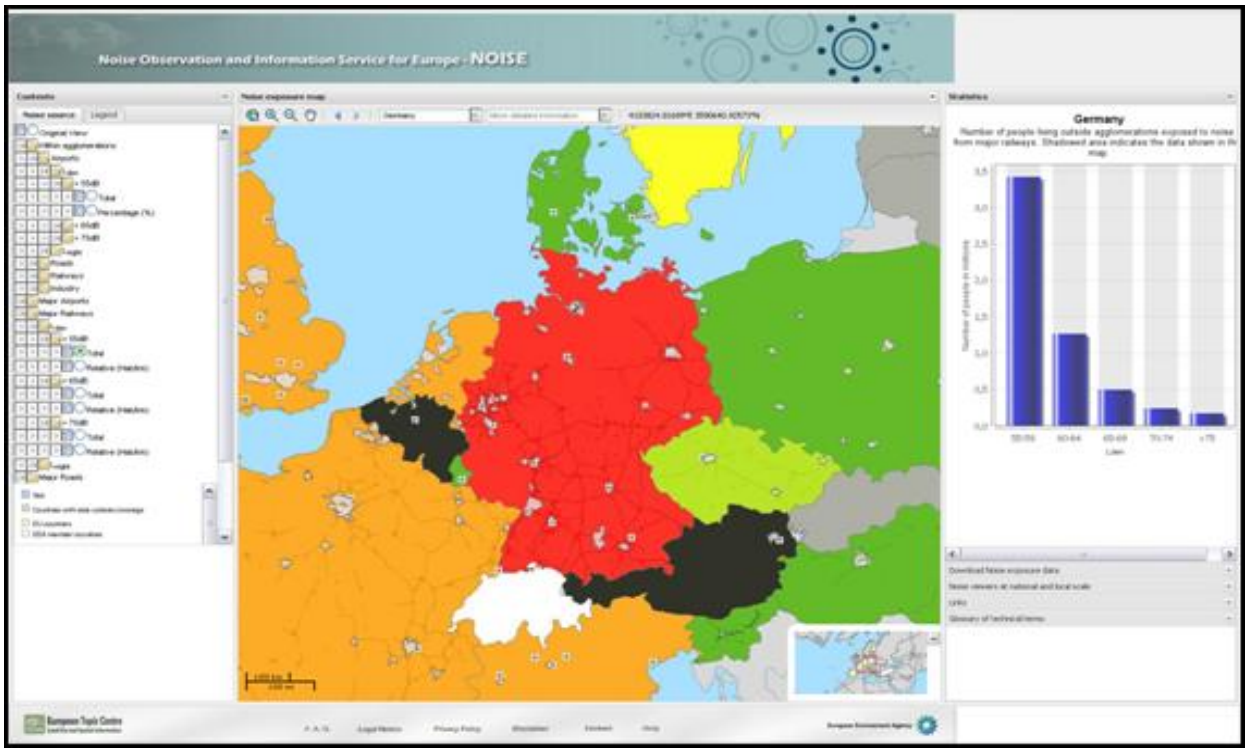



Noise Observation and Information Service for Europe - NOISE
Poland

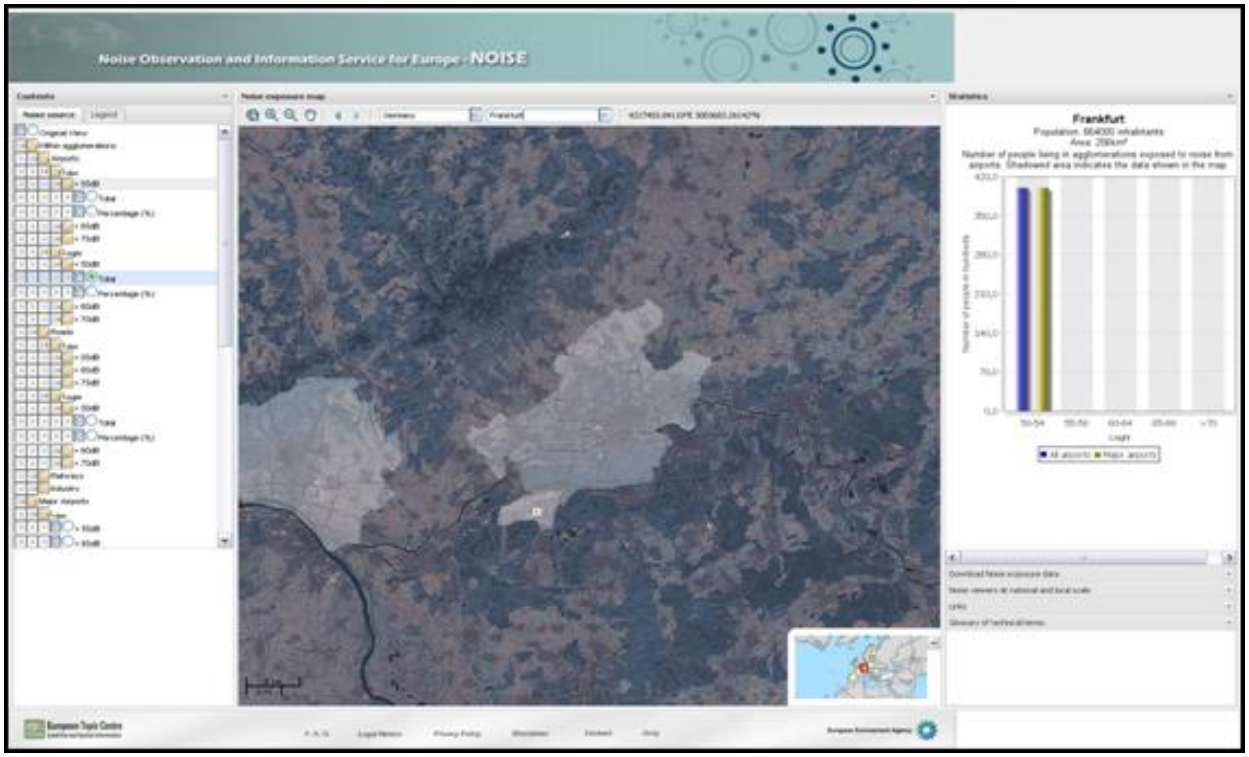

Noise Observation and Information Service for Europe - NOISE
Bielefeld

Amb el *servlet* (http://158.109.49.37/web_noise/index.html) funcionant apareixen així:






Noise Observation and Information Service for Europe - NOISE
Poland


Noise Observation and Information Service for Europe - NOISE
Frankfurt

9. Conclusions

El projecte a nivell personal m'ha permès posar en pràctica aquells coneixements que havia après durant el màster i m'ha donat la possibilitat de seguir-los ampliant. També suposa l'oportunitat d'aprendre com es realitza un projecte a nivell professional.

Al principi va ser difícil haver d'entrar a formar part d'un projecte ja creat, en el qual s'hi treballava des de ja feia temps. Ha suposat un esforç posar-se al dia de tot el que s'havia fet fins aleshores, tant pel que fa la part del visor com pel que fa les dades i la seva estructura.

El projecte consta en gran part d'una recopilació documental de:

- Projecte *Noise Map Viewer*: funcionament del visor, estructura de dades, etc.
- Metodologia a seguir.
- Tecnologies utilitzades fins el moment: *MapServer*, *MapFish*, *Apache*, etc.
- Tecnologies que s'haurien d'utilitzar a partir de l'inici del projecte: *JFreeChart*, *Tomcat*, *Eclipse*, *Spoon*, etc.
- Llenguatges de programació: *Java* i *JavaScript*.

La metodologia utilitzada ha estat d'allò més útil per donar una estructura al projecte, tot i que s'ha adaptat a aquest de la forma més adequada. També ha estat important saber exactament quins eren els objectius de l'ETC-LUSI pel que respecta el mòdul que havia de generar els gràfics.

La fase d'anàlisi ha estat clau per entendre el funcionament del visor i ha marcat els passos que s'haurien de seguir posteriorment, ja que s'han definit les tecnologies que s'estaven utilitzant, els requisits funcionals i tecnològics, i s'han definit les interfícies d'usuari que serveixen per entendre tant el comportament intern del sistema com la resposta que espera l'usuari en interactuar amb ell.

El disseny del software, on hi trobem els casos d'ús, ha estat força complex de realitzar ja que calia tenir molt clar quines eren totes les peticions que pot fer un usuari a la web, quin tipus de gràfic ha d'obtenir, i els múltiples casos i les múltiples excepcions existents.

La fase de desenvolupament del procés que ha de generar els gràfics ha estat la més llarga. En aquest punt és on s'han acabat de prendre vàries decisions que marquen el quefer del projecte. S'ha hagut de tenir molt clar quin era el funcionament i paper de cada un dels diferents servidors que s'utilitzen, també ha calgut decidir quina base de dades era la més adequada per a un funcionament ràpid, sòlid i amb suport per a un gran nombre de dades dins d'un servidor. És aquí

on s'ha programat el procés des de l'*Eclipse*, el qual ens ofereix una gran ajuda i una consolidada API que en molts casos ens ha facilitat la complicada tasca de programar. Cal destacar d'aquesta IDE el mode *debug* tant important per detectar els errors comesos en la creació del *servlet*.

Per últim la implementació del *servlet* ha estat la part més ràpida i senzilla. És aquí que s'ha vist la importància de treballar des d'un bon principi amb els mateixos codis d'imatge, que ja estaven predefinits dins del codi *JavaScript*, i en el qual s'ha basat la programació del nou procés que generarà els gràfics.

El projecte s'ha realitzat amb èxit en el temps establert. Això suposarà un gran estalvi de temps i recursos a l'hora de crear els gràfics que a l'acabar el present projecte es generen automàticament. També suposarà una manera molt més ràpida de poder actualitzar les dades a la web. A falta d'algunes millores degut a la complexitat i gran volum de dades que es manipulen, i del poc temps de duració del projecte, i a falta d'algunes decisions de l'ETC-LUSI alienes al present projecte, en breu es podran veure els resultats d'aquest accedint a la pàgina web del visor.

10. Glossari

- ***Apache**: Apache HTTP server* és un servidor HTTP (de pàgines web) de codi obert multi-plataforma.
- **API**: una Interfície de Programació d'Aplicacions (*Application Programming Interface*) és el conjunt de funcions i procediments (o mètodes per la programació orientada a objectes) que dóna una llibreria per ser utilitzada per un altre software.
- **Classe**: són declaracions o abstraccions d'objectes, és la definició d'un objecte. Al programar un objecte es defineixen les seves característiques i funcionalitats, estem doncs, programant una classe.
- ***Eclipse**:* és un entorn de desenvolupament integrat (*Integrated Development Environment*) de codi obert, multi-plataforma per desenvolupar Aplicacions de Client Enriquit (*Rich Client Platform*).
- ***Framework**:* és un entorn de treball. Pot incloure programari de suport, llibreries, llenguatges de programació i programari extra que ens ajudi a desenvolupar i integrar diversos components d'un projecte de programari.
- ***Geodatabase**:* una base de dades geogràfica o base de dades espacial és una base de dades dissenyada per emmagatzemar, consultar i manipular informació geogràfica i dades espacials.
- **IDE** (*Integrated Development Environment*): un entorn de desenvolupament integrat és un programa compost per un conjunt d'eines (un editor de codi, un compilador, un depurador i un constructor d'interfícies gràfiques) que ajuden a la tasca de programar. Pot funcionar amb diversos llenguatges de programació.
- **Interfície d'usuari**: és el medi amb que un usuari pot comunicar-se amb una màquina.
- **Interfície gràfica d'usuari** (*Graphics User Interfaces* - GUI): permeten a l'usuari comunicar-se amb l'ordinador d'una manera ràpida, senzilla i intuïtiva representant gràficament els elements de control.
- **Interoperabilitat**: condició mitjançant la qual sistemes heterogenis poden intercanviar processos i dades.
- **ISS** (*Internet Information Services*): consta d'una sèrie de serveis per als ordinadors que funcionen amb Windows. Va ser integrat en sistemes operatius de Microsoft destinats a oferir serveis com p.e. Windows Server 2003. Aquest servei converteix un ordinador en un servidor d'Internet per poder publicar pàgines web.
- ***Java**:* és un llenguatge de programació orientat a objectes desenvolupat per *Sun Microsystems*.

- ***Java EE*** (*Java Platform Enterprise Edition*): anteriorment conegut com *Java 2 Platform Enterprise Edition* o *J2EE*. És una plataforma de programació per desenvolupar i executar software d'aplicacions en llenguatge de programació *Java*.
- **Javadoc**: és una utilitat de *Sun Microsystems* per la generació de documentació d'API's en format HTML a partir del codi font *Java*. La majoria d'IDE's els generen automàticament.
- ***JavaScript****:* és un llenguatge *script* (que controla aplicacions) que s'utilitza per accedir a objectes en aplicacions. Principalment s'utilitza integrat en navegadors web permetent el desenvolupament d'interfícies d'usuari millorades i pàgines web dinàmiques
- ***JCommon****:* és una llibreria de classe *Java* que s'utilitza amb *JFreeChar*t.
- **JDBC** (*Java DataBase Connectivity*): és una API que permet l'execució d'operacions sobre bases de dades des de *Java*, independentment del sistema operatiu que s'utilitzi o de la base de dades a la qual es vulgui accedir, utilitzant el llenguatge SQL (Llenguatge de Consulta Estructurat - *Structured Query Language*) del model de la base de dades que s'utilitzi.
- ***JFreeChart****:* llibreria totalment lliure que facilita als programadors mostrar gràfics de qualitat professional en les seves aplicacions.
- **Llibreria**: conjunt de subprogrames utilitzats per desenvolupar software. Contenen codi i dades que proporcionen serveis a programes independents i passen a formar part d'ells. Això permet que el codi i les dades es comparteixin i puguin modificar-se de forma modular.
- ***MapFish****:* és un web 2.0 *mapping application framework*, compost per dues parts: *MapFish* client i *MapFish server*. El *MapFish* client és un entorn *JavaScript*, basat en OpenLayers per la part de la cartografia, i ExtJS i GeoExt per la part GUI (*Graphical User Interface*). En quant a servidor, és tractat com a tal i es composa de diferents mòduls que poden utilitzar diferents llenguatges. Pot ser utilitzat amb *MapServer* o qualsevol servidor cartogràfic que sigui capaç de comunicar-se amb protocols oberts com WMS o WFS.
- ***MapServer****:* és un entorn de desenvolupament de codi obert per la creació d'aplicacions SIG a Internet amb la finalitat de visualitzar, consultar i analitzar informació geogràfica a través de la xarxa.
- **Mètrica**: és una Metodologia de Planificació, Desenvolupament i Manteniment de Sistemes d'Informació, promoguda pel Ministeri d'Administracions Públiques del Govern Espanyol.
- **Multi-plataforma**: terme que s'utilitza per referir-se a programes, sistemes operatius, llenguatges de programació, o altre classe de software, que pugui funcionar en diverses plataformes.
- **Objecte**: en la programació orientada a objectes, un objecte és la unitat que en temps d'execució realitza les tasques d'un programa. Es defineix també com la instància d'una classe.

- **ODBC** (*Open DataBase Connectivity*): és un estàndard d'accés a bases de dades, el seu objectiu és fer possible accedir a qualsevol dada, d'una determinada base de dades, des de qualsevol aplicació.
- ***Open Source**:* de l'anglès, significa codi obert. És el terme amb que es coneix el software distribuït i desenvolupat lliurement.
- **Paquet**: un paquet en *Java* és un contenidor de classes que permet agrupar les diferents parts d'un programa.
- ***pgAdminIII**:* interfície gràfica del *PostgreSQL*, que serveix com a eina d'administració de les bases de dades incloses dins d'aquest. Permet, entre d'altres, fer consultes SQL per al desenvolupament de bases de dades complexes.
- ***Plug*-in***:* significa complement. És una aplicació que es relaciona amb una altra per aportar-li una funció nova i específica. Aquesta aplicació addicional s'executa per l'aplicació principal i interactua a través de l'API.
- ***PostgreSQL**:* és un sistema de gestió de base de dades relacional orientada a objectes de software lliure.
- **Programació orientada a objectes**: és un tipus de programació que utilitza objectes i les seves interaccions per dissenyar aplicacions i programes. Actualment són molts els llenguatges de programació que suporten la orientació a objectes.
- ***Servlet**:* són objectes *Java* que es troben dins d'un contenidor de *servlets* (p.e. *Tomcat*) o servidor d'aplicacions, que responen a peticions HTTP servint pàgines dinàmiques. És capaç de rebre una petició i tornar una resposta a aquesta.
- **SIG** (Sistema d'Informació Geogràfica): és un sistema informàtic capaç d'integrar, emmagatzemar, editar, analitzar, compartir i mostrar informació amb referències geogràfiques.
- ***Spoon**:* és una interfície gràfica d'usuari que permet dissenyar les transformacions i feines que es poden executar amb les eines de *Kettle* (programa d'integració de dades que ajuda a l'extracció, transformació, transport i càrrega de dades).
- **SQL** (*Structured Query Language*): el llenguatge de consulta estructurat és un llenguatge que ens dóna accés a bases de dades relacionals, per tal de recuperar la informació que contenen o modificar-la.
- ***Tomcat**:* servidor d'aplicacions que funciona com un contenidor de *servlets*.
- **WMS** (*Web Map Service*): servei definit per l'OGC (*Open Geospatial Consortium*) que produeix mapes de dades referenciades de forma dinàmica a partir d'informació geogràfica.

11. Bibliografia

API's (Consulta octubre de 2009):

- *JFreeChart*: http://www.jfree.org/jfreechart/api/gjdoc/index.html
- *Sun Microsystems:* http://java.sun.com/j2se/1.5.0/docs/api/

Bases de dades (Consulta octubre de 2009):

- *Pentaho*: http://www.pentaho.com/products/data_integration/
- *Pentaho-Kettle*: http://kettle.pentaho.org/
- *Pentaho-Spoon*: http://wiki.pentaho.com/display/EAI/.01+Introduction+to+Spoon#.01IntroductiontoSpoon-InstallingSpoon
- *pgAdminIII*: http://www.pgadmin.org/
- *PostgreSQL*: http://www.postgresql.org/

Enciclopèdies (Consulta setembre de 2009):

- http://es.wikipedia.org/wiki/Wikipedia:Portada

ETC-LUSI (Consulta setembre de 2009):

- http://etc-lusi.eionet.europa.eu/

LIGIT – Màster en Tecnologies de la Informació Geogràfica (Consulta maig de 2008):

- LIGIT: http://ligit0.uab.es/web/
- MTIG: http://ligit0.uab.es/mtig/index.htm

Manuals *Java* (Consulta octubre de 2009):

- *Sun Microsystems*: http://java.sun.com/docs/books/tutorial/
- http://www.itapizaco.edu.mx/paginas/JavaTut/froufe/introduccion/indice1.html#dos
- http://www.programacion.com/java/tutorial/java_basico/

Medi Ambient (Consulta setembre de 2009):

- *European Environment Agency*: http://www.eea.europa.eu/
- *The Directive on Environmental Noise*: http://ec.europa.eu/environment/noise/directive.htm

Mètrica versió 3 (Consulta setembre de 2009):

- Metodologia de Planificació, Desenvolupament i Manteniment de Sistemes d'Informació - Ministeri d'Administracions Públiques del Govern Espanyol; CSAE (Consell Superior d'Administració Electrònica): http://www.csae.map.es/csi/metrica3/index.html

Noise Map Viewer (Consulta setembre de 2009):

- http://noise.eionet.europa.eu/index.html

Normalització i estandardització (Consulta setembre de 2009):

- CEN - *The European Committee for Standardization*: http://www.cen.eu/cenorm/homepage.htm
- ISO - *International Organization for Standardization*: http://www.iso.org/iso/home.htm

- ISO/TC 211: http://www.isotc211.org/
- OGC - *Open Geospatial Consortium:* http://www.opengeospatial.org/
- W3C - *World Wide Web Consortium*: http://www.w3.org/ i http://www.w3c.es/

SQL (Consulta octubre de 2009):

- Microsoft: http://msdn.microsoft.com/en-us/library/bb510741.aspx
- http://www.maestrosdelweb.com/editorial/tutsql1/
- http://www.w3schools.com/sql/sql_intro.asp

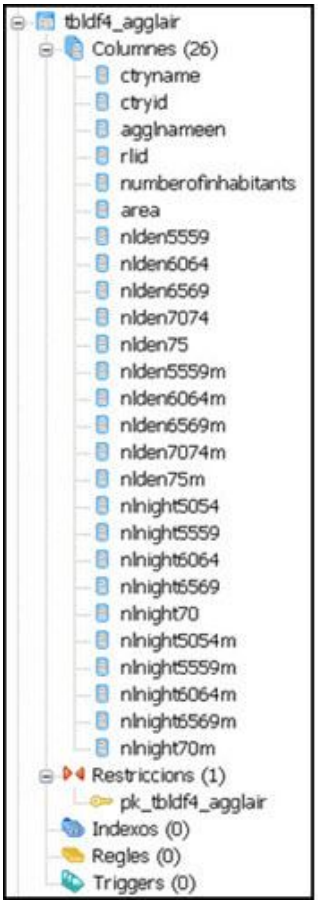
Tecnologies *Open Source* (Consulta setembre de 2009):

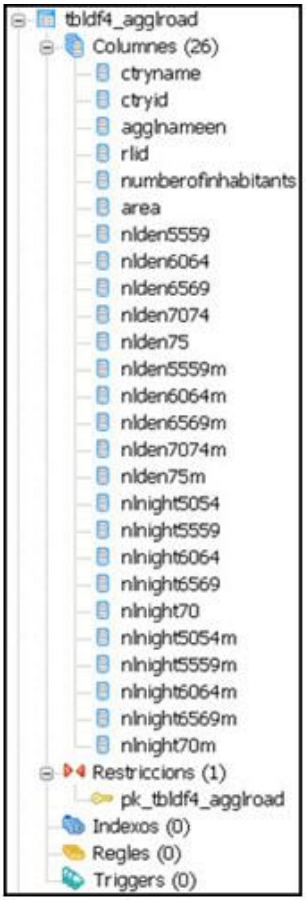
- *Eclipse*: http://www.eclipse.org/
- Java-Source.net: http://java-source.net/open-source/charting-and-reporting
- *JFreeChart*: http://www.jfree.org/
- *MapFish*: http://www.mapfish.org/
- MapTools – *MS4W*: http://maptools.org/

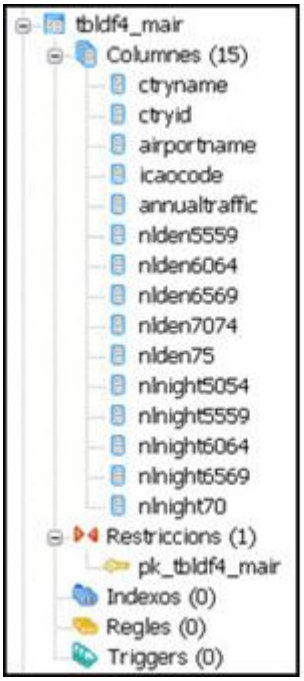
12. Índex de figures

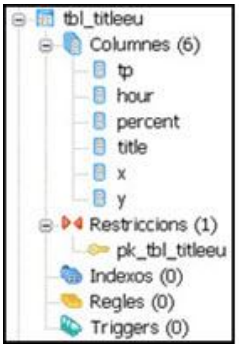
13. Annexos

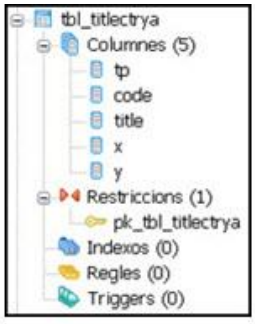
13.1. Taules i camps de *postgres*

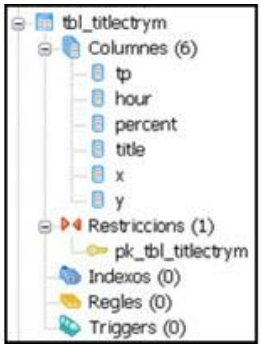


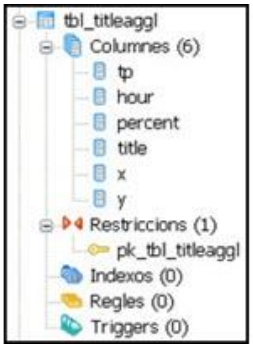






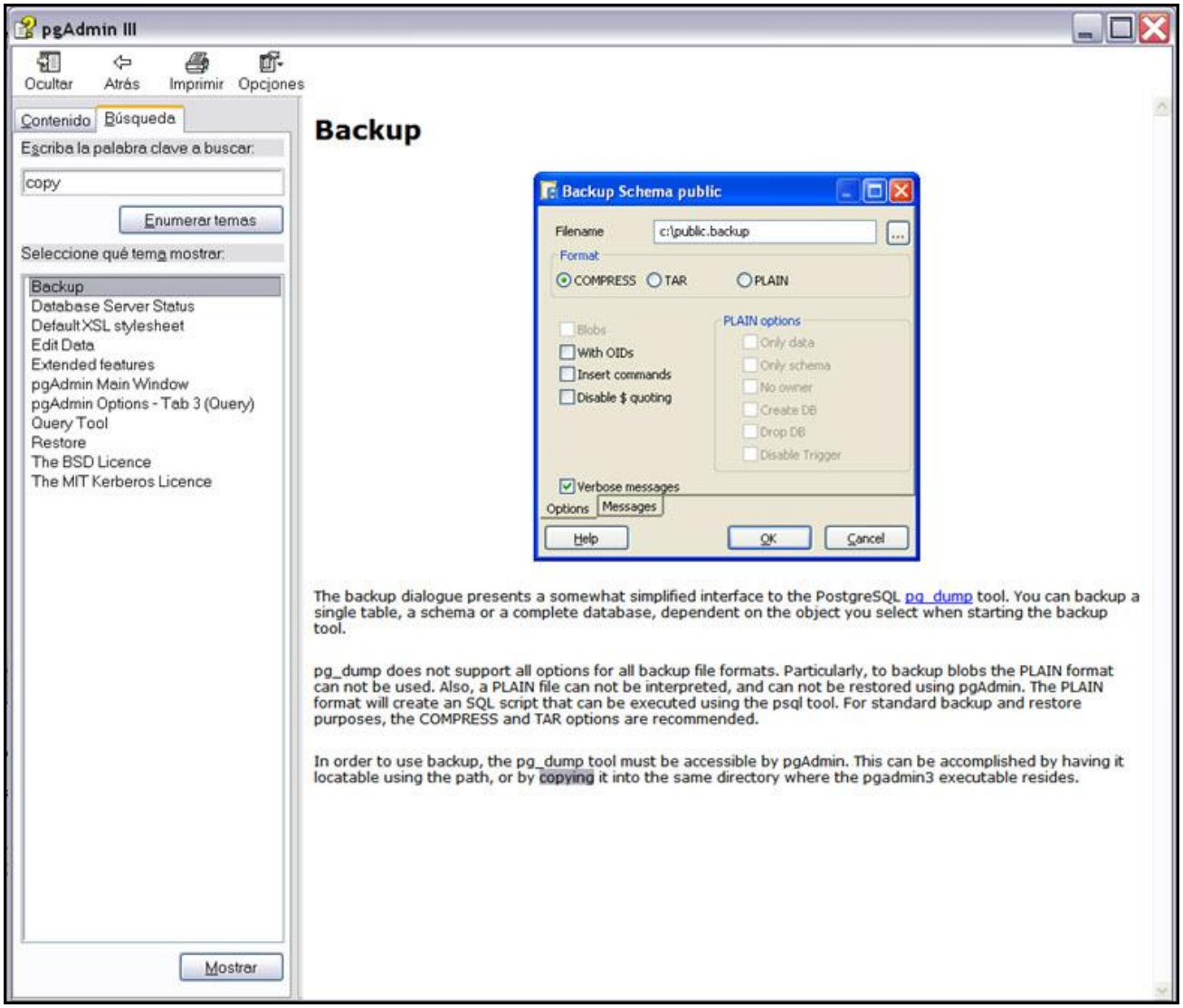




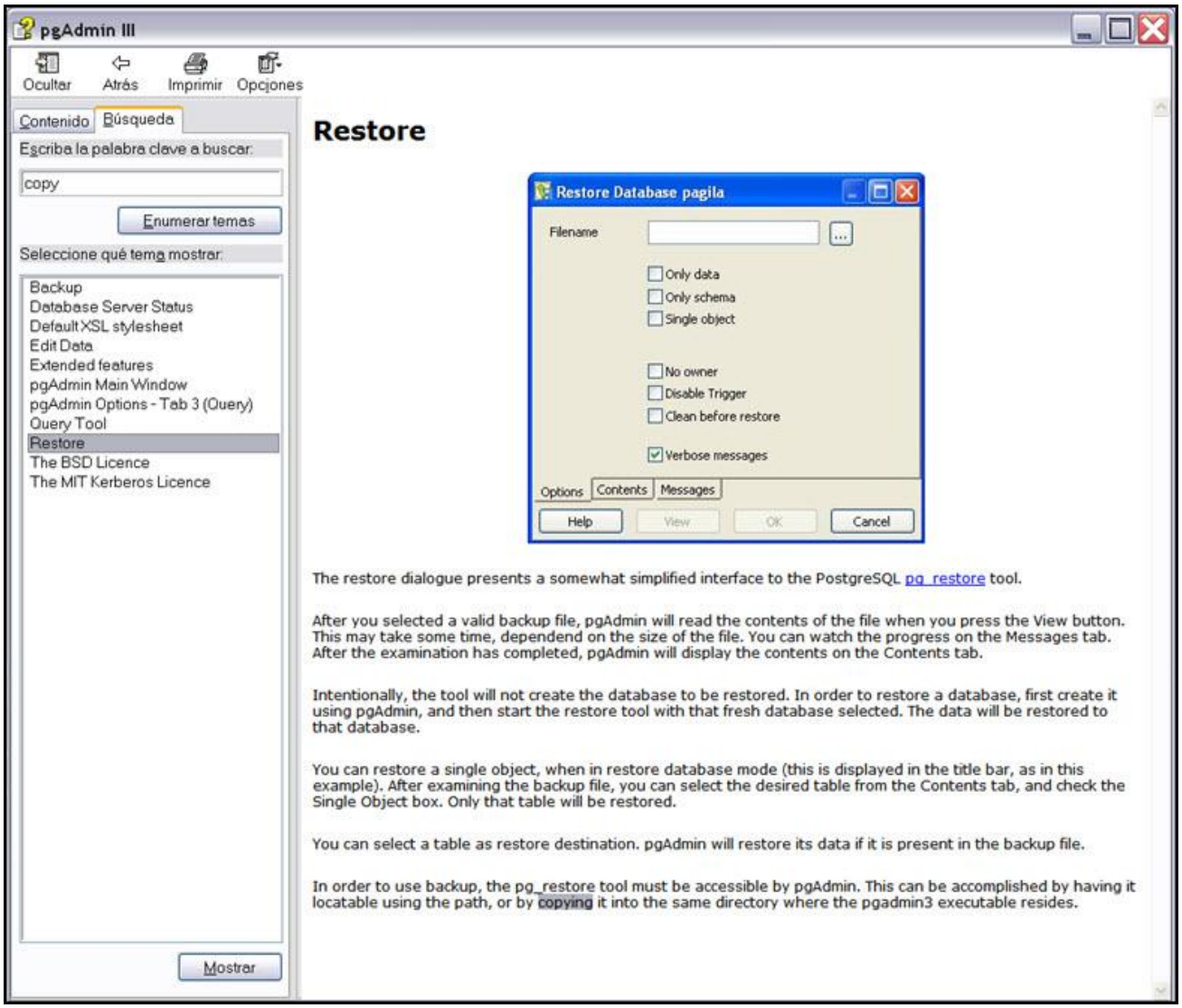


13.2. *Backup* i *restore* de *postgres*

Per fer la còpia de seguretat de la base de dades ***postgres***:



Per instal·lar la base de dades ***postgres*** al servidor o a altres màquines:



13.3. CD: *web_noise* (pàgina web), *postgres* (base de dades), *noise.war* (*servlet*) i *workspace* (*Eclipse*)