Turist@ v2.0

Xavier Descarrega de la Vega
(xavier.descarrega@estudiants.urv.cat)

Dirigit per:

Sr. David Isern Alarcón
(david.isern@urv.cat)
Dr. Antoni Moreno Ribas
(antonio.moreno@urv.cat)

juny de 2008

Índex

Capítol 1

Objectius, antecedents i extensió del projecte

1.1 Introducció i objectius

Són moltes les formes que un turista té d'accedir a la informació turística d'una determinada zona i, encara en són més si som capaços de planificar el viatge amb antel·lació a la nostra estança. Probablement també disposarem d'un dispositiu de posicionament o *GPS* que ens vagi marcant l'itinerari i és possible que ens permeti localitzar algunes activitats, si més no, podem aprofitar el nostre viatge d'una manera "eficient".
Tot això és vàlid i realment les tecnologies de la informació ens ajuden i molt però, de fet, tant buscar la informació abans del viatge, *in-situ* com el *GPS* no ens són prou útils per improvitzar un viatge. Què passaria si només indicant les nostres preferències, rebéssim un itinerari precís de les activitats d'acord amb el nostre perfil turístic?
Amb l'increment de les capacitats dels dispositius mòbils (Assistents Digitals Personals o *PDA*, telèfons mòbils...) s'incrementen les possibilitats de crear aplicacions més polivalents. Si a això hi sumem la implementació d'un terminal modern a gran escala, tenim que podem arribar a quasi qualsevol turista (veure apèndix B per a més informació sobre l'extensió de terminals moderns).
El projecte parteix de tres projectes de final de carrera anteriors (tots relacionats amb els sistemes multi-agent aplicats a la informació turística), el primer és un projecte creat per l'Àlex Viejo en el qual es descentralitzava el sistema (hi havia un back-end que gestionava els propis agents que proporcionaven la informació i un front-end que s'encarregava de fer de nexe entre l'usuari i la plataforma dels agents); el segon va ser creat per la Montse Batet i oferia un servei de recomanació al turista; el tercer, creat per Marc Redondo, oferia l'habilitat de construïr un itinerari turístic.
La meva feina consistiria en:

- Veure quina feina realitzaven tots tres projectes
- Ajuntar totes tres habilitats
- Veure quines millores es podien aplicar (GPS, gràfics, portabilitat...)

1.2 Antecedents

1.2.1 Plataforma mòbil

L'objectiu del primer PFC estudiat és donar un pas més en la cerca de l'assistent personal basat en agents, obtenint un producte que de veritat ofereixi les funcionalitats requerides per un usuari real. Les claus del projecte són:

- Funcionament en un mòbil senzill: Aplicació capaç d'executar-se en la majoria dels terminals.
- Connexió GPRS: Llibertat de moviments per a l'usuari per tota la ciutat.

- Sistema de localització GSM: L'usuari sap en tot moment on es troba.
- Mapes de la ciutat: L'assistent mostra mapes per a facilitar l'orientació de l'usuari.
- Sistema intel·ligent de recomanació: Recomanacions adaptades als gustos del client i actualització automàtica de les preferències de l'usuari.

El que es pretenia conseguir era tindre un SMA funcionant en un entorn distribuït, en el qual el client intervingués utilitzant un telèfon mòbil. Aquest client seria avisat en temps real de la presència d'activitats en el lloc on es trobés, permetent realitzar peticions personalitzades i mostrar al client la seva localització en la ciutat.
Per a tal efecte, primer s'ha aconseguit obtenir la posició a partir del sistema GSM del telèfon mòbil.
Posteriorment es va establir la conexió via GPRS entre l'agent personal i les estacions fixes i, per acabar, entrem en l'implementació de l'aplicatiu realitzant un exhaustiu estudi de les possibilitats que l'entorn J2ME ens ofereix per a poder obtenir totes les funcionalitats desitjades per a l'assistent de turisme. Cal recordar que totes les anteriors versions de l'aplicatiu utilitzaven J2SE i, per tant, un conjunt més ric en llibreries per a l'aplicació.

1.2.2 Recomanació turística

L'objectiu d'aquest projecte és realitzar un estudi profund dels sistemes de recomanació existents i dissenyar un recomanador per activitats turístiques i incorporar-lo en un prototipus basat en la metodologia de sistemes multi-agent (SMA). Es construirà un agent que permetrà realitzar deferents tipus de recomanacions d'activitats personalitzades segons el perfil de l'usuari. Per una banda, les recomanacions es faran usant tècniques basades en els sistemes recomanadors estudiats i de altres tècniques d'intel·ligència artificial, i per l'altra banda s'implementaran varies d?aquestes tècniques per a poder fer una comparació entre elles.
Una altre objectiu és millorar un prototipus que es va implementar en altres projectes final de carrera anteriors. La nova versió disposarà d'una gamma molt més àmplia de tipus d'activitats. També es podran realitzar cerques d'activitats d'un o varis tipus d'activitat, tenint la possibilitat d'afegir, a més, restriccions a aquestes cerques. Es disposarà, per tot això, d'una interfície per a l'usuari que li permetrà introduir les seves dades o modificar-les i fer peticions de cerques d'activitats i de recomanacions. El prototipus que es vol implementar s?usarà com a idioma l'anglès, per tal de tenir un sistema que pugui ser usat pels turistes estrangers.

1.2.3 Creació itinerari

El principal objectiu d'aquest projecte és ampliar el sistema existent per a que permeti a l'usuari organitza-se l'agenda del viatge. A partir d'un conjunt d'activitats prèviament pre-seleccionades. Aquestes activitats es poden obtenir a través de cerques o recomanacions obtingudes del sistema. També ens permet veure el resultat de la planificació a diari amb el seu recorregut i representació gràfica. A més, evita la concentració d'activitats en un mateix dia.

1.3 Extensió del projecte

L'extensió del projecte és factible si la tecnologia ens permet noves possibilitats, tot i tenir la idea, l'essència de l'eina. El poder disposar d'un GPS integrat en el mòbil ens permetrà major mobilitat, una conexió o una capacitat de memòria ens permetrà guardar, per exemple, mostres de les activitats en forma d'àudio, vídeo o fotografia. Conforme s'extén l'ús de la plataforma Java, es millora les seves capacitats per tal d'aprofitar cada terminal al màxim.

1.4 Estructura del document

En el capítol 2 trobareu tota la informació relativa als antecedens d'aquest projecte: un breu resum de cadascuna de les funcionalitats dels anteriors Projectes de Final de Carrera. En el tercer capítol agrupo totes les funcionalitats de l'aplicatiu, distingint entre el FrontEnd i BackEnd. En el quart capítol indico totes les decisions de disseny preses durant el desenvolupament del

projecte. Dins el capítol 5 exposo alguns dels algorismes utilitzats. En arribar al capítol 6 finalitza l'etapa de desenvolupament i comença l'avaluació, exposant un cas pràctic. El resum i les conclusions es troben en el capítol 7 i els recursos i bibliografia en el capítol 8. Com a manual d'ús hi ha el capítol 9 i, per a més informació, l'annex descriu les eines utilitzades: JADE, JADE-LEAP i uns petita introducció a J2ME.

Capítol 2

Especificacions del projecte

Un dels punts forts del projecte és l'adaptació que s'ha fet per tal de facilitar la feina a l'usuari (turista) i al proveïdor del servei (en aquest cas podria ser una agència turística).

2.1 BackEnd

El BackEnd serà qui crei els agents i els recursos necessaris per a que el dispositiu mòbil hagi d'efectuar el mínim de càlculs possibles. Tot el sistema d'agents, així com les característiques de la base de dades resten definides en un fitxer XML que es carrega en temps d'execució i que activa els agents tal i com estan definits en aquest fitxer.
A més, el BackEnd dóna la possibilitat de veure les conexions i les transferències que hi ha entre el BackEnd i el FrontEnd. Per finalitzar les característiques del BackEnd tenim les opcions de crear, modificar i eliminar agents i d'eliminar (expulsar) usuaris.

2.2 FrontEnd

Aquí resideix en gran part la senzillesa del treball:

- Creació/Modificació del perfil de l'usuari. Una finestra ens permet establir el nostre nom i les nostres preferències, a més del dispositiu GPS bluetooth (si en disposem d'un).
- Petició de recomanació turística: A partir del nostre perfil, el BackEnd ens el·labora un seguit de recomanacions d'activitats i ens les retorna per tal de poder seleccionar quines ens interessen més (o totes).
- Petició creació itinerari: Un cop sleccionades les activitats, les guardem i automàticament rebrem un itinerari planificat de les activitats triades anteriorment.
- Posicionament: Si disposem d'un GPS, ens mostrarà la nostra posició en el mapa i, si no en disposem, el programa únicament ens posicionarà cadascuna de les activitats.

Capítol 3

Disseny

3.1 Descripció general

La segona tasca a realitzar (la primera era veure com funcionaven els antecedents a aquest projecte) és veure com podem ajuntar el tres projectes. Per una sèrie de motius decideixo fer el programari de cap i de nou, descartant qualsevol porció de codi i únicament reutilitzant les idees ja que:

- Falta de mnemotècnia. És prou difícil amb les limitacions de les llibreries J2ME i la propia tasca del programador com per tenir que averiguar què signifiquen les variables quan el seu nom no ens condueix a res.
- Falta de cohesió entre treball. Tot i tenir la mateixa idea en diversos casos, m'he trobat en punts en que el codi era totalment diferent.
- Repetició de codi. Una cosa que ens pot fer perdre hores és veure codi exactament igual que realitza dues accions diferents.

A més, el FrontEnd no creïa que fóra el suficient simple per a que un turista sense coneixements podés utilitzar-lo si més no, retallavem una mica l'accessibilitat de l'aplicatiu.

3.2 Decisions de disseny

3.2.1 Agents

Dins al sistema existeixen 2 tipus d'agents i, dins de cada tipus, tenim un subtipus d'agent:

- Agent de servei. És un agent que es carrega per defecte sigui quina sigui la nostra configuració d'activitats. Dins d'aquest grup, els tres subtipus d'agent de servei són:
  - SessionManager. Gestiona les sessions obertes entre BackEnd i terminals i gestiona també els perfils dels usuaris conectats.
  - Recommender. És qui efectuarà les recomanacions d'acord al perfil que emmagatzemi el SessionManager.
  - Scheduler. Serà qui, quan rebi un llistat d'activitats, retorni les activitats planificades al llarg del dia.
- Agent d'activitat. Tindrà una taula associada que, prèviament, conté totes les activitats amb la seva informació essencial:
  - CoordX. Coordenada X o longitud del lloc on es realitza l'activitat.
  - CoordY. Coordenada Y o latitud del lloc on es realitza l'activitat
  - Id. Identificador únic per a l'activitat dins la taula.
  - Descripció. Breu descripció que li apareixerà a l'usuari quan li sigui retornada la llista d'activitats recomanades.

- Data. Data de l'activitat.
- Hora. Hora inici de l'activitat.
- Durada. Durada de l'activitat. Si aquest camp és nul (0) se considerarà l'activitat com a *no-fixa* (veure apartat 3.2.4).
- Qualificació (en funció del nombre d'activitats). Conté tants camp com tipus d'activitats disponibles hi hagi dins el sistema. Cada camp tindrà una qualificació associada que valora l'activitat per a poder fer la recomanació.

A més, contindrà tant camps opcionals com es desitji i, aquests, seran utilitzats per a donar informació addicional sobre l'activitat al turista.

3.2.2 Gestió perfil usuari

Quan s'incorpora un nou usuari al sistema de recomanació, el sistema recollirà informació sobre les seves preferències i característiques per a la generació del perfil inicial preguntant a l'usuari manualment, és a dir, obtenint la informació explícitament. Utilitzarem el mètode que es defineix a continuació: Oferir a l'usuari un formulari que li permeti donar les seves dades personals i descriure algunes de les seves preferències culturals i d'oci.
Aquest formulari és el que es mostra a continuació.

Dades del formulari

A continuació s'enumeraran quins atributs del perfil d'usuari es preguntaran directament a l'usuari, és a dir, quins atributs estaran en el formulari inicial que haurà d'omplir l'usuari :

- Nom. Identificador de l'usuari, per a questions de gestió de perfil d'usuaris.
- Edat. L'edat del turista, per quaestions relacionades amb la planificació.
- Interès en activitat. Aquí hi haurà un llistat de totes les activitats disponibles i el turista valorarà de l'1 al 5 la seva preferència. Aquesta Preferència es traduïrà llavors a un valor:
  - 1 = -0.1
  - 2 = -0.05
  - 3 = 0
  - 4 = 0.05
  - 5 = 0.1

El sistema de recomanació gestionarà les dades del perfil inicial creant grups d'usuaris amb perfils semblants o estereotips per a poder realitzar recomanacions col·laboratives. És a dir, es classificaran els usuaris en grups que representen les característiques de les diferents tipologies d'usuaris. Les dades usades per realitzar aquesta classificació seran dades de tipus personal de l'usuari i interessos.

Actualització o aprenentatge del perfil d'usuari

El sistema recollirà informació per l'actualització del perfil d'un usuari utilitzant els dos mètodes que es defineixen a continuació:

- **Explícitament:** L'usuari indicarà els seus interessos de la següent manera. L'usuari indicarà quines activitats d'entre les recomanades vol realitzar i en quina data. El sistema recomanador cada dia mirarà quines activitats va realitzar l'usuari el dia anterior i li demanarà que en faci una valoració. Llavors l'usuari farà la valoració personal d'aquestes activitats i les transmetrà al Recomanador, que és qui s'encarregarà de actualitzar el perfil.
- **Implícitament:** S'observarà i guardarà el comportament de l'usuari i després s'analitzarà per distingir entre el que li agrada i el que no. És a dir, s'analitzaran els gustos i preferències d'un usuari per realitzar un aprenentatge dels seus gustos. Aquesta informació s'obtindrà observant les seleccions d'activitats d'entre les recomanades. El funcionament consistirà en que l'usuari farà una cerca d'activitats i, quan l'usuari enviï al Planificador les activitats seleccionades, també ho enviarà al recomanador per tal de que actualitzi el seu eprfil.

Actualització explícita del perfil a través de valoracions de l'usuari

L'actualització del perfil d'usuari és realitza quan un usuari fa una valoració general d'una activitat que ha realitzat. Així doncs, l'agent Recomanador rebrà una llista dels identificadors de les activitats amb una valoració per cadascuna d'elles i actualitzarà els interessos dels usuaris. L'actualització dels interessos dels usuaris es farà sumant diferents valors al vector d'interessos dels usuaris en cas que l'activitat hagi agradat o restant aquestes mateixes quantitats en cas que l'activitat no hagi agradat. La traducció numèrica de les valoracions és la que es llista a continuació:

- 1 = -0.1
- 2 = -0.05
- 3 = 0
- 4 = 0.05
- 5 = 0.1

Per cadascun dels atributs de l'activitat que indiquen el grau d'interès, hem dissenyat el següent procés d'actualització.
Suposem que tenim un vector u amb els graus d'interès de l'usuari en diferents aspectes culturals i d'oci, on $\vec{u_i}$ és un valor entre 0 i 1 que representa l'interès en l'aspecte i i un vector $\vec{a}$ amb les característiques d'una activitat, on a_i és la valoració de la característica de la activitat.
Considerem l'atribut i-èssim, que ha rebut una certa valoració i de l'usuari, aleshores:

$$\text{Si } a_i \geq Medium \Rightarrow \vec{u_i} = \vec{u_i} + valoracio_i * a_i$$

És a dir, per l'atribut i, si l'interès de l'activitat és com a mínim Medium, aleshores sumarem/restarem a l'interès de l'usuari una quantitat que depèn de la valoració que hagi fet, i de l'interès de l'activitat per aquest atribut concret.
Així, si la valoració és dolenta restarem el valor numèric indicat anteriorment, en canvi si és bona sumarem punts a l'interès per aquest atribut.
Si l'interès de l'activitat en aquest atribut és baix (0 o 1) aleshores no fem res, no fem un increment ni un decrement l'interès de l'usuari, perquè aquest criteri no és rellevant per aquesta activitat.
D'aquesta manera s'aconsegueix que per les activitats amb un interès alt en art, per exemple, però mal valorades per l'usuari es disminueixi aquest interès. Per les activitats amb un interès alt en art però ben valorades per l'usuari s'augmenta aquest interès.
Per exemple, suposem que l'interès de l'activitat en art és de 0.75 i l'interès en art de l'usuari és de 0.5. Suposem que l'usuari ha valorat la activitat com a Horrible (valor numèric ?0.1). Com que l'interès en l'activitat és major que 0.5, decrementerem l'interès de l'usuari. L'interès de l'usuari en art actualitzat quedaria de la següent manera:

$$Interesactualitzat = 0.5 + (-0.1 * 0.75) = 0.425$$

Evidentment, com a l'usuari no li ha agradat una activitat ben valorada en art, se li ha disminuït en el seu perfil l'interès en art.
Posem un altre exemple. Suposem que l'interès de l'activitat en música és de 0.75 i l'interès en música de l'usuari és de 0.65. Suposem que l'usuari ha valorat la activitat com a Excellent (valor numèric +0.1). Com que l'interès en l'activitat és major que 0.5, incrementarem l'interès de l'usuari. L'interès de l'usuari en música actualitzat quedaria de la següent manera:

$$Interesactualitzat = 0.65 + (+0.1 * 0.75) = 0.725$$

Com que a l'usuari li ha agradat una activitat ben valorada en música, se li ha incrementat en el seu perfil l'interès en música.

3.2.3 Bases de dades

Aquí rau també, part de la senzillesa del projecte. S'ha intentat minimitzar la quantitat d'informació repetida així com tenir més d'una taula per agent.
De manera que existeix una única base de dades (configurable) amb una única taula per a cada agent d'activitat, una taula per als perfils dels usuaris i una altra que conté referncies a les activitats disponibles.

Esquema entitat-relació

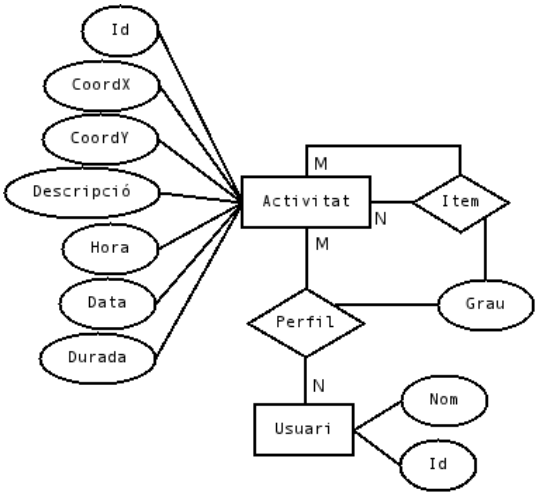


Figura 3.1: Esquema entitat-relació base de dades

Com a segona decisió de disseny, respecte la base de dades, he cregut convenient separar l'aplicatiu de gestió de recomanacions turístiques de la gestió de la base de dades de manera que, suposem que la base de dades ve d'alguna manera, amb les activitats actualitzades per un programari extern. D'aquesta manera, els agents no poden modificar cap de les taules amb les fonts d'informació sobre les activitats, únicament l'agent de gestió de sessions i el recomanador poden ajustar les seves taules.

3.2.4 Back End

Sistema recomanador

En el disseny d'un sistema recomanador cal tenir en compte molts aspectes relacionats principalment amb la representació dels perfils d'usuari i amb la manera en que es realitzarà el procés de recomanació. A continuació es mostren els aspectes més importants a tenir en compte.
Concretament cal decidir:

- Com representar i emmagatzemar les dades del perfil d'usuari. Existeixen diferents tècniques per representar el model d'usuari, com es detallarà més endavant en aquest document.
- uines han de ser les característiques que ens interessa emmagatzemar de l'usuari: interessos i preferències i/o dades demogràfiques, etc.
- A priori no es té coneixement dels usuaris (Startup problem). S?ha de decidir com es solucionarà aquest problema.
- La manera d'obtenció de les dades. Aquestes poden ser obtingudes de manera implícita (inferint els interessos i preferències a partir del coneixement del domini) o explícita (obtenint les dades directament de l'usuari).

- Si mantindrem grups d?usuaris. Si hi són, cal tenir en compte quina classificació es farà dels usuaris, i de quina mida seran els grups.
- Els continguts d?una recomanació. Poden ser des de un sol bit fins anotacions textuals no estructurades.
- Com guardar, actualitzar i esborrar informació obtinguda explícitament i implícitament.
- Com aprendre o adaptar els interessos i preferències dels usuaris.
- La tècnica de recomanació a utilitzar. Existeixen múltiples tècniques per realitzar una recomanació. Es poden predir interessos i preferències d'usuaris individuals basats en informació pròpia de l'usuari o basats en els usuaris similars, i en suposicions sobre subgrups homogenis (estereotips).
- Quin és el domini de la recomanació. És important definir en quin domini o àmbit es mouran las recomanacions d'un determinat sistema recomanador. En aquest àmbit és important estudiar el volum d'ítems amb el que hem de treballar. No és el mateix treballar amb pocs ítems que no pateixen modificacions en un interval de temps mitjà o gran, com per exemple les pel·lícules de la cartellera d'un cinema, que amb grans volums d'ítems que canviaran ràpidament, com per exemple, les noticies a nivell internacional.
- La rapidesa amb la que el sistema recomanador ha de poder fer les noves recomanacions. Usant l'exemple anterior, el sistema que recomana pel·lícules d?una cartellera que es modifica un cop per setmana no ha d'actualitzar-se amb la mateixa velocitat que ho fan milers de notícies en poques hores.
- Quan realitzar l'adaptació del perfil d?usuari, depenent del domini i de la velocitat de recomanacions necessària.
- Qui ha de realitzar les recomanacions i per què. Si les recomanacions es fan en grups d'usuaris amb característiques similars pot passar que alguns usuaris generin muntanyes de recomanacions que després seran positives per als seus interessos, ja que el sistema estarà entrenat per complaure els gustos d?aquests. Però per a la resta d'usuaris, que hauran generat poques recomanacions probablement diferents, les recomanacions generades pels usuaris anteriors els afectaran negativament.
- Com incentivar als usuaris per a obtenir les seves dades i així poder realitzar recomanacions. En alguns casos el fet d'obtenir recomanacions haurà de tenir alguna mena d'incentiu, per exemple, obligar a fer recomanacions abans d'obtenir-ne una per un mateix, o oferir alguna compensació econòmica.
- Com tenir cura de la consistència, seguretat i privacitat del contingut del model d'usuari. La privacitat és un aspecte important a tenir en compte. En moltes ocasions els usuaris no volen que els seus hàbits o interessos siguin coneguts per tothom. Tenint en compte en quin grau poden recelar els usuaris en el moment d'indicar la seva identitat, les recomanacions poden ser anònimes, es poden usar pseudònims o es poden etiquetar amb la identitat real de l'usuari.

En un sistema recomanador per a turistes esporàdics com aquest, l'usuari proveeix al sistema d'informació personal. Les recomanacions també es poden basar en els gustos i preferències de turistes semblants que han visitat la ciutat anteriorment. Per aquest motiu aquest sistema serà un sistema recomanador híbrid entre un sistema recomanador basat en el contingut i un sistema recomanador col·laboratiu.
Una recomanavió serà col·laborativa o no depenent de la quantitat d'usuaris que hi hagi en el sistema. Por exemple, si el nombre d'usuaris del sistema és petit el més probable és que la recomanació col·laborativa no tingui sentit. Llavors la recomanació es farà en base a la recomanació personalitzada. La motivació principal pel que el sistema de recomanació sigui col·laboratiu és que probablement un turista no estarà en el lloc on fa les vacances els dies suficients com per a poder aprendre els seus gustos personals.
Representarem un museu, monument, itinerari, exposició, etc. amb un vector de característiques.

Aquestes característiques representen el grau d'interès de l'activitat en diferents àmbits, així com informació bàsica per poder realitzar una recomanació com, por exemple, el preu de l'activitat o els idiomes disponibles. Per altra banda, el perfil d'usuari el representarem mitjançant un vector d'interessos en els diferents àmbits turístics. Aquest interessos son un valor entre 0 i 1, on el 0 representa que l'usuari no hi està interessat i el 1 representa que l'usuari hi està totalment interessat.
Per les recomanacions basades en el contingut utilitzarem els interessos de l'usuari mentre que per les recomanacions col·laboratives utilitzarem les dades personals i els interessos. A partir d'aquí enumerarem les diferents tècniques que hem estudiat per a realitzar les recomanacions.
Per a les recomanacions basades en contingut s'ha decidit buscar la distància Ecludiana dels atributs dels vectors (de l'activitat i de les preferències de l'usuari). Podem calcular la similitud entre els gustos d'un usuari i la descripció d'una activitat realitzant el càlcul de la distància euclidiana entre aquests dos vectors i dividint per la arrel de la màxima distància entre els dos vectors, és a dir, el nombre d'atributs dels vectors que tenim en compte (atributs d'activitat igual a 0 no es tenen en compte en el càlcul) i que anomenem n.

$$Distancia = \frac{\sqrt{\sum (u_i - a_i)^2}}{\sqrt{n}}$$

Sistema planificador

El sistema planificador neix amb la idea de facilitar a l'usuari la feina de realitzar l'agenda del seu viatge. Tal com s'ha dit, el sistema dóna prioritat a activitats que siguin al gust de l'usuari per tal de que gaudeixi del viatge a la ciutat.
Per implementar-ho, s'ha construït l'agent "SchedulerAgent" que és l'encarregat de la planificació de les activitats que un usuari voldria realitzar durant el període de temps que està en una ciutat, com Tarragona.
Dins la planificació, distingirem dos tipus d'activitats:

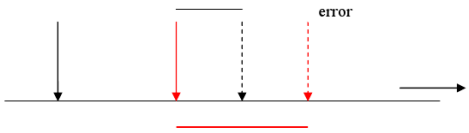
- Activitats fixes. Són activitats de les quals en tenim l'horari i la durada i podem planificar en seguretat quan i durant el turista podrà assistir a l'activitat. Aquestes activitats tenen major prioritat a l'hora de realitzar la planificació.
- Activitats no-fixes. Són activitats de les quals no en disposem informació sobre el seu horari concret (per exemple, un museu, una exposició...) i podem encabir-les en qualsevol buit de l'agenda.

De manera que, l'agent planificador realitzarà dues accions en diferent temps: Crear itinerari a través de les activitats seleccionades pel turista i que són fixes i, després, omplirà els buits de l'horari amb els activitats no-fixes
Tenim per tant, que la informació bàsica per a l'agent planificador és: identificador activitat, hora/data inici, durada i si és fixa o no-fixa.
La segona decisió de disseny està relacionada amb el càlcul de temps dels desplaçaments entre els llocs de les activitats en el qual es té en compte l'edat del viatger ja que condiciona la velocitat per desplaçar-se entre activitats. Per compensar el temps s'aplica un factor al temps estimat segons l'edat que es calcula de la següent forma:

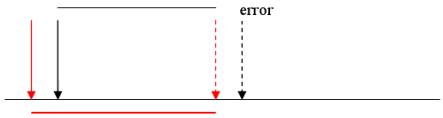
```
factor = 1.0;
si (edat<3) factor = 1;
si (edat>=3 i edat<7) factor = 3;
si (edat>=7 i edat<12) factor = 2;
si (edat>=50 i edat<60) factor = 1.5;
si (edat>=65 i edat<75) factor = 2.0;
si (edat>=75) factor = 3;
```

Una altra de les decisions de disseny preses ha estat que quan s'acabi una activitat, s'afegeix deu minuts a l'hora de finalització d'una activitat per tal de que l'usuari pugui descansar i deixar un petit marge d'error.
Es poden donar quatre casos en que les activitats no siguin planificables, ho il·lustrarem a través de diagrames. En negre les activitats planificades, en vermell l'activitat que volem afegir. En fletxa continua inici activitat i en fletxa discontínua final d'activitat.

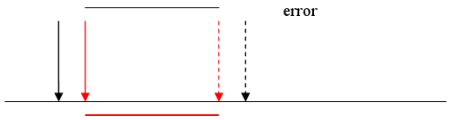
Cas 1) Activitat comença abans d'acabar-se l'anterior



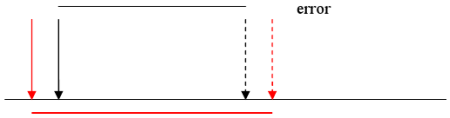
Cas 2) Activitat comença abans però acaba dins una activitat planificada



Cas 3) L'activitat es solapa completament amb una activitat planificada



Cas 4) Idem Cas 3) però el horaris inici i fi correctes



Fitxer configuració

Els antecedents tenien una classe per a cada agent d'activitat (agent museu, agent teatre, agent...). El que he decidit fer és crear un arxiu XML com l'exemple:

```
<?xml version="1.0" encoding="UTF-8"?>
<SystemDescription Name="Turist@1">
 <Database Name="turista">
  <Host Port="3306"  Name="localhost"></Host>
  <User Name="admin" Pwd="password"></User>
 </Database>
 <AgentProperties>
  <Property Name="ID_ACTIVITY" Type="INTEGER NOT NULL AUTO_INCREMENT"></Property>
  <Property Name="COORDX" Type="INTEGER"></Property>
  <Property Name="COORDY" Type="INTEGER"></Property>
  <Property Name="NAME" Type="VARCHAR(32)"></Property>
  <Property Name="ADDRESS" Type="VARCHAR(100) NULL"></Property>
  <Property Name="TIME" Type="VARCHAR(100) NULL"></Property>
  <Property Name="DATE" Type="VARCHAR(100) NULL"></Property>
  <Property Name="LENGTH" Type="VARCHAR(100) NULL"></Property>
 </AgentProperties>
 <AgentsDescription>
  <Agent Name="NOMAGENT1" ActName="NOMACT1" TableName="NOMTAULA1"></Agent>
<Field Name="NOMCAMP" Type="TIPUS"></Field>
  <Agent Name="NOMAGENT2" ActName="NOMACT2" TableName="NOMTAULA2"></Agent>
  ...
  <Agent Name="NOMAGENTN" ActName="NOMACTN" TableName="NOMTAULAN"></Agent>
 </AgentsDescription>
</SystemDescription>
```

I crear cadascun dels agents en temps d'execució. Això, per començar, ens reduïa molt de codi ja que el comportament per cada agent d'activitat era exactament el mateix. A més, la quantitat d'agents d'activitats pot ser quasi infinit. La definició de l'estructura és la següent:

- Database. Defineix totes les propietats per a conectar-nos amb la base de dades.
- AgentProperties. Defineix quin seran els camps imperatius per tal de que el sistema funcioni i puguis fer les recomanacions.
- AgentsDescription. Aquí definim cadascun dels agents d'activitat. Aquí dins podrem definir els camp addicionals per a informació complementaria per al turista.

Interfície gràfica

El segon problema que volia afrontar era l'interfície gràfica. Fins ara, cada agent s'engegava des del JADE, on s'especificava si engegar o no l'interfície gràfica associada. Jo he decidit que la modificació de la base de dades ha de venir donada per un altre programa (el qual no és l'extensió d'aquest PFC) i que les funcionalitats de l'interfície s'havien de reduïr a:

- Carregar/Guardar XML.
- Veure/Expulsar usuaris connectats.
- Crear/Modificar/Eliminar agent.

Diagrama UML

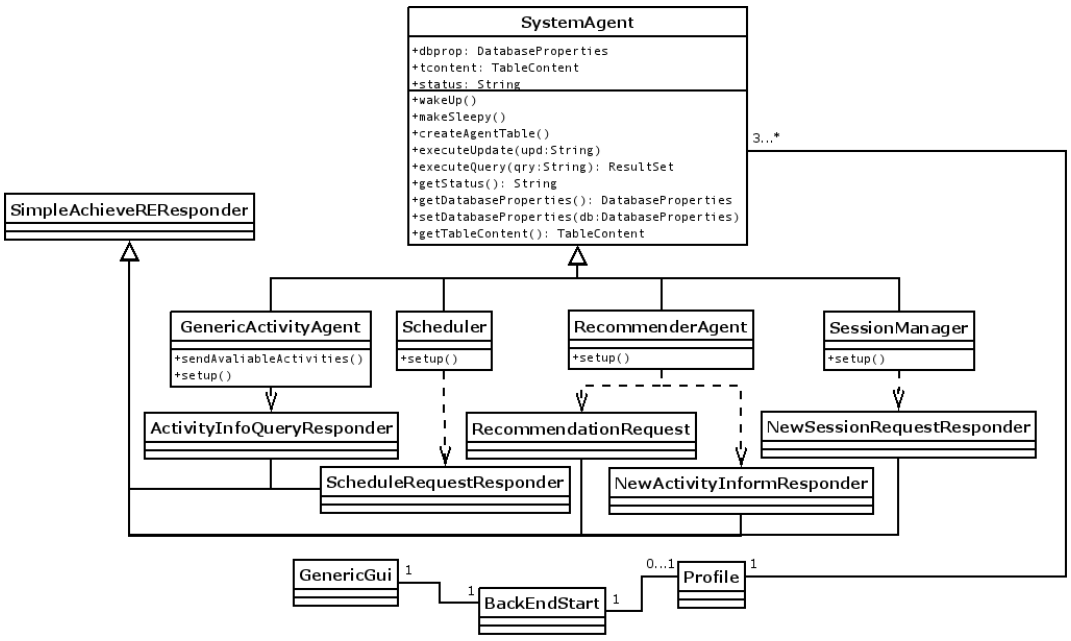


Figura 3.2: Diagrama UML BackEnd

3.2.5 Front End

Totes les capacitats del PFC anterior[1] van ser tingudes amb comte però, crec que s'havien d'eliminar funcionalitats, si més no, simplificar el menú principal de l'aplicatiu (que és el punt de partida del turista). Vaig decidir deixar únicament els següents punts:

- Modificació perfil.
- Sol·licitud recomanació.

- Veure agenda.
- Veure alertes.

A més, per tal de que els mapes poguéssin ser mostrats aprofitant al màxim la pantalla, necessitaria crear (o buscar) una nova interfície amb els menús menys exagerats que els que duu la J2ME. Llavors, vaig decidir utilitzar la File (Flexible Interface Rendering Engine). File és una libreria que conté els components bàsic per a desenvolupar aplicacions gràfiques per a J2ME amb una interfície simple i elegant. El que també era una qualitat important és que aquesta llibreria ens permet desenvolupar l'interfície sense preocupar-nos en adaptar-ho a cada dispositiu en particular ja que se'n encarrega la pròpia llibreria.

Posicionament

Un dels factors clau per a poder rebre informació sobre activitats, veure la nostra posició en el mapa i la de les activitats, és el posicionament. Hi haurà dos modes de funcionament.
En l'antecedent que utilitzava la plataforma mòbil [1], s'utilitzava la informació rebuda a través de la cel·la GSM. S'ha decidit no utilitzar aquest mètode per el gran error que podíem tindre en la nostra posició a més de que hauriem de tindre emmagatzemades totes les posicions associades a un identificador de cel·la amb el conseqüent malbaratament de memòria, quan, el GPS, ens retorna directament la nostra posició en longitud i latitud.

- Mode GPS. Funciona quan tenim un dispositiu GPS configurat en el nostre perfil, el dispositiu està disponible i la qualitat del senyal és superior al 30 per cent.
- Mode off-line. Funciona quan no tenim un dispositiu configurat o bé, si hi és, aquest no està disponible o bé la qualitat del senyal és inferior al 30 per cent. Es posiciona llegint l'última posició vàlida rebuda des del dispositiu GPS. És com un GPS "estàtic" on únicament podem navegar pel mapa i veure les activitats que hi ha en cada fragment d'aquest que tinguem actualment en la pantalla.

Aquesta tasca la durà a terme un fil independent de la nostra aplicació que, es dedicarà exclusivament a actualitzar la nostra posició al mapa, al perfil i a fer-ho saber a l'agent recomanador (per a poder rebre possibles activitats disponibles en la nova àrea).
Cada cop que s'actualitzi la posició de l'usuari, l'interfície s'encarregarà de carregar el nou mapa a la pantalla (i de mostrar les noves activitats si s'escau).
El tractament de mapes s'explica a la següent secció.

Tractament de mapes

El tractament de mapes és la part més important per a la orientació del turista. Això comportava 2 passos:

- Selecció del mapa i mostrar-ho a pantalla.
- Crear una capa al damunt per tal de marcar la nostra posició i la de les activitats.

Cada mapa té una coordenada associada, és a dir, quan s'actualitzen les coordenades de la nostra posició, es mira en quin rang de coordenades està el mapa i en funció d'això es carrega el fitxer .png que té les coordenades associades i es mostra a pantalla amb el corresponent desplaçament per a que la nostra posició en el mapa quedi centrat a pantalla.
Es poden utilitzar els mapes que es desitjin, en aquest treball, s'ha utilitzat com a font Google Maps. Per adaptar-ho a pantalla, hem d'agafar una combinació de 96 d'ample per 54 de llargada (veure apèndix per saber més sobre les especificacions J2ME). Com en el nostre exemple hem utilitzat Tarragona, s'ha decidit com a una mida del mapa de 1152 d'ample per 648 de llargada. S'ha tingut en compte també no sobrepassar els 512KB en la mida de la imatge de manera que la càrrega sigui relativament ràpida sense que l'usuari se'n doni compte ja que, quan arribem als extrems dels mapes, es carrega el seu contigu en segon pla per a que la transició sigui més suau.

Diagrama UML

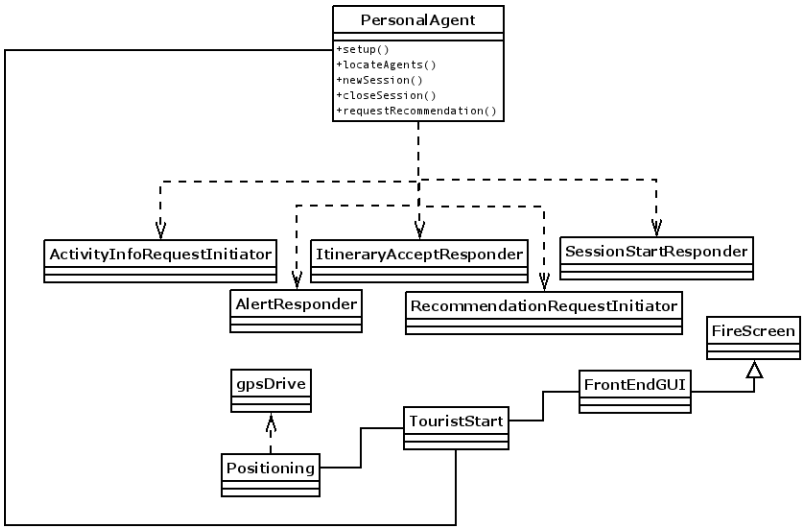


Figura 3.3: Diagrama UML FrontEnd

3.2.6 Sistema Multi-Agent

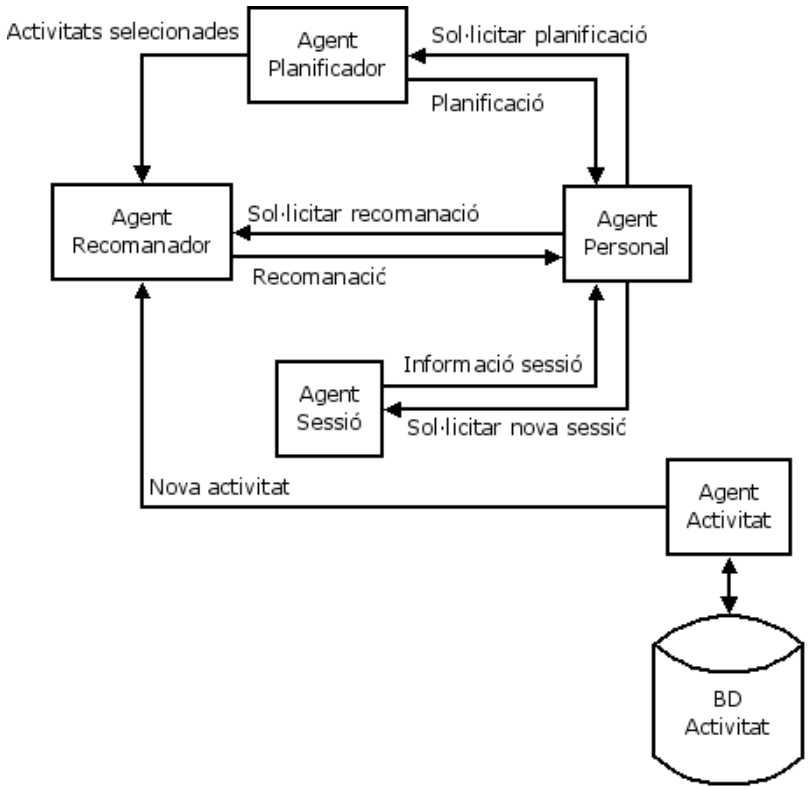


Figura 3.4: Esquema arquitectura Sistema Multi-Agent

Inicialització BackEnd

Cada cop que carreguem un arxiu de configuració i conectem el sistema, cadascun dels agents d'activitat, enviarà un XML semblant al següent:

```
<activitylist Agent="NOMAGENT">
 <activity Name="NOMACTIVITAT1" Id="ID" Time="TEMPS" DB="NOMBBDD">
  <grade Name="TIPUSACTIVITAT1" Value="VALORACIÓ1"></grade>
  <grade Name="TIPUSACTIVITAT2" Value="VALORACIÓ2"></grade>
  ...
  <grade Name="TIPUSACTIVITATN" Value="VALORACIÓN"></grade>
 </activity>
</activitylist>
```

Amb el següent contingut:

- NOMAGENT. Nom de l'Agent qui envia la informació (usat per crear identificador d'activitat únic dins el sistema).
- NOMACTIVITAT1. Nom de l'activitat a afegir.
- ID. Identificador únic, amb combinació amb el nom de l'agent és la clau primària.
- TEMPS. Horari de l'activitat
- NOMBBDD. Nom de la base de dades de l'Agent (usat només en depuració).
- TIPUSACTIVITAT1. Nom del tipus d'activitat

- VALORACIÓ1. Valoració de l'activitat

De tal manera que l'agent recomanador guardarà dins la seva base de dades quines són les activitats disponibles actualment. Els camps "TIPUSACTIVITAT1" i "VALORACIO1" són els que utilitzarà el Recomanador i n'hi haura tant parells com agents Activitat hi hagi (o diferent tipus d'activitats hi hagi definits).

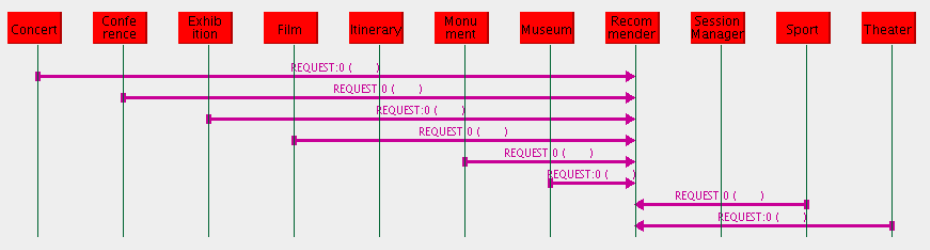


Figura 3.5: Enviament activitats disponibles a l'agents recomanador

Inici sessió usuari

Primer, l'agent Personal envia un missatge de petició de nova sessió, llavors, l'agent Sessió, en cas d'acceptar la sessió nova, li envia quines són les activitats disponibles actualment per a que l'usuari pugui crear el seu perfil.

```
<newsession>
<activity Name="ACTIVITAT1"></activity>
<activity Name="ACTIVITAT2"></activity>
...
<activity Name="ACTIVITATN"></activity>
</newsession>
```

Un cop l'usuari ha decidit quines són les seves preferències, li contesta amb el següent contingut:

```
<profile User="NOMUSUARI" Id="IDENTIFICADOR">
<activity Name="ACTIVITAT1" Grade="INTERÉS1"></activity>
<activity Name="ACTIVITAT2" Grade="INTERÉS2"></activity>
...
<activity Name="ACTIVITATN" Grade="INTERÉSN"></activity>
</profile>
```

On s'identifica amb el seu Nom i un Identificador únic. Dins l'XML envia els parells Nom activitat - Interès (del 0 al 5) que, posteriorment, es transformaran del 0 a l'1 per a poder aplicar l'algorisme de recomanació.

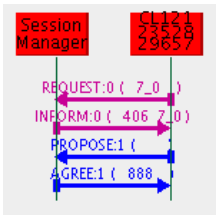


Figura 3.6: Inici sessió i enviament del perfil d'usuari

Tancament sessió usuari

L'usuari envia un missatge amb el codi de tancament de sessió per a que l'agent Sessió elimini de la seva taula el client.

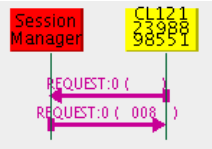


Figura 3.7: Tancament de sessió

Petició recomanació

La petició de recomanació també és una de les comunicacions més simples ja que, la gran major part de la feina la realitza l'agent Recomanador i, pràcticament, no hi ha més comunicació entre l'agent Recomanador i el Personal.

```
<activitylist>
<activity Name="10:00 Soul Machine" Id="CONCERT.1"></activity>
<activity Name="17:00 Extremoduro" Id="CONCERT.5"></activity>
...
<activity Name="22:00 Conchita" Id="CONCERT.4"></activity>
</activitylist>
```

Aquí tenim un exemple del fitxer XML que reb un determinat client amb una determinada recomanació.

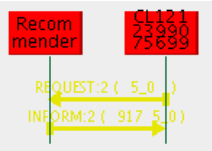


Figura 3.8: Petició recomanació

Petició informació detallada activitat

Sempre i quan hi hagin camps opcionals definits en algun dels agents activitats (i la base de dades així estigui definida), es podrà obtenir informació complementària sobre una determinada activitat que el turista pot seleccionar del llistat d'activitats planificades en la seva agenda.

```
<activityinfo Name="ACTIVITAT">
<field Name="FIELD1" Content="CONTINGUT1"></field>
<field Name="FIELD2" Content="CONTINGUT2"></field>
...
<field Name="FIELDN" Content="CONTINGUTN"></field>
</activityinfo>
```

Actualització posició (només en mode GPS)

Per a no sobrecarregar la xarxa amb paquets d'aquest tipus (ja que la posició en el terminal s'actualitza cada segon), s'ha decidit que aquesta informació únicament s'envia cada cop que el turista és mou la meitat de la mida de la pantalla, és a dir quan, es mou la finestra (o *viewport*) la meitat de la longitud o l'amplitud, l'agent Personal envia la informació sobre la posició.

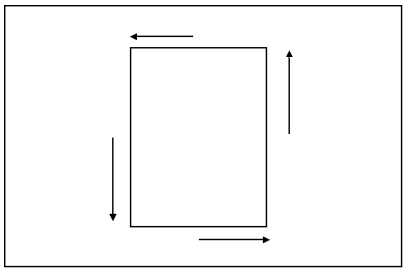


Figura 3.9: Actualització posició

```
<position lat="LAT" long="LONG">
</position>
```

Capítol 4

Manuals d'ús

Primer que tot, si volem que el sistema pugui ser capaç de recomanar activitats, hem de tindre una base de dades amb informació sobre aquestes. En segon lloc, ja podrem posar en marxa el BackEnd.
S'executa amb la comanda Turist@.bat (tant en GNU/linux com en Windows). Després procedim a carregar el fitxer .xml amb la definició del sistema. Tot seguit, ens apareixerà un llistat de tots

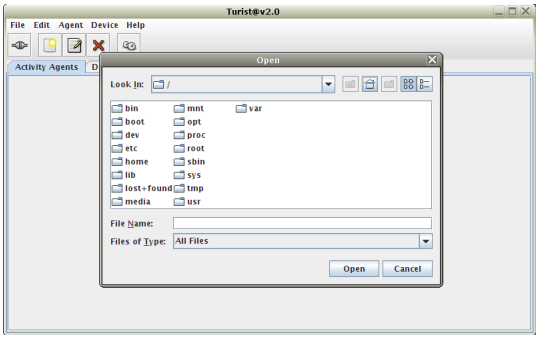


Figura 4.1: Càrrega fitxer de configuració

els agents configurats en el sistema.

4.0.7 Finestra principal BackEnd

Tret de carregar/guardar fitxer de configuració, la resta de menús estan replicats en la barra d'eines. La primera icona de la barra d'eines ens serveix per engegar/apagar el sistema. Les tres següents per afegir/modificar/expulsar un agent o bé un usuari del sistema. L'última ens posa en marxa el "sniffer" que permet seguir les conexions i el seu contingut a través dels agents.

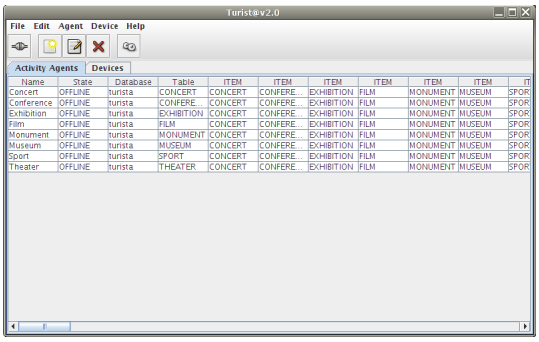


Figura 4.2: Càrrega fitxer de configuració

4.0.8 Finestra principal FrontEnd

La finestra principal del FrontEnd conté, bàsicament, el fons que serà on es pintarà el mapa de la nostra localització i les acitivitats planificades i el menú principal que contindrà les següents opcions:

- Recommend. Sol·licita una recomanació d'acord amb els nostre perfil.
- Schedule. Visualitza la nostra planificació (agenda).
- Alerts. Visualitza les alertes pendents de noves activitats (trobades o afegides al sistema).
- Profile. Visualitza/Modifica el perfil de l'usuari actual.
- Exit. Finalitza l'aplicació

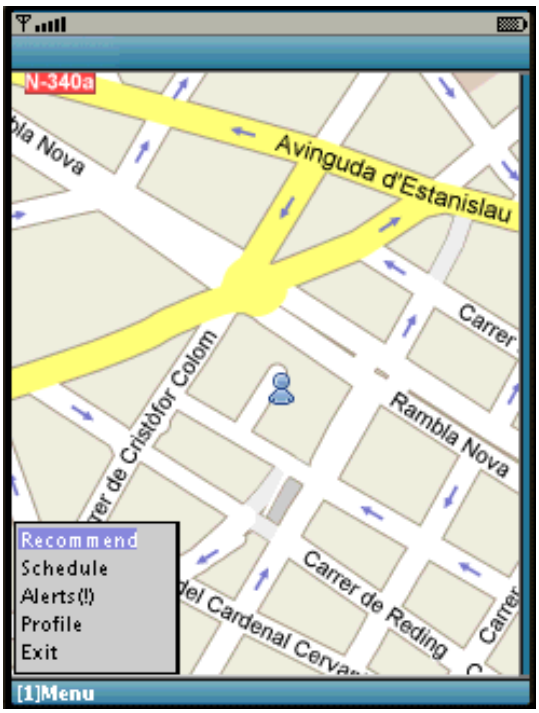


Figura 4.3: Càrrega fitxer de configuració

Capítol 5

Avaluació

5.1 Casos pràctics

Per als nostrs casos pràctics, suposarem configurat el sistema amb 8 activitats:

- Concert

```
+--------------+-----------+------------+----------+--------+
| NAME         | ADDRESS   | DATE       | TIME     | LENGTH |
+--------------+-----------+------------+----------+--------+
| Soul Machine | Tarragona | 0000-00-00 | 10:00:00 |      0 |
| Extremoduro  | Tarragona | 0000-00-00 | 17:00:00 |      0 |
| Conchita     | Tarragona | 0000-00-00 | 22:00:00 |      0 |
+--------------+-----------+------------+----------+--------+
```

- Conference

```
+---------------------------------------+-----------+------------+----------+--------+
| NAME                                  | ADDRESS   | DATE       | TIME     | LENGTH |
+---------------------------------------+-----------+------------+----------+--------+
| Quatre campanyes del P. Claret a      | Tarragona | 0000-00-00 | 00:00:05 |      0 |
| L'estÃmul a la creaci artstica       | Tarragona | 0000-00-00 | 00:00:05 |      0 |
| El P. Claret, confessor reial, m      | Tarragona | 0000-00-00 | 00:00:05 |      0 |
+---------------------------------------+-----------+------------+----------+--------+
```

- Exhibition

```
+-----------------------------------+-----------+------------+----------+--------+
| NAME                              | ADDRESS   | DATE       | TIME     | LENGTH |
+-----------------------------------+-----------+------------+----------+--------+
| Estan jugant a Meccano al Parc C  | Tarragona | 0000-00-00 | 00:00:05 | NULL   |
| ExhibiciÃ Club NataciÃn Tarraco   | Tarragona | 0000-00-00 | 00:00:05 | NULL   |
| Tuning Rave Catalonia             | Tarragona | 0000-00-00 | 00:00:05 | NULL   |
+-----------------------------------+-----------+------------+----------+--------+
```

- Film

```
+----------------------+-----------+------+------+--------+
| NAME                 | ADDRESS   | DATE | TIME | LENGTH |
+----------------------+-----------+------+------+--------+
| 88 minutos           | Tarragona | NULL | NULL | NULL   |
| Algo pasa en Las Vegas | Tarragona | NULL | NULL | NULL   |
| Iron Man             | Tarragona | NULL | NULL | NULL   |
+----------------------+-----------+------+------+--------+
```

- Monument

```
+-------------+-----------+------+----------+--------+
| NAME        | ADDRESS   | DATE | TIME     | LENGTH |
+-------------+-----------+------+----------+--------+
| La Muralla  | Tarragona | NULL | 00:00:00 |        |
| FÃrum       | Tarragona | NULL | 00:00:00 |        |
| Amfiteatre  | Tarragona | NULL | 00:00:00 |        |
+-------------+-----------+------+----------+--------+
```

- Museum

```
+------------------------------+-----------+------+------+--------+
| NAME                         | ADDRESS   | DATE | TIME | LENGTH |
+------------------------------+-----------+------+------+--------+
| Museu d'art moderm           | Tarragona | NULL | NULL | NULL   |
| Museu Naciopnal AruqeolÃgic  | Tarragona | NULL | NULL | NULL   |
| Museu paleocristiÃ           | Tarragona | NULL | NULL | NULL   |
+------------------------------+-----------+------+------+--------+
```

- Sport

```
+---------------------+-----------+------------+----------+--------+
| NAME                | ADDRESS   | DATE       | TIME     | LENGTH |
+---------------------+-----------+------------+----------+--------+
| Bicicletada popular | Tarragona | 0000-00-00 | 00:00:05 | NULL   |
| Caminada            | Tarragona | 0000-00-00 | 00:00:05 | NULL   |
| Mitja maratÃ        | Tarragona | 0000-00-00 | 00:00:05 | NULL   |
+---------------------+-----------+------------+----------+--------+
```

- Theater

```
+---------------------------------+-----------+------+------+--------+
| NAME                            | ADDRESS   | DATE | TIME | LENGTH |
+---------------------------------+-----------+------+------+--------+
| El Rock, reflex d'una societat  | Tarragona | NULL | NULL | NULL   |
| AixÃ no Ã©s vida                 | Tarragona | NULL | NULL | NULL   |
| Contraban                       | Tarragona | NULL | NULL | NULL   |
+---------------------------------+-----------+------+------+--------+
```

Suposem també que s'ha carregat el sistema de configuració:

```
 <?xml version="1.0" encoding="UTF-8"?>
<!-- Agents Description -->
<!-- Decide whether one different database per agent could be possible or not --
<SystemDescription Name="Turist@1">
    <Database Name="turista">
        <Host Port="3306"  Name="localhost"></Host>
        <User Name="admin" Pwd="passwd"></User>
    </Database>
    <!-- Agent description fields -->
    <AgentProperties>
        <Property Name="ID_ACTIVITY" Type="INTEGER NOT NULL AUTO_INCREMENT"></Pr
        <Property Name="COORDX" Type="INTEGER"></Property>
        <Property Name="COORDY" Type="INTEGER"></Property>
        <Property Name="NAME" Type="VARCHAR(32)"></Property>
```

```
            <Property Name="ADDRESS" Type="VARCHAR(100) NULL"></Property>
            <Property Name="TIME" Type="VARCHAR(100) NULL"></Property>
            <Property Name="DATE" Type="VARCHAR(100) NULL"></Property>
            <Property Name="LENGTH" Type="VARCHAR(100) NULL"></Property>
        </AgentProperties>
        <AgentsDescription>
            <Agent Name="Concert" ActName="CONCERT" TableName="CONCERT"></Agent>
            <Agent Name="Conference" ActName="CONFERENCE" TableName="CONFERENCE"></A
            <Agent Name="Exhibition" ActName="EXHIBITION" TableName="EXHIBITION"></A
            <Agent Name="Film" ActName="FILM" TableName="FILM"></Agent>
            <Agent Name="Monument" ActName="MONUMENT" TableName="MONUMENT"></Agent>
            <Agent Name="Museum" ActName="MUSEUM" TableName="MUSEUM"></Agent>
            <Agent Name="Sport" ActName="SPORT" TableName="SPORT"></Agent>
            <Agent Name="Theater" ActName="THEATER" TableName="THEATER"></Agent>
        </AgentsDescription>
    </SystemDescription>
```

Un cop fet això i, amb el sistema en marxa i tots els agents connectats, procedim a fer les demostracions.

5.1.1 Cas pràctic 1, Tarragona ciutat

El primer cas pràctic se centra en la ciutat de Tarragona. Configurem un perfil amb les següent característiques:

- Nom: Cas1
- Edat: 25
- Interès en Concerts: 5
- Interès en Conferències: 1
- Interès en Exibicions: 1
- Interès en Pel·lícules: 4
- Interès en Monuments: 1
- Interès en Museus: 1
- Interès en Esports: 4
- Interès en Teatres: 1

A més de l'adreça MAC del dispositiu GPS Bluetooth (si en disposem d'un). Tot seguit quan guardem el perfil, ens mostrarà la nostra ubicació en el mapa de Tarragona. Ara tenim el FrontEnd connectat amb el BackEnd on, aquest, té el perfil que hem creat, preparat per a fer una recomanació quan nosaltres li sol·licitem. Demanen una recomanació turística i ens retorna el llistat d'on podem seleccionar les activitats que volem incloure en la nostra agenda. Les activitats queden programades i, automàticament, el mapa ens mostra la ubicació d'aquestes com a icones de banderes blaves.

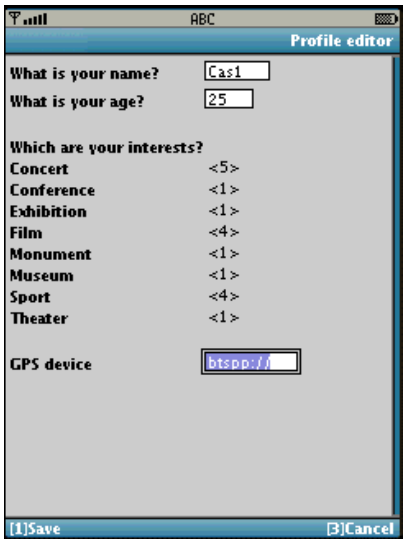


Figura 5.1: Configuració perfil usuari

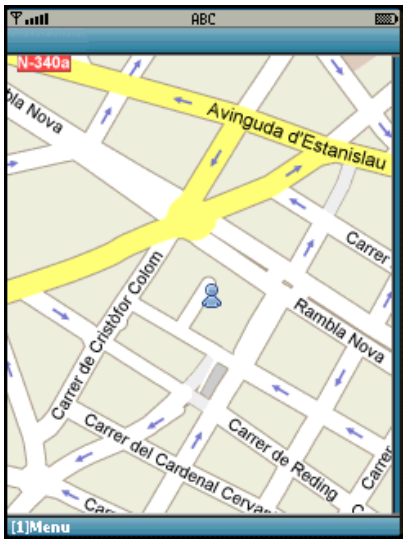


Figura 5.2: Mapa Tarragona

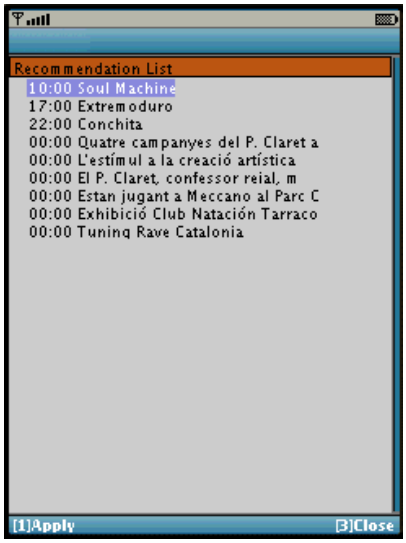


Figura 5.3: Llistat activitats recomanades

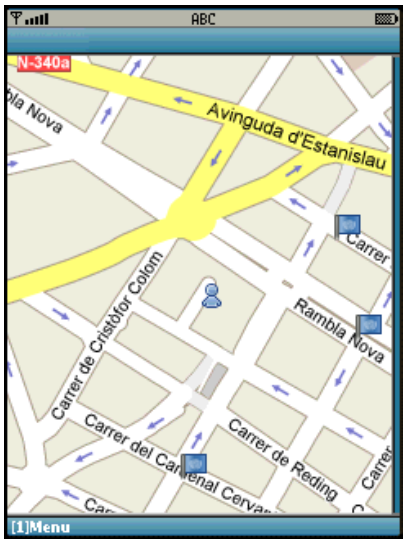


Figura 5.4: Mapa amb les activitats planificades senyalades

Capítol 6

Conclusions i agraïments

Com era d'esperar, no ha estat una tasca senzilla i, encara menys, quan tens uns antecedents del projecte. Suposant que el codi fóra prou bo, que els algorismes es puguéssin reutilitzar i que, els objectes que havien creats estiguéssin el suficientment encapsulats, la meva feina hagués sigut molt més simple. No obstant, m'he trobat en codi poc comentat i poc intuïtiu, molta repetició del mateix codi per a donar solucions diferents. L'única excusa que trobo, és que els treballs estaven purament orientats als Sistemes Multi-Agents i prescindien de tot els que ens han ensenyat durant la carrera (esquemes entitat-relació, esquemes UML, encapsulat i abstracció...). Òbviament el que es valora (i més en l'ambit de l'informàtica) és el resultat final i, amb això, no menyspreo en cap moment cap dels treballs anteriors ni tan sols puc dir que hagin comès un error sinò que han creat aplicacions de cara al consumidor i no de cara a una (possible) futura reutilització del codi. En quan a les idees, és amb el que m'he basat per donar continuïtat al projecte.

6.1 J2ME i dispositius mòvils

Probablement, aquest ha sigut un problema amb el que tant l'Àlex Viejo [1] com jo ens hem topat i és la falta d'homogeneïtat que hi ha dins el món del desenvolupament d'aplicacions per a entorns mòbils. Sun, amb J2ME, està intentant facilitar-nos una mica més la feina però sempre hi haurà la mateixa utopia que la que tenim per desenvolupar una mateixa eina que funcioni tant en Windows com en Linux i, això encara és molt mes exagerat quan hi ha diversos fabricants de dispositius mòbils, diverses llibreries, sistemes operatius... Una eina com la que s'ha desenvolupat, és viable utilitzar J2ME no obstant, per a aplicacions d'alt rendiment (com ara jocs 3D) resulta completament frustrant utilitar aquesta tecnologia.

6.2 Sistemes multi-agent

El projecte en conjunt és bastant més que un sistema multi-agent ja que, si ens centrem únicament en la part dels sistemes multi-agent, aquest projecte és un exemple molt senzill d'un sistema multi-agent ja que únicament implementat un model client-servidor on, un agent (servidor) té la font d'informació i l'altre (el client) realitza la petició per obtenir aquesta informació per a processar-la posteriorment. En aquest cas disposavem d'una arquitectura descentralitzada que ens permet executar la part més feixuga de la feina en el BackEnd i utilitzar un FrontEnd per a (com el nom indica) mostrar la informació a l'usuari.

Per acabar m'agradaria fer patent la feina que estan fent dins el "Grup de Tecnologies per a la Gestió Avançada del Coneixement, iTAKA" així com en la resta de grups de recerca de la Universitat Rovira i Virgili; crec que totes i cadascunes de les petites idees que creen és el que perdura i fa que tot plegat cobri una mica més de sentit. I, per acabar, agraïr sobretot la paciència que David Isern ha volcat sobre la meva feina.

Capítol 7

Recursos utilitzats

7.1 J2SE

Java ha sigut la principal eina utlitzada per a tot el desenvolupament del programa, en concret, s'ha utilitzat l'última versió (1.6.06) de la JDK.

7.2 J2ME

Donat que l'aplicatiu (tant el FrontEnd com el BackEnd) ha estat desenvolupat sota GNU/linux, s'ha utilitzat la versió 2.2 de la llibreria per a dispositius CLDC i MIDP per problemes de compatibilitat. No he necessitat utilitzat cap recurs que em brindés una llibreria més nova (per exemple la 2.5.2) ja que he utilitzat unes llibreries pròpies per al sistema de posicionament que més endavant detallo.

7.3 JADE i JADE-LEAP

JADE ha estat la principal eina amb combinació amb LEAP que m'ha permès muntar tota la infraestructura del sistema multi-agent, per a més informació sobre ambdos paquets, veure els apèndix A.1 i A.2.

7.4 FireScreen

FireScreen és una llibreria gràfica, de baix nivell, que ens permet una sèrie de característiques:

- Pantalla completa.
- Aprofitament pantalla.
- Animació menús.

7.5 GNUBox

GNUBox és una petita utilitat amb la que ens podem conectar a internet amb el nostre mòvil Symbian a través d'un ordinador i el port Bluetooth o USB.

7.6 Làtex

(o LaTeX en escriptura normal, habitualment pronunciat [láteX] o bé [léiteX], però no acabat en [ks]) és un conjunt de macros de TeX, escrites per Leslie Lamport (LamportTeX) el 1984, amb la intenció de facilitar l'ús del llenguatge creat per Donald Knuth, (TeX), al qual no només modifica sinó que complementa.

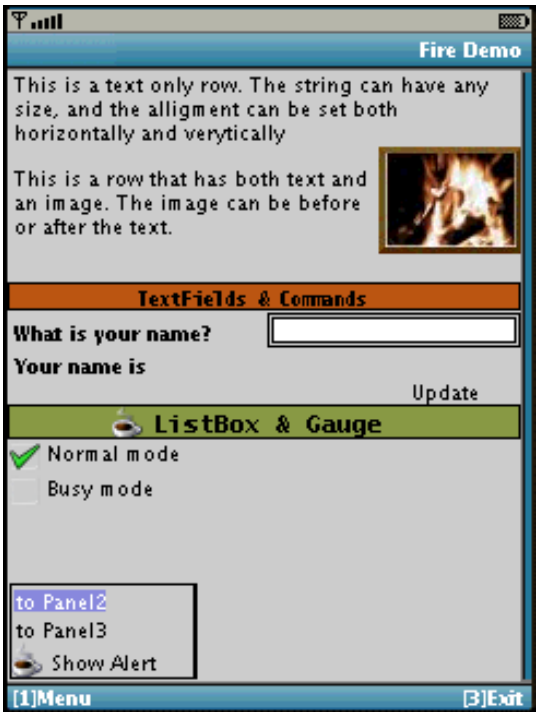


Figura 7.1: Exemple FireScreen

La idea principal de LaTeX és ajudar qui escriu un document a centrar-se en el contingut més que en la forma. És molt utilitzat per a la composició de tesis i llibres tècnics ja que la qualitat tipogràfica dels documents realitzats amb LaTeX és comparable a la d'una editorial científica de primera línia. LaTeX és programari lliure sota llicència LPPL i està disponible per a la majoria de plataformes actuals (Linux, Unix, Windows, Mac, etc.)
La manera en què LaTeX interpreta el format que ha de tenir el document és mitjançant «etiquetes». Per exemple, "documentclassarticle" li diu a LaTeX que el document que processarà és un article. Pot resultar estrany que avui en dia es continuï usant un sistema que no és WYSIWYG (what you see is what you get, «el que veus és el que obtens») però les característiques de LaTeX continuen sent moltes i molt variades i, en molts casos, supera àmpliament en qualitat tipogràfica els paquets més habituals. També hi ha diverses eines (aplicacions) que ajuden una persona a escriure aquests documents d'una manera més visual (LyX, TeXmacs, TeXnic Center i d'altres); aquestes eines se les podria anomenar WYSIWYM (what you see is what you mean, «el que veus és el que volies dir»). Cal remarcar que l'escriptura i la visualització de fórmules matemàtiques de la Viquipèdia es realitza amb LaTeX.
LaTeX es pronuncia habitualment com ?latekh? amb la j final com el so alemany de la ch: en realitat aquest darrer caràcter és un lletra grega ? majúscula, ja que el nom TeX deriva del grec ????? (habilitat, art, tècnica). Normalment s'imprimeix amb el logo tipogràfic de la imatge superior, però si no es pot escriure així s'acostuma a representar com LaTeX per evitar confusió amb el nom comú l'làtex".

Apèndix A

A.1 JADE

Per implementar aquest projecte s'ha usat Jade. JADE (Java Agent Development Enviroment) és un entorn de construcció de Sistemes Multiagent, conté un conjunt de llibreries escrites en JAVA i proporciona un conjunt de packages JAVA que permeten crear agents. A més de proporcionar les funcions de manejament d?agents a baix nivell i les interfícies que faciliten la programació, també ens presenta un entorn d?execució on els agents podran executar-se i interactuar. La versió utilitzada per a la implementació d?aquest projecte és la 3.01. JADE es composa d'un conjunt de paquets que descriurem a continuació:

- jade.core: conté el nucli del sistema. Proporciona la classe jade.core.Agent que és imprescindible per a la creació de SMA. També conté el paquet jade.core.behaviours que inclou les classes de comportaments suportats.
- jade.lang: conté les classes específiques per a suportar un llenguatge de comunicació entre agents. Està format pel paquet jade.lang.acl.
- jade.content: conté les classes específiques per a suportar un llenguatge de comunicació entre agents (ACL), així com la utilització i definició d?ontologies. Està format pels paquets jade.content.lang i jade.content.onto.
- jade.domain: conté totes les classes per representar la plataforma d?agents i els models del domini, tal com entitats per a l?administració d?agents, llenguatges i ontologies (AMS, DF, ACC ...).
- jade.proto: paquet que conté els protocols d?interacció entre agents FIPA.
- jade.tools: trobem algunes eines que faciliten el desenvolupament d?aplicacions i l?administració de la plataforma:
  - Remote Management: Agent (RMA): actua com una consola gràfica per a l?administració i control de la plataforma.
  - Dummy Agent: és una eina de monitorització i depuració, composta per una interfície gràfica i un agent JADE. Permet crear missatges ACL i enviar-los a altres agents, podent visualitzar-los.
  - Introspector Agent: eina que permet la monotorització del cicle de vida d?un agent.
  - Sniffer: agent que intercepta missatges ACL i els visualitza.
  - SocketProxyAgent: agent que actua com un gateway bidireccional entre la plataforma JADE i una connexió TCP/IP.
- jade.gui: conté eines per construir interfícies gràfiques.

A.1.1 Plataforma d'agents

A JADE, els agents resideixen en un entorn predefinit anomenat plataforma. Cada plataforma està dividida en contenidors i cadascun d?ells contindrà agents. Els contenidors poden entendre?s com dominis d?agents. La plataforma s?engega en una màquina determinada, mentre que

els agents poden residir en màquines diferents o no. Jade proporciona un entorn segur perquè els agents es puguin comunicar. Tal com es defineix a la FIPA [FIPA, 1998b], en cada plataforma es defineixen dos agents vitals pel funcionament del sistema: el Domain Facilitator (DF), que proporciona un servei de Pàgines Grogues i l? Agent Management System (AMS, que proporciona un servei de Pàgines Blanques). Gràcies a aquests serveis, si un agent A vol enviar un missatge a un agent B que desenvolupa la tasca T, l?agent A preguntarà al DF quin o quins agents poder fer-se càrrec de la tasca T. Quan es vulgui conèixer l?adreça d?un agent determinat, preguntarà a l?AMS que és qui coneix tots els agents de la plataforma.

A.1.2 Agents

Una agent JADE és un objecte de la classe Agent. Per crear un agent hem d?estendre la classe Agent i implementar els mètodes que són obligatoris de la classe. Dos mètodes molt importants que s?han d?implementar necessàriament són:

- setup() : aquest mètode s?executa quan l?agent entra al sistema per primera vegada. Inicialitza l?agent i posa en marxa comportaments.
- takeDown() : aquest mètode s?executa quan se surt del sistema. Finalitza l?agent.

A.1.3 Llenguatge de comunicació

El llenguatge FIPA ?SL (Semantic Language) és un llenguatge formal amb el qual es pretén poder especificar una gran varietat de problemes. En FIPA-SL es poden formar expressions lògiques, intercanviar informació entre agents i expressar accions a realitzar. La notació utilitzada és semblant al Lisp: es defineixen un conjunt de predicats genèrics i cadascun té una sèrie d?atributs i paràmetres.

A.1.4 Comportaments

Per a donar funcionalitat a un agent s?hauran de definir comportaments (behaviours). Cal implementar com a mínim un comportament per agent. Per cada servei que hagi de donar un agent s?implementarà un o més comportaments. Jade proporciona un conjunt de comportaments amb els quals podem treballar. Un scheduler intern controlarà l?execució de tots els comportaments. N?hi ha de dos tipus:

- Comportaments simples: no tenen fills. S?agrupen en la classe abstracta SimpleBehaviour. representa un comportament atòmic que s?executa sense interrupcions. Pot ser dels tipus:
  - OneShotBehaviour: si s?executa una sola vegada.
  - SenderBehaviour: comportament per enviar missatges ACL. S?executa el mètode send() de manera atòmica.
  - CyclicBehaviour: si executa una acció cíclicament.
  - TickerBehaviour : periòdicament executa un tros de codi indicat per l?usuari.
  - WakerBehaviour: és una OneShot task que s?executarà només després de produir-se un time-out.
- Comportaments compostos: poden tenir fills, s?agrupen en la classe abstracta
  - CompositeBehaviour. En té tres subclasses:
  - SequencialBehaviour: executa tots els seus fills de manera seqüencial i acaba quan tots els fills acaben.
  - ParallelBehaviour: executa els fills concurrentment, Acaba quan es compleix una condició, per exemple, quan tots els fills han acabat, N fills han acabat o algun fill ha acabat.
  - FSMBehaviour: executa els fills d?acord a una Màquina d?Estats Finits definida per l?usuari.
- ReceiverBehaviour: és un comportament per rebre missatges ACL. S?executa el mètode receive() de manera atòmica.

A.1.5 Ontologia

El contingut dels missatges entre agents són d?una determinada ontologia i s?especifica al camp ontology. L'ontologia és molt important, perquè és el vocabulari que s?usarà en els missatges. Una ontologia està formada per tres tipus d?elements: Una ontologia està formada per objectes o frames: cada frame està compost per atributs. Cada atribut o slot és un camp d?informació en un objecte, un slot pot ser un altre frame, un objecte bàsic o un conjunt d?elements. Cada slot està etiquetat i s?hi podrà accedir pel nom i pel tipus. Els objectes estan formats per dues parts:

- Definició dels slots: Els slots contenen el nom del slot, el tipus, l?estructura de dades que el manegarà, i si és un slot que serà obligatori o opcional en els missatges.
- Definició de la classe Java: s?ha de definir una classe que s?encarregarà de manegar cada objecte o predicat. Aquesta classe haurà de contenir mètodes set i get per a cada slot que haguem definit. Aquestes funcions seran les que ens permetran consultar i omplir les dades del missatge.

Amb tots aquest elements es poden crear totes les entitats que formaran la ontologia i que es classifiquen en tres grups:

- Conceptes: defineixen els objectes
- Predicats: defineixen una sintaxi lògica que permet fer peticions en base a un conjunt de restriccions.
- Accions: es refereixen als serveix que poden oferir els agents.

A.1.6 Protocols de comunicació

JADE proporciona els elements necessaris (behaviours i performatives) per definir els protocols descrits per la FIPA [FIPA, 1998b]. Bàsicament els protocols tenen dos parts, la part de qui fa la crida del protocol (initiator) i la part de qui manega la resposta d?aquest (responder). Cada agent haurà d?estendre els comportaments en funció del seu paper. JADE no implementa tots els protocols que defineix la FIPA. Veiem els més importants:

Protocol FIPA Request

En aquest protocol un agent demana a un altre agent que realitzi una acció i que li retorni els resultats. En la Fig. 34 podem veure el protocol FIPA Request. Els quadres blancs representen l'initiator i els grisos el responder.

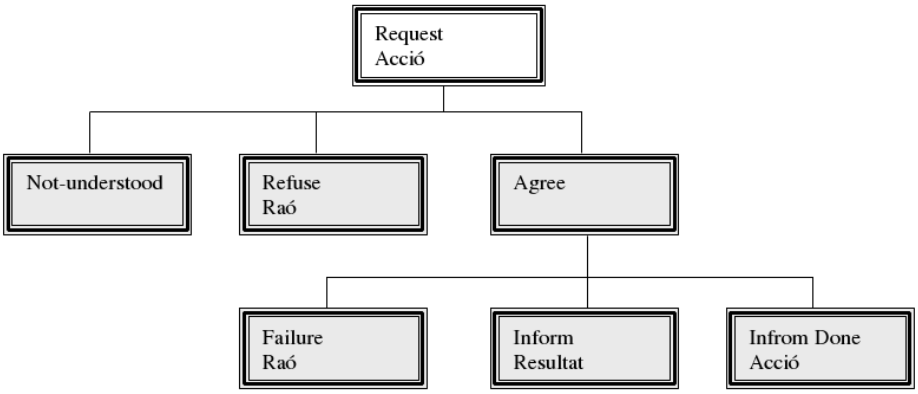


Figura A.1:

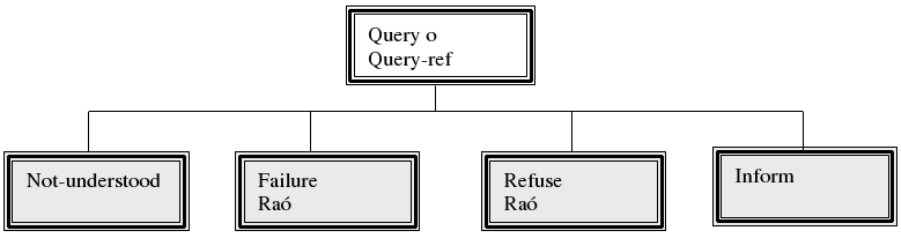


Figura A.2:

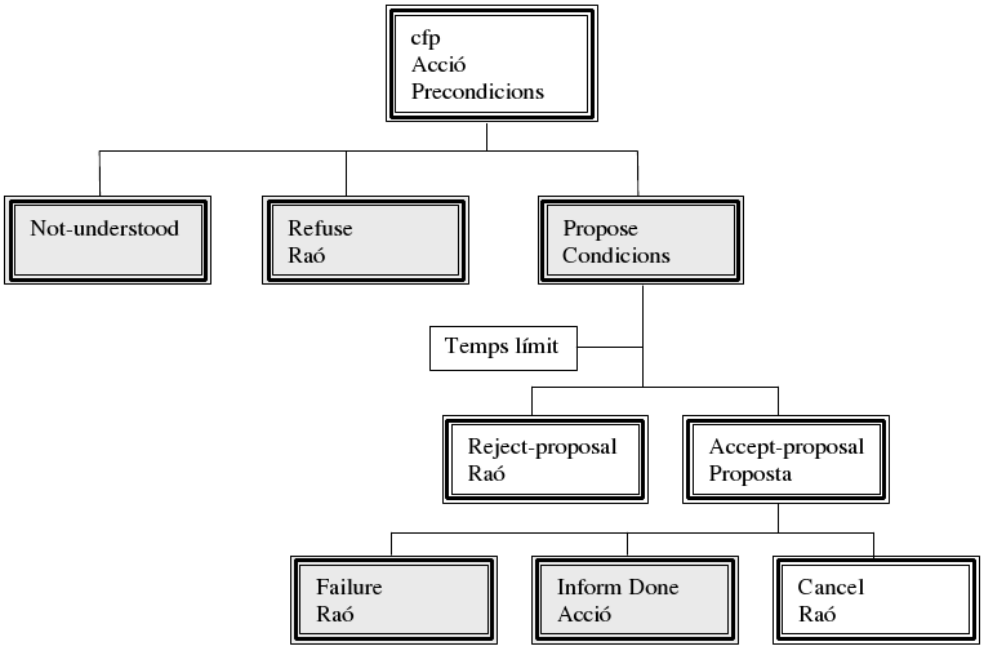


Figura A.3:

Protocol FIPA Query

S'utilitza quan un agent vol demanar algun tipus d'informació a un altre agent. En la Fig. 35 podem veure el protocol FIPA Query. Els quadres blancs representen l'initiator i els grisos el responder.

Protocol FIPA Contract Net

Aquest protocol s'utilitza quan es vol realitzar una negociació entre diversos agents. En la Fig. 36 podem veure el protocol FIPA Contract Net. Els quadres blancs representen l'initiator i els grisos el responder. Jade proporciona diferents classes per a poder manipular els protocols de comunicació:

- AchieveREInitiator /AchieveREResponder: implementen la banda del initiator i del responder respectivament per els protocols FIPA-Request, FIPA-Query, FIPA-Propose, FIPA-Request-When, FIPA-Recruiting, FIPA-Brokering. Existeix una variant simplificada d'aquestes classes: SimpleAchieveREInitiator i SimpleAchieveREResponder.
- ContractNetInitiator / ContractNetResponder: implementen l?initiator i el responder del protocol FIPA-Contract-Net.

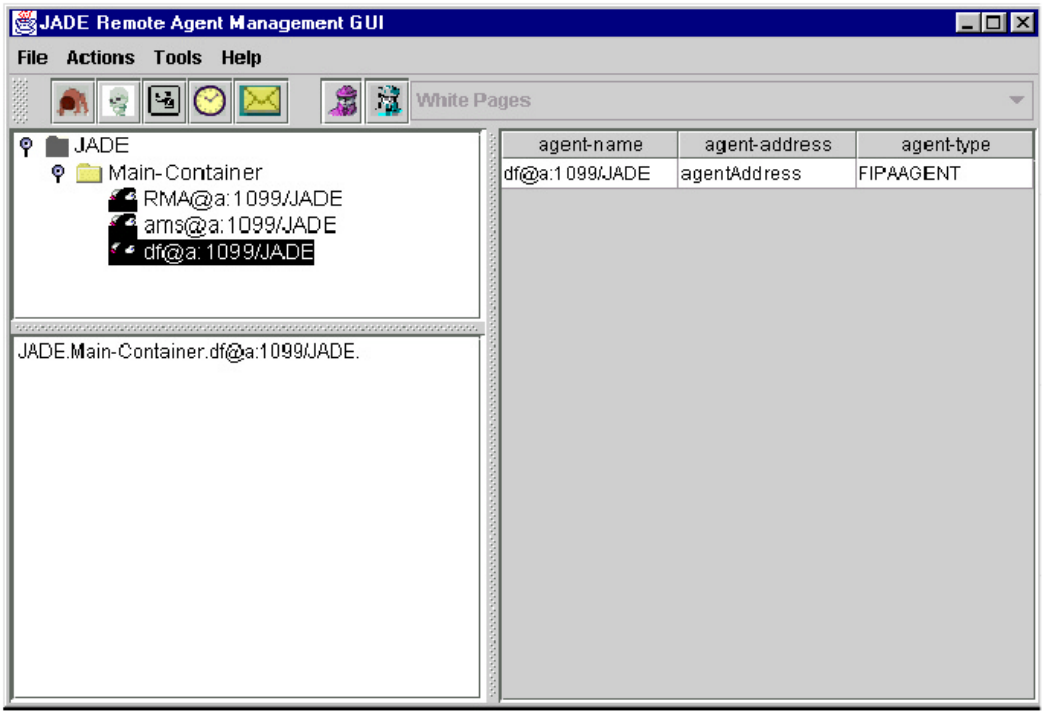


Figura A.4:

- SubscriptionInitiator / SubscriptionResponder : implementen l?initiator i el rsponder del protocol FIPA-Subscribe.

A.1.7 Eines de desenvolupament JADE

JADE ofereix una interfície gràfica per a l'administració de la plataforma, així com eines per seguir el funcionament del sistema Multi-Agent. Per iniciar la sessió i els agents hem d'escriure la comanda: java jade.Boot -platform [opcions] [llista d'agents] Si es vol inicialitzar l?entorn gràfic cal posar a les opcions el paràmetre -gui. En engegar-se l'entorn gràfic es creen uns quants agents automàticament:

- L'agent RMA: és l'encarregat de controlar la interfície.
- El DF: on es subscriuran els serveis dels agents.
- L'AMS: on es guardaran les adreces dels agents.

La llista d?agents a engegar ha de seguir els seguent format: ¡nom agent¿:ClasseJava

RMA

L'agent RMA (Remote Agent Management) s'ocupa de la interfície gràfica d'una plataforma d'agents. Mostra l'estat de la plataforma a la que pertany i proporciona eines per a demanar accions i per depurar i fer un test d'aplicacions basades en JADE. A continuació podem veure la interfície gràfica d'un RMA:

Sniffer

És una aplicació JAVA que rastreja l'intercanvi de missatges en un entorn basat en JADE. Permet veure en la seva interfície gràfica (Fig. 38) els missatges que es van passant entre els agents. Si seleccionem alguna de les fletxes que simbolitzen aquest intercanvi, se'ns mostrarà el missatge amb tots els seus paràmetres. És una aplicació JAVA que rastreja l'intercanvi de missatges en un entorn basat en JADE..

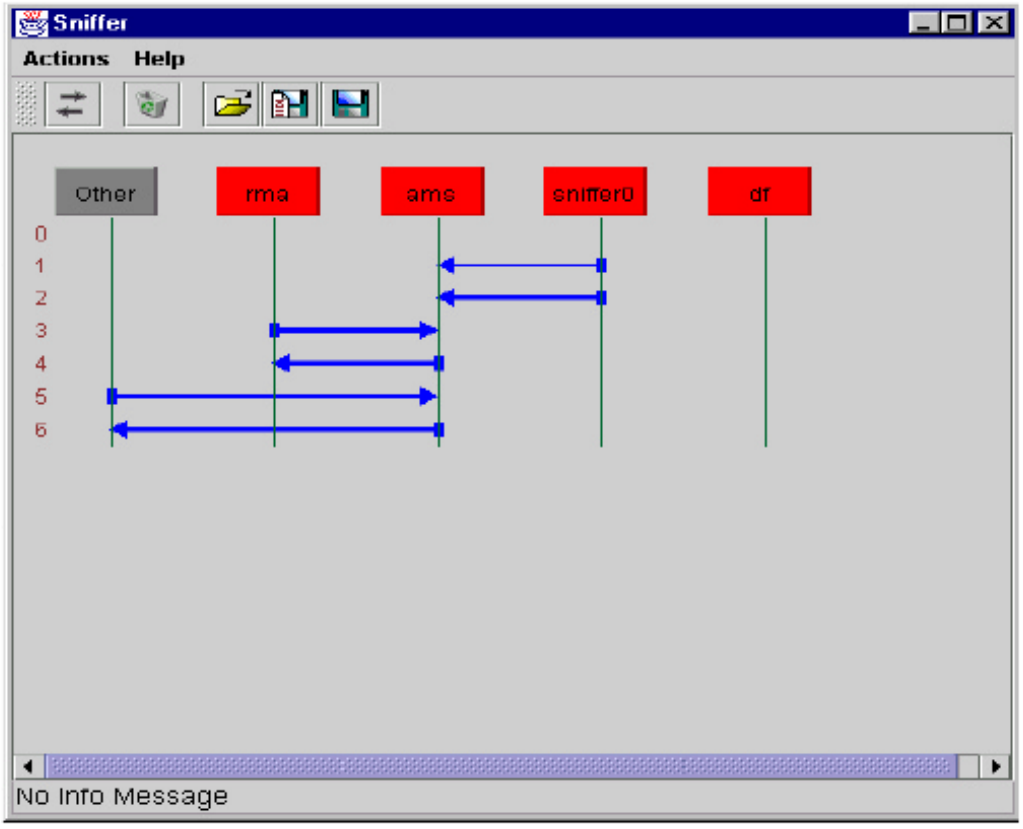


Figura A.5:

Dummy Agent

Aquest agent permet enviar missatges ACL entre els agents, rebre'ls i inspeccionar-los i llegir i salvar-los a un fitxer.

Introspector Agent

Ens permet monotoritzar el cicle de vida del agent.

DF

També podem activar l'interfície de l'agent DF (des del menú tools), on podrem observar els agents que s'han registrat i quin serveis ofereixen.

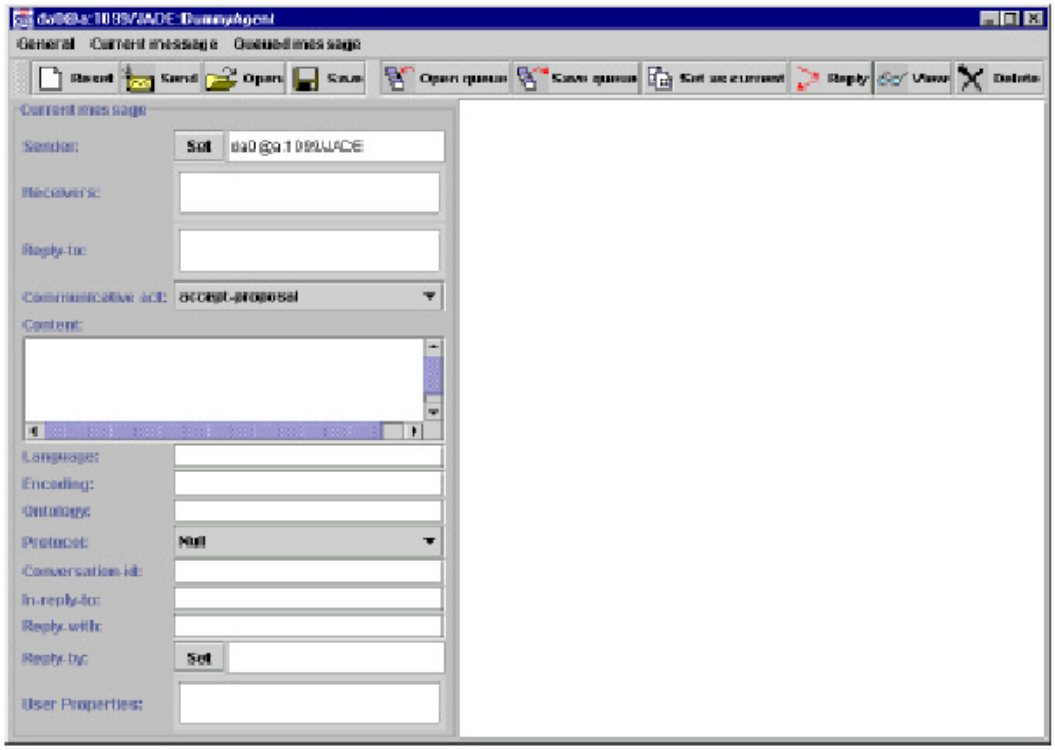


Figura A.6:

A.2 JADE-LEAP

L'afegitó "LEAP", combinat amb JADE, reemplaça algunes part del nucli de JADE formant un entorn d'execució que identifiquem com a JADE-LEAP ("JADE potenciat per JADE") i pot ésser implementat en un extens rang de dispositius des de servidors fins a telèfons mòbils amb capacitat per executar programes Java. PEr tal d'aconseguir això, JADE-LEAP pot ser "moldejat" de tres difetents formes per a tres entorns Java diferents:

- j2se: per a execució en ordinadors d'ús personal i servidors amb JDK1.4 o superior.
- pjava: per a execució en dispositius mòbils que suportin J2ME CDC o PersonalJava, com quasi totes les PDA existent en el mercat. Tenir en compte que, al 2003, l'especificació de "Personal Java" va ser declarada obsoleta i va ser substituïda per la configuració CDC de l'edició J2ME de Java. Des del punt de vista de l'execució de JADE-LEAP, no hi ha cap diferència.
- midp: per a l'execució en dispositius mòbils que suportin únicament MIDP1.0 (o superior), la gran majoria dels telèfons mòbils existent en el mercat.

A més a més, existeix una versió de JADE-LEAP per a la plataforma .NET de Microsoft no obstant, no ens serà del nostre interès pel que obviarem parlar-ne. Poques capacitats que estan presents en les versions j2se i pjava de JADE-LEAP no hi estan en la plataforma midp per les pròpies limitacions de les llibreries de J2ME. El punt de vista de programador i usuaris JADE-LEAP per a la plataforma j2se i per .NET és pràcticament el mateix en termes de l'API i de l'administració de l'execució. Per això, els programadors poden desenvolupar els seus agents JADE sota l'afegitó JADE-LEAP i viceversa sense canviar ni una línia de codi. Per altra banda, s'ha de tenir en compte que els contenidors JADE i JADE-LEAP no poden ser barrejats en una mateixa plataforma. Per poder comunicar una plataforma JADE amb una JADE-LEAP sino que ho hem der fer tal i com indica la FIPA utilitzant el HTTP MessageTransportProtocol.

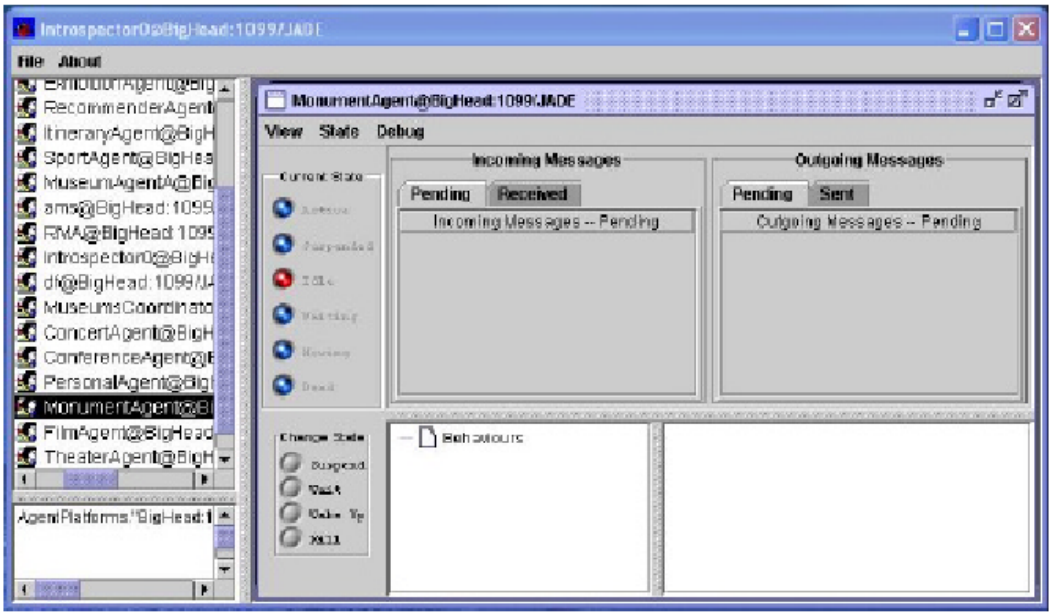


Figura A.7:

A.2.1 Execució de JADE-LEAP en PCs i servidors

Quan treballem en PCs i servidors que utilitzen JDK1.4 o posterior, s'ha d'utilitzar la versió j2se de JADE-LEAP, que es quasi idèntic a JADE. Llavors, per a iniciat el "main container", teclejem:

```
java jade.Boot [opcions] [descripció agents]
```

on les opcions poden ésser especificades:

```
<local-name>:<agent-class>[(<arg1>,<arg2>...)]
```

Noteu la diferència respecte JADE on l'agent és separat per espais. A més, a diferència de JADE, no hi d'haver espais entre els arguments de l'agent. Per exemple:

```
java jade.Boot -gui -nomtp
        Peter:myPackage.MyClass1;John:myPackage.MyClass2(anarg)
```

carregarà un "Main container" sense MTP i activarà l'interfície visual del RMA a més d'un agent anomenat "Peter" de la classe myPackage.MyClass1 (sense arguments) i un agent anomenat "John" de la classe myPackage.MyClass2 (amb un argument amb valor "anarg"). De manera semblant, teclejant:

```
java jade.Boot ?container ?host myHost
```

Crearà un contenedor perifèric (sense cap agent dins) que es registrarà al "Main container" executant-se a la màquina "myHost". A més, per tal de mantenir compatibilitat amb versions anteriors de JADE-LEAP, podem utilitzar l'antiga forma:

```
java jade.Boot <bootstrap properties file name>
```

és vàlida i equivalent a:

```
java jade.Boot ?conf <bootstrap properties file name>
```

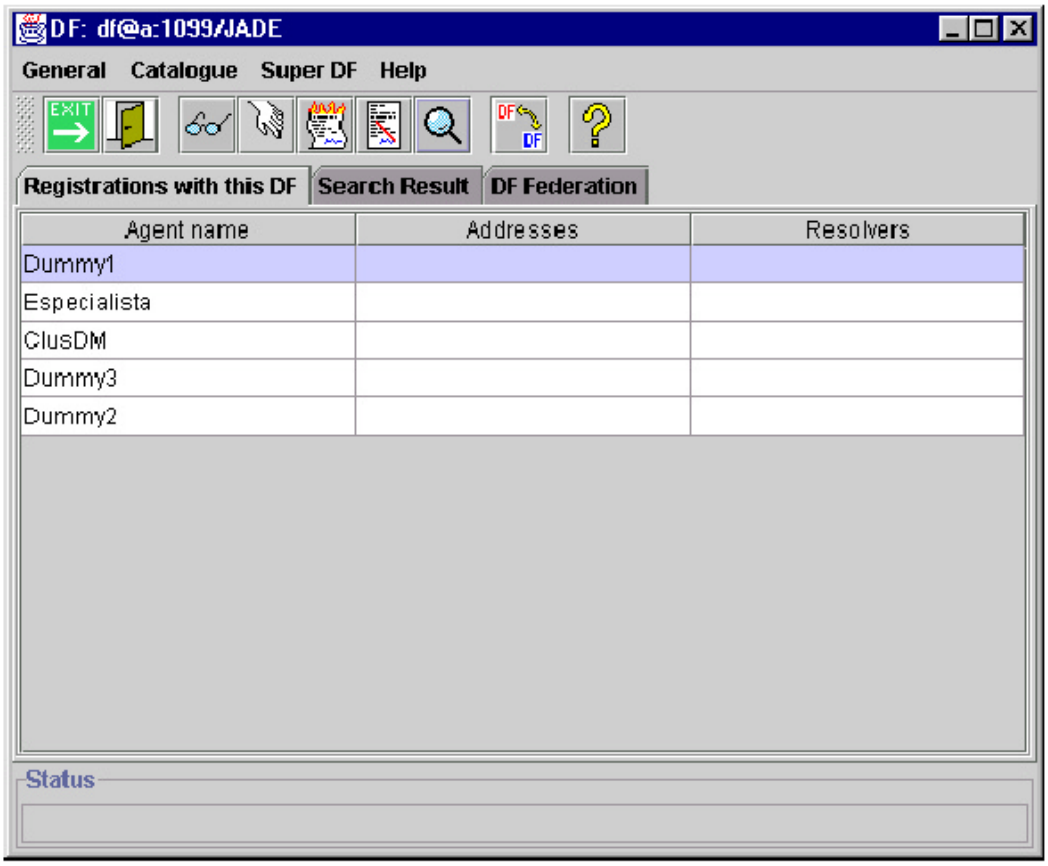


Figura A.8:

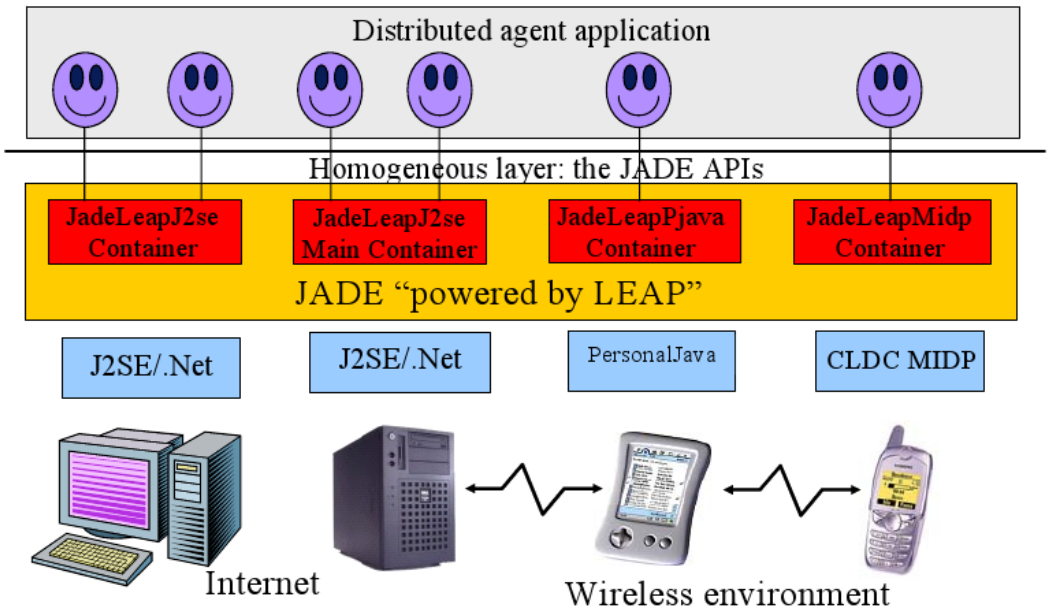


Figura A.9: Entorns d'execució de JADE-LEAP

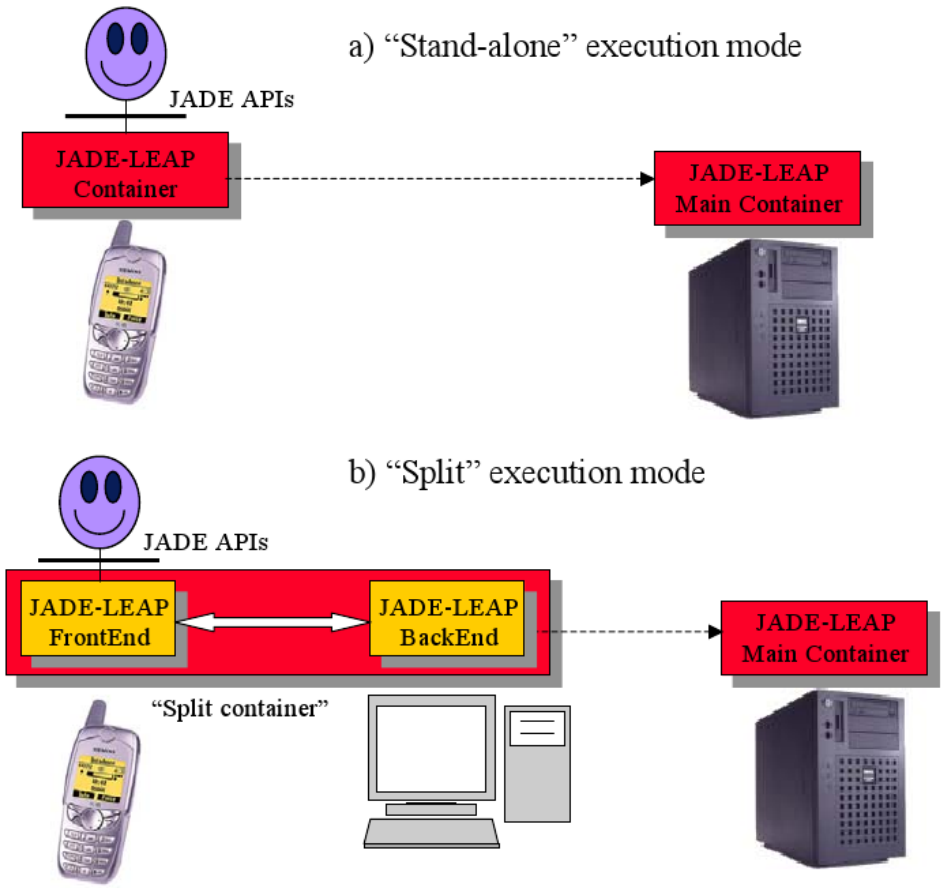


Figura A.10: Models d'execució de JADE-LEAP

A.2.2 Execució de JADE-LEAP en dispositius mòvils

En general, l'execució de JADE-LEAP pot fer-se de dues formes. L'execució normal o "Stand-alone", on es crea el contenedor principal dins la mateixa màquina on s'executa JADE. L'execució separada o "Split", es crea un contenedor principal en un servidor o *backend* i en el dispositiu mòbil o *frontend*. La següent taula resumeix com es soporten els 2 modes en funció del dispositiu mòbil.

	j2se	.NET	pjava	midp
Stand-alone	Recomanat	Suggested	Soportat	No soportat
Split	Soportat	Soportat	Recomanat	Obligatori

L'execuió partida o *split* es recomanada per dispositius mòbils amb poca capacitat ja que:

- El *frontend* és més lleuger que un contenedor complet
- La fase d'arrancada és més ràpida ja que qui carrega el contenedor principal és un *backend* que, suposadament, és molt més potent i no necessita comunicar-se amb el *backend* per a fer-ho
- L'ús de les comunicacions està optimitzat.

És important remarcar que el programador no necessita distingir entre un contedor i l'altre, únicament s'ha de preocupar de configurar correctament les direccions IP i ports i la resta se n'encarregarà el sistema. S'han de tindre en compte les següents qüestions:

- Quan executem un *split container*, hi ha d'haver un contenedor funcionant en mode j2se (no necessariament el contenedor principal) que ens actuarà com a mitjancer.

- Un contenedor principal no es pot dividir.
- La mobilitat/clonació d0agents no està suportada en un *split container*

Quan executem JADE-LEAP en dispositius PersonalJava/CDC, es recomana l'execució dividida a més del comportament dinàmic soportat per la classe

`jade.core.behaviours.LoaderBehaviour`

. Únicament si això no s'adapta a les nostres necessitats utilitzarem l'execució autònoma o *stand-alone*.

A.2.3 PDA i telèfons executant Personalava/CDC J2ME

Quan treballem amb PDA i telèfons que tenen capacitat J2ME CDC o PersonalJava, la versió pjava de JADE-LEAP és la idònea.

Execució dividida

Un *split container* es creat teclejant:

```
java jade.MicroBoot [opcions]
```

on els agents es criden de la mateixa manera que en JADE i les següent opcions estan disponibles:

`-host <nom_host/adreça>`

indica la màquina on s'està executant el contenedor principal (per defecte localhost).

`-port <port-number>`

indica el port (per defecte 1099).

`-agents <agents specification>`

activa els agents especificats.

`-exitwhenempty <true|false>`

determina si s'ha de destruir el contenedor si no queden més agents dins seu.

Execució autònoma

Es crea un contenedor, exactament igual que un contenedor j2se, teclejant:

```
java jade.Boot [opcions] [especificacions agents]
```

on són vàlides les mateixes opcions i especificacions d'agents que l versió j2se. Per a mantenir la compatibilitat amb versions anteriors, l'antiga comanda:

```
java jade.Boot <arxiu_propietats>
```

és vàlida i compatible amb:

```
java jade.Boot -conf <arxiu_propietats>
```

A.2.4 Telèfons mòbils amb Java

Quan estem treballan en telèfons mòbils, hem d'utilitzar la versió *midp* de JADE-LEAP. Per tal de garantir el funcionament en dispositius MIDP, s'inclouen els següents MIDlets:

- jade.MicroBoot. Crea un contenedor dividid o *split container*.
- jade.util.leap.Config. Edita la configuració (hostname, port...).
- jade.util.leap.OutputViewer. Mostra el log de JADE-LEAP de l'última sessió.

Tal i com està implementat en la versió *pjava*, aquesta versió també soporta les opcions de configuració *host*, *port* i *exitwhenempty*. Aquestes poden configurar-se de dues formes:

- En l'arxiu JAD o el MANIFEST.
- A través del MIDlet jade.util.leap.Config.

Load the JADE MIDlet suite and select the Config MIDlet.

Press the `Set` button to activate the `Set property` form and set a new property specifying its key and value.

Press the `Exit` button to leave the Config MIDlet and save the configuration.

Figura A.11: Models d'execució de JADE-LEAP

Configurar JADE-LEAP a través del JAD o MANIFEST

Per tal de que JADE-LEAP llegeixi les propietats de la configuració des del JAD o des del MANIFEST, cal configurar la clau LEAP-conf clau1 en el JAD o el MANIFEST especificant l'opció de configuració:

```
LEAP-<key>: <value>
```

A mode d'exemple, les següent línies formes part del JAD o el MANIFEST:

```
...
LEAP-conf: jad
LEAP-host: host1
LEAP-agents: Peter:MyClass1;John:MyClass2
...
```

Indiquen que es crearà el BackEnd en el host "host1" i arrencarà els agents "Peter" i "John". Aquesta forma de configurar el JADE-LEAP és simple però ha de fer-se abans d'instal·lar el programa en qüestió al dispositiu mòbil.

Configurar JADE-LEAP utlitzant el MIDlet "Config"

Per tal de que JADE-LEAP llegeixi la configuració que hem creat a través del MIDlet "Config", hem de seguir les següents pantalles:

A.3 Java

El conjunt de l'aplicació, tant del FrontEnd com del BackEnd, ha estat realitzada íntegrament en Java. El Frontend està creat amb la versió J2ME de Java (amb les restriccions que això comporta com ara el tipus *float*) i el BackEnd amb la J2SE. J2SE conté pràcticament totes les llibreries bàsiques per a poder crear el programari a mida per a les nostres especificacions. L'inconvenient ens el trobem amb J2ME. Donat que és una tecnologia molt recent i que, no existeix encara un estàndard de funcionament i/o llibreries per a tots els telèfons mòbils, ens trobem amb un inconvenient.

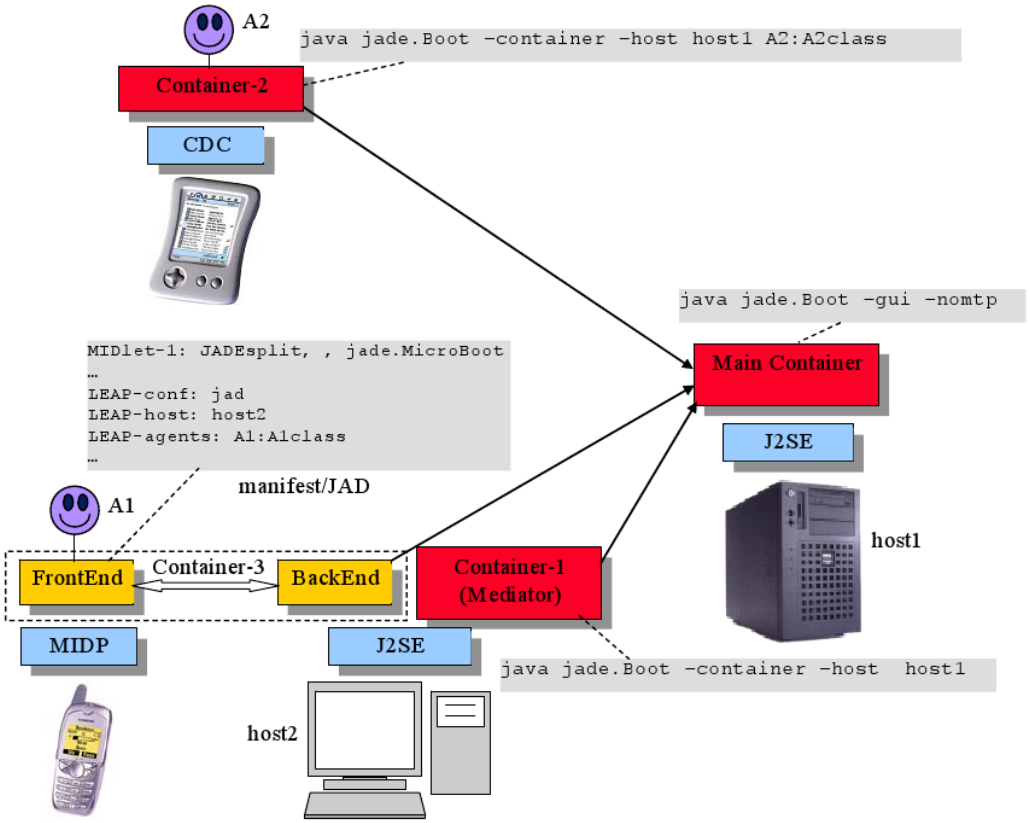


Figura A.12: Models d'execució de JADE-LEAP

A.3.1 MIDlets

El model "MIDlet"

Un MIDlet és una aplicació *Mobile Information Device Profile (MIDP)*. Com en un applet, un MIDlet és una aplicació controlada. En lloc de ser controlada per un navegador web, és controlada per un manegdor d'aplicacions de proposit especial o *AMS* incrustada dins el dispositiu, sovint un telèfon mòbil (com és el cas) o altres dispositius compatibles amb J2ME. Un control extern té sentit en aquest context ja que ens permet interrompre l'execució cada cop que rebem event externs. Seria un comportament nefast que una aplicació no ens deixés contestar una trucada. El model MIDlet també està soportat per el *Personal Digital Assistant Profile (PDAP)*.

Cicle de vida del "MIDlet"

La classe principal d'un MIDlet ve heredada de javax.microedition.midlet.MIDlet. La classe principal defineix els tres mètodes dels cicles: startApp(), pauseApp() i destroyApp(). Hi han 3 possibles estats en un MIDlet:

- paused: L'instància del MIDlet està inactiva.
- active: El MIDlet està actiu.
- destroyed: El MIDlet ha finalitzat i està preparat per al "Garbage Collector".

Cal tenir en compte que no hi ha equivalent amb la càrrega d'un applet ja que no hi ha cap mètode d'inicialització. Normalment, un MIDlet s'inicialitza ell mateix cada cop que el mètode startApp() es invocat. Els MIDlet són creats per l'AMS, típicament en resposta a una petició de l'usuari. Per exemple, l'AMS mostra tots els MIDlets instal·lat en el sistema i deixa a l'usuari que en seleccioni un. L'estat inicial d'un MIDlet és "pausat". Com un applet, un MIDlet hauria de fer, o no, una inicialització en el seu constructor, ja que el seu context d'execució encara no ha estat creat. En algun punt després de cridar al constructor, l'AMS activa el MIDlet i invoca el mètode startApp(). Després d'inicialitzar, startAp() crea i mostra l'interfície gràfica. Després de retornar del startApp(), el MIDlet canvia l'estat de pausat a activat. Si el MIDlet no pot ésser inicialitzat per qualsevol motiu, llença una excepció del tipus javax.microedition.midlet.MIDletStateChangeException i, tot seguit, canvia a l'estat "destruït". La desactivació es produeix quan hi ha qualsevol transició de l'estat "activat" al "pausat". El MIDlet no es destruït, per suposat, però allibera tants recursos com sigui possible. Si la desactivació és iniciada per l'AMS, el mètode pauseApp() és invocat. Si el MIDlet es desactiva ell mateix, utilitzant el context d'execució del MIDlet, el mètode pauseApp() no és invocat. El codi bàsic d'un MIDlet té la següent aparença:

```
import javax.microedition.midlet.*;

public class BasicMIDlet extends MIDlet {
    public BasicMIDlet(){
// constructor - don't do much here
    }

    protected void destroyApp( boolean unconditional )
       throws MIDletStateChangeException {
// called when the system destroys the MIDlet
    }

    protected void pauseApp(){
// called when the system pauses the MIDlet
    }

    protected void startApp()
       throws MIDletStateChangeException {
// called when the system activates the MIDlet
    }
```

```
}
```

Context d'un "MIDlet"

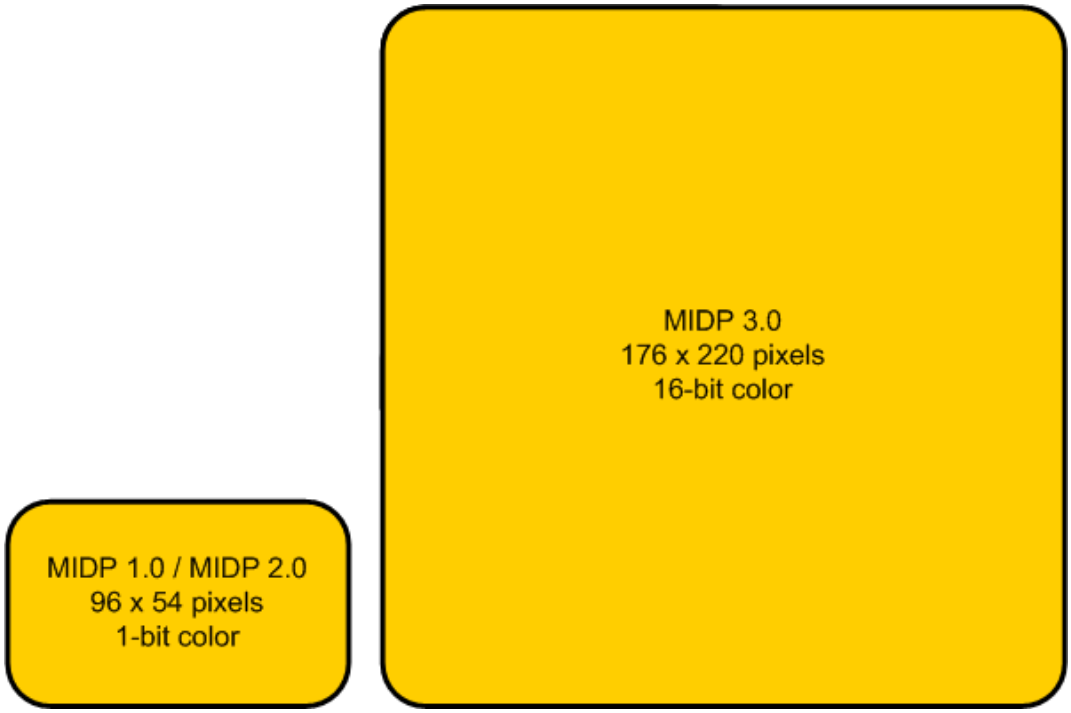
La classe javax.microedition.midlet.MIDlet també defineix mètodes que permet que un MIDlet interactui amb els seu context d'execució: getAppProperty() retorna els valors de les propietats d'inicialització; resumeRequest() demana a l'AMS que reactivi el MIDlet; notifyPaused() desplaça el MIDlet cap a l'estat "pausa"; i notifyDestroyed() desplaça el MIDlet cap a l'estat "destruït". Les propietats d'inicialització del MIDlet són parells nom-valor en el descriptor del MIDlet o bé en el MANIFEST. El descriptor de l'aplicació és un fitxer de text a part que informa sobre el contingut de MIDlets del fitxer JAR (MIDlet suite). Quan s'invoca un getAppProperty(), busca primer en el descriptor (JAD), després al MANIFEST. Els mètodes restants afecten directament sobre el cicle de vida del MIDlet. Un MIDlet pausat, cirda resumeRequest() per a ser reactivat. Un MIDlet actiu crida a notifyPaused() per a ser desactivat. El resumeRequest() únicament sol·licita a l'AMS ser reactivat; l'AMS decideix si i quan reactivar el MIDlet. La reactivació invoca el mètode startApp(). Per contra, notifyPaused() o notifyDestroyed() provoquen un canvi d'estat; com a conseqüència, ni pauseApp() ni destroyApp() són invocats. El codi millorat d'un MIDlet és el següent:

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

public class BetterMIDlet extends MIDlet {

    private Display display;

    public BetterMIDlet(){
    }

    protected void destroyApp( boolean unconditional )
       throws MIDletStateChangeException {
 exitApp(); // call cleanup code
    }

    protected void pauseApp(){
 // add pause code here
    }

    protected void startApp()
       throws MIDletStateChangeException {
 if( display == null ){
     initApp(); // perform one-time initialization
 }
 // add per-activation code here
    }

    private void initApp(){
 display = Display.getDisplay( this );
 // add initialization code here
    }

    public void exitApp(){
 // add cleanup code here
 notifyDestroyed(); // destroys MIDlet
    }
}
```

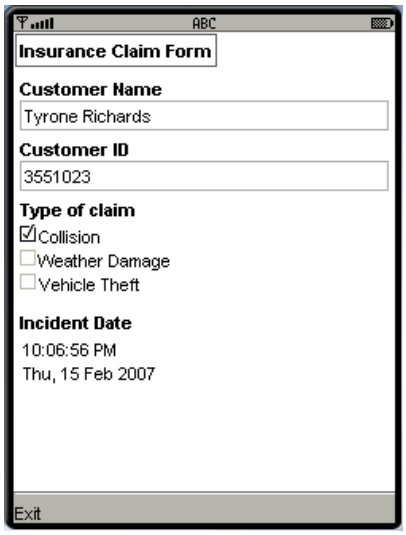
A diferrència de la classe BasciMIDlet, BetterMIDlet defineix un mètode initApp() per a la inicialització en un sol pas (cridat el primer cop que startApp() és invocat) i un mètode exitApp() per a centralitzar l'alliberament de recursos.

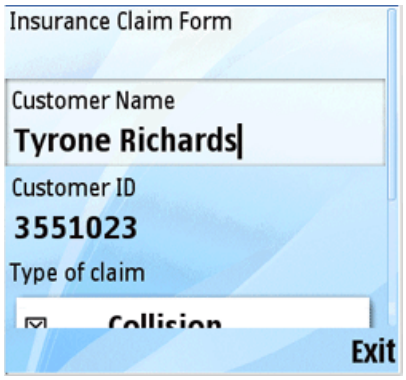
A.3.2 Gràfics en J2ME

Quan es va crear Java ME en l'any 2000 (llavors anomenat J2ME) no hi havien massa alternatives disponibles per als programadors d'interfícies gràfiques. De fet, la única API disponible era l'interfície estàndard per a LCD javax.microediation.lcdui de la *Mobile Information Device Profile (MIDP)*. En aquest moment, MIDP estava a la versió 1.0 i no hi havien eines disponibles com la *Sun Java Wireless Toolkit*.
Actualment, amb tantes eines gràfiques per a Java ME, els programadors per a dispositius mòbils tenen un altre problema, la massificació d'APIs gràfiques.
Amb la gran dispersió de tipus de terminals mòbils, s'ha decidit crear un estàndard en quant a la mida de pantalla, sobretot orientat a les aplicacions gràfiques. Aquí tenim una sèrie d'exemples



de l'ús de les modernes llibreries gràfiques de Java ME.





A.4 XML

XML, de l'anglès eXtensible Markup Language (≪llenguatge de marques extensible≫), és un metallenguatge extensible, d'etiquetes, desenvolupat pel World Wide Web Consortium (W3C). És una simplificació i adaptació de l?experimentat SGML, i permet definir la gramàtica de llenguatges específics (de la mateixa manera que HTML és, alhora, un llenguatge definit per SGML). Per tant, XML no és realment un llenguatge en particular, sinó una manera de definir llenguatges per a diferents necessitats. Alguns dels llenguatges que empren XML per a la seva definició són XHTML, SVG, MathML. XML no ha nascut només per a la seva aplicació a Internet, sinó que es proposa com a un estàndard per a l'intercanvi d'informació estructurada entre diferents plataformes. Es pot utilitzar per a bases de dades, editors de text, fulls de càlcul i per moltes altres aplicacions diverses. XML és una tecnologia relativament senzilla que té al seu voltant altres que la complementen i la fan notablement més extensa, a més de proporcionar-li unes possibilitats molt més grans. A l'actualitat té un paper molt important, ja que permet la compatibilitat entre sistemes, permetent de compartir informació d?una manera segura, fiable i fàcil.

Bibliografia

[1] Viejo, A., Estudio de la implementación de agentes en dispositivos móviles., Projecte Final de Carrera, Enginyeria Tècnica en Informàtica de Sistemes, Universitat Rovira i Virgili (2003).

[2] Batet, M., Sistema multi-agent de recomanació turistica., Projecte Final de Carrera, Enginyeria Tècnica en Informàtica de Sistemes, Universitat Rovira i Virgili (2005).

[3] Redondo, M., Generació d'itineraris turístics personalitzats., Projecte Final de Carrera, Enginyeria Tècnica en Informàtica de Sistemes, Universitat Rovira i Virgili (2005).

[4] Bellifemine, F., Caire, G., Greenwood, D., Developing Multi-Agent Systems with JADE., Wiley (2005).

[5] Nikraz, M., Caire, G., Bahri, P.A., A Methodology for the Analysis and Design of Multi-Agent Systems using JADE, School of Engineering Science and Parker Center, Murdoch University.

[6] Viquipèdia