EPS
Escola Politècnica Superior
UdG

Projecte/Treball Fi de Carrera

Estudi: Eng. Tècn. Informàtica de Gestió. Pla 2001

Títol: INFOHABITANTS
Laboratori per la Simulació d'Infohabitants com Agents Recomanadors (SIRA)

Document: Memòria P/TFC

Alumne: Francesc Josep Gómez López

Director/Tutor: Josep Lluís de la Rosa i Esteva
Departament: Enginyeria Elèctrica, Electrònica i Automàtica
Àrea: Enginyeria de Sistemes i Automàtica

Convocatòria (mes/any): Juny/08

ÍNDEX

1. OBJECTIUS I PREPARACIÓ

1.1. MOTIVACIÓ

Una de les línies d'investigació d' ArLab (Agents Research Lab), grup de recerca de la Universitat de Girona, son els sistemes recomanadors.

Un dels problemes amb què es troben és que no s'ha trobat cap plataforma que permeti experimentar la recerca d'agents recomanadors. Al 2003 es va crear la plataforma IRES [Montaner, et. al, 03], però és una plataforma a mida per recomanacions de restaurants que no permet ser usada en un altre domini. Un dels temes de recerca es la privacitat en sistemes recomanadors. Per poder fer experiments es necessari crear una plataforma en la qual els agents recomanadors no estiguin relacionats amb cap usuari, que es el que ara fan els agents recomanadors en els sistemes actuals. També hi ha la necessitat d'experimentar amb gran quantitat d'informació.

Fent una recerca per la literatura no s'ha trobat cap simulador o sistema de recomanació amb infohabitants - entitats que processen molta informació i de vàries fonts - . El simuladors existents treballen amb perfils d'usuari centralitzats, és a dir, treballen amb perfils associats o, com a molt, amb agents recomanadors associats directament a un perfil d'usuari. Per això, sorgeix la necessitat de disposar una eina que permeti experimentar amb varies tècniques de recomanació i tipus d'interacció amb usuaris, a partir de la utilització d'infohabitants als quals se'ls pugui configurar el seu comportament de recomanació.

1.2. FITES A ASSOLIR

L'objectiu del projecte serà la implementació d'un simulador de sistema de recomanació, amb agents recomanadors.

Per a l'elaboració del projecte es crearà un sistema de recomanació basat en agents software i el disseny d'aquest sistema estarà pensat per ser controlat exclusivament per agents. Es construirà un hàbitat virtual d'agents i s'implementaran aquests agents software amb diversos comportaments, creences i tipus de decisions per desenvolupar un sistema recomanador que es pugui fer servir per investigar, amb grans volums de dades, els model teòrics de recomanació.

D'aquests agents, es posarà èmfasi en el disseny i implementació d'infohabitants recomanadors que seran els encarregats de consultar les dades necessàries per realitzar les recomanacions i calcular els valors que mesurin l'èxit o fracàs de cada experiment de recomanació. Es programaran, doncs, diverses tècniques de recomanació i tipus d'interacció entre

infohabitants i usuaris/compradors, perquè els infohabitants puguin contenir un únic perfil o varis perfils d'usuari, així com distintes mesures per poder calcular l'eficàcia d'aquests algorismes.

Per poder testejar el simulador s'implementarà, també, una interfície gràfica que permeti configurar un seguit de paràmetres per regir els experiments de recomanació i el comportament de cada procés i permeti, també, monitoritzar i gestionar els resultats que s'obtindran de cada experiment.

Aquestes consideracions serviran per crear una aplicació o laboratori d'experimentació amb agents recomanadors a fi de poder investigar i analitzar el comportament de cada infohabitant recomanador amb les diverses tècniques implementades - a partir d'aquest punt es farà referència al laboratori amb el nom de **SIRA**, sorgit de la nomenclatura anglesa del títol del projecte -.

2. REQUERIMENTS I MARC TEÒRIC

En aquest apartat s'explicarà la teoria requerida i aplicada en SIRA. Aquesta teoria servirà per entendre la base del present projecte.

