# GWT RAD Base:
## Desenvolupament d'un framework JEE .
## Aplicació pràctica: Registre E/S de la policia local.

**Eduard Ramon Escrihuela Planells.**
ETIG

**Verònica Peña Pastor.**
Consultora.

31 de desembre de 2008

# Dedicatòria i agraïments

Aquest treball de fi de carrera va dedicat a totes aquelles persones que estan al meu costat, i que tots els dies en donen el seu suport de manera incondicional.

Sobre tot a la meva esposa Maria José, que sent de la Mancha, ha tingut el coratge de matricular-se també a la UOC, fent dret (en català!) per a estar al meu costat estudiant en dies festius i fins a altes hores de la matinada.

Al meu fill Eduard, de tant sols disset mesos, que em troba a faltar moltes estones en les èpoques de lliuraments de PACs. Espero que agafi dels pares l'esperit d'estudi i sacrifici que li estem inculcant.

Als meus pares i sogres, per estar al meu costat en el moments difícils de la meva vida, i per haver-me transmet els seu valors ètics.

Als companys de treball Josep Enric, Juan Emilio , Joan Enric i també al meu cap César perquè fins ara han cregut en mi i han valorat la meva feina.

Als professors que he tingut en els estudis d'enginyeria de la UOC no presencials per donar-me un millor tracte personal com a l'alumne que en altres escoles universitàries presencials per les que he passat.

A Javier Paniza de Gestión 400 per la seva brillant tasca en el framework OpenXava, en el que he estat inspirat.

Als companys de altres administracions locals (Alfred de Cullera, Francesc de Bellreguart, Santiago de Sueca, Julio d'Alzira, David de Carlet, Amparo de Moncada, i una llarga llista més) per la seva inestimable ajuda.

A Sang Shin per la seva gran tasca divulgadora i docent de Java i JEE a la seva pàgina web *www.javapassion.com*.

A tota la comunitat Java Hispano, i en especial a Abraham per la seva tasca informadora en *www.**javahispano**.org* .

Gràcies.

# Resum

Crec que aquest projecte és una mica atípic tant pel contingut com per les eines que s'empren. Dissenyar un petit framework JEE per a desenvolupar aplicacions de gestió escapa una mica de l'abast d'una assignatura de 7,5 crèdits. A més emprar GWT a la part de la vista sembla una opció poc vista fins ara. Per tant em considero un "quixot" del segle XXI.

La finalitat d'aquest treball és fer-lo servir per a l'Ajuntament de Tavernes de la Valldigna, i aconseguir estar d'acord amb la llei 11/2007, de 22 de juny, d'accés electrònic dels ciutadans als Serveis Públics. Tant mateix ha de complir la llei de Protecció de Dades de Caràcter personal.

De moment a l'hora de redactar aquesta memòria, no s'han implementat encara moltes de les exigències de prestacions i seguretat que les lleis esmentades abans exigeixen. Per tant aquest projecte és un projecte viu i espero que real. També el meus coneixements limitats de les matèries que cal dominar han fet que aquest projecte encara deixi molt que desitjar.

Aquest treball consta de 2 parts, la primera consisteix en desenvolupar el framework JEE amb arquitectura MVC que he denominat GWT RAD Base (qualsevol altre nom hagués valgut també), i la segona és una aplicació pràctica de registre de entrada i sortida de la policia local implementada mitjançant aquest framework.

Aquest projecte és un "Mash-up" de tecnologies:

- GWT s'ha emprat per a la part de client, a més s'ha combinat amb diferents llibreries com "drag'n drop", "mosaic" i "gwt-incubator".
- Per a comunicació navegador-servidor d'aplicacions s'ha emprat el "gwt-rpc" que és una ampliació dels serveis http.
- A la part del servidor d'aplicacions s'ha fet servir un Tomcat i la part de controlador s'ha implementat amb servlets.
- Per a la part de persistència s'ha fet servir un SGBD Postgres que és atacat mitjançant Hibernate que és una implementació de JPA.
- Per a fer llistats s'ha emprat la llibreria Jasper Reports.
- S'han fet servir controls Google Maps amb algunes funcions GIS

Pel que fa a l'aplicació pràctica del registre d'entrada i sortida de policia, s'ha hagut d'implementar prèviament un sistema de control d'accessos, una gestió del territori i persones. Posteriorment, es demana també que s'implementi un registre d'accidents. S'ha fet servir un entorn multi idioma (de moment en castellà, català, valencià, francès, anglès i alemany). A més també està prevista una petita gestió d'expedients.

Es clar que tot aquests objectius són pràcticament impossibles d'aconseguir, així que els resultats es veuran molt després de que s'acabi aquest semestre.

# Índex de contingut

# Índex de figures

# Capítol I: Introducció

Les administracions locals són empreses de serveis, i una de les tasques primordials és el manejament d'informació. La informació és un recurs molt apreciat, i que té moltes similituds amb un altre be, que és l'aigua.

- En efecte, tots volem tenir la SUFICIENT informació, volem tenir coneixement de l'entorn per a poder actuar amb conseqüència.
- Però també la volem tenir CONTROLADA (no és suficient ser bombardejats amb cartes de correu ordinari i electrònic, publicitat, *banners*, etc.) sinó que hem de tenir-la a l'abast per a ser emprada quan ens faci falta.
- La QUALITAT d'aquesta ha de ser bona. Si aquesta informació no té claredat, precisió o no es explícita, doncs tal vegada no es podrà fer servir.
- Ha de tenir uns canals de DISTRIBUCIÓ, que la completen, depuren i la facin arribar allà on sigui útil, i en el moment indicat.
- Ha de tenir uns CONTROLS de QUALITAT de manera que arribi de manera controlada als llocs on es demandi i que arribi justament allò que es demani.
- L'ACCÉS ha de ser controlat, de manera que cadascú ha d'accedir a aquella informació que sigui necessària per a ell o el seu lloc de treball i que no sigui emprada per a altres fins.
- Al igual que el cicle de l'aigua, la informació a les administracions locals té un circuit que es denomina EXPEDIENT. Cada expedient rep la informació de nombroses fonts i es destina a diferents seccions o negociats on es tracta, s'avalua i es fan servir PROCEDIMENTS tipificats i normalitzats fins que es pren una resolució que es notificada a cada afectat.

Al mateix temps una empresa, considerada com un sistema, qualsevol actuació realitzada per un dels seus membres, pot repercutir en qualsevol subsistema de l'empresa. Per tant la informació ha d'estar INTEGRADA, o sigui, les fonts d'informació han de ser compartides, i per tant no ha de tenir duplicitats ni incongruències.

Pel que respecta al personal de seguretat d'un ajuntament, s'ha de dir que degut a la naturalesa del lloc de treball, on es combinen una gran quantitat de tasques diferents (actuacions al carrer, trànsit, festes, delinqüència, campanyes escolars, antidroga, socorrisme, escolta, judicis, torns de treball excessivament llarg acompanyats de llargues absències, informes de treball, ..) han de poder emprar sistemes de informació que siguin molt SIMPLES i intuïtius.

A més el sistema ha de ser de MANTENIMENT BAIX, la qual cosa evita costos i pèrdues de temps.

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

Avui en dia el registre de la policia local està sent portat mitjançant un full de càlcul. No es multi usuari, ni permet control adequat d'accessos, i li manca molta operatòria.

L'elecció d'aquest TFC també segué motivada per l'obligatorietat de compliment de la llei 11/2007, que obliga a les administracions locals a permetre als ciutadans la consulta dels expedients que estan afectats i a més poder fer la gran majoria de tràmits sense haver-se de

personar-se a les dependències municipals, i a més amb llibertat d'horari.

Ja són moltes les entitats que han vist Internet com una oportunitat de negoci i de oferir serveis còmodes i de qualitat. A més la informació no ha de ser patrimoni exclusiu de ningú; és per això que s'ha decidit per programari lliure amplament emprat.

És evident que una aplicació de registre policial no té cabuda en aquest decret 11/2007, ja que la informació que es maneja és confidencial i de alta protecció, i per tant no pot estar a l'abast dels ciutadans, però marca moltes pautes a seguir com és la possibilitat de ser multi idioma, multi usuari, té diferents nivells d'accessos a les opcions de menú, permet provar el framework, té una part de GIS, a més es veuen aspectes de tramitació d'expedients i *workflow*.

A més el nombre d'usuaris és petit, la qual cosa permet ser un bon banc de proves per a avaluar rendiments, mancances de seguretat, estabilitat i establir les pautes de millora en una primera fase.

A més és un bon moment de migrar les aplicacions en plataforma client servidor a la de "*cloud computing*" com es diu a les aplicacions *muti-tier* que poden fer servir diversos servidors amb possibilitats de virtualització.

## 1.2 Objectius del TFC

### 1.2.1 Objectius generals

Els objectius generals d'aquest TFC són:

- Entendre la filosofia de JEE (servidors d'aplicacions, *frameworks*, *servlets*, contenidors EJB, i altres tecnologies)

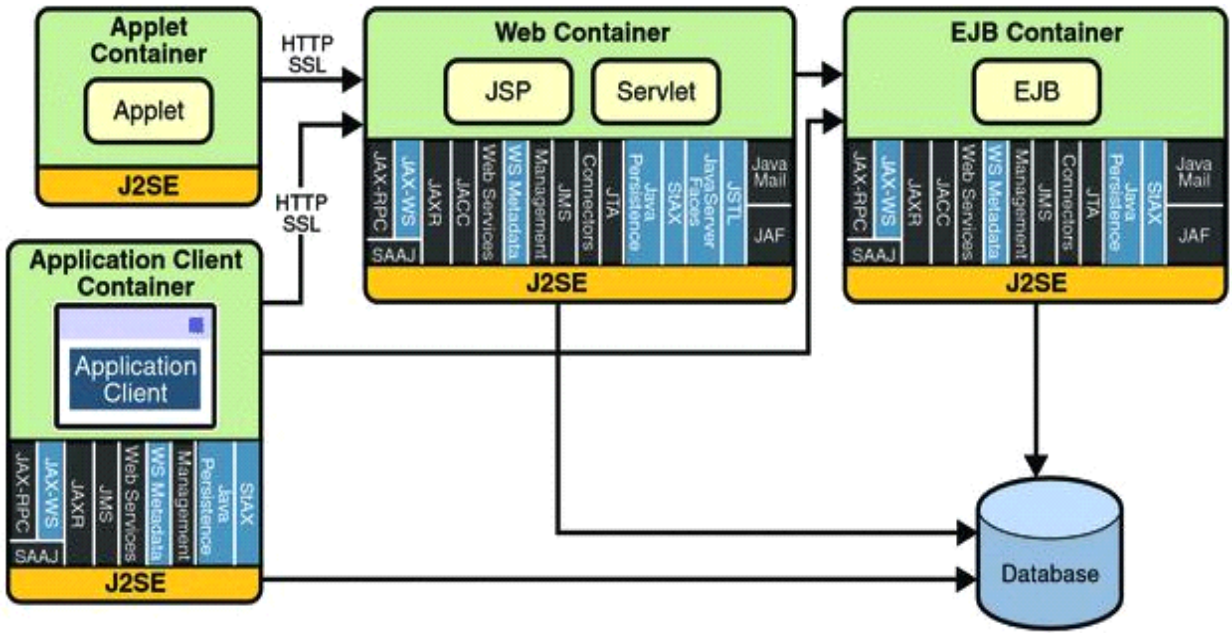



**Fig1: Diferents tecnologies emprades en JEE (font: Sun Microsystems)**

- Investigar com es pot implementar el procés de transaccions a les BBDD en JEE.
- Entendre l'ús d'alguns patrons com el ***façade***, ***MVC***, ***composite*** etc.

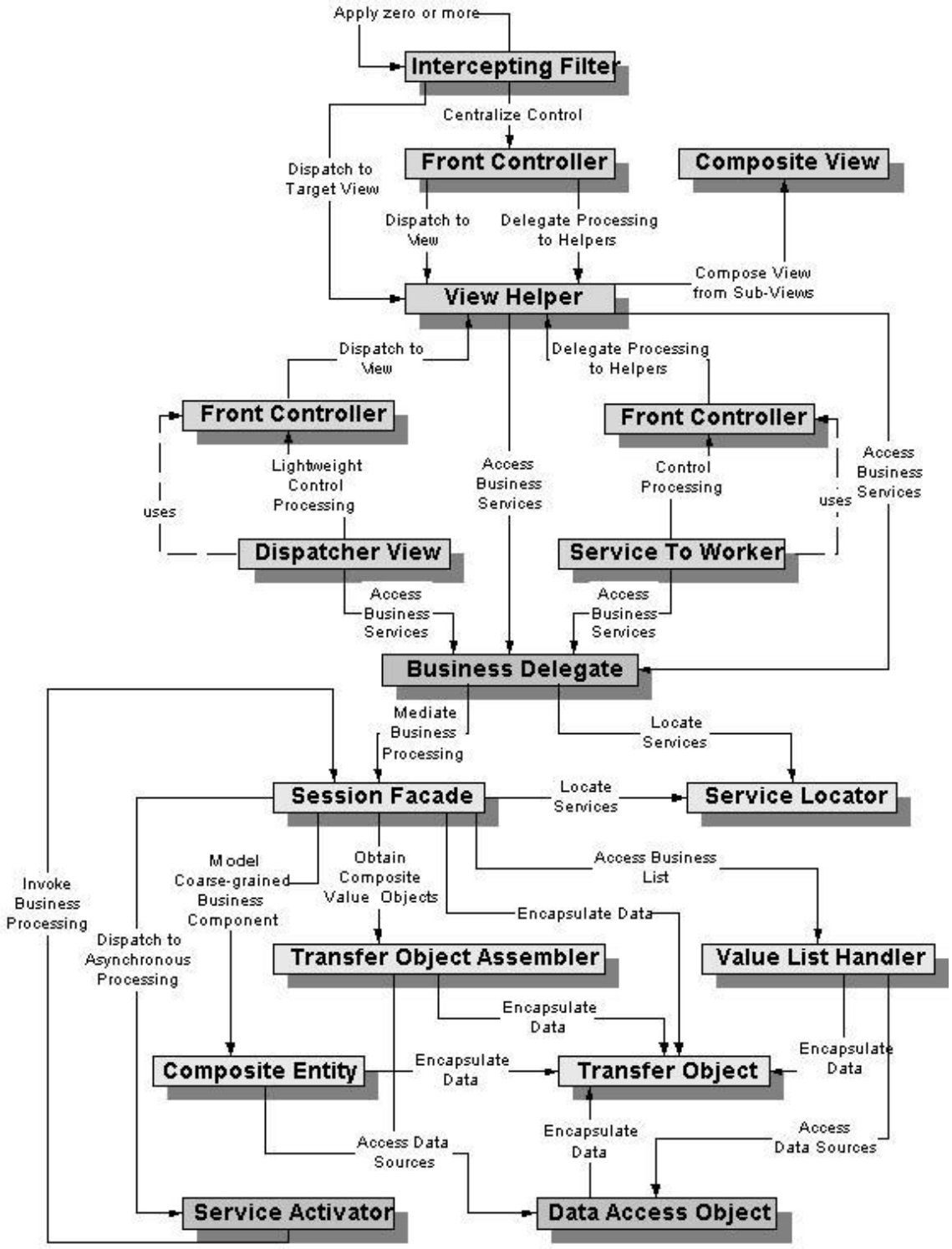



**Fig2: Diferents patrons emprats en JEE (font:Sun Microsystems)**

- Analitzar les diferents tecnologies que hi ha a la capa de client i a ser possible de tipus RIA (*Rich Internet Applications)*, i de software lliure (*JSF, GWT, Open Laszlo, Eco2, ItsNat* ..).
- Aprendre el funcionament de la signatura electrònica a aplicacions Web.

- Veure l'àmbit de treball de les filosofies de treball OO i sobre tot les Orientades a Components.
- Agafar confiança amb sistemes Linux.
- Comprendre com es crea una aplicació multi idioma.
- Fer servir metodologies àgils per al desenvolupament d'aplicacions, A ser possible es planteja desenvolupar un mini framework per a poder treballar amb especificacions de disseny, independitzant el desenvolupament de la tecnologia emprada en la part del client .
- Per altra part el TFC, segons les especificacions de l'assignatura ha de ser un treball eminentment pràctic, i que sintetitzi gran part dels coneixements adquirits durant la carrera.

## 1.2.2 Objectius particulars

Analitzem somerament els objectius d'alguns mòduls i etapes del projecte:

### 1.2.2.1 Encetament del projecte

Aquesta fase tractarà els següents objectius:

- Instal·lació del servidor
- Instal·lació del servidor d'aplicacions i tot el programari necessari del servidor
- Definició de permisos, tallafocs, etc.

### 1.2.2.2 Mòdul de signatura electrònica

Aquest mòdul tractarà sobre la posta en marxa de l'accés per signatura electrònica.

### 1.2.2.3 Mòdul multi idioma

Es veurà com la implementació de la internacionalització a nivell de client o de servidor.

### 1.2.2.4 Mòdul de taules secundàries o de codificacions.

Es definirà un procés de tractament d'aquelles taules o entitats que contenen poca informació (taules descriptives) i que tenen pocs registres i també pocs camps o atributs

### 1.2.2.5 Mòdul de bitàcola o log.

Es registrarà l'usuari que ha accedit a la aplicació, les operacions realitzades i els registres afectats i si cal si el resultat ha segut correcte o no. D'aquesta manera es compleix la LORTAD, i tant mateix es podran fer estadístiques d'ús del programari i si cal anotar els errors que s'han produït. D'aquesta manera es pot tenir un control de qualitat de l'aplicació.

### 1.2.2.6 Mòdul de definició d'accessos.

Es definirà la política d'accessos i permisos al sistema i els rols de cada usuari/grup.

### 1.2.2.7 Mòdul de proves de l'aplicació.

S'intentarà definir jocs de proves per a provar l'aplicació. Si hi ha temps es faran servir diferents eines per a provar l'aplicació (*JUnit* o altres).

### 1.2.2.8. Mòdul de SIG

Com a utilitat opcional es planteja emprar algun component per a situar en l'espai l'objecte de la petició rebuda.

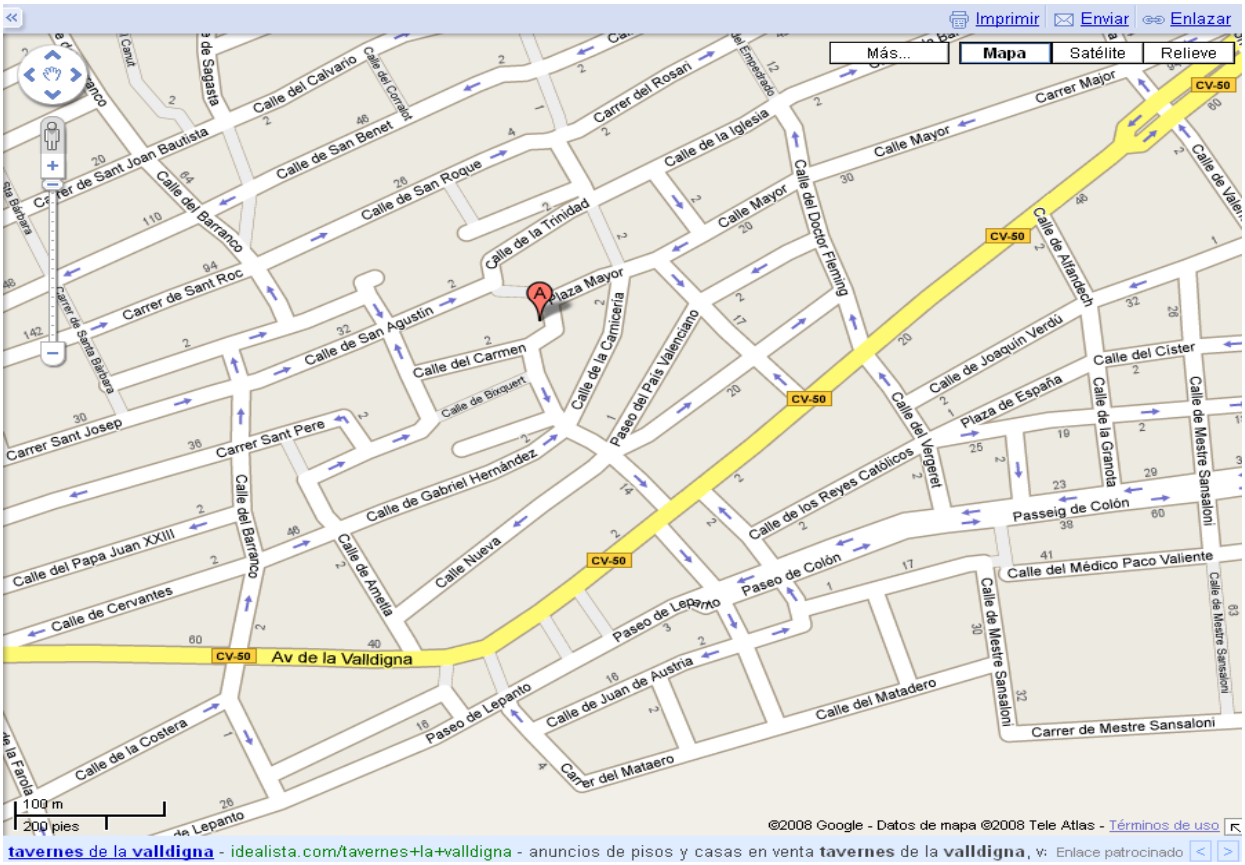



**Fig3: Google Maps (font: Google)**

### 1.2.2.9 Mòdul del framework

Es planteja els següents objectius:

- Crear una interfície d'usuari molt similar per a totes les pantalles. La qual cosa reduirà l'aprenentatge dels usuaris.
- Simplificar la tasca de programació. Una vegada realitzat l'anàlisi, ha de ser fàcilment implementat. Amb la qual cosa implica que:
  - S'ha obtingut una eina RAD (*Rapid Application Development*)
  - Els programadors poden ser persones amb baix perfil acadèmic.
  - Augment de la productivitat.
- Desenvolupament orientat a objectes: A cada classe disposa de informació de la vista i del comportament mitjançant mètodes i esdeveniments

- Facilitat de manteniment. Si evitem la generació de codi, es simplificaran les tasques de manteniment de codi, desplegament, compilació i jocs de proves.
- Evitar al màxim l'ús del críptic XML a favor de les anotacions. S'aconseguirà major claredat del codi.
- Independència del motor de base de dades.
- Poder fer servir un editor de dades per a cada tipus de dades (quadres de text, llistes desplegables, visualitzadors de coordenades espacials, calendaris..) i poder definir-ne de nous si cal.
- Establir validacions de dades .
- Poder navegar entre diferents pantalles per a obtenir o inserir dades momentàniament.
- Fer informes i exportar la informació a diferents formats.

#### 1.2.2.10 Mòdul de registre.

Es planteja les següents funcionalitats:

- Enregistrar les peticions rebudes .
- Fer esmenes a les dades enregistrades (ocasionalment)
- Agrupar en una sola les peticions duplicades (que tal vegada han vingut per diferents vies)
- Fer consultes, filtres, llistats i estadístiques.
- Tancar les peticions que ja han segut ateses.
- Poder ser emprat des de diferents ubicacions mitjançant un navegador.
- Establir un sistema d'accessos ben controlat amb enregistrament del treball de l'usuari.
- Tenir una bona gestió de la documentació usada i a ser possible tenir-la en format electrònic per a evitar pèrdues .
- Definir el *workflow* de tramitació d'expedients.
- Fer un seguiment d'expedients on controlar estats.
- Generar estadístiques per a poder prendre decisions.
- Control de terminis de documentació a subministrar als jutjats.
- Gestionar un registre d'accidents.
- Poder visualitzar informació geogràfica.

#### 1.2.2.11 Mòdul d'actuacions.

Cada petició enregistrada pot ser la base d'una sèrie d'actuacions policials. Cadascuna d'aquestes actuacions ha de ser guardada al sistema amb informació detallada d'inici i fi ,recursos emprats i situacions especials provocades.

Normalment les actuacions s'efectuen en base d'un procediment preestablert. Si el TFC es complica molt, tal vegada s'obviarà els procediments establerts, però no la tipificació de les accions. Per tant es podrà:

- Enregistrar actuacions

- Definir els recursos (materials i humans) emprats en l'actuació
- Indicar els desperfectes causats i altres dades a destacar
- Donar comptes al negociat de patrimoni dels desperfectes produïts
- Donar comptes al negociat de personal dels recursos humans emprats en horari extra laboral.

Com es pot veure, aquest és un projecte una mica ambiciós, per tant si hem de ser realistes, moltes d'aquestes funcionalitats no podran ser desenvolupades ja que el temps requerit sobrepassa al disponible.

## 1.3 Enfocament i mètode seguit

El camí general que he emprat és el següent:

1. Veure quins productes hi ha disponibles. Fer una selecció inicial en base a les crítiques dels usuaris, basant-me en les prestacions i corba d'aprenentatge.
2. Instal·lar-los i provar-los. Descartar aquells que donen molts problemes o tenen una alta complexitat, o estan mal documentats. Si no hi ha alternatives a productes candidats a descartar, no hi ha més remei que fer la dedicació suficient per a poder-los usar.
3. Una vegada decidit quins son els productes mínims a emprar, que permetran tenir una aplicació amb unes funcionalitats mínimes es planteja una metodologia ***top-down*** inicial, mitjançant la qual, es defineix un sistema amb funcionalitats mínimes i he intentat dividir en problema en altres mes senzills. Una vegada dissenyats els petits subsistemes, he intentat acoblar-los i fer els ajustes per a que els sistema funcioni amb les funcionalitats mínimes.
4. Una vegada comprovat que el sistema compleix els mínims, es passa a una fase d'**increment de funcionalitats** en base a l'esquelet prèviament construït i provant nous productes que permetin implementar aquestes funcionalitats.
5. Aquestes noves funcionalitats exigeixen algunes vegades replantejar els elements existents per a que siguin mes oberts o generals. La qual cosa obliga a redissenyar els components originals, però en base al principi d'obert-tancat. Per tant s'està fent servir un **cicle de vida en espiral**.
6. Com es parteix de controladors genèrics que arrepleguen les instruccions de creació de vistes o de comportament, la fase inicial ha segut molt dura ja que els errors d'execució eren difícils de depurar. El principal problema és que si qualsevol dels controladors creats, després d'un canvi de versió té un "*bug*", a les hores totes les classes de l'aplicació agafen el "bug". Per tant cal tenir molta cura en testar les aplicacions abans de ficar-les a disposició dels usuaris.
7. Cal tenir especial cura amb el tractament i emissió d'excepcions, ja que el sistema es torna molt inestable i fan molt difícil el procés de depuració.
8. Altre tema a tenir en compte és el de la documentació. Cal tenir una gran disciplina per a documentar. He intentat documentar en anglès, per a que es pogués fer servir per una comunitat mes gran de gent, però com el projecte ha segut realitzat a estones per motius de família i de treball, a vegades s'ha documentat en català i altres s'ha deixat la documentació per a posteriors moments, i ha segut difícil entendre el codi una vegada ha passat el temps!.
9. Respecte al disseny gràfic de les pantalles, s'ha intentat refinar el "*look*" per a que siguin agradables a l'usuari, però jo no soc dissenyador per tant és pot millorar la imatge

d'aquestes.

Els productes estudiats han segut:

1. **Postgres**. Per a avaluar la migració de SQL Server a un SGBD lliure sobre Linux.
2. **GWT**. La seva filosofia és molt similar a les llibreries Java *Swing* i *AWT*. Per a comunicació amb el servidor empra *RPC* (similar a Java RMI). L'únic problema va ser entendre la forma de treballar asíncrona *d'Ajax* que si no s'empra correctament, els jocs de proves donen resultats molt imprevisibles.
3. **CSS i HTML**: Per a definir estils o tenir una idea del DOM que hi ha per sota de GWT.
4. **Java Script**: Sols somerament.
5. **Servlets**. Per a estudiar les funcions d'un servidor d'aplicacions i com podia passar dades al client. Un dels pilars dels *servlets* són les sessions, que he descobert al final.
6. **Hibernate**. És una implementació JPA molt difosa. Independitza l'aplicació del SGBD. Cal tenir en compte la configuració de les connexions JDBC que empra per sota. Altre aspecte important és l'ús d'anotacions.
7. **Anotacions**. He estudiat com fer anotacions i he dissenyat algunes per a definir els valors per omissió dels atributs, així com els editors de camp, i algunes restriccions.
8. **Reflection**. És una llibreria Java molt poderosa per a treballar en tipus abstractes de dades, i sense aquesta, hagués segut molt difícil implementar el framework.
9. **Jaspert Reports**. Que és una gran eina en Java per a dissenyar reports.
10. **Google Maps i Google Gadgets**: Per a donar suport mínim de GIS i altres vistositats.

## 1.4 Planificació del projecte

### 1.4.1 Consideracions inicials.

Tal com està marcada la planificació sembla un projecte amb cicle de vida en cascada, la qual cosa no es gens certa. Com es pot apreciar l'estudi dels components del JEE dura tot el semestre. És important arribar a la fase d'implementació amb coneixements suficients per a començar a produir codi.

Un altre factor a tenir en compte és el efecte piramidal entre el que es vol inicialment i el que s'aconsegueix. Es a dir, aquest projecte no s'acaba amb la memòria del TFC, sinó que va a durar més temps, per tant el producte final te uns terminis superiors als del TFC. Prego doncs al tribunal que tingui en compte aquest aspecte.

Les fases que entren en joc en aquesta planificació són:

### 1.4.2 FASE 0: Estudi del JEE

Aquesta fase dura tot el projecte, és una fase de formació i investigació, on es proven diferents ferramentes. Les tècniques de desenvolupament de codi estaran basades en aquesta fase. Per tant és una fase molt important i que cal fer la suficient dedicació. A més en aquesta fase es desenvoluparan petits components de manera que s'acurti el període d'implementació.

### 1.4.3 FASE 1: Pla de treball

Aquesta fase pràcticament enceta el projecte, per tant s'ha de preparar el pla de treball, saber de quina disponibilitat de temps hi ha, els recursos a emprar etc. Al final es genera el producte que és la PAC1. Les tasques de que consta són:

- Elecció del TFC: 4 dies.

- Definició del TFC: 6 dies.
- Planificació del TFC: 2 dies.
- Revisió de productes: 1 dia.
- Lliurament de la PAC1: 1 dia

### 1.4.4 FASE 2: Anàlisi de sistemes

Aquesta fase és molt important, aquí emprarem les tècniques d'anàlisi orientades a objectes que hem aprés durant la carrera. S'han de plantejar les entrevistes als usuaris que treballen al negociat, esbrinar quins usuaris estan disposats a assumir la responsabilitat del procés de recollida d'informació, veure quins problemes tenen avui en dia (de productivitat, organització, seguretat, disponibilitat espacial i temporal), quines entitats entren en joc, quines utilitats volen que implementi l'aplicació, quins són els actors i els seus rols bàsics, etc.

Per tant conèixer l'estat actual del sistema i saber on volem arribar és sense dubte la fase més important. El producte final d'aquesta segona fase és la PAC 2.Les tasques que consta aquesta fase són:

- Definició de requisits: 7 dies.
- Creació del model de casos d'us: 7 dies.
- Diagrama de classes: 8 dies.
- Interfície gràfica d'usuari: 9 dies.
- Revisió del producte: 1 dia.
- Lliurament de la PAC2: 1 dia.

### 1.4.5 FASE 3: Disseny de sistemes

Una vegada realitzat l'anàlisi de sistemes, ens queda refinar aquest de manera que les entitats i altres elements aportats pels usuaris, quedin ben estructurats i normalitzats en base a les tècniques estudiades en la carrera. Per tant aquesta fase, i les que esdevindran no aconseguiran cap qualitat si la fase anterior presenta moltes deficiències. Al final d'aquesta fase, el producte aconseguit és la PAC3. Les tasques que conformen aquesta fase són:

- Definició de la capa de domini: 14 dies.
- Disseny de la persistència: 5 dies.
- Disseny del joc de proves: 3 dies.
- Revisió de productes: 1 dia.
- Lliurament de la PAC3: 1 dia.

### 1.4.6 FASE 4: Implementació

Es suposa que en aquesta tasca, el temps serà el principal enemic del projecte. Estarem al final del semestre, amb una gran càrrega de treball d'altres assignatures amb PACS que lliurar. A més conèixer les ferramentes JEE i aconseguir un domini d'aquestes serà un llindar difícil de superar. Al final els productes aconseguits són:

- La memòria del TFC
- La presentació del TFC
- La implementació del programari
- Document adjunt de compilació i desplegament de l'aplicació.

Les tasques que defineixen aquesta fase són:

- Preparació del maquinari i S.O.: 1 dia.
- Instal·lació del programari de desenvolupament: 1 dia.
- Desenvolupament del codi: 20 dies.
- Creació de jocs de proves: 2 dies.
- Redacció de la memòria i documents annexos: 20 dies.
- Elaboració de la presentació: 2 dies.
- Revisió de productes: 1 dia.
- Lliurament dels productes: 1 dia.

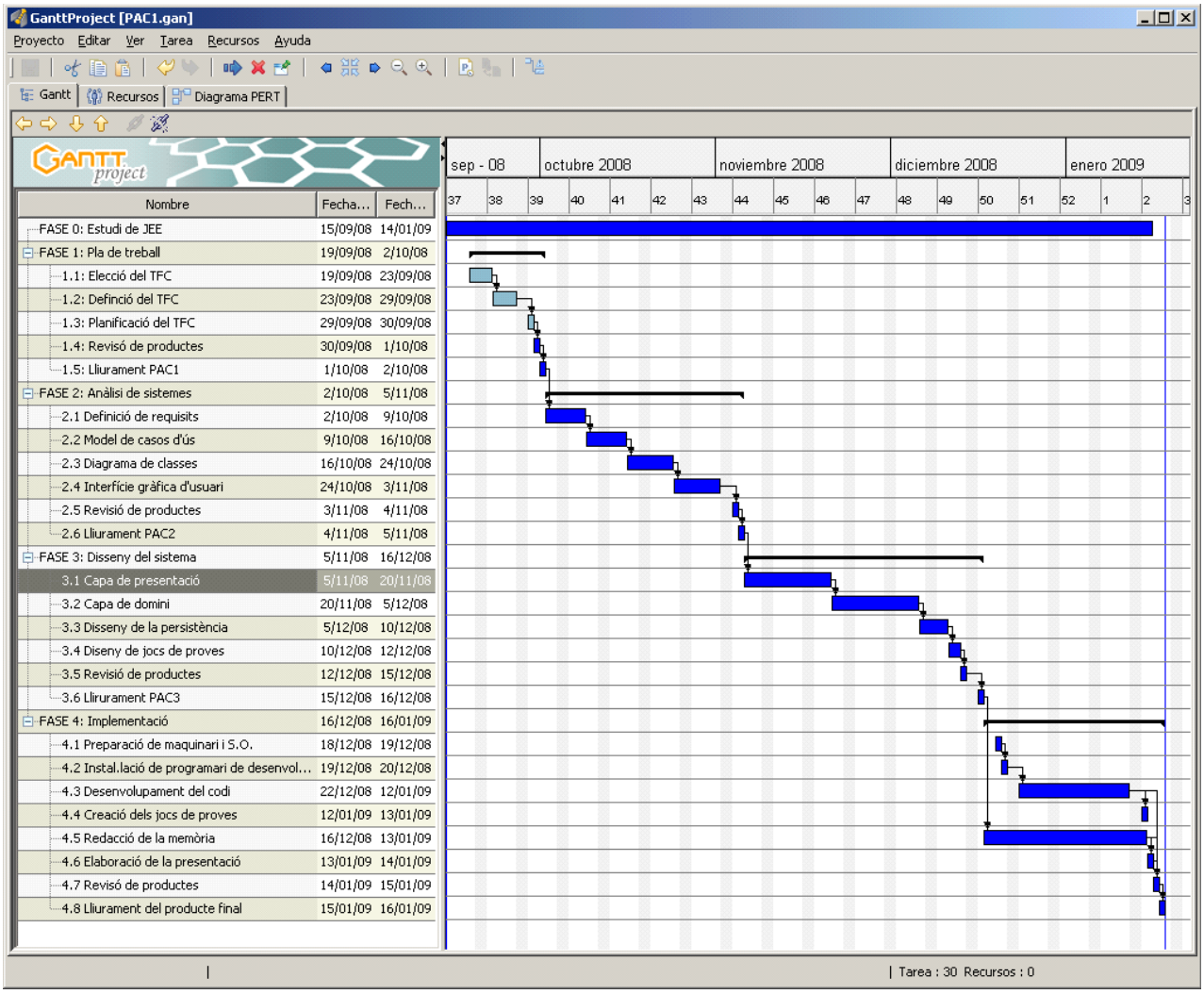



**Fig4: Diagrama de Gantt amb la temporització.**

## 1.4.7 Conclusions sobre la planificació.

Per a resumir aquest pla de treball és interessant fer les següents puntualitzacions:

- Aquest és un treball real
- Acomplir els requeriments inicials sembla impossible amb el temps i recursos destinats.
- La finalització definitiva d'aquest projecte serà posterior a la data de lliurament del TFC.
- Per tant molts dels requeriments plantejats seran opcionals per a incloure en el TFC.
- La planificació feta a priori pot ser poc acurada ja que es desconeix aquesta tecnologia.

- Aprendre les tecnologies emprades en JEE és un gran repte que requereix gran quantitat de temps, però que és molt important per al meu futur professional.

## 1.5 Producte obtingut

Els productes obtinguts seran:

### *1.5.1 Framework (GWT RAD Base)*

Aquest framework es el producte estrella del TFC. Està pensat per a :

- Definir des del model (classe) les especificacions de la vista.
- Definir en un atribut de l'objecte aplicació, el seu subsistema de menús.
- Poder emprar diferents editors de atributs i fàcilment definir-ne d'altres implementant una interfície.

### 1.5.2 Aplicació de Registre E/S de la policia local

Per a confeccionar aquest producte, s'ha emprat el framework anterior. La aplicació en concret disposa de diverses opcions:

- Administració de l'aplicació: control d'accessos, definicions de subaplicacions i menús, usuaris, grups etc.
- Definició de taules de suport i codificacions.
- Sistema de registre de E/S.
- Sistema de registre d'accidents.
- Estadístiques, llistats, i sistema de presa de decisions.
- Tramitació d'expedients.

## 1.6 Breu descripció dels altres capítols de la memòria.

D'ara endavant s'estudiaran els següents aspectes:

- L'arquitectura de l'aplicació (arquitectura de xarxa, patrons emprats (sobre tot MVC))
- Components principals del framework (paquets, classes i justificació)
- Descripció del disseny orientat al model mitjançant el frameworK GWT RAD Base.
- Anàlisi de l'aplicació de Registre d'E/S de la Policia Local.
- Instruccions bàsiques de compilació i desplegament.
- Aspectes particulars del disseny orientat al model de l'aplicació de Registre E/S.
- Conclusions
- Glossari
- Bibliografia.

# Capítol II: Arquitectura de l'aplicació.

## 2.1 Especificacions generals de disseny

Els tres aspectes més importants a tenir en compte per a dissenyar una aplicació són:

- La **reutilització**: Com a forma de aprofitar i amortitzar la inversió de recursos en el desenvolupament del model conceptual (classes) i de components. El fet de reutilitzar les classes ve donat de la ma de tenir altres aplicacions anteriors, les quals fan servir classes del model conceptual que en certa manera, intervenen en la nova aplicació. El fet de reutilitzar components surt del plantejament de fer cada component amb el nivell d'abstracció mes alt possible
- **Patrons**: Són per antonomàsia, el recull de l'experiència grans experts, i que constitueixen una recepta efectiva per a resoldre amb èxit problemes que sovint es plantegen. Es poden comparar a plantilles on el detall i contingut específic de cada cas concret queda a càrrec del dissenyador.
- ***Frameworks***: Són una infraestructura de base composta per classes i components (reutilitzables) que ajuden a construir una aplicació. Normalment tenen una alta especialització, però tot depèn en gran mesura de l'abast i el mitjans del projecte. En concret, un framework com el *.net* de *Microsoft*, té una gran complexitat ja que té un propòsit més general. Altres com *OpenXava*, fa ús d'altres *frameworks* (JSF, JPA) i utilitats (ANT) i components (*Jasper Reports*) per a desenvolupar aplicacions de gestió. Normalment els *frameworks* fan servir patrons (ja que son mes abstractes), però també poden fer servir d'altres *frameworks*. A part dels avantatges que tenen els *frameworks* pel que respecta a l'ús i reutilització de classes i components ja desenvolupats, presenta una sèrie de desavantatges:
  - Corba d'aprenentatge molt alta. La quantitat de conceptes a dependre és considerable, i alguns són difícils de entendre.
  - Poden fer servir gran quantitat de recursos del sistema, ja que tal vegada no farem servir tota la funcionalitat que ens permet el framework. Altres vegades, els *frameworks* no fan un ús òptim dels recursos de maquinari.
  - Cal anar a versions estables: A vegades hi ha canvis en els components de base, i si no estan ben acabats i testats, poden provocar errades en aplicacions que fan servir aquests components. A vegades, aplicacions que funcionen adequadament en versions anteriors, no tenen un comportament estable amb les noves versions.

Entre els principis de disseny cal esmentar els següents:

- **Baix acoblament**: Es preferible que la dependència entre classes, paquets, etc., ha de ser mínima, així es manté a un mínim les dependències.
- **Alta cohesió**: Cal tenir una alta relació entre les responsabilitats de les classes . Amb aquest principi s'aconsegueix facilitat per a entendre, de reutilitzar, mantenir i gran immunitat front a canvis del sistema.
- Principi d'**obert-tancat**: Una entitat de programari ha d'estar oberta a l'extensió i tancada a modificacions. Si fos en cas contrari, qualsevol canvi ocasionat en una classe de la que es depèn, provocaria mal funcionament a les classes que depenen d'aquesta.

- **No repetició**: Cal evitar tant la duplicació d'informació com de responsabilitats. El principal avantatge és que es reduirà el cost de manteniment.
- **Segregació d'interfícies:** Mitjançant interfícies cal evitar que una classe o subclasse faci ús d'operacions que estan definides però que no s'han de fer servir.
- **Inversió de dependències**: Els mòduls o classes d'alt nivell, no han de dependre dels mòduls o classes de baix nivell, sinó d'una abstracció. Així si es canvia el servidor de fitxers o de bases de dades, no ha ser difícil redireccionar l'aplicació cap a un altre servidor.

## 2.2 Arquitectura de xarxa

L'arquitectura de la xarxa a emprar en principi no té massa complicació, ja que com el nombre d'usuaris a emprar el sistema és baix (inferior a 5), no cal un disseny de xarxa ni de servidors que sigui molt espectacular. De tota manera, per al cas dels servidors es pot créixer per virtualització en cas que es vegi un descens del rendiment ja sigui del servidor d'aplicacions com del servidor de BD o altres elements. També s'ha de tenir les precaucions adequades per a evitar intrusions. Un sistema DMZ podria ser aconsellable.

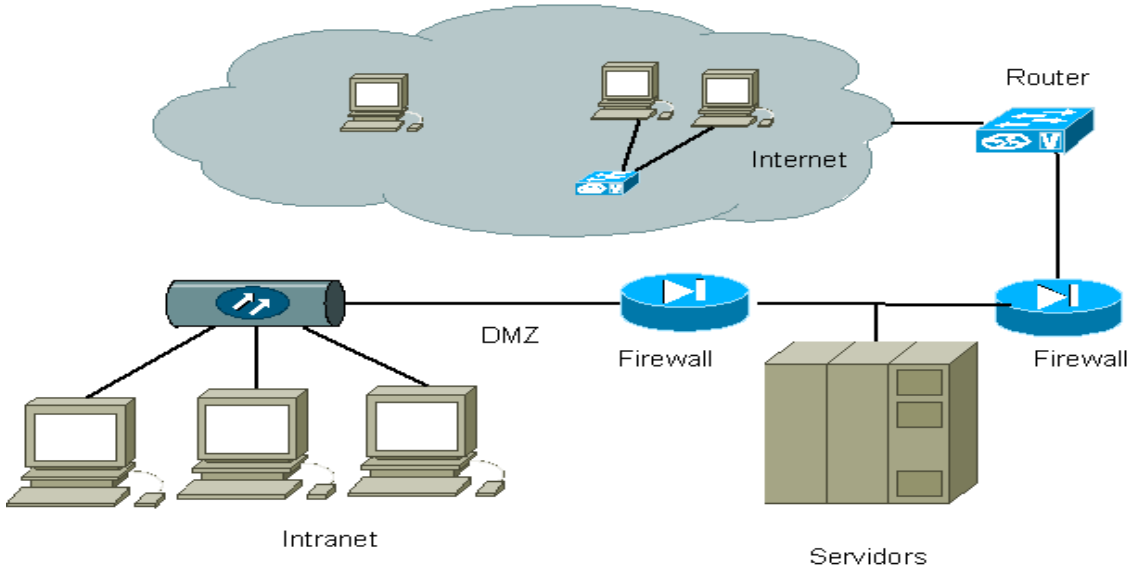



**Fig5: Esquema d'una estructura DMZ.**

## 2.3 Arquitectura JEE amb patró MVC

Es farà servir una arquitectura en tres capes, (*n-tier*) **presentació**, **domini** (servidor d'aplicacions) i **serveis tècnics** (persistència, llistats), segons les especificacions de JEE.

JEE és una definició d'arquitectura creada per a la integració en Internet, on el nombre d'usuaris pot ser alt, i es delimiten molt clarament les funcions de cada servidor i dels components a emprar.

Estem fent servir el patró **MVC** ja que permet separar la capa de la lògica de negoci de la de presentació, simplificant la implementació, ja que els components es poden implementar per separat i amb independència. Vegem breument els components d'aquest patró:

- **Model**: Són les classes que contindran les dades a manipular i candidates a la persistència . A mes inclourà totes les regles de negoci. Això es farà a nivell de POJO i

anotacions i esdeveniments de persistència.

- **Vista**: Es la que forma la interfície d'usuari. Es realitzarà amb GWT i romandrà en el navegador del client!.
- **Controlador**: És la part que interactua amb les dues anteriors. Els components principals són els ***servlets***, encarregats de rebre les ***XMLHttpRequest*** del client.

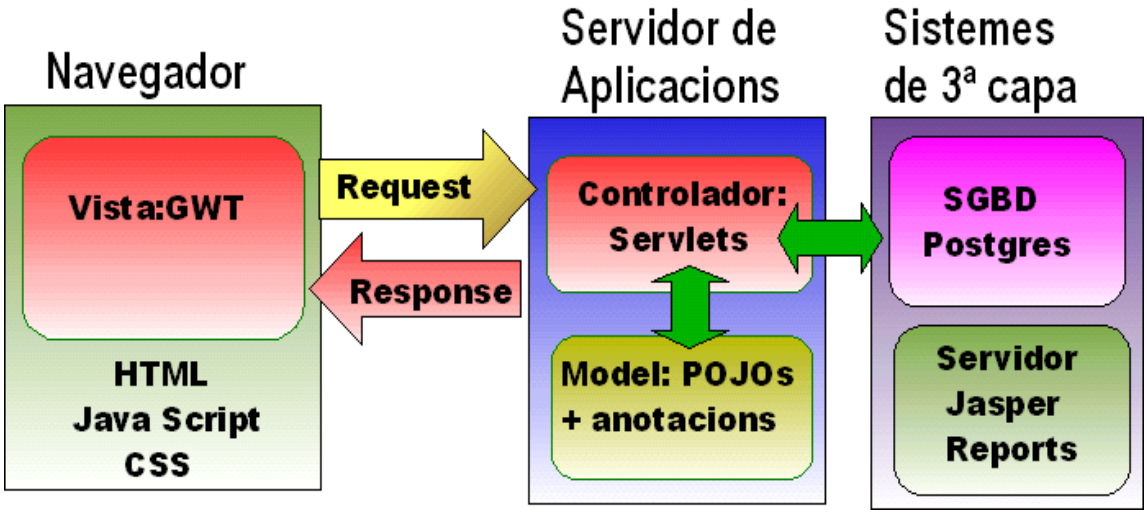



**Fig6: Diagrama del patró MVC emprat.**

## 2.4 Nivell de la capa de presentació

Per a la capa de presentació es farà servir el *framework GWT* (*Google Web Toolkit*). Aquest *framework* té les següents característiques:

- Disposa d'us sistema de components molt similar al *Java Swing o Java AWT.*
- Es desenvolupa amb codi font *Java* compatible amb el *run-time* 1.5, i hi es compila a HTML i *Java Script* compatible amb els navegadors més emprats: *Microsoft Explorer, Mozilla Firefox, Safari* i *Opera* i també per descomptat pel nou navegador *Google Chrome*.
- Disposa d'un sistema de *RPC* (*remote procedure calls*) que fa servir peticions asíncrones del tipus *XMLHttpRequest*, que són la base de *AJAX* , per a desenvolupar *RIA* (*Rich Internet Aplications*)
- També es poden fer servir peticions *HttpRequest*
- A la part del servidor es fan servir "*servlets*" o programes *Java* que es s'executa dintre d'un servidor d'aplicacions. Per tant el controlador estarà centrat de forma majoritària al servidor.
- Disposa d'un complet sistema de controls d'esdeveniments, la qual cosa fa que tingui un subsistema propi MVC. Els components GWT seran dissenyats fent servir aquest patró.
- Es pot accedir al DOM que hi ha per sota dels components, i canviar l'aspecte i comportament.

## 2.5 Nivell de persistència.

Tal com s'ha dit abans, s'opta pel SGBD *Postgres*, ja que inicialment s'havia plantejat per a substituir l'antic MS-SQL Server. Un dels avantatges de *Postgres* és que es pot fer servir tant en servidors Linux (en producció) com en entorns de desenvolupament Windows. Disposa de bones eines d'administració tant de domini públic com de pagament, la documentació és prou completa, i hi ha una comunitat relativament gran al voltant d'aquesta BD.

A més les prestacions semblen molt bones tant respecte a velocitat, estabilitat, robustesa i seguretat. També, un dels aspectes mes atractius de *Postgres*, que de moment no l'ha fet servir, és la seva capacitat de tractament de dades i procediments GIS (*PostGIS*).

Tal vegada s'hagués pogut optat per MySQL que ha segut adquirida per Sun Microsystems, la qual cosa dona un gran marge de confiança.

Però com ja s'ha dit abans, el fet d'emprar JPA ens permet fer la inversió de dependència respecte al SGBD, per tant, en qualsevol moment, donades les ferramentes de migració de dades adequades, es pot fer servir un altre SGBD. Com *Hibernate* empra connexions JDBC, cal disposar dels *drivers* JDBC adequats i configurar la connexió. En aquest cas, la connexió s'ha definit a la classe ***HibernateUtil.*** Més endavant s'explicarà tot el procés de configuració.

## 2.6 Nivell de lògica de negoci.

Sempre s'actuarà mitjançant transaccions, ja que és un dels components de *Hibernate* i a més s'intenta que es faci completament les accions.

A més s'ha centralitzat en objectes específics, tota la gestió de extracció, inserció, modificació i esborrat de dades. La qual cosa permet configurar la lògica de negoci des de la capa de model, a partir de 3 maneres.

- La primera és mitjançant les anotacions de *Hibernate* i *Hibernate Validator*, les quals permetin fer tant un control simple a nivell d'atribut, com un control de la integritat referencial a nivell de BD.
- Mitjançant anotacions pròpies. Per a definir valors per omissió, editors, etc.
- La segona és implementant la interfície ***IBusinessLogic***, mitjançant la qual es pot definir la lògica de classe per als següents esdeveniments:
    - ***beforeBeanRetrieval***, ***afterBeanRetrieval***, per a realitzar comprovacions o accions abans i després de recuperar l'objecte des del SGBD.
    - ***beforeBeanUpdate***, ***afterBeanUpdate***, per a realitzar comprovacions o accions abans i després de actualitzar l'objecte al SGBD.
    - ***BeforeBeanInsert***, ***afterBeanInsert***, per a realitzar comprovacions o accions abans i després d'inserir l'objecte al SGBD.
    - ***BeforeBeanDelete***, ***afterBeanDelete***, per a realitzar comprovacions o accions abans i després de esborrar l'objecte del SGBD.

## 2.7 Nivell Web, servlets i RPC.

El nivell Web és la part del servidor que interactua directament amb el client. El client realitzarà les peticions i serà un *servlet* l'encarregat de processar les dades i retornar la informació demandada pel client després de consultar si fos necessari a la BD o actuar amb

altres subsistemes com el de llistats. La comunicació es fa mitjançant invocacions RPC de GWT. Analitzem doncs aquests components:

### 2.7.1 Servlets

Amb l'objecte de evitar redundàncies a l'hora de definir interfícies, s'ha optat per tenir un únic *servlet* que atengui tot tipus de peticions. Es clar que aquest *servlet* tindrà molts procediments definits, i serà l'encarregat de desviar les accions cap als controladors adequats; i tot en base als paràmetres passats al *servlet*. Aquest *servlet* ha de implementar una interfície que sigui reconeguda des del client.

A més s'han emprat altres *servlets*, principalment per al subsistema de llistats, però no fan ús del mecanisme RPC.

### 2.7.2 Mecanisme de transmissió RPC.

Aquest mecanisme permet que el client GWT(*Java Script*) i servidor (Java) es puguin conèixer i intercanviar informació. Per a que es pugui realitzar tot aquest procés s'han de definir un conjunt d'interfícies i establir un servei (amb una classe ***proxi***) que apuntarà cal al servlet del servidor d'aplicacions. Vegem un esquema d'aquesta estructura

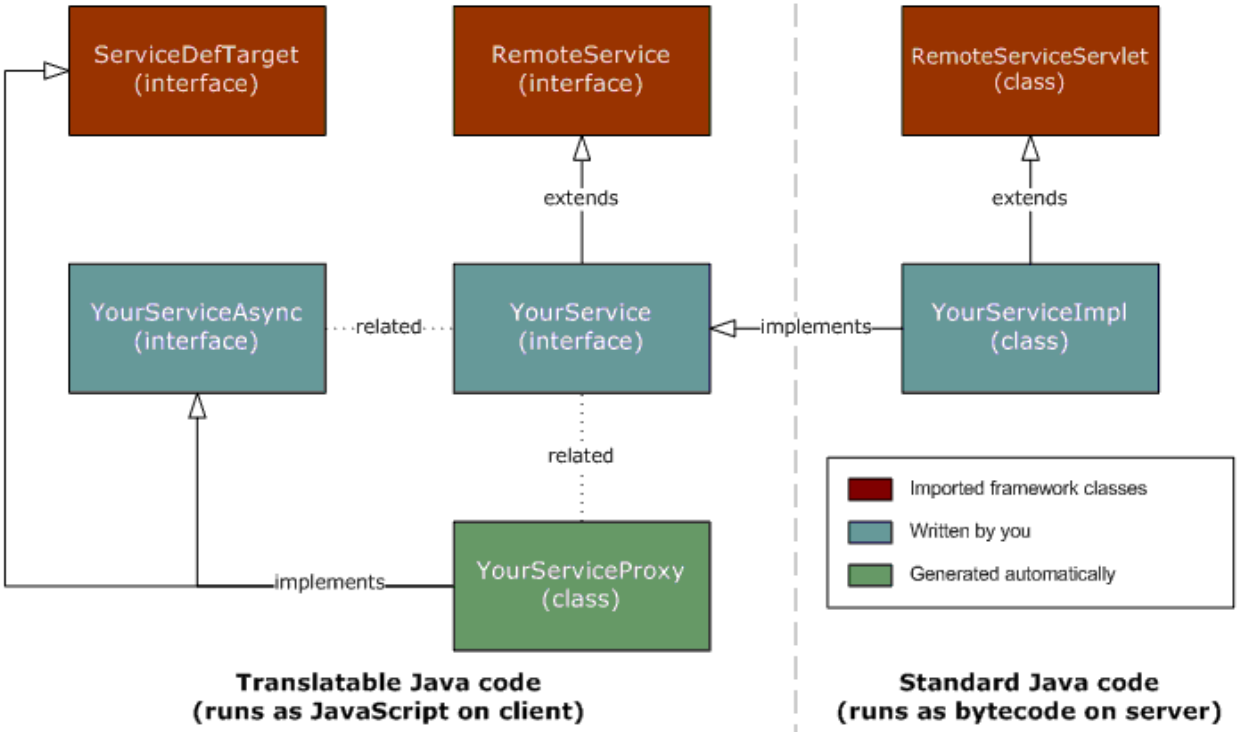



**Fig7: Configuració de les classes i interfícies de client i servidor en GWT-RPC. (Font Google)**

# Capítol III: Components i paquets.

Com es pot veure des de l'*Eclipse IDE* l'aplicació té un paquet general denominat ***tav***. Aquest paquet té altres subpaquets i elements independents. A part ,hi ha altres arxius que estan fora d'aquest paquet.

Abans de detallar cada component, farem una petita introducció del framework, i això es veu com encaixa cada component dins del entorn del framework.

## 3.1 Anàlisis introductori del framework.

Bàsicament es fan servir 2 grups de classes, un grup que actua de *vista* (del model MVC) que és el que rep del domini tota la informació de l'estructura de la vista i crea els components necessaris.

L'altre grup de classes és el que fa de controlador, que rep el control quan es produeix un esdeveniment i es dedica a prendre les accions oportunes.

Dintre del grup de la vista, hi han diferents classes, i cadascuna té una funcionalitat concreta: definició de menús, formularis de classe, formulari d'accions, formularis de conformació, controls o editors dins de cada formulari o menú.

Però en el grup del controlador i ha també diferents elements que actuen de controladors, i en base al tipus d'esdeveniment a tractar. Hi ha controladors de tractament del focus, d'accions de botons, de crides entre formularis, de crides a persistència i de control d'errors.

Un exemple típic de creació d'un formulari (Vista) és el següent. Des del menú es selecciona fer un manteniment d'un objecte concret, a les hores els controladors actuen per a aconseguir la informació des del model i crear un objecte "*vista*" però completament *customitzat* a partir d'aquesta informació. Aquesta informació de la vista està composta per una col·lecció de components a incrustar en el formulari a mostrar.

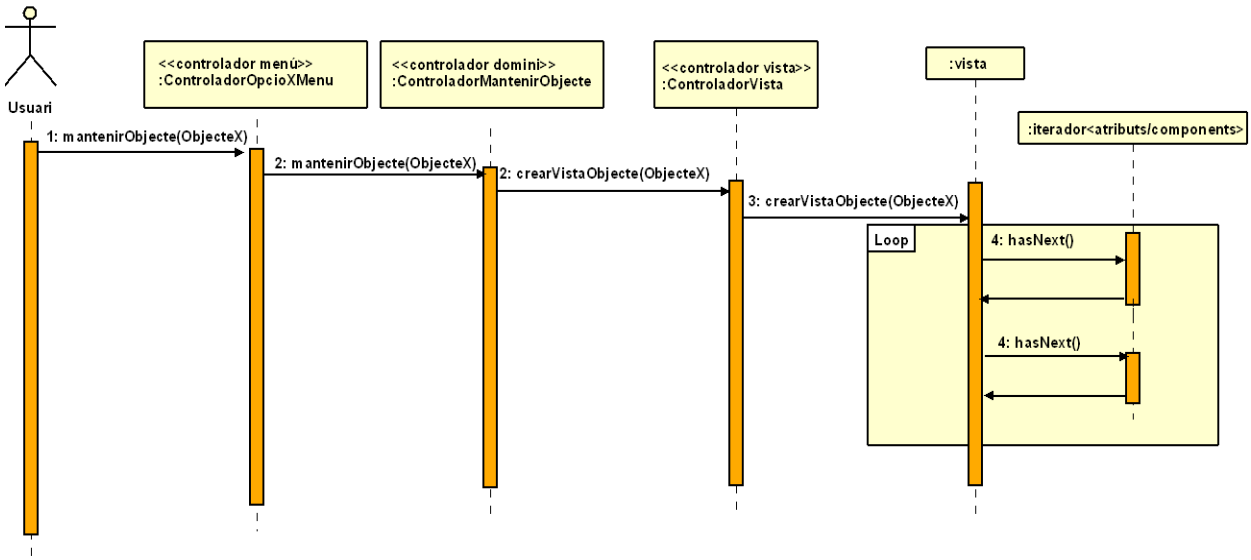



**Fig8: Diagrama de seqüències .**

Pel que fa al funcionament de les classe vista i controlador (per al manteniment d'un

objecte) cal indicar que hi ha unes opcions bàsiques com manteniment (altes- baixes – modificacions), fer llistats, fer detall de col·leccions, i a més es poden adjuntar altres opcions definides en el *domini.*

Queda més clar en un diagrama de col·laboració simplificat (on no es mostren els fluxos de retorn d'informació):

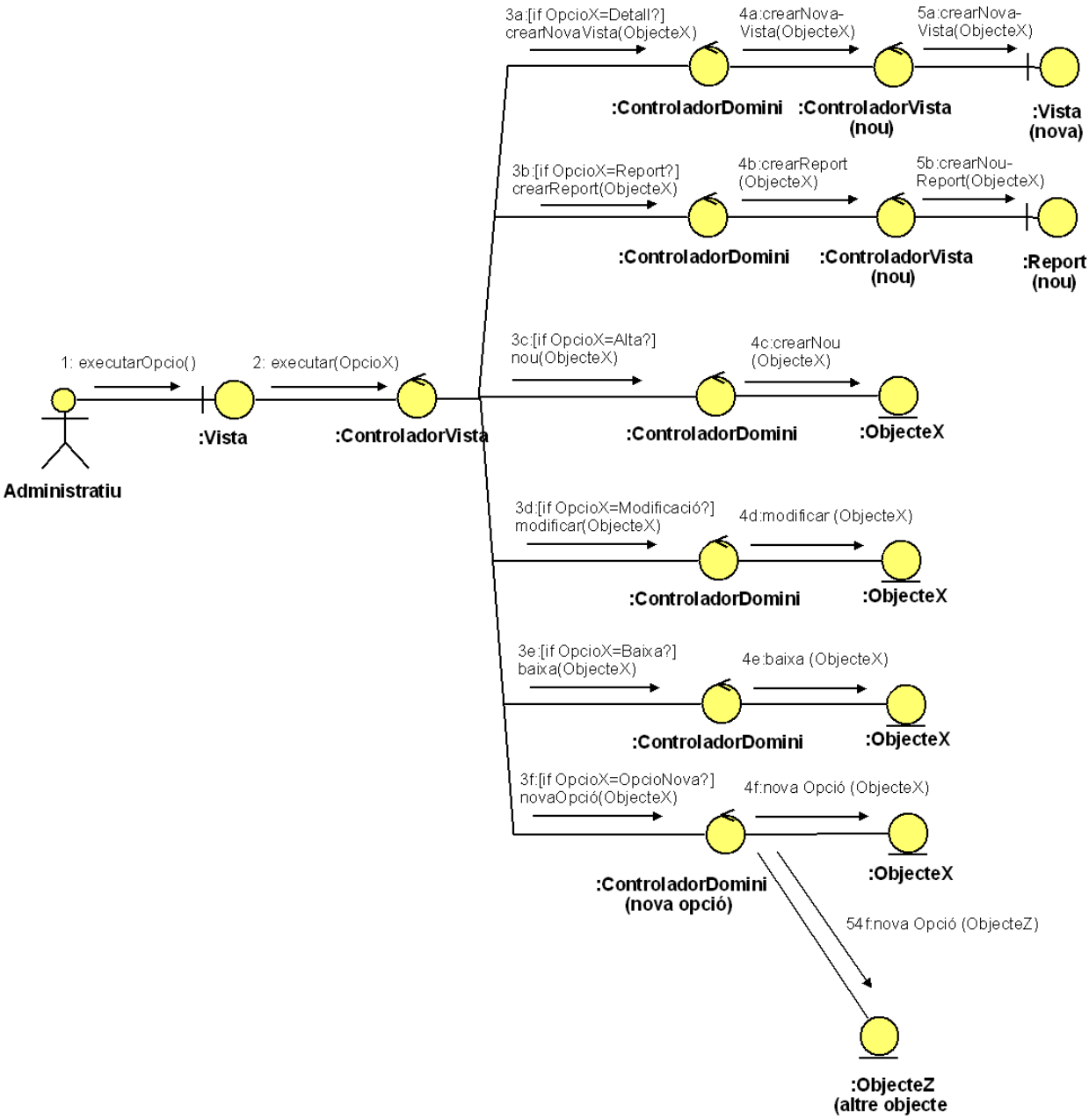



**Fig9: Diagrama de col·laboració.**

Pel que fa al tema de patrons, es farà servir el patró ***Strategy***, com una manera de unificar el comportament dels grups de components iguals que llancen esdeveniments (grups de botons etc), simplificant per tant les crides als controladors. També per a fer la comunicació entre la vista i el controlador es pot fer servir el patró ***Observer*** mitjançant una interfície.

## 3.2 Elements externs al paquet *tav*.

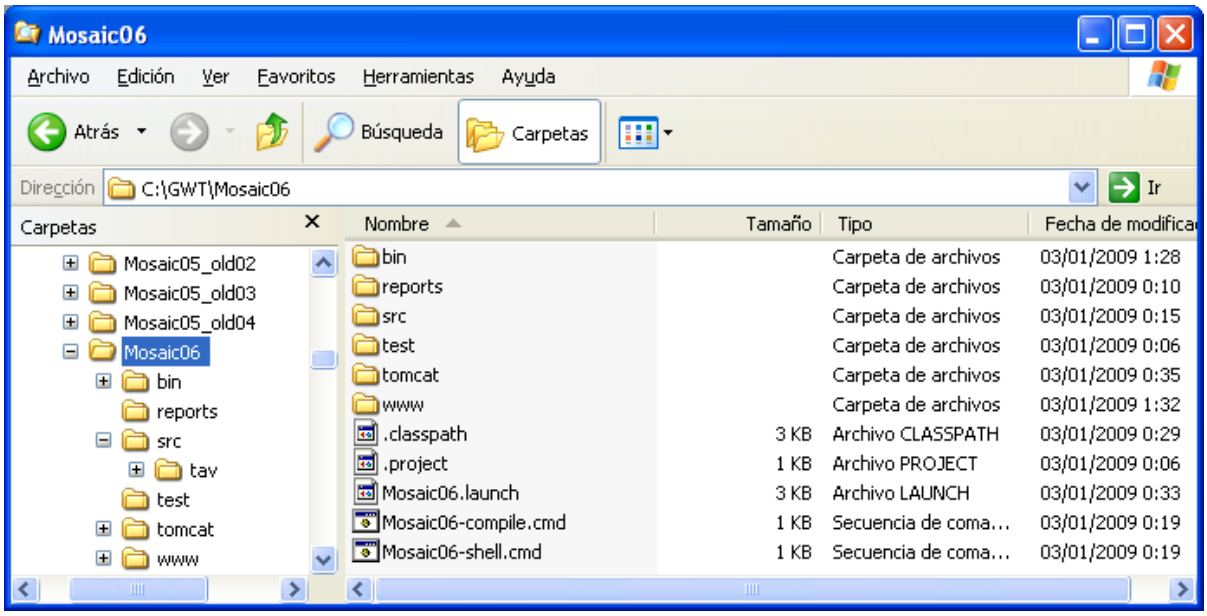



**Fig10: Vista parcial dels elements del projecte Mosaic06.**

Cal destacar els següents arxius que formen part de la compilació i execució (en mode host) del codi GWT, i que prenen el nom del projecte al que pertanyen:

- ***Mosaic06-compile.cmd***: Com el seu nom indica, és per a compilar, i conté una comanda de compilació que entre altres informació detalla les rutes de les llibreries(en format jar) de GWT a emprar. Per al nostre cas, a part de les llibreries estàndard de GWT, hem fet ús de les llibreries ***drag'n drop***, ***incubator, mosaic i maps***.

  Es mostra com a exemple il·lustratiu el contingut del fitxer *Mosaic06-compile.cmd*

```
@java -Xmx256M -cp "%~dp0\src;%~dp0\bin; C:/GWT/gwt-windows-1.5.3/gwt-user.jar; C:/GWT/gwt-windows-1.5.3/gwt-dev-windows.jar; C:/GWT/jars20081212/gwt-dnd-2.5.6.jar; C:/GWT/jars20081212/gwt-incubator.jar; C:/GWT/jars20081212/gwt-mosaic-0.1.4.jar; C:/GWT/jars20081212/gwt-maps.jar" com.google.gwt.dev.GWTCompiler -out "%~dp0\www" %* tav.Mosaic06
```

**Fig11: Contingut del fitxer de compilació Mosaic-06.cmd**

- ***Mosaic06-shell.cmd***:És l'encarregat de l'execució del programa dintre de l'IDE. També ha de referenciar aquestes llibreries emprades.

  A més també cal esmentar les següents carpetes:

- ***tomcat****:* És la carpeta on es crea la configuració de un servidor d'aplicacions *tomcat* reduït per a ús intern en *hosted-mode* per a fer depuració amb l'IDE.
- ***www***: És on es guarda tot el codi HTML i *Java Script* generat pel compilador, i que després s'ha de desplegar al servidor d'aplicacions en producció.
- ***reports***: És la carpeta temporal on *Jasper Reports* crearà dinàmicament els llistats.

## 3.3 Elements interns del paquet *tav*.

Entre els elements solts que hi ha al paquet ***tav*** destaquem:

### *3.3.1 web.xml.*

Aquest fitxer s'empra com a descriptor dels *servlets* emprats. Aquí s'han de definir obligatòriament tant els *servlets* que empren el mecanisme RPC de GWT com els servlets clàssics que deriven directament de *javax.serlet.* Vegem doncs aquest fitxer.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

  <!-- Standard Action Servlet Configuration -->
  <!-- RPC -->
  <servlet>
    <servlet-name>FormService Service</servlet-name>
    <servlet-class>tav.server.services.FormServiceImpl </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>FormService Service</servlet-name>
    <url-pattern>/Services_EDU</url-pattern>
  </servlet-mapping>

  <!-- Altres servelts NO RPC(1): Veure un fitxer per l'explorer -->
  <servlet>
    <servlet-name>ServletFiles</servlet-name>
    <servlet-class>tav.server.jasper.ServletFiles</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ServletFiles</servlet-name>
    <url-pattern>/ServletFiles</url-pattern>
  </servlet-mapping>

  <!-- Altres servelts NO RPC(2) Crida a Jasper Report -->
  <servlet>
    <servlet-name>ProvaJasperReport</servlet-name>
    <servlet-class>tav.server.jasper.ReportSendToBrowserServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ProvaJasperReport</servlet-name>
    <url-pattern>/ProvaJasperReport</url-pattern>
  </servlet-mapping>

  <!-- Pàgina de benvinguda -->
  <welcome-file-list>
    <welcome-file>Mosaic06.html</welcome-file>
  </welcome-file-list>
```

```
</web-app>
```

**Fig12: Descriptors de servlets i pàgina de benvinguda al fitxer web.xml**

S'ha marcat en vermell el servlet que fa servir el GWT per al RPC i s'adjuntarà el nom del servlet, la classe que el representa i la URL que cal indicar per a poder-lo cridar. Tant mateix s'ha marcat en verd la resta de *servlets* que no són de GWT i que s'empren per als llistats. Per a la selecció de la pàgina de benvinguda s'empra el color blau.

### 3.3.2 Arxius de recursos (internacionalització i18n).

S'ha emprat Netbeans per a administrar els arxius de recursos, ja que és molt pràctic.

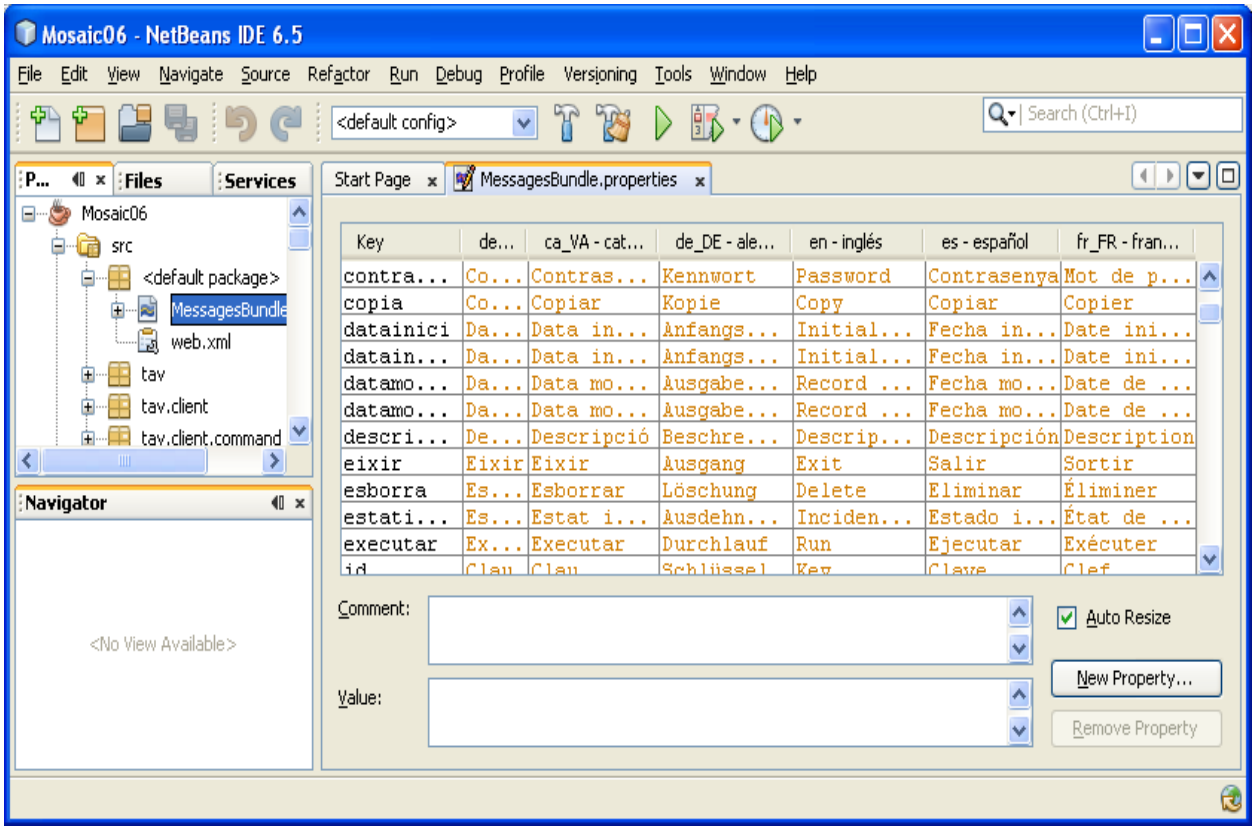



**Fig13: Ús de Netbeans per als arxius de recursos i18n.**

Com es pot apreciar, s'ha definit recursos per a valencià, català, castellà, francès, anglès i alemany. Per a trobar les paraules s'ha fet servir el traductor de Google, que encara no està massa ben acabat, per tant ens podem trobar alguna sorpresa.

S'han ficat claus per a cada nom d'atribut i de classe, i també per a missatges de navegació, avisos i errors.

## 3.4 Paquets dintre de *tav*.

### *3.4.1 tav.public*

Conté el descriptor d'herències *Mosaic06-gwt.xml* i també la carpeta *public* amb les imatges que s'empren en els botons per a que siguin mes intuïtius.

### 3.4.2 tav.client

Conté les 2 classes principals de la interfície d'usuari:

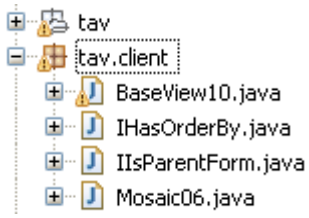



Fig. 14: Paquet tav.client

- ***Mosaic06***: És la classe de benvinguda i que engega l'aplicació (ja que implementa la interfície GWT ***Entrypoint*** i a més està definida al fitxer *web.xml*), empra un component ***CompositeBody***, que implementa el patró ***Composite*** per a reutilitzar el codi.

- Primerament demana a l'usuari que s'identifiqui i després sobre aquesta classe es representa el menú fent servir la classe*ApplicationMenu*, que més endavant explicarem. Com a més, pot fer crides a altres pantalles i no ha de perdre el control de les pantalles filles, ha d'implementar la interfície ***IIsParentForm***.

- ***BaseView10***: És la classe que s'encarrega de dibuixar les pantalles i fer el manteniment de les dades amb tant sols donant-li el nom de la classe, i unes condicions d'acotació de les cerques. El nombre 10 indica la versió que està actualment aquesta classe. Ella s'encarrega de fer crides a altres classes (***WidgetUtilities*** i ***FormCollection***) per a crear els editors i les col·leccions.
- A més conté 2 interfícies: ***IIsParentForm*** i ***IhasOrderBy***, la primera és per a aquelles pantalles que necessiten tenir control dels formularis fills i la segon és per aquells formularis han de mostrar la informació de la BD en base a un criteri d'ordenació. Recordem que les interfícies les fem servir per a l'especificació de disseny *segregació d'interfícies* i també com una manera d'implementar l'herència múltiple.

### 3.4.3 tav.client.command.

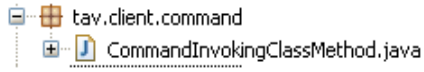



Fig. 15: Paquet tav.client.command

Conté la classe ***CommandInvokingClassMethod*** per a executar mètodes d'una classe des de el menú.

### 3.4.4 tav.client.command.

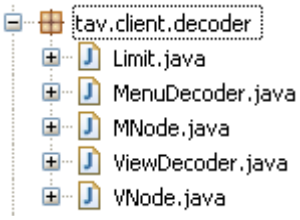



Fig. 16: Paquet tav.client.decoder

Conté classes per a descodificar la informació per a generar la vista (formularis) i els menús. La informació té una estructura d'arbre, ja que hi ha elements de la vista que poden contenir altres elements (panells i altres contenidors per al cas dels formularis, i submenús per al cas dels menús)

Per als menús emprem les classes:

- ***MenuDecoder***: Que s'encarrega de fer tota la descodificació del menú i ficar-la a una estructura de nodes
- ***MNode***: Cadascun dels nodes del menú.

Anàlogament per a la vista tenim 2 classes similars:

- ***ViewDecoder***: Que s'encarrega de fer tota la descodificació de la pantalla i ficar-la a una estructura de nodes
- ***VNode***: Cadascun dels nodes del formulari.

La classe ***Limit*** és per a trobar el primer delimitador en una tira de caràcters de entre un conjunt de caràcters que es fan servir de delimitadors.

### 3.4.5 tav.client.form.

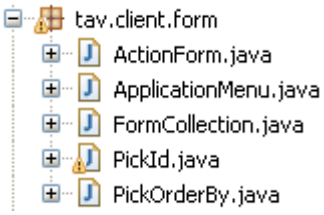



**Fig. 17: Paquet tav.client.form**

Conté classes que representen formularis menys complexos que es fan servir per les classes principals ***BaseView10*** i ***Mosaic06***.

- **ActionForm**: representa un formulari es mostra a l'usuari per a que introdueixi els paràmetres necessaris per a executar el mètode d'una classe que ací es defineix com acció.
- ***ApplicationMenú:*** Serveix per a definir el menú d'una aplicació i serà representat sobre la classe ***Mosaic06***.
- ***FormCollection***: Per a mostrar les col·leccions sobre un formulari.
- ***PickId***: Per a seleccionar la clau primària d'un registre.
- ***PickOrderBy***: Per a generar la clàusula HQL *"order by"*.

### 3.4.6 tav.client.form.buttons.

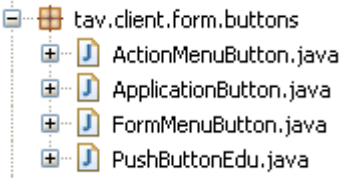



**Fig. 18: Paquet tav.client.form.buttons**

Conté botons que es fan servir a qualsevol formulari.

- ***ActionMenuButton*****:** Per a ser emprat en les accions del menú.
- ***ApplicationButton***: Per a les accions del menú general de les aplicacions.
- ***FormMenuButton***: Per als menús secundaris de les aplicacions.
- ***PushButtonEdu***: Botó general que conté una imatge.

### 3.4.7 tav.client.form.editors.

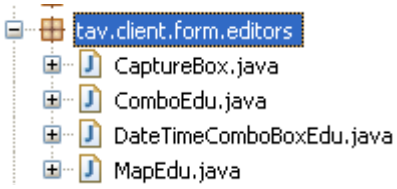



**Fig. 19: Paquet tav.client.form.editors**

Conté els editors (controls) que farem servir per a mostrar i introduir les dades en la interfície d'usuari. Implementaran les interfícies *HasName* i *HasText* de GWT.

- ***CaptureBox*****:** Similar a una llista desplegable però que s'empra per a conjunts grans de dades.
- ***ComboEdu***: És una llista desplegable que està lligada a la BD.
- ***DateTimeComboBoxEdu***: Per a seleccionar dates i hores en un calendari.

- ***MapEdu***: Control que hereta de *Google Maps* per a donar certes prestacions GIS.

### 3.4.8 tav.client.form.formactions.

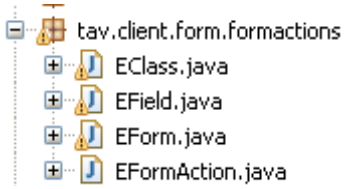



**Fig. 20:Paquet tav.client.form.formactions**

Conté les classes necessàries per a mostrar un formulari que demani a l'usuari els paràmetres necessaris per a ser passats a un mètode d'una classe i executar-lo.

- ***EClass***: Per a guardar la classe i el mètode a executar.

- ***Efield:*** Guarda informació necessària per a mostrar el control o editor que es fa falta per a que l'usuari entri la informació.
- ***Eform***: Té la informació d'un formulari i la seva col·lecció de camps ***Efield*** .
- ***EformAction***: És el conjunt del formulari ***Eform*** i la classe que conté el mètode a executar ***Eclass***.

### 3.4.9 tav.client.form.misc.

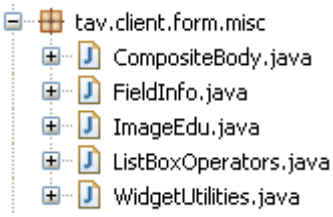



**Fig. 21: Paquet tav.client.form.misc**

Conté classes de diversa índole que s'han agrupat en un paquet miscel·lani.

- ***CompisiteBody***: És un panell que disposa del títol i es fa servir el patró ***composite***, per a que totes les pantalles principals tinguin el mateix aspecte.
- ***FieldInfo***: Guarda tota la informació que té un Atribut de una classe tal com longitud, tipus, editor(control) a emprar, etc.

- ***ImageEdu:*** Un component tipus imatge que es fa servir per als botons principalment.
- ***ListBoxOperators***: Conjunt de tots els operadors que es poden fer servir en una instrucció HSLQ a ser generada des de un control tipus ***ComboEdu***.
- ***WidgetUtilities:*** Classe que agafa la informació d'un ***FieldInfo*** i s'encarrega de mostrar el camp amb el seu editor corresponent i ficar i traure informació al editor (control). Cada vegada que es crea un editor nou, sols s'ha de modificar aquesta classe per a que tingui constància.

### 3.4.10 tav.client.form.services.


**Fig. 22: Paquet tav.client.form.services**

Conté les interfícies i la definició del proxi per al procediment de connexió RPC al servidor d'aplicacions i establir la comunicació. Cal revisar el punt 2.7.2 que explica amb detall aquest procés. Vegem dons els components que interactuïn:

- ***IformService***: Interfície que indica el tipus de crides a fer en el cas que no siguin asíncrones.
- ***IformServiceAsync***: Interfície igual a l'anterior excepte que no s'espera rebre cap paràmetre (tipus ***void***) i a més els arguments de les crides són els mateixos i se li afegeix una component tipus ***AsyncCallBack<Object>*** per a que es porti a terme

aquesta tasca de forma asíncrona.

- ***ServerService***: És crea un servei que ha de ser únic amb el patró ***Façade*** que és realment un ***proxi***. Crea una classe te tal manera que implementa les 2 interfícies anteriors i a més les relaciona amb el *servlet* (***FormServiceImp*** del servidor) fent referència a l'adreça indicada en *web.xml*.

### 3.4.11 tav.client.utility

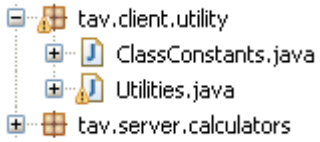



Fig. 23: Paquet tav.client.utility

Conté classes d'utilitat amb mètodes estàtics amb la fi de no usar aquestes classes i no tenir que crear instàncies. Es a dir, fem servir els mètodes com mètodes de classe i no d'objecte. Les classes a fer servir son:

- ***ClassConstants***: Conté el nom del paquet que conté les classes que són les que alimenten al framework. En concret és el paquet ***tav.server.classes***.
- ***Utilities***: Té un munt d'utilitat que es fan servir a la part del client com tractament de dates, conversió de dades, obtenir el nom d'una classe a partir del nom complet, etc.

### 3.4.12 tav.server.calculator i tav.server.validator.

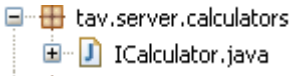



Fig. 24: Paquet tav.server.calculator.

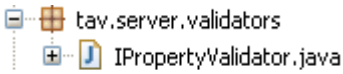



Fig. 25: Paquet tav.server.validator.

Són paquets experimentals que contenen interfícies que encara no estan desenvolupades. Són per a fer camps calculats i per a fer validacions tipificades. La idea d'aquestes classes ja segut importada del framework *OpenXava*.

### 3.4.13 tav.server.edu

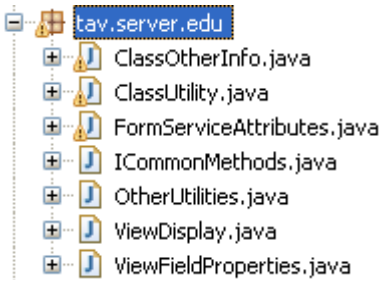



Fig. 26: Paquet tav.server.edu

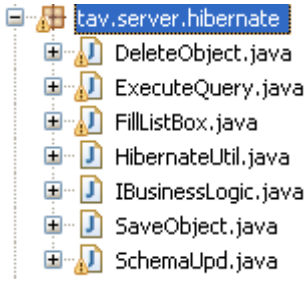
Conté un conjunt miscel·lani de classes i interfícies, per a tractar la informació de les classes que alimenten al framework. Vegem doncs aquestes:

- ***ClassOtherInfo***: En principi sols obté l'atribut que guarda la descripció de la classe. Més endavant s'implementaran altres funcionalitats. Sols emprar-se per a les llistes desplegables (***ComboEdu***).
- ***ClassUtility***: És la utilitat per excel·lència de la part del servidor..

- Extrau quasi tota la informació de la classe i la fica en classes ***FieldInfo***.
- ***FormServiceAtributes***: Conté tota la informació d'una classe (del paquet ***tav.server.classes***) que es guarda al servidor. A més conté tots els objectes obtinguts de les crides JPA d'aquesta classe i el registre actual mostrat al l'usuari. Aquesta informació es guarda en la ***sessió*** d'usuari, per a evitar actualitzacions innecessàries i evitar que l'ús de la memòria sigui més eficient.

- ***IcommonMethods***:Interfície que tenen que implementar totes les classes del paquet ***tav.server.classes*** per a que siguin emprades pel framework.
- ***OtherUtilities***: Altres utilitats per a la part del servidor. Utilitats de dates, de classes i de cerca en *arxius de recursos*.
- ***ViewDisplay***: Una super classe que tenen de la que han d'heretar totes les classes del paquet ***tav.server.classes*** en la que han de sobreescriure el mètode getViewDisplay en el cas que la classe en qüestió no sigui una classe simple (amb sols els atributs:clau, descripció, usuari i data de modificació).
- ***ViewFieldProperties***: És la classe per excel·lència que fem servir a la part del domini per a implementar les ***anotacions***, o sigui al paquet ***tav.server.classes***, per a definir valors per omissió dels atributs, editor, valors màxims i mínims, si és de sols lectura etc.

### 3.4.14 tav.server.hibernate.




Conté les classes que actuen directament sobre Hibernate. Per tant qualsevol altra classe que vulgui fer alguna actuació sobre la BD ha de fer ús d'aquestes classes, i així el nivell d'abstracció és més alt i a més fem servir la especificació de disseny inversió de dependència, permetent així canviar d'especificació JPA més fàcilment. Vegem doncs els components d'aquest paquet.

**Fig. 27: Paquet tav.server.hibernate.**

- ***DeleteObject***: Fem servir les crides de *Hibernate* per a esborrar un objecte de la BD. Abans i després de esborrar, fem servir els mètodes ***beforeBeanDelete*** i ***afterBeanDelete*** respectivament de la interfície ***IBusinessLogic*** que implementen les classes del paquet ***tav.server.classes***.
- ***ExecuteQuery***: Per a treure informació de la BD. Si afecta a tota una classe del paquet tav.server.classes,,farà servir els mètodes ***beforeBeanRetrieval*** i **afterBean-Retrieval** de la interfície ***IbusuinessLogic***, abans i després de l'extracció de cada objecte.
- ***FillListBox***: S'empra explícitament per a omplir la informació d'una llista desplegable (***ComboEdu***) mitjançant crides a *Hibernate*.
- ***HibernateUtil***: Aquest és la classe més important del paquet, ja que aquí definim:
  - La connexió al SGBD (per sota empra JDBC, per tant cal dir on estan els *drivers* de la BD i l'adreça IP i la BD a emprar l'esquema.
  - Les classes (del paquet ***tav.server.classes***) que es faran servir a la persistència (mitjançant anotacions). Per exemple per a afegir la classe GNAplicacio es farà servir la instrucció: **.addAnnotatedClass(GNAplicacio.class)**
  - Es crea un servei *Hibernate* mitjançant un patró façade.
- ***IBussinesLogic***: És una interfície que han d'implementar les classes del paquet tav.server.classes per a establir la lògica de negoci abans i després de fer qualsevol actuació a la BD.
- ***SaveObject***: Per a guardar una alta o modificació d'un objecte. Implementa els mètodes adients de la interfície ***IbussinesLogic*** per a establir també la lògica de negoci abans i després de esborrar.

- ***SchemaUpd***: Aquest mètode és per a actualitzar l'esquema de taules de la BD. Evidentment, sols s'ha de poder emprar per l'administrador de l'aplicació.

### 3.4.15 tav.server.jasper

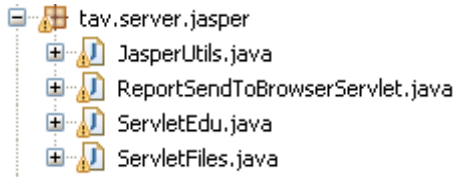



**Fig. 28: Paquet tav.server.jasper.**

Conté classes que es fan servir al generar reports mitjançant la llibreria Java *Jasper Reports*. Les classes són:

- ***JasperUtils***: Per a compilar i visualitzar un llistat i generar un fitxer a partir d'aquest llistat.
- *ReportSendToBrowserServlet*: Un *servlet* e prova per a lliurar el report al client.
- ServletEdu: Una classe de prova per a provar els *servlets*.
- ***ServletFiles***:El *servlet* que lliura un report en el format desitjat (txt, pdf, excel, etc)

### 3.4.16 tav.server.services.

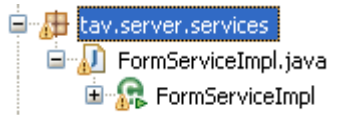



**Fig. 29: Paquet tav.server.services.**

Conté ***FormServiceImp*** que és l'únic *servlet* que s'empra a RPC, que ha de implementar la interfície ***IformService,*** i estendre la classe ***RemoteService-Servlet***. Sobre aquest servlet correrà el servei RPC i haurà una classe ***proxi*** al client que s'encarregarà de derivar el servei.

Aquesta classe te gran quantitat de mètodes, alguns són:

- Per a obtenir els valors internacionals des de l'arxiu de recursos.
- Per fe operacions bàsiques de persistència d'un objecte (altes, modificacions, consultes i baixes)
- Per a obtenir informació de les pantalles a mostrar.
- Per a obtenir missatges informatius i d'error.
- Per afer llistats.
- Per a fer crides a actualització de l'esquema de la BD.
- Extraure informació de les classes i els respectius atributs i mètodes.

# Capítol IV: Disseny orientat al model. El paquet *tav.server.classes*

Aquest és el paquet que un programador ha de manipular per a crear una aplicació de gestió. Una vegada creada o esborrada una classe, cal anar a la classe ***HibernateUtil*** per a donar-la de alta o de baixa en el servei de *Hibernate*. Analitzem primerament com es defineixen menús i formularis i després veurem cada grup de classes que s'han de revisar o crear.

## 4.1 Disseny de menús d'una aplicació.

Els menús de cada aplicació es defineixen a l'atribut ***menuDisplay*** de cada objecte de la classe ***GNAplicació***. Per tant s'ha de recuperar aquesta informació des de la gestió de persistència. És a dir, s'ha de actualitzar ja sigui directament des de la BD o com un formulari sobre la classe ***GNAplicació***. Vegem la pantalla del formulari generada pel framework per a l'aplicació de territori.

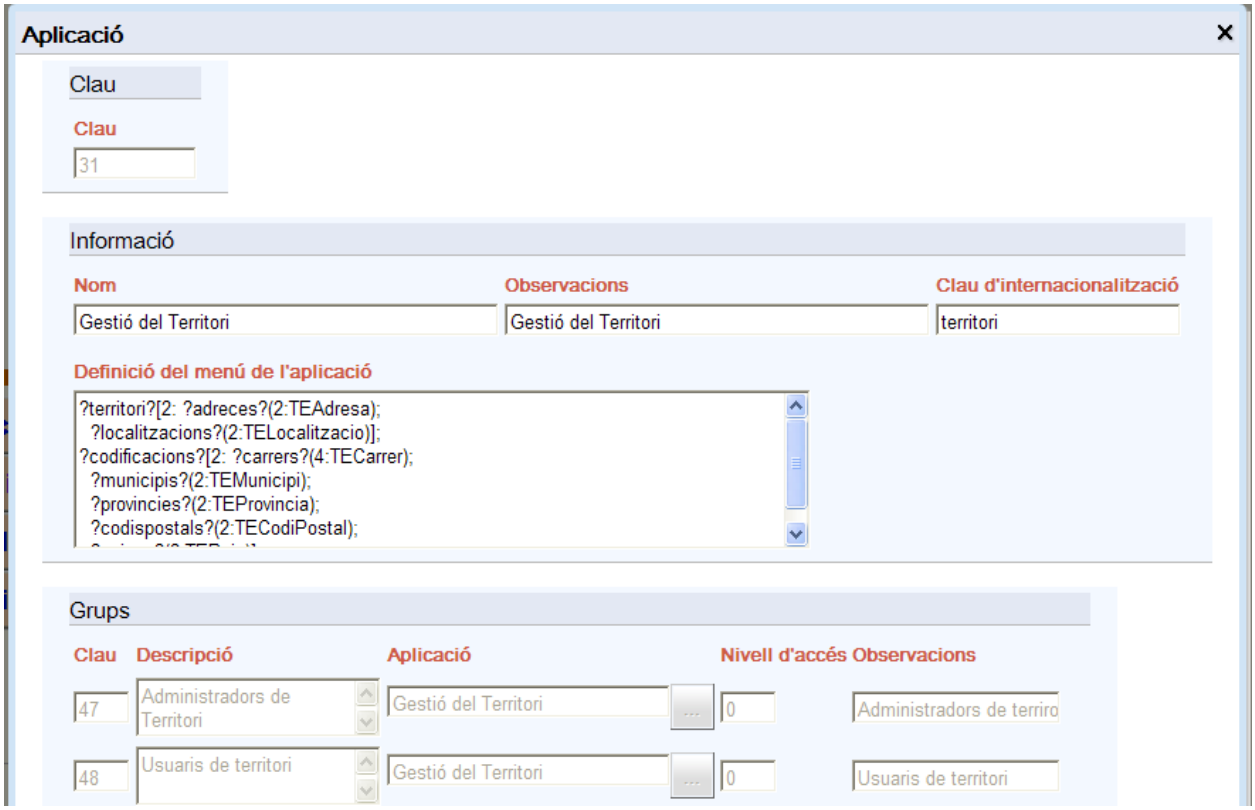



**Fig. 30: Formulari de modificació d'aplicacions.**

El que hem de destacar és el camp “Definició del menú de l'aplicació” on es mostra com s'ha definit els sistema de menús per a aquesta aplicació de territori.

Si s'arregla millor el contingut d'aquest camp (que és el atribut ***menuDisplay***) i fiquem comentaris i algun detall més, podem esbrinar com definir els menús.

```
?territori? [ 2:                                  1.-Submenú "territori" amb nivell de seguretat 2
                                                    i conté 2 formularis i 1 comandament:
        ?adreces? ( 2: TEAdresa :                 1.1.-Formulari "adreces" de nivell 2 de seguretat
                id>3                                que treballa sobre la classe TEAdreça i a més sols
         );                                         mostrarem els registres que la clau sigui major que 3

        ?localitzacions?( 2:TELocalitzacio)       1.2.- Formulari "localitzacions" de nivell 2 de seguretat
                                                    que treballa sobre la classe TELocalització.
        ?llistar_adreces_locals? { 4:             1.3- Comandament "llistar_adreces_locals"de nivell 4 de
                TEAdresa:llistar@                   seguretat, que executa el mètode llistar de la classe
                1 @ 2@ No                           TEAdresa, i agafa 3 paràmetres de valor: 1,2 i No
        }
];                                                Fi del submenú "territori".

?codificacions?[ 2:                               2.- Submenú "codificacions" amb nivell 2 de seguretat
                                                    i conté 5 formularis:
        ?carrers? ( 4: TECarrer) ;                2.1.- Formulari "carrers" amb nivell 4 de seguretat
                                                    que treballa sobre la classe TECarrer.
        ?municipis? ( 2: TEMunicipi);             2.2.- Formulari "municipis" amb nivell 2 de seguretat
                                                    que treballa sobre la classe TEMunicipi
        ?provincies?( 2: TEProvincia);            2.3.- Formulari "provincies" amb nivell 2 de seguretat
                                                    que treballa amb la classe TEProvincia.
        ?codispostals? ( 2: TECodiPostal);        2.4- Formulari "codispostals" amb nivell 2 de seguretat
                                                    que treballa amb la classe TECodiPostal.
        ?paisos?( 2: TEPais)                      2.5.- Formulari "paisos" amb nivell 2 de seguretat
                                                    que treballa amb la classe TEPais.
];                                                Fi del submenú "codificacions".
```

**Fig. 31:Com fer un menú d'aplicació.**

**Definició de submenús:**

- **(1) Títol del submenú entre "??":** per exemple **"?*territori*?"**. El valor entre interrogacions és una clau per a consultar ***l'arxiu de recursos,*** per tant cal fer servir preferentment ***Netbeans*** per a donar d'alta cadascuna de les referències als idiomes de treball.

- **(2) Claudàtors "[ ]" per a indicar l'abast del submenú** (al igual que s'empra les claus "{}" en Java per a indicar l'abast d'un procediment.

- **(3) Nivell mínim de seguretat** que ha de tenir l'usuari per a poder entrar al submenú, i després hem de ficar dos punts "**:**" de separador. Per exemple "***? submenu1?[2:];***" indica que el submenú "*?submenu1?*" té un nivell mínim de seguretat de 2, i no té cap element associat.

- **(4) Llista de formularis i accions de submenú,** separats per punt i coma "**;**". ara s'indicarà com definir formularis i accions de submenú.

- **Definició de formularis:**

- **(1) Títol del formulari entre "??":** per exemple **"?*carrers*?"**. El valor entre interrogacions és una clau per a consultar ***l'arxiu de recursos,***

- **(2) Parèntesis "( )" per a indicar l'abast del formulari** (al igual que s'empra les claus "{}" en Java per a indicar l'abast d'un procediment.

- **(3) Nivell mínim de seguretat** que ha de tenir l'usuari per a poder entrar al formulari, i després separem de la resta d'informació amb dos punts "**:**". Per exemple "**?formulari1?[5: Classe1:id<100];**" indica que el formulari "*?formulari1?*" té un nivell mínim de seguretat de 5, i treballa sobre la classe Classe1 i sols en aquells objectes de la classe que l'atribut id sigui menor de 100.

- (**4) Nom de la classe sobre la que treballa**, que ha de ser una classe del paquet t***av.server.classes.*** Si a continuació i ha més informació es ficaran 2 punts de separació "**:**".

- **(5) Clàusula where de HQL (opcional)**.

- ***Definició d'accions (comandament) de menú:***

  - **(1) Títol del comandament entre "??":** per exemple **"?*carrers*?"**. El valor entre interrogacions és una clau per a consultar ***l'arxiu de recursos,***

  - **(2) Claus "{ }" per a indicar l'abast del comandament** (al igual que s'empra les claus "{}" en Java per a indicar l'abast d'un procediment.

  - **(3) Nivell mínim de seguretat** que ha de tenir l'usuari per a poder entrar al formulari, i després separem de la resta d'informació amb dos punts "**:**". Per exemple "**?esborrar?[5: Classe1:esborra@3@ABCDEFG];**" indica que el comandament "*?esborrar?*" té un nivell mínim de seguretat de 5, i executarà el mètode esborra de la Classe1 i li passarà 2 paràmetres, el primer és "3" i el segon és "ABCDEFG".

  - (**4) Nom de la classe sobre la que treballa**, que ha de ser una classe del paquet t***av.server.classes.*** Si a continuació i ha més informació es ficaran 2 punts de separació "**:**".

  - **(5) Nom del mètode de la classe a executar.**

  - (**6) Relació de paràmetres** separats per "**@**" que precisa el mètode per a ser executat.

La pantalla de la figura 30 generarà un menú de la classe territori que es mostra parcialment a la figura següent:

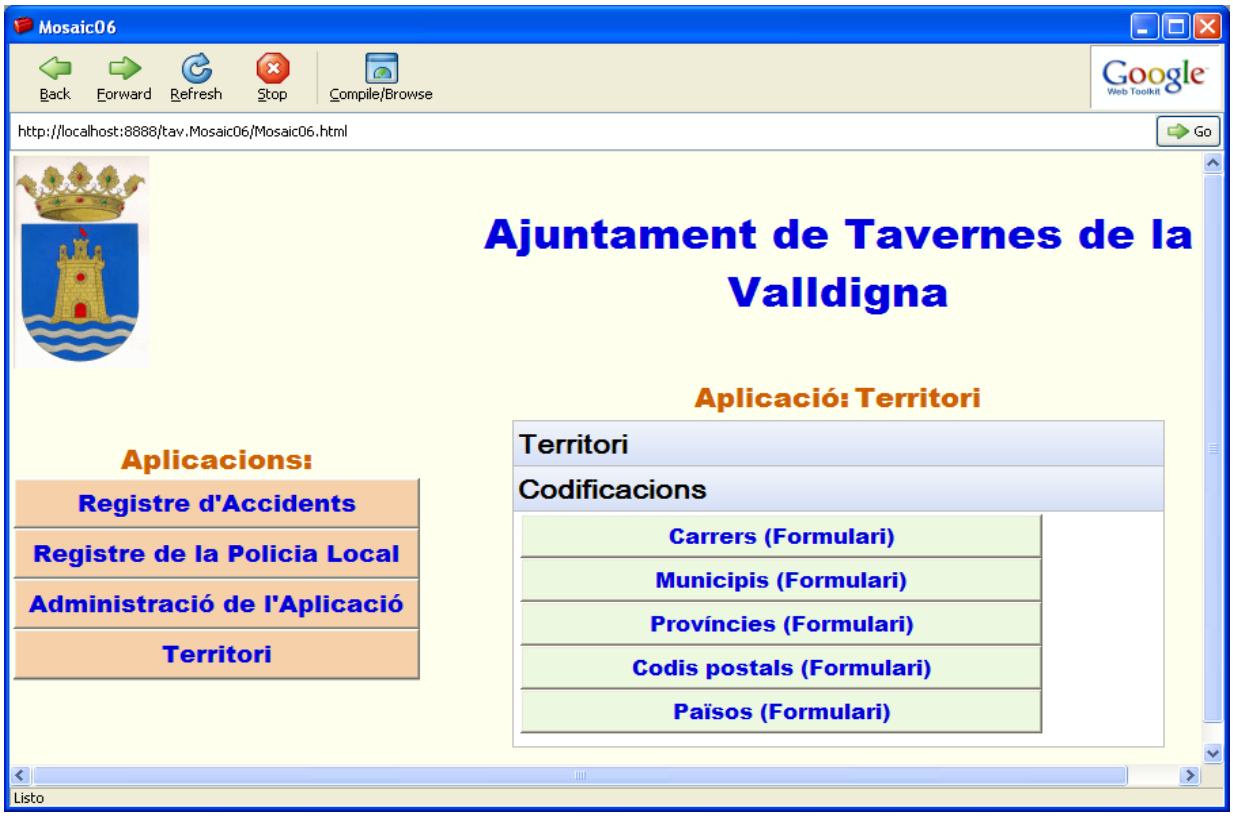



Fig. 32: Vista del menú generat.

## 4.2 Disseny de la vista (formularis).

El disseny d'un formulari (Vista) es fa mitjançant el mètode de classe ***getViewDisplay*** de cada classe del paquet ***tav.server.classes***. Per tant s'ha d'emprar un entorn IDE com *l'Eclipse* tant per a dissenyar la classe com per a dissenyar la vista. Vegem un exemple d'aquest mètode de la classe ***GNAplicacio***: al qual s'han afegit comentaris per a mostrar mes claredat.

```
public static String getViewDisplay() {
   String viewDisplay=
         // 1.- Fem un panell de títol "clau" amb l'atribut id. El següent element anirà a altra línia
         "¿clau? [ id ];" +

         // 2.- Fem un panell de títol "informació" i atributs: nom, observacions o i18nkey a la mateixa línia
         //     l'atribut menuDisplay anirà a la següent línia.
         "¿informacio? [ nom, observacions, i18nKey;  menuDisplay];" +

         // 3.- A la següent línia fem un panell de títol "grups" d'una col·lecció d'objectes GNGrup, que serà
         //     copiable, en una línia mostra els atributs: id, descripció, aplicació, nivellAcces i observacions
         "¿grups? ( GNGrup: Y: id, descripcio, aplicacio, nivellAcces, observacions);" +

         // 4.- A la següent línia fem un panell de títol "registre" i amb 2 atributs: usuari i dataModificacio
         "¿registre? [ usuari, dataModificacio]";
   return viewDisplay;}
```

Fig. 33: Mètode getViewDisplay de la classe GNAplicació.

Els resultats es poden veure a la figura 30, de la qual hem retallat l'últim panell(el de títol "registre"). Com es pot veure aquest llenguatge és molt similar al de fer menús. Fem una descripció completa:

- **Definició de panells:**
  - **(1) Títol del panell entre "¿?":** per exemple **"*¿informació*?"**. El valor entre interrogacions és una clau per a consultar ***l'arxiu de recursos,*** per tant cal fer servir preferentment ***Netbeans*** per a donar d'alta cadascuna de les referències als idiomes de treball.
  - **(2) Claudàtors "[ ]" per a indicar l'abast del panell** (al igual que s'empra les claus "{}" en Java per a indicar l'abast d'un procediment.
  - **(3) Llista d'atributs de la classe a mostrar** separats per punt i coma "**;**" si volem provocar un salt de línia, o fiquem coma "**,**" si volem que vagin a la mateixa línia. Exemple "***¿informacio?[nom, observacions; menuDisplay];***" tindrem a la mateixa línia els atributs nom i observacions i a la següent l'atribut menuDisplay. A més el següent element a mostrar després del panell aniria a la següent línia ja que després del claudàtor "**]**" hi ha un punt i coma "**;**".

- **Definició de fitxes desplegables (tabs):** S'empra en pantalles que tenen molta informació, i quan s'activa una fitxa, esta queda visible i oculta a les demés. És igual al cas dels panells però canviant els claudàtors per claus. Encara no ha segut implementat a l'hora de fer la memòria.
  - **(1) Títol de la fitxa entre "¿?":** per exemple **"*¿informació*?"**. El valor entre interrogacions és una clau per a consultar ***l'arxiu de recursos,*** per tant cal fer servir preferentment ***Netbeans*** per a donar d'alta cadascuna de les referències als idiomes de treball.
  - **(2) Claus "{ }" per a indicar l'abast de la fitxa** (al igual que s'empra les claus "{}" en Java per a indicar l'abast d'un procediment.
  - (3) **Llista d'atributs de la classe a mostrar** separats per punt i coma "**;**" si volem provocar un salt de línia, o fiquem coma "**,**" si volem que vagin a la mateixa línia. Exemple "*¿grups?{nom, observacions; menuDisplay},*" tindrem a la mateixa línia els atributs nom i observacions i a la següent l'atribut menuDisplay.

- **Definició de col·leccions:** S'empra per a veure els elements d'una col·lecció.
  - **(1) Títol de la col·lecció entre "¿?":** per exemple **"*¿grups*?"**. El valor entre interrogacions és una clau per a consultar ***l'arxiu de recursos,*** per tant cal fer servir preferentment ***Netbeans*** per a donar d'alta cadascuna de les referències als idiomes de treball.
  - **(2) Parèntesis "( )" per a indicar l'abast de la col·lecció** (al igual que s'empra les claus "{}" en Java per a indicar l'abast d'un procediment.
  - **(3) Nom de la classe** que conforma la col·lecció, i 2 punts de separador "**:**".

- **(4) Si es copiable (Y/N)** la col·lecció al fer una “alta copiant”, i 2 punts de separador “**:**”.

- **(5) Llista d'atributs de la classe a mostrar** separats per punt i coma “**;**” si volem provocar un salt de línia, o fiquem coma “**,**” si volem que vagin a la mateixa línia. Exemple “***¿igrups?{ GNGrups: Y: id, descripcio};***” tindrem una col·lecció de títol grups que afecta a la classe GNGrups, que es copiable i sols mostrem els atributs id i descripció a la mateixa línia.

## 4.3 Altres especificacions mitjançant anotacions pròpies.

La classe ***ViewFieldProperties*** és la que ens permet definir anotacions importants, com les següents:

- ***ReadOnly***: Si es de sols lectura.
- ***Visible***: Si es pot veure o no el camp.
- ***DefaultValue***: Valor per omissió.
- ***Editor***: Control a emprar per a la seva edició. Ha d'implementar les interfícies de GWT hasText i hasName.
- ***IsCounter:*** Si es comptador (per al comptador del registre).
- ***GroupCounterMembers:*** Agrupació de comptador (cada any es reinicia el comptador)

Hi ha altres que encara no estan implementades i són per a tractar esdeveniments com canvis de valor del camp, agafament i pèrdua del focus etc.

## 4.4 Administració: Aplicacions, usuaris i grups.

Són al paquet ***tav.server.classes***, les classes encarregades i tenen el prefix GN:

- ***GNAplicació***: Les aplicacions es guarden en cada objecte d'aquesta classe i:
  - L'atribut ***i18nKey*** és l'encarregat d'enregistrar la clau per a obtenir el títol de l'aplicació consultant a ***l'arxiu de recursos,*** Per tant cal fer servir preferentment **Netbeans** per a donar d'alta cadascuna de les referències als idiomes de treball.
  - A més és l'encarregada de guardar la informació dels menús de les aplicacions a l'atribut ***menuDisplay***, mitjançant un petit llenguatge de símbols senzill explicat anteriorment. A la figura 30 es mostra com crear l'aplicació de territori.

- ***GNGrup****:* Definim els grups associats a cada aplicació, amb els seus permisos.
- ***GNUsuari***: Definim els usuaris generals, amb independència de l'aplicació.
- ***GNUsuariGrup***: Definim els usuaris que integren un grup,i per tant adquireixen els permisos del grup per a l'ús de l'aplicació.

# Capítol V: Anàlisi de l'aplicació de Registre E/S de la Policia Local.

## 5.1 Definició del model de negoci

El model de negoci defineix molt somerament els processos i entitats mes destacables. Aquest model, un tant simplificat seria:

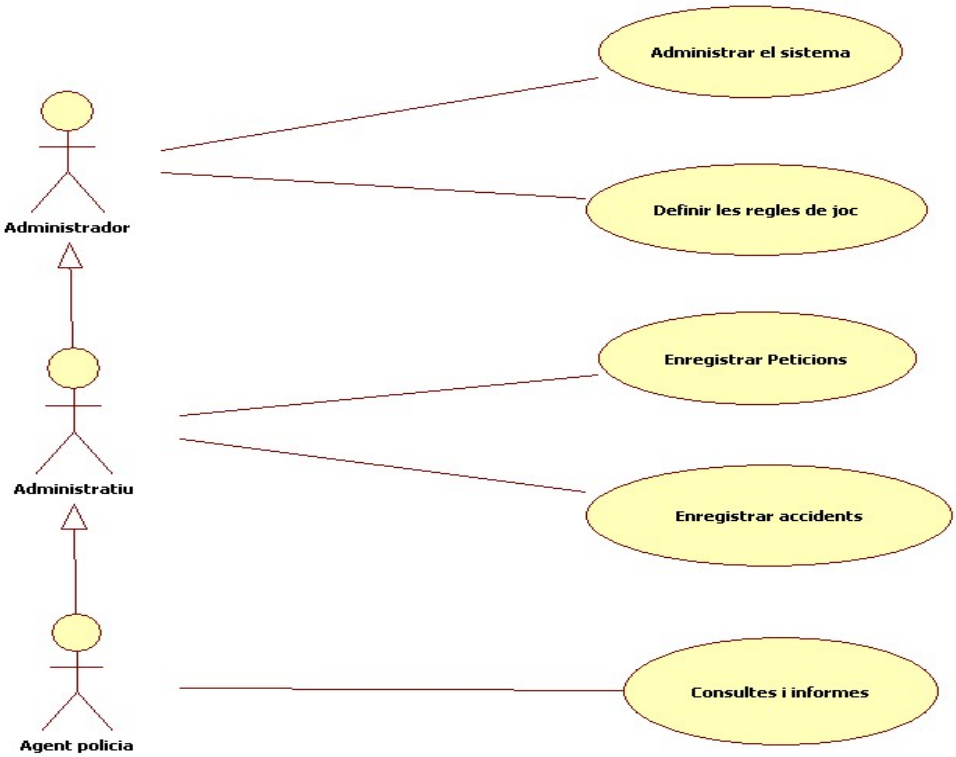



**Fig. 34: Model de negoci simplificat (casos d'ús)**

## 5.2 Els guions

Els guions són les operacions mes habituals que els usuaris realitzen i que són la base per a identificar el casos d'us. Vegem-los en funció dels diferents tipus d'usuaris:

- **Administrador**: Identificar-se i entrar al sistema (*login*), manteniment d'usuaris, grups i rols, manteniment dels subsistemes de codificacions y regles de joc. A mes si escau podrà realitzar feina administrativa, com els següents usuaris:
- **Administratiu**: Identificar-se i entrar al sistema (*login*), enregistrar peticions (registre), i accidents, definir localitzacions dels fets, gestionar persones, tramitació, documentació.
- **Agent policia**: Identificar-se i entrar al sistema (*login*), fer informes d'accidents i altres successos, consulta de registre i accidents.

## 5.3 Identificació dels actors

Els actors són elements externs al sistema i que interactuïnn amb ell per a l'entrada i sortida de dades. Solen estar associats a persones, i mes concretament als usuaris. No cal enumerar quins seran els actors, sinó identificar els tipus d'actors bàsics. Vegem-los:

- **Agent de policia**: És l'usuari del sistema que té les funcions més limitades, ja que és un element d'actuació i no de gestió.
- **Administratiu**: És el que més va a tenir contacte amb l'aplicació i el que més informació subministrarà al sistema.
- **Administrador**: Serà el que establirà el que pot fer cada usuari, o sigui, serà l'encarregat d'administrar l'aplicació.

## 5.4 Identificació dels casos d'ús

En primer lloc, s'ha de tenir clara la definició de cas d'ús:

- Són les funcions del sistema amb els seus *inputs* i *outputs* clarament definits.
- Descriuen clarament l'objecte d'aquestes funcions (***que fan?***) però no la implementació (***com ho fan?***)
- Són processos que es caracteritzen per la seva l'autonomia. Sols poden ser engegats per un actor o per altre cas d'ús.
- Són un dels objectius bàsics dels jocs de proves de l'aplicació, en concret del tipus capsa negra.

Donat que aquesta aplicació és molt extensa, sols s'indicaran els casos d'us de manera simplificada. Com ja s'ha dit, s'emparà la paraula *manteniment* per al conjunt d'operacions *alta - baixa – modificació*. També s'emprarà la paraula *gestionar* per a indicar les operacions bàsiques de un subsistema. Els casos d'ús mes importants a destacar són:

- *Login*.
- Manteniment d'usuaris.
- Manteniment de grups.
- Manteniment de rols.
- Gestionar registre.
- Gestionar accidents.
- Gestionar localitzacions.
- Gestió de taules bàsiques.
- Gestió de persones i altres taules.
- Gestió de tramitació.
- Realització de parts i informes.
- Consulta al registre.
- Consulta als accidents.
- Consulta als tràmits.

### 5.4.1 Identificació de les relacions de casos d'ús.

Els casos d'ús poden estar relacionats amb 3 tipus de relacions que es detallen:

- **Relacions d'extensió**: Aquesta relació indica que el comportament del cas d'ús extensió pot ser inserit en el cas d'ús extingit sota determinades condicions. Aquestes relacions s'han aplicat principalment als casos d'ús de consultes.
- **Relacions d'inclusió**: Aquestes relacions indiquen que un cas d'ús pot incloure a un altre. Aquesta relació és molt útil per a extraure comportament comuns, i distingir de manera clara els comportaments de cada cas d'ús. L'exemple més significatiu és el del

cas d'ús login que és un prerequisit per a poder fer qualsevol altre cas d'ús.

- **Relació de generalització / especialització**: Es presenta quan un cas d'ús és una forma especialitzada d'un altre existent, concepte similar al concepte d'orientació a objectes de *subclasse*. En el nostre diagrama no hi ha cap definida.

### 5.4.2 Diagrama de casos d'ús.

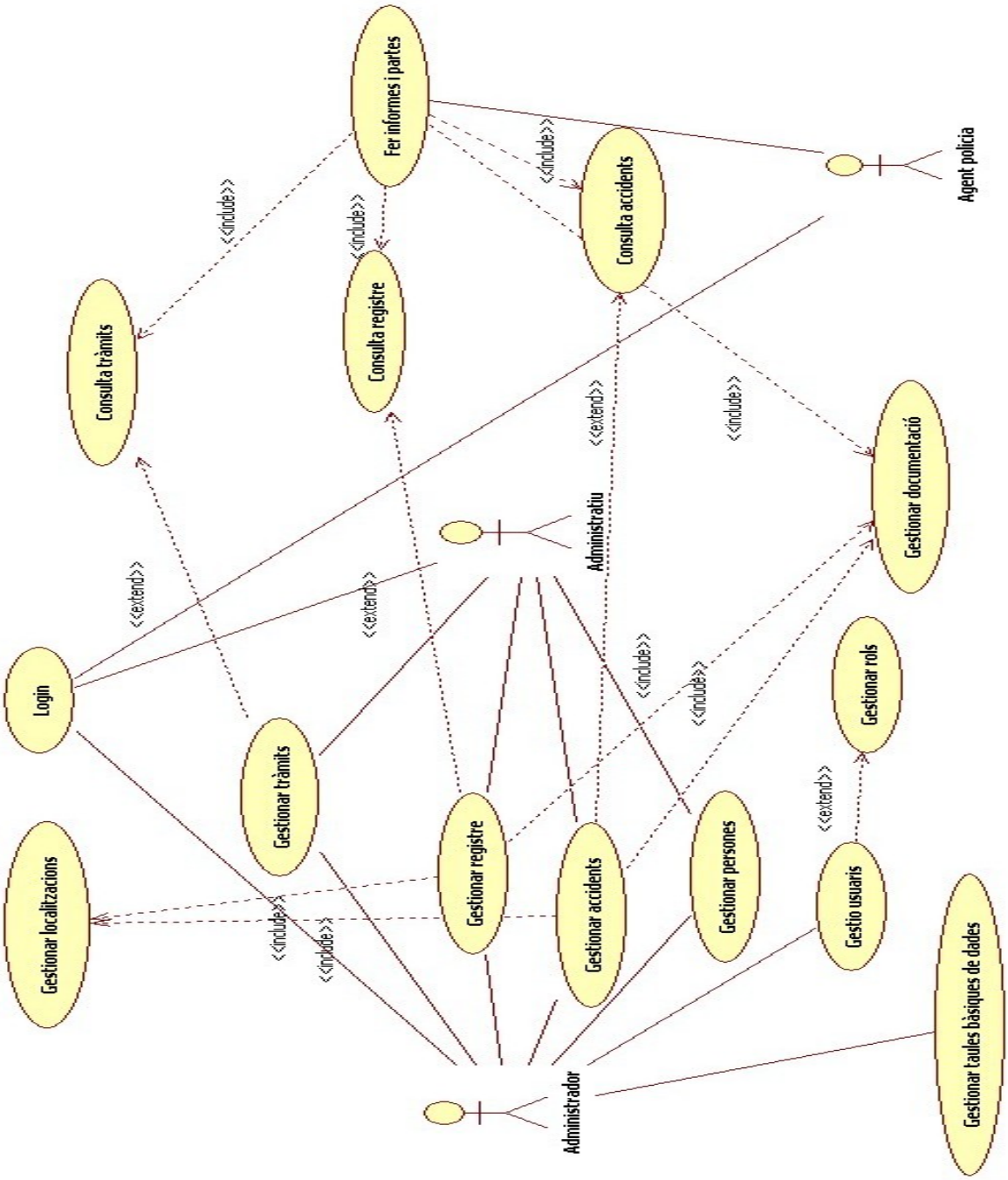



**Fig. 35: Diagrama de casos d'ús.**

### 5.4.3 Documentació dels casos d'ús.

Vegem alguns dels casos d'ús mes importants. **Ara ja no té sentit definir el cas d'ús manteniment, ja que engloba a 3 casos d'ús que tenen particularitats molt diferents**, per tant s'han de considerar cada cas d'ús en detall.

| **Cas d'ús** | **Login** |
|---|---|
| **Objecte** | Identificació dels usuaris per a aconseguir l'accés al sistema. |
| **Actors** | Administrador, Administratiu, Agent de policia |
| **Casos relacionats** | Qualsevol funcionalitat ha de passar per aquest cas d'ús. |
| **Precondició** | L'usuari ha d'estar donat d'alta. |
| **Postcondició** | L'usuari queda identificat al sistema i per tant està en disposició d'efectuar les accions pròpies amb acord als permisos assignats. |
| **Descripció:** | L'usuari subministrarà la signatura electrònica (o alternativament les paraules d'usuari i de pas) . A continuació el sistema validarà les seves dades i li ficarà al seu abast un menú amb les opcions que te disponibles. |
| **Observacions** | És el primer que ha de fer un usuari, abans de fer altra acció. |

Ara es detalla cadascun dels casos d'ús relacionats amb la tasca *manteniment d'usuaris*.

| **Cas d'ús** | **Alta d'un usuari** |
|---|---|
| **Objecte** | L'administrador donarà d'alta aquells usuaris que entre les seves tasques de treball estigui el fer ús d'aquesta aplicació. |
| **Actors** | Administrador |
| **Casos relacionats** | Login , Baixa d'usuaris, Modificació d'usuaris |
| **Precondició** | L'usuari no ha d'estar prèviament donat d'alta. L'Administrador ha d'estar registrat i identificat. |
| **Postcondició** | L'usuari queda donat d'alta del sistema |
| **Descripció:** | L'Administrador elegeix l'opció de fer altes del *submenú* usuaris. Emplena les dades i les enregistra al sistema. |

| | |
|---|---|
| **Observacions** | Un usuari no pot accedir al sistema si prèviament no ha segut enregistrat per l'Administrador. A més per al tema de control d'accessos de la Llei de Protecció de Dades, és imperatiu legal tenir enregistrats cadascun dels accessos realitzats per cada usuari. |

| **Cas d'ús** | **Baixa d'un usuari** |
|---|---|
| **Objecte** | L'administrador donarà de baixa aquells usuaris que de forma definitiva o temporal queden apartats de tasques relacionades amb el programari. |
| **Actors** | Administrador |
| **Casos relacionats** | Login, Alta d'usuaris, Modificació d'usuaris |
| **Precondició** | L'usuari ha d'estar prèviament donat d'alta. L'Administrador ha d'estar registrat i identificat. |
| **Postcondició** | L'usuari queda donat de baixa del sistema |
| **Descripció:** | L'Administrador elegeix l'opció de fer baixes del *submenú* usuaris. A continuació escull aquell usuari que va a ser donat de baixa del sistema. |
| **Observacions** | Els agents de policia presenten una alta mobilitat entre ajuntaments. A mes si aquesta aplicació es pot emprar des d'Internet, s'ha de tenir una política clara de baixes d'usuaris de l'aplicació ja que es tracten dades d'alt grau de protecció segons la llei, per tal d'evitar accessos no desitjats. |

| **Cas d'ús** | **Modificació d'un usuari** |
|---|---|
| **Objecte** | L'administrador modificarà les dades bàsiques d'un usuari. |
| **Actors** | Administrador |
| **Casos relacionats** | Login, Alta d'usuaris, Baixa d'usuaris |
| **Precondició** | L'usuari ha d'estar prèviament donat d'alta. L'Administrador ha d'estar registrat i identificat. |
| **Postcondició** | L'usuari queda donat de baixa del sistema |
| **Descripció:** | L'Administrador elegeix l'opció de fer modificacions del *submenú* usuaris. A |

| | |
|---|---|
| | continuació escull aquell usuari que va a ser modificat i realitza les esmenes adients. |
| **Observacions** | Segons la Llei de Protecció de Dades, la informació ha de ser precisa. A part de ser un imperatiu legal, a ningú li interessa tenir dades que no son reals. |

Pel que respecta al tema *manteniment de grups*, amb els casos d'ús particulars de alta, baixa, modificació de grups, assignació i desassignació de grups a usuaris; per simplificar sols es descriurà l'opció d'assignació de un usuari a un grup.

| **Cas d'ús** | **Assignació d'un usuari a un grup** |
|---|---|
| **Objecte** | L'administrador assignarà un usuari a un grup, i a les hores, aquest usuari aconseguirà permisos per a tenir els rols assignats a cada grup. |
| **Actors** | Administrador |
| **Casos relacionats** | Login , Alta d'usuaris, Baixa d'usuaris, Modificació d'usuaris, Alta de grups, Baixa de Grups, Modificació de Grups, Assignació de rols a grups, .. |
| **Precondició** | L'usuari ha d'estar donat d'alta. El grup ha d'estar donat d'alta. L'usuari no ha de tenir assignat aquest grup.. |
| **Postcondició** | L'usuari queda assignat a un grup. |
| **Descripció:** | L'Administrador elegeix l'opció de assignació de grups a l'usuari del *submenú* grups. Selecciona el grup en qüestió i busca l'usuari que ha de ser assignat. |
| **Observacions** | Amb aquesta funcionalitat es simplifica molt el procés d'assignació de permisos als usuaris per a realitzar les tasques concretes. Un usuari pot ser assignat a diferents grups, amb la qual cosa agafa permís per a obtenir les funcionalitats assignades a cada grup. |

Per a finalitzar s'indicarà sols els casos d'ús enregistrar una petició i enregistrar un accident, que són els eixos centrals de l'aplicació:

| **Cas d'ús** | **Enregistrar una petició** |
|---|---|
| **Objecte** | L'administratiu enregistrarà una petició de servei que replsigui per telèfon d'un particular, a través del 112, per escrit o altres fonts d'entrada. Ha de tenir dades suficients com el lloc on s'ha de fer l'actuació, la data i hora, els agents mobilitzats, documentació aportada i ha d'obrir un expedient. |
| **Actors** | Administratiu |
| **Casos relacionats** | Login , Manteniment de localitzacions, Manteniment de les regles de joc, |

| | |
|---|---|
| | Manteniment de taules bàsiques, Manteniment de Localitzacions .. |
| **Precondició** | Ha d'estar donat d'alta l'administratiu i ha de tenir estar inclòs en el grup o grups que tinguin el rol d'enregistrar peticions. Han d'estar donades d'alta els registres corresponents de les taules bàsiques de codificacions per al tipus de petició a enregistrar (tipus de procedència, tipus de document, tipus d'afectat, tipus de documentació etc.), a més s'ha de produir una petició d'actuació que cal enregistrar, i aquesta petició pot tenir una localització per a l'actuació. |
| **Postcondició** | La petició queda enregistrada, tant mateix s'enregistra la localització on fer l'actuació i s'obri si escau un expedient per a fer el seguiment i tramitació. |
| **Descripció:** | L'Administratiu elegeix l'opció de enregistrar una petició del *submenú* "registre". Mitjançant llistes desplegables selecciona aquelles codificacions per a definir de forma precisa el tipus de petició, procedència, assignació, a més s'introduirà un extracte de la petició, observacions, data i hora de la petició, el nombre de registre general d'entrada si escau; també, amb un control tipus *Google Maps* es definirà la localització exacta dels fets, i també, la documentació aportada. |
| **Observacions** | Una vegada enregistrada, pot donar pas a un expedient (depenent del tipus de petició). Ja hi ha altres casos d'ús relacionats amb el seguiment d'expedients. |

| **Cas d'ús** | **Enregistrar un accident** |
|---|---|
| **Objecte** | L'administratiu enregistrarà l'actuació realitzada com a conseqüència d'un accident (normalment de transit). |
| **Actors** | Administratiu |
| **Casos relacionats** | Login , Manteniment de localitzacions, de les regles de joc, de conceptes,manteniment de Localitzacions, Enregistrar peticions .. |
| **Precondició** | Ha d'estar donat d'alta l'administratiu i ha de tenir estar inclòs en el grup o grups que tinguin el rol d'enregistrar peticions. Han d'estar donades d'alta els registres corresponents de les taules bàsiques relacionades, s'ha de tenir la relació d'afectats i danys produïts, s'ha de saber la localització, i com es produïren els fets. Han d'existir informes dels agents i metges desplaçats. |
| **Postcondició** | L'accident queda enregistrat, i també la localització on s'ha produït l'accident i s'obri si escau un expedient per a fer el seguiment i tramitació oportuna. |
| **Descripció:** | L'Administratiu elegeix l'opció de enregistrar un accident del *submenú* "accidents". Mitjançant llistes desplegables selecciona aquelles codificacions |

| | |
|---|---|
| | per a definir de forma precisa el tipus d'accident, un extracte de com s'ha produït l'accident, observacions, data i hora, relació de persones i elements afectats; també, amb un control tipus *Google Maps* es definirà la localització exacta dels fets, i també, s'adjuntarà les *URL* de la documentació aportada. |
| **Observacions** | Un accident és un fet que involucra responsabilitats, i per tant origina un expedient que ha de tenir molta mobilitat, ja que sol intervenir els jutjats. És doncs molt important tenir controlada molta documentació que es demanda per part dels jutges i assegurances, i al mateix temps servirà aquesta documentació per a fer estadístiques de *punts negres* del terme municipal que poden requerir actuacions urbanístiques per a eliminar-los. . |

## 5.5 Diagrama d'estats.

El diagrama de estats ens mostra el comportament d'un objecte. A tal efecte ens indica la relació d'estats per on passa l'objecte al llarg de la seva vida. En concret per al cas del registre de peticions es té:

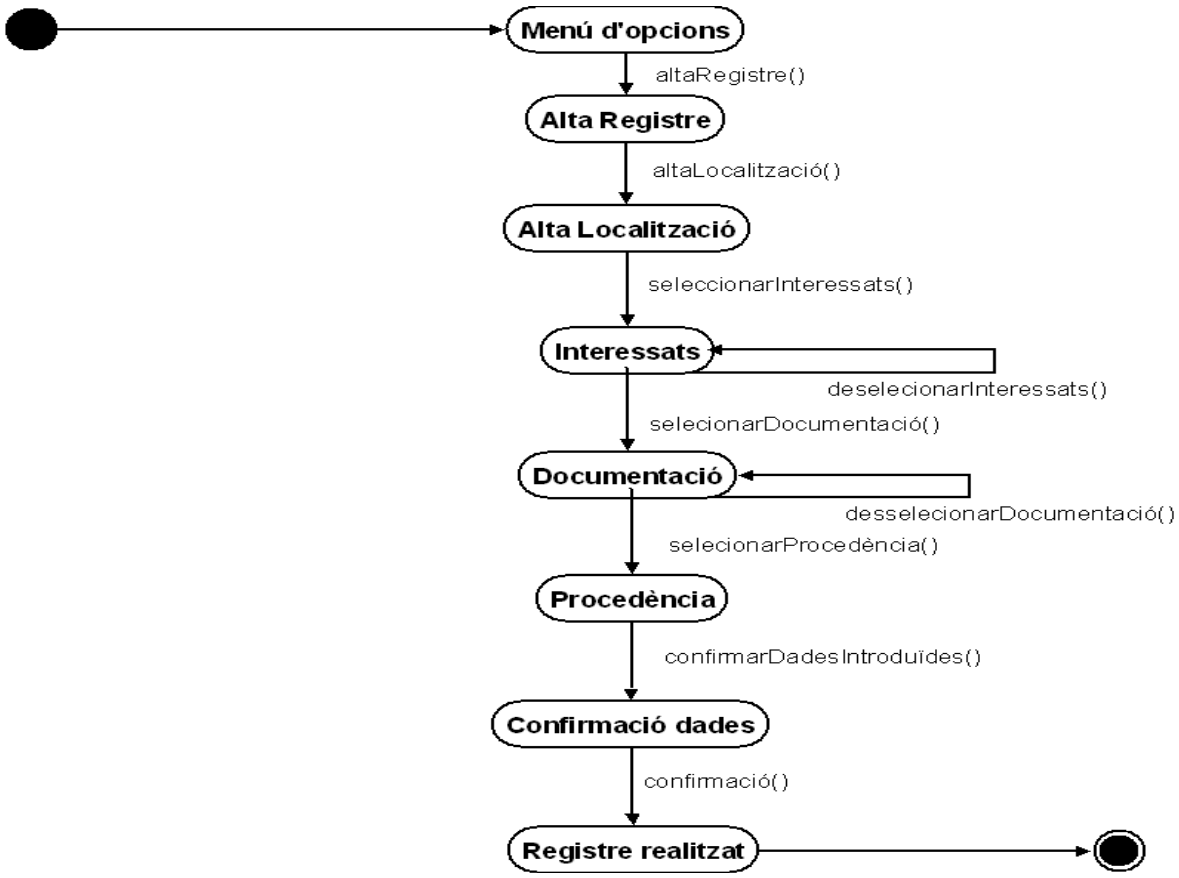



**Fig. 36: Diagrama d'estats.**

## 5.6 Diagrama de paquets

Si no considerem els elements del framework per motiu de simplicitat i claredat, podem establir el següent diagrama de paquets. Un dels objectius és desenvolupar un petit *framework* per a generar la part de la Vista del model MVC. Però, per motius de simplicitat d'idees, no es convenient incloure'l dins del diagrama de paquets, ja que tal vegada és més convenient definir-lo a la fase de disseny i no pas aquesta d'anàlisi.

Per tant es considerarà un únic paquet d'anàlisi que és l'aplicació en si de "Registre de la Policia Local", i es consideraran diferents paquets de servei, en base a les diferents funcionalitats de l'aplicació. Vegem doncs el diagrama de desplegament:

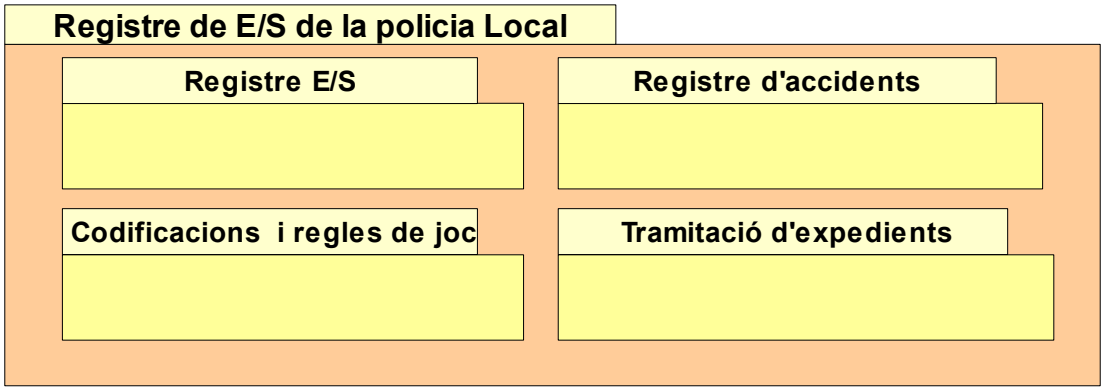



**Fig. 37: Diagrama de paquets.**

## 5.7 Identificació de les classes

Com ja se sap, les classes es poden classificar en funció del seu paper dins de l'aplicació. Per tant es te:

- **Classes de frontera**: Son les encarregades de comunicar-se amb els actors, per tant ens indiquen la frontera del sistema. Contenen la interfície d'usuari, (pantalles d'introducció de dades, menús, etc.).
- **Classes d'entitats**: Contenen els objectes bàsics de modelat del procés a informatitzar. Són candidats al procés de persistència, i tenen mobilitat (són la informació que flueix) per les classes de frontera i control.
- **Classes de control**: Són les encarregades de establir el flux d'informació entre els diferents rutes dins de l'aplicació.

Com es pot apreciar hi ha una certa similitud d'aquest plantejament amb el del patró **Model – Vista - Controlador** (les classes de frontera són semblants en el concepte de vista, les classes d'entitat tenen certes similituds amb el model, i les classes de control per la seva part tenen aspectes del control).

El nostre objectiu en aquesta fase d'anàlisi es centra en les classes d'entitat, ja que les classes de frontera i control es veuen en la fase de disseny.

### 5.7.1 Diagrames de col·laboració.

Mitjançant els diagrames de col·laboració, es pot veure com interactuïn entre si les diferents classes. Es una forma d'identificar les classes de frontera, control i entitat. Per motius de temps, sols s'inclourà el diagrama del cas d'ús inserir una petició al registre de Policia Local.

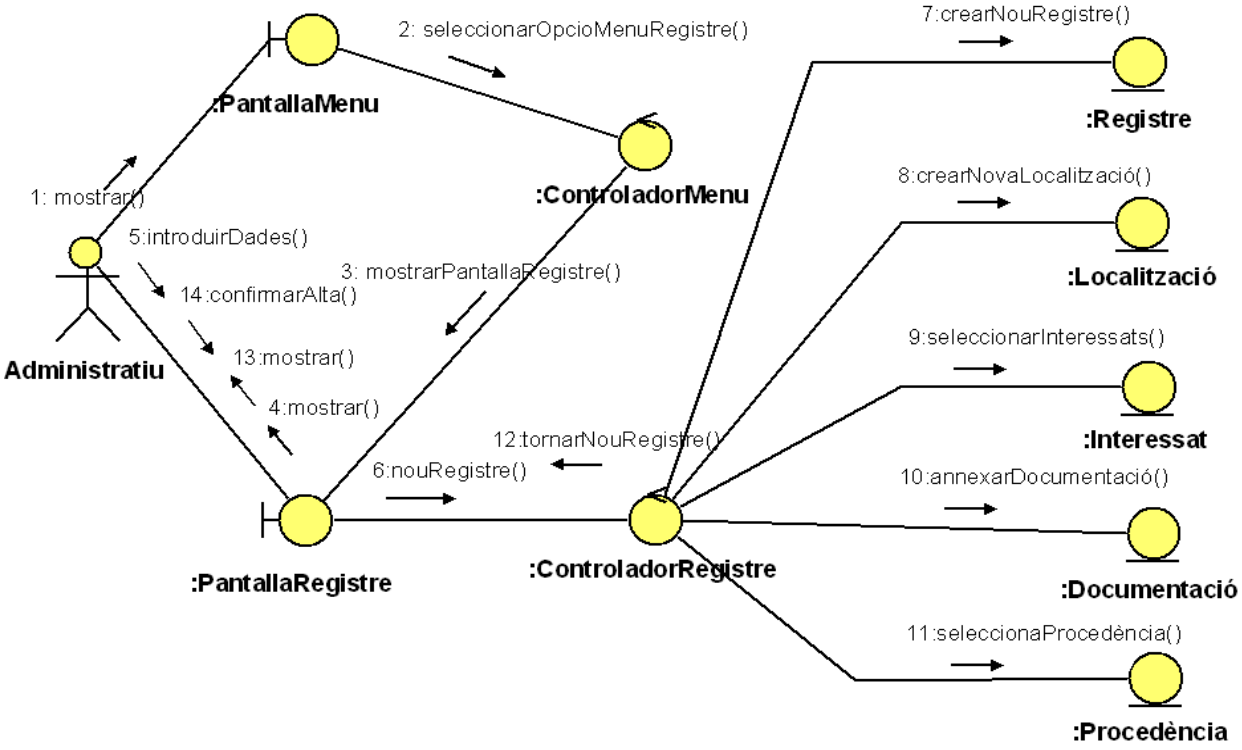



Fig. 38: Diagrama de col·laboració

## 5.7.2 Identificació de les classes d'entitat.

Mitjançant els casos d'ús es poden esbrinar quines seran les classes d'entitat. Cal destacar que aquestes classes, rebran un tractament més acurat en la fase de disseny.

S'emprarà els casos d'ús generalitzats de *manteniment* o *gestió* en compte de *alta - baixa – modificació*. Al cercar les classes d'entitat, s'empra un asterisc (*) si la classe ha segut prèviament identificada anteriorment, mentre que si hi ha dubte es ficarà una interrogació (?).

| Cas d'ús | Classes d'entitat emprades |
|---|---|
| **Login** | Administratiu?, Administrador?, Agent de Policia? |
| **Manteniment d'usuaris** | Usuari |
| **Manteniment de grups** | Grup, Usuari* |
| **Gestionar registre.** | Registre, Localitzacions, Documentació, Interessats, Procedència |
| **Gestionar accidents** | Accidents, Registre*, Localitzacions*, Afectats, Documentació* |
| **Gestionar localitzacions** | Localitzacions* |

| | |
|---|---|
| **Gestió de conceptes i regles joc** | Tipus de documents?, Tipus de destinacions? Regles |
| **Gestió de persones i altres** | Persones, Afectats*, Interessats* |
| **Gestió de tramitació** | Tràmits, Tramitant |
| **Realització de parts i informes** | Documentació* |
| **Consulta al registre** | Registre*, Localitzacions*, Documentació*, Interessats* |
| **Consulta als accidents** | Accidents*, Registre*, Localitzacions*, Afectats*, Documentació* |
| **Consulta als tràmits** | Tràmits*, Tramitant* |

S'establirà els següents criteris: No te cap sentit fer un anàlisi exhaustiu d'una aplicació de seguretat basada en usuaris, grups i rols, ja que ja existeixen molts anàlisis realitzats al respecte. A més per simplicitat es descartaran en aquesta primera fase la major part de les classes de codificacions, ja que aporten un excés de complicació per aquesta fase d'anàlisi.

### 5.7.3 Atributs de les classes d'entitat.

Una vegada simplificat el conjunt de classes, es fa la proposta d'atributs de les classes:

| Classe d'entitat | Atributs |
|---|---|
| **Usuari** | nom (string), cognom (string), password(string), grups (llista), dni (string) |
| **Grup** | nom (string), rols (llista) |
| **Rol** | nom (string), funcionalitat (string) |
| **Registre** | Id (integer), extracte (string), data (datetime),observacions (string), documentació (llista), localització (localització) |
| **Accident** | Id (integer), extracte (string), data (datetime),observacions (string), documentació (llista), localització (localització), afectats (llista) |
| **Persona o Interessat** | Nom (string) |
| **Tràmit** | Data (date), descripcio, documentació (llista), tramitant (persona) |
| **Documentació** | URL (string), nom(string) |
| **Procedència** | Nom (string) |

# Capítol VI: Disseny de l'aplicació de Registre E/S de la policia local mitjançant el framework GWT RAD Base.

## 6.1 Disseny de les classes d'entitat

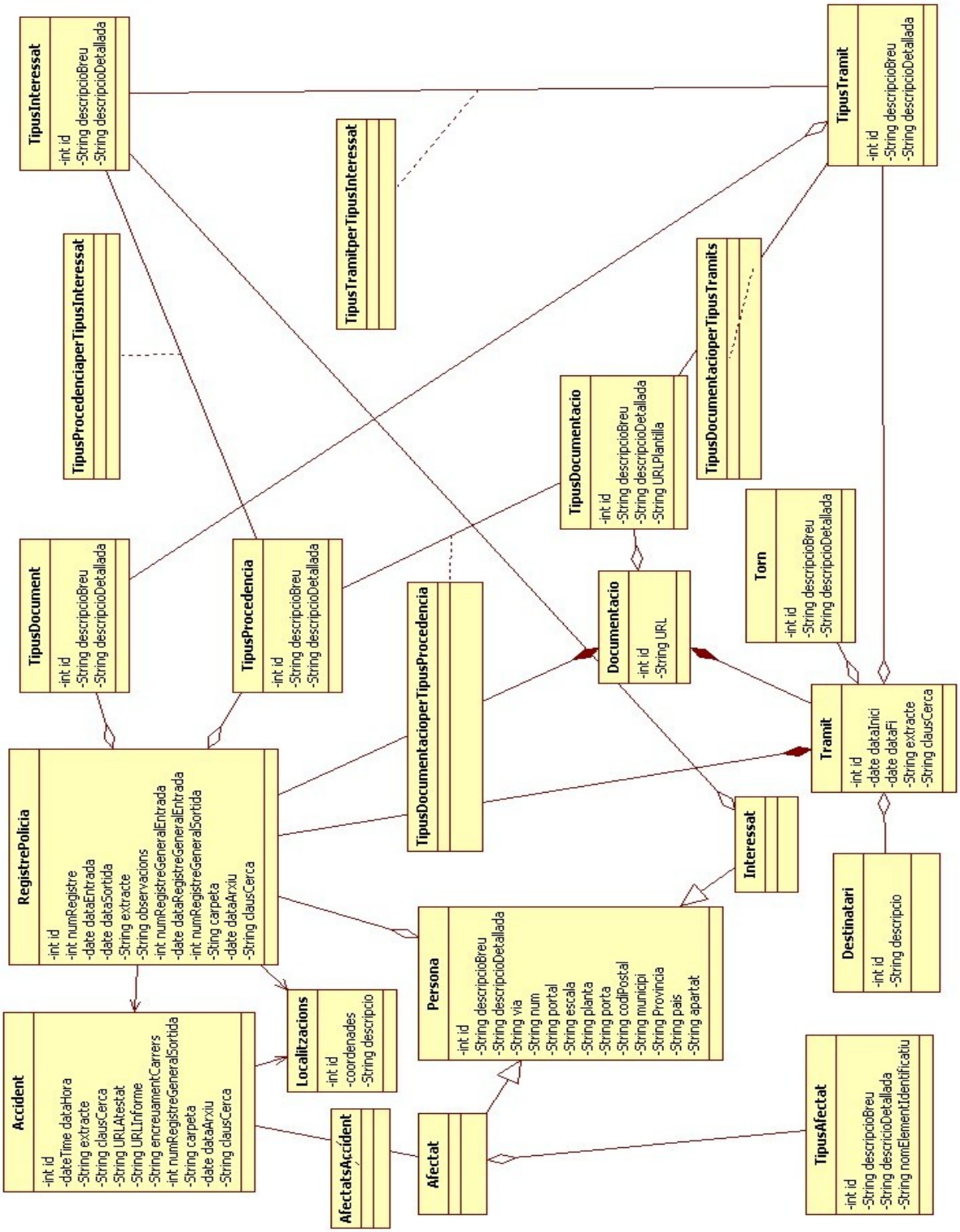


**Fig. 39: Diagrama de classes**

En la figura d'abans es mostra el diagrama de classes amb sols principals classes d'entitat que intervenen. Per simplicitat, no s'inclouen totes les classes que fan referència a usuaris, grups.

## 6.2 Disseny de la persistència.

Seria important fer un diagrama E-R per a veure com quedaria el disseny de la BBDD. No vull justificar que no l'inclogui ja que un bon disseny de BBDD (taules, índexs, relacions, vistes, procediments emmagatzemats, funcions, disparadors (*triggers*)) es peça fonamental en el rendiment i estabilitat de la base de dades, i per tant de l'aplicació.

Però he preferit delegar aquesta tasca en la *JPA (Java Persistent Api)* i que sigui *Hibernate* qui s'encarregui de fer el mapeig objecte-relacional

## 6.3 Disseny de la interfície d'usuari.

El disseny de la interfície d'usuari ve estrictament encorsetat pel *framework*, per tant en centraré a fer el disseny des del model al paquet ***tav.server.classes.***

## 6.4 Disseny des del model.

En els darrers punts, s'ha vist les instruccions de com dissenyar des del model. Vegem sols els detalls més importants que han sorgit a aquesta aplicació.

### 6.4.1 PASSA 1: Disseny de l'esquema de menús.

Es tracta de crear el següent esquema (Menú) de treball de la fig. 40. Partim d'una pantalla de login i a continuació hem de tenir un menú principal a l'esquerra amb totes les aplicacions (veure la fig.32 per a més aclariment) . Hem de recordar que les aplicacions s'han de donar d'alta al formulari d'aplicacions que apareix a l'esmentada figura amb la clau "1.1.1 Aplicacions". Per a dissenyar cadascun dels submenús de cada aplicació ens basarem en la informació aportada el punt 4.1 d'aquesta memòria. Cal recordar que aquesta definició disposa d'una nomenclatura pròpia i a més es guarda a la base de dades, i més concretament a la taula d'aplicacions.

### 6.4.2 PASSA 2: Definir els atributs i mètodes de les classes .

Aquest pas ja s'ha fet pràcticament al punt 6.1, no més cal remarcar que cal evitar les següents situacions:

- Relacions molts a molts: Cal simplificar-les per a tenir una estructura d'arbre.
- Classes molt complexes: Cal simplificar al màxim la grandària de les classes. Un disseny clar i breu és la clau de tenir una aplicació clara.
- Mètodes complicats. Ja que poden perdre la compatibilitat amb GWT.
- Mètodes que consumeixen molts recursos dins de transaccions. A l'hora d'emprar els mètodes definits en la interfície ***IbussinesLogic*** cal tenir en compte que s'empren dins d'una transacció, per tant cal optimitzar-los al màxim amb la fi d'evitar bloquejos no desitjats quan el nombre d'usuaris creix, fent que la nostra aplicació no funcioni adequadament. Per tant cal tenir molta cura amb la definició de la lògica de negoci. De tota manera això es veurà al punt 6.4.4.

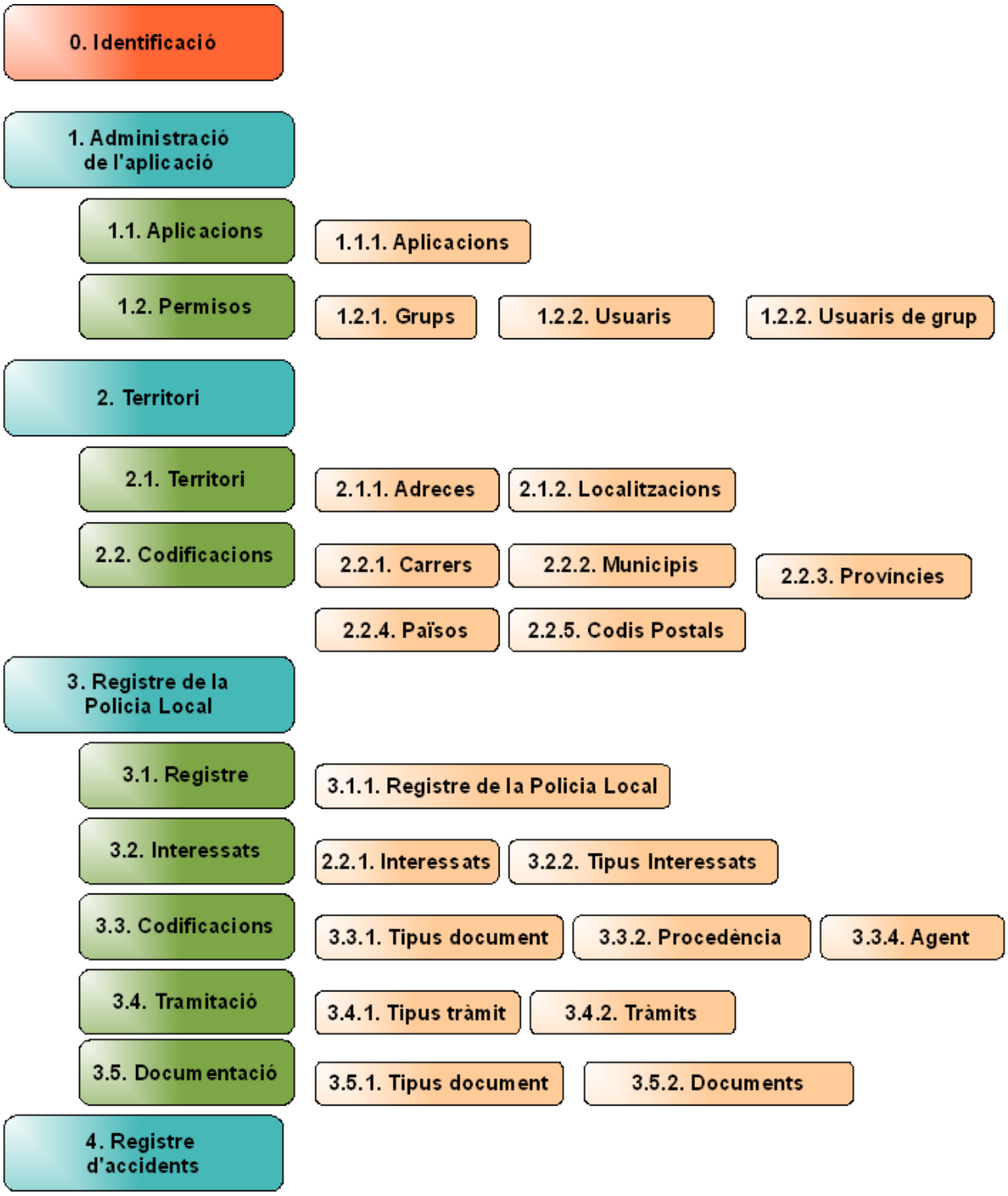


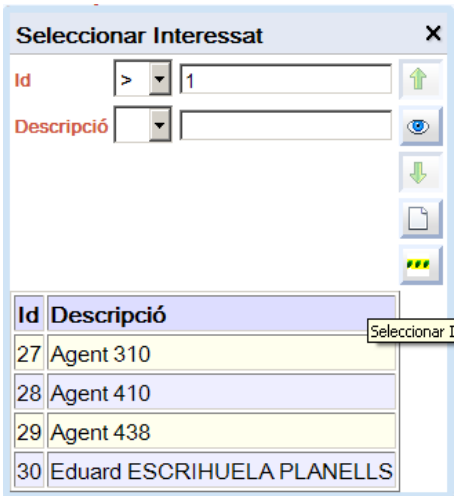

**Fig. 40: Esquema de menús**

### 6.4.3 PASSA 3: Definir els editors dels camps.

Per omissió l'editor és un quadre de text. Amb anotacions de la classe *ViewDisplay*, es poden seleccionar editors com *Calendaris*( que s'empra a l'atribut *dataModificació* de totes les classes), *Google Maps* (a l'atribut *localització* de la classe *RPRegistre*), *Llistes desplegables* (quan hi ha pocs camps a seleccionar en una anotació *@ManytoOne*) o també un Selector quan el nombre de registres a esbrinar és gran. Vegem exemples emprats en *RPRegistre*:

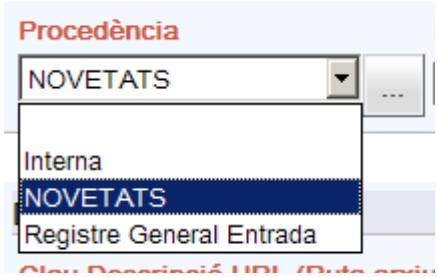
```
@ManyToOne(targetEntity=RPInteressat.class)
@JoinColumn(name="rprprepr")
@ViewFieldProperties(Editor="CaptureBox", DefaultValue="1")
private RPInteressat representant;
```


**Fig. 41: Control Selector "CaptureBox"**



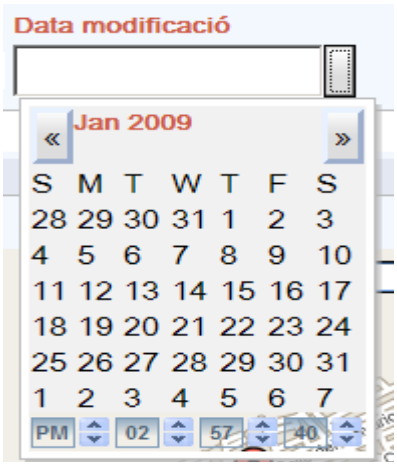

```
@ManyToOne(targetEntity=RPProcedencia.class)
@JoinColumn(name="rprprppr")
@ViewFieldProperties(Editor="ComboEdu")
private RPProcedencia procedencia;
```

**Fig. 42: Control "ComboEdu" per omisió.**



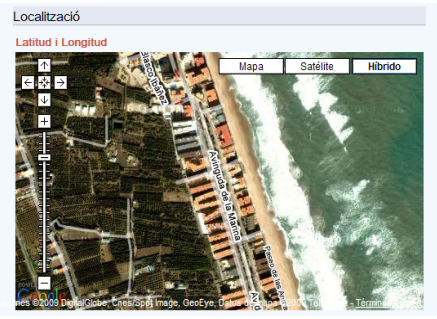

```
@Column(name = "rprpfmod")
@ViewFieldProperties(ReadOnly=true,  DefaultValue="getDateTime",
                     Editor="DateTimeComboBox")
private Date dataModificacio;
```

**Fig. 43: Control "DataTimeComboBox"**




```
@Column(name = "rprpltlg", length=40)
@NotNull
@NotEmpty
@ViewFieldProperties(Editor="MapEdu",
                DefaultValue="39.07237819997761; -0.2691650390625")
private String latitudLongitud;
```

**Fig. 44: Control "MapEdu" de Google Maps**



### 6.4.4 PASSA 4: Definir la lògica de negoci.

Al punt 2.6 s'explica com es pot definir la lògica de negoci. Per al nostre cas concret, s'ha fet servir els següents recursos:

- Anotacions ***@NotNull*** i ***@NotEmpty*** per alguns camps de tipus String que no interessi que es queden buits o a nul. (Gran part dels d'atributs de les classes empren aquestes anotacions. Cal advertir que aquestes dues anotacions no es poden fer servir a tipus de dades numèrics, ja que Hibernate llença una excepció al guardar l'objecte.
- A la classe *RPRegistre*: S'ha fet servir una anotació pròpia de la classe ***ViewDisplay*** per a definir un camp ***comptador*** (el camp en qüestió es *numRegistre*, i a més aquest camp depèn del *any,* és a dir cada vegada que canviï l'any, es renumeri des de 0 el número de registre). Aquesta lògica de negoci ja ve implementada de sèrie en el framework, i no cal fer cap programació addicional de cada classes en concret.

```
@Column(name = "rprpnreg")
//Every year numRegistre is initialized
@ViewFieldProperties(isCounter=true, GroupCounterMembers="any_", ReadOnly=true)
private Long numRegistre;
```

**Fig. 45: Expressió d'un comptador amb anotacions a la classe RPRegistre.**

- Per contra, no s'han fet servir els mètodes tant útils de ***beforeBeanUpdate***, ***afterBeanUpdate ..*** que li donen una gran potència a la lògica de negoci i a més fan ús de transaccions, la qual cosa inicialment sembla que evita emprar ***EJB.***

### 6.4.5 PASSA 5: Definir l'estructura de la pantalla.

Com ja s'ha dit abans en el punt 4.2, hi ha un petit llenguatge per a estructurar la pantalla. En concret, per a la classe més important, que és la que té més atributs, s'ha aconseguit la següent estructura de pantalla, amb els controls que s'han mencionat al punt 6.4.3.

Per a definir aquesta pantalla hem de mostrar el mètode ***getViewDisplay*** de la classe ***RPRegistre:***

```
public static String getViewDisplay() {
 String viewDisplay=
  "¿clau?              [id, any_, numRegistre ]," +
  "¿registre?          [usuari,dataModificacio]" +
  "¿informacio?        [dataDocument, dataEntrada, tipusDocument;" +
                       "descripcio; observacions; cercar]," +
  "¿localitzacio?      [latitudLongitud];" +
  "¿procedencia?       [agents; representant, interessat;" +
                       "procedencia, numRegistreEntrada]," +
  "¿assignacio?        [agent, torn, assignacio,dataComplimentat;" +
                       "correus, numRegistreSortida, dataRegistreSortida, expedient];" +
  "¿documentacio?      (RPDocument: N : id, descripcio, url, observacions)," +
  "¿tramitacio?        (RPTramit: N : id, rpTipusTramit, rpInteressat,descripcio,observacions)" ;
 return viewDisplay;}
```

**Fig. 46: Mètode getViewdisplay de la classe RPRegistre**

Com es pot apreciar, els 2 primers panells (clau i registre) es situen un al costat de

l'altre, ja que es separen amb una coma, els altres 2 panells (informació i localització) es situen mes cap avall. A continuació venen els 2 panells següents(procedència i assignació). Recordeu que un panell es defineix amb claudàtors"[ ]"

Després fem referència a 2 col·leccions de classes, la primera és la documentació aportada a aquest registre, i la segona és la documentació. Recordeu que les col·leccions es defineixen amb parèntesis "( )".

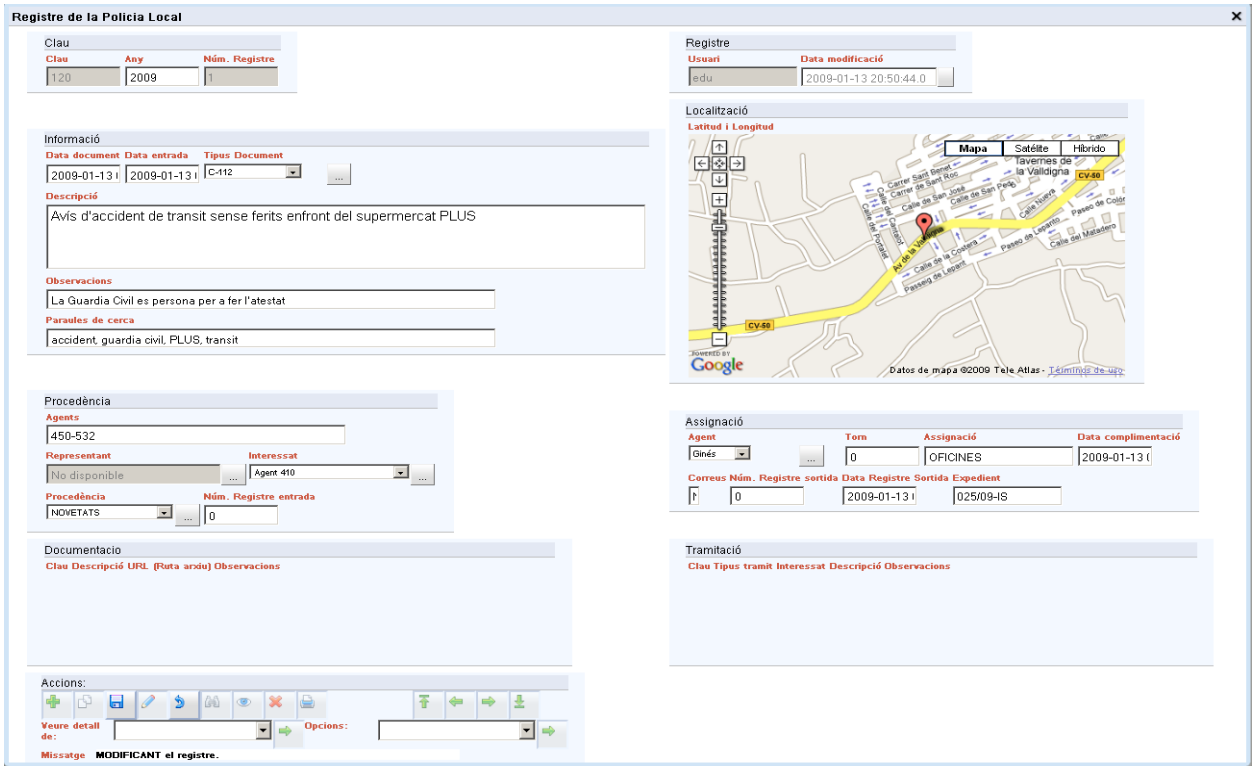



**Fig. 47: Pantalla final resultat a la classe RPRegistre.**

## 6.4.6 PASSA 6: Definir les col·leccions.

Per a definir una col·lecció, hem de fer:

- Definir (opcionalment, però molt recomanable) la col·lecció en el mètode getViewDisplay de la taula mare (veure fig.47). Per exemple a la figura 47 tenim 2 col·leccions (documentació i tramitació).
- A les taules de la col·leccio (taules filles), definir una anotació @ManyToOne amb la taula mare.
- A la taula mare, cal definir el mètode ***getDetailActions***, per a poder mantenir aquestes classes de la col·lecció. Per exemple a la taula ***RPRegistre*** definim l'enllaç a les taules de la col·lecció (RPDocument i RPTramit), i queda a la part baixa de la pantalla

```
public static String[][] getDetailActions() {
  String s[][]={{"RPDocument",""},{"RPTramit",""}};
  return s;}
```

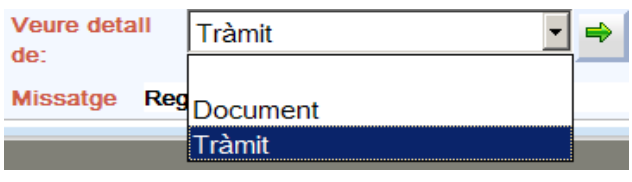



**Fig. 48: Mètode getViewdisplay de la classe RPRegistre**

### 6.4.7 PASSA 7: Definir les accions

No s'ha definit cap acció en aquesta aplicació, ni a nivell de taula ni a nivell de menú. Tan sols remarcar les accions que s'han definit en el menú inicial de l'administrador del sistema, per a crear les taules al SGBD i per a configurar aplicacions i permisos.

### 6.4.8 PASSA 8: Provar l'aplicació, depurar, i desplegar

Aquest passa s'ha definit amb detall amb un altre document que s'adjunta, ja que per motius d'extensió no es pot incloure a la memòria

# Conclusions

Aquesta ha segut una gran experiència, ja que he aconseguit entendre gran quatitat de conceptes tant de Java com de les tecnologies Web. Encara falta ultimar algunes coses com refinar els llistats, fer utilitats de migracions de dades, optimitzar i documentar millor el framework.

Si es tingués que fer una valoració econòmica d'aquest framework, no se si ho sabria fer, ja que aquest treball ha segut per a mi un repte, i que m'ha servit per a dependre, per tant, el millor és ficar-lo a disposició de la comunitat, i no entrar en aquest tipus de valoracions sobre un producte queda molt aviat obsolet per la ràpida evolució de la tecnologia.

Doncs, agraeixo a aquelles persones que s'interessin per aquest framework i tinguin la amabilitat de llegir aquesta memòria, facin els comentaris i crítiques adients per a la millora d'aquest treball a la meva bústia de correu eescrihuela@gmail.com**,** ja que aquest projecte te per objectius continuar després del lliurament d'aquesta memòria. Gràcies.

# Glossari

**Ajax:** Conjunt de tècniques basada en Java Script i crides asíncrones que tenen com a misió aconseguir un temps de resposta més curt de les aplicacions web.

**Anotació**: Eina per a completar la informació d'una classe, atribut o mètode. Es poden crear noves classes que són les encarregades a tipificar aquesta informació.

**Cloud computing**: Internet es compara metafòricament amb un núvol (cloud), on no queda clar on resideixen els servidors, i les funcionalitats són oferides com serveis.

**Disseny orientat al Model**: És una manera de dissenyar aplicacions de forma que pràcticament tota l'aplicació es genera des de la capa de model emprant POJOs i anotacions, com *OpenXava*.

**Editor:** Objecte en una pantalla que permet al usuari veure i entrar dades d'un camp.

**Expedient:** És la base de la tramitació en les administracions, i les fases són: iniciació, ordenació, instrucció, finalització i execució haver-hi procediments especials.

**Framework:** Bastiment que permet simplificar el desenvolupament de programari. Normalment es basa en plantilles. Funcionalment hi ha 2 tipus de frameworks, els que generen codi abans del desplegament, i els que generen els components en temps d'execució. El segón tipus s'està imposant per la seva simplicitat de desplegament.

**Gestió:** Veure *manteniment*.

**GWT:** (Google Web Toolkit) Framework de programari lliure liderat per Google, per a desenvolupar aplicacions web amb Ajax a la part del client de manera que el codi font és Java, i quan es compila es genera Java Script, HTML i CSS a la part del client.

**Manteniment:** Altra forma de cridar al conjunt d'operacions d'alta, baixa, modificació, consultes i llistats.

**POJO**: (Plain Old Java Object). Objectes classics de Java.

**Registre E/S:** És la manera de tenir guardades i ordenades tota les entrades i sortides de documentació trascendent. Està subjecte a normativa legal de obligat compliment. Una de les maneres de nàixer un expedient és a través de la documentació aportada per registre d'entrada.

**R.I.A.:** Segles corresponents a *Rich Internet Application*. És una manera d'emular al l'arquitectura *Client-Server* amb clients que disposen de controls de alta vistositat i prestacións, des de clients Web tipus Thin Client (que cada vegada són més complicats des de l'aparició d'Ajax)

# Bibliografia

**Benet Campderich Falgueras i altres** (2004-UOC): Enginyeria del programari

**Benet Campderich Falgueras i altres** (2005-UOC): Enginyeria del programari orientada a objectes.

**Jordi Cabot Sogrera i altres** (2005-UOC): Ingenieria del programari orientada a objectes.

**Jordi Cabot Sogrera i altres** (2006-UOC): Ingenieria del programari de components i sistemes distribuits.

**Joshua Bloch**(2001-Sun): Effective Java programming language guide.

**Jeff Frisen** (2007-Apress): Beginning Java SE Platform: From novice to professional.

**John Zulowski** (2006-Apress): Java 6 Platform revealed.

**H.M.Deitel** (2004-Prentice Hall): Java how to program.

**Cay S. Horstmann i Gari Cornell** (2002-Sun): Java 2 (Volúmens 1 i 2)

**John R. Hubbard** (2007-Schaum): Data structures with Java.

**Herb Schildt** (2008-McGraw Hill): Herb Schildt's Java programming cookbook.

**Bruce Eckel** (2006-Prentice Hall): Thinking in Java.

**Eric Jendrok i altres** (2008-Sun): The Java EE5 tutorial.

**Jason Hunter** (2000-O'Reilly): Java Servlets programming.

**Javier Garcia de Jalón i altres** (1999-Tecnum): Aprenda servlets de Java.

**Tim Downey** (2007-Springer): Web development with Java using Hibernate, JSP and servlets.

**Neal Ford** (2004-Manning): Art of Java web development (Struts, Tapestry, Commons,Velocity).

**Marti Hall i altres**(2008-Sun): Core servlets and Java server pages.

**Hibernate Reference Documentation** (2008): Core, Annotations, Validator, Tools.

**Christian Bauer i Gavin King** (2007-Manning): Java persistence with Hibernate.

**Dave Minter i altre** (2006-Apress): Beginning Hibernate, from novice to professional

**Eric Pugh i Joseph D. Gradecki** (2004-Wrox): Professional Hibernate.

**The Postgres SQL Global Development Group** (2008): Postgres 8.3.5 documentation.

**Peter Eisentraut, Bernd Helmle** (2008-O'Reilly): PostgresSQL Administation.

**Bruce Momjian** (2001-Addison Wesley): Postgres introducction and concepts.

**Leonard Richardson i Sam Ruby** (2007-O'Reilly):Restful web services.

**David R.Heffelfinger**(2005-Packt): Jasper Reports for Java developers.

**Giulio Toffoli** (2007-Apress): The definitive guide to i-Reports.

**Teodor Danciu i Lucian Chinta** (2008-Apress): The definitive guide to Jasper Reports.

**James W. Cooper** (1998-Addison Wesley): The design patterns.

**James R. Trott** (2000-Addison Wesley): Desing patterns explained.

**Craig Larman** (1995-Prentice Hall): Applying UML and patterns.

**Jason Brittain** (2008-O'Reilly): Tomcat: The definitive guide.

**Vibul Gupta** (2008-Apress): Accelerated GWT.

**Robert Hanson i Adam Tacy** (2007-Manning): GWT in action.

**Robert T. Cooper i Charlie E. Collins** (2008-Manning): GWT in practice.

**Prabhakar Chaganti** (2007-Packt): Google Web Toolkit and Java Ajax programming.

**Ryan Dewsbury** (2008-Prentice Hall): Google Web Toolkit applications.

**David Geary** (2007-Prentice Hall): Google Web Toolkit solutions.

**Jeff Dwyer** (2008-Apres): Pro Web 2.0 application development with GWT.