I, entrant en l'elaboració del projecte, s'explicarà quines especificacions ha de complir, quina serà la base d'estudi del mateix i s'exposaran les primeres funcionalitats i algorismes que s'implementaran.

2.1. IDENTIFICACIÓ DE REQUERIMENTS

Un primer requeriment del projecte és el de trobar algoritmes de recomanació que permetessin establir seguretat de dades d'usuari respecte a algoritmes tradicionals de recomanació. Els tradicionals, estableixen que hi ha un vincle entre un agent recomanador i un usuari. Aquests algorismes son usats per HabitatPro™ per elaborar les recomanacions personalitzades a un usuari. Els nous tipus estableixen que a un usuari el pot recomanar més d'un agent, o un agent pot recomanar a més d'un usuari, trancant així el vincle existent entre la parella agent i usuari.

Partint d'un treball d'investigació de nous algorismes de recomanació [Delfin, 07], no ha calgut fer cap mena de recerca per el present projecte. Només s'han hagut d'implementar algoritmes presentats per el treball d'investigació mencionat.

Els requeriments del projecte estan encarats a assolir l'objectiu d'implementació d'algoritmes de recomanació, desenvolupant agents software, amb la finalitat que un usuari de SIRA pugui fer simulacions de recomanació. Per tant, a continuació s'identificaran requeriments que s'han de complir per aquests propòsits.

La implementació de SIRA es divideix en dues parts , per tant, es crearà un hàbitat d'agents independent de l'aplicació gràfica.

L'hàbitat ha de ser *obert* i complir *escalabilitat*, per permetre implementar noves funcionalitats o mòduls en futures ampliacions. Així com també haurà de complir *estabilitat*, que no tingui errades que aturin la seva execució. I ha de ser prou *general* per permetre executar varis tipus d'interacció entre agent i usuari i varies tècniques de recomanació, cosa que els simuladors actuals no ho permeten. Per la implementació de l'hàbitat no cal inventar res, *s'aprofiten llenguatges o eines* existents per la creació d'agents.

Pel que respecta a l'aplicació gràfica, en un principi ha de ser *plataforma dependent*: es dissenyarà una única interfície gràfica, a fi de ser utilitzat per un usuari final que li permeti controlar la funcionalitat dels infohabitants i les recomanacions facin. Aquesta eina ha de permetre *monitoritzar, testejar* amb varis paràmetres de configuració de recomanació i *analitzar* experiments de recomanació, essent fàcil la seva *utilització.*

2.2. AGENT SOFTWARE

S'ha parlat anteriorment de la creació d'agents recomanadors que seran els encarregats de realitzar les consultes i càlculs de cada experiment de recomanació. Cal, doncs, programar el sistema recomanador orientat al desenvolupament d'agents software amb comportaments propis, a fi de dotar al sistema d'autonomia i dinamisme per afrontar els diferents experiments a realitzar gràcies a SIRA

2.2.1. Concepte o definició

Hi ha molts autors que defineixen el concepte d'agent, encara que totes les definicions tenen trets comuns. De les moltes definicions existents, les més properes a una definició de caire informàtic dirien que els agents software son un paradigma de programació, consistent en algorismes autònoms que actuen segons varis rols de comportament. Aquests algorismes formen entitats software a les quals se les implementa comportaments similars al comportament humà. Tenen autonomia i capacitat de decisió, raonen, aprenen, es comuniquen amb d'altres agents, poden organitzar-se y desplaçar-se d'un computador a un altre. Utilitzen les seves capacitats per ajudar a un usuari final a resoldre els seus problemes de manera intel·ligent.

Com que donar una definició teòrica no és gaire trivial, totes elles concorden en exposar les mateixes o similars característiques que ha de complir un agent software.

2.2.2. Característiques

Per tal de classificar els agents, o definir les seves característiques, cal fixar-se en les seves característiques individuals, fixar-se en el seu hàbitat i comportament extern i fixar-se en la seva utilitat.

Segons les seves característiques individual, es poden definir els agents com **reactius**, que realitzen procediments o rutines a partir de la recepció d'estímuls externs; o **cognitius**, que realitzen tasques complexes, de raonament o d'aprenentatge.

També es distingeixen tipus d'agents segons el tipus d'**interacció** que realitzen amb d'altres agents o entitats: s'han de poder comunicar entre ells o han de proporcionar una interfície per establir comunicació amb persones. I també es poden classificar segons la manera amb què s'organitzen entre ells, existint agent **individualistes**, que resolen tasques per si sols, o **cooperants**, que resolen tasques amb l'ajuda d'altres agents amb processos com la negociació.

Els agents a desenvolupar en el present projecte han d'estar dotats d'intel·ligència i no ser d'un únic tipus, ja que han de poder reaccionar a estímuls o missatges que puguin rebre, a més de prendre decisions.

Per un primer disseny dels agents recomanadors, cadascun executarà les tècniques de recomanació amb la informació que puguin recollir d'un magatzem de dades sense necessitat d'intercanviar informació amb d'altres agents. Mentre que es desenvoluparan d'altres agents que s'encarreguin de la gestió global de l'hàbitat on resideixin els agents, a fi de rebre les peticions sol·licitades per l'usuari de SIRA: consultar informació, crear els agents, etc.

Per tant, els agents han de complir amb una sèrie de característiques - no només els recomanadors, si no els altres tipus d'agents - perquè pugin ser catalogats com a agents intel·ligents:

- *Autonomia*

  Un mateix agent ha de saber controlar les seves pròpies accions i el seu estat intern, així com decidir quines accions executar o quins objectius ha d'assolir.

- *Comunicació*

  Un agent ha de ser sociable i tenir capacitat de comunicació amb d'altres agents a través d'un llenguatge comú, entès tant per l'emissor com pel receptor de la comunicació.

- *Aprenentatge*

  És la capacitat que te l'agent de canviar el seu comportament segons la seva experiència, a partir de la recopilació d'informació de l'entorn - fer consultes a un magatzem de dades - o de la interacció amb d'altres agents.

- *Reactivitat*

  L'agent ha de tenir en compte els canvis que es produeixen en l'entorn o dins del seu comportament, i respondre de manera adequada a les seves percepcions.

- *Proactivitat:*

  Un agent ha de ser capaç de prendre iniciatives pròpies, no només executant el seu comportament, si no perseguint un objectiu, a partir d'intencions i desitjos.

Existeixen altres característiques que serveixen per acabar de completar la definició d'agent, com la mobilitat: poden canviar d'entorn o xarxa, adaptables: varien el seu comportament segons estímuls de l'entorn; l'encapsulació, habilitat social, etc.

2.2.3. Arquitectures d'agents

Com a paradigma de programació, cal estudiar arquitectures o models d'implementació per tal de crear els agents necessaris i que cada tipus d'agent respongui a unes tasques determinades:

- *Deliberativa*

  Utilitzen models de representació simbòlica del coneixement. L'agent pren decisions utilitzant mecanismes de raonament lògic per assolir el seu objectiu. En l'arquitectura *Beliefs, Desires, Intentions, BDI (Creences, Desitjos i Intencions)* [Rao and Georgeff, 91], els agents estan dotats de Creences, Desitjos i Intencions, les quals les creences el domini de sapiència que pot aprendre un agent, els desitjos representen els objectius o les accions que un agent ha de fer, assolint un estat final, i les intencions representen la planificació que segueix un agent per aconseguir els seus objectius partint de les seves creences.

- *Reactiva*

  Aquesta, gestiona un seguit de tasques, sense necessitat de crear un model simbòlic, que defineixen el comportament d'un agent i estan organitzades per capes de baix nivell fins a un més alt nivell d'abstracció.

  Tenen una major aplicació en el disseny de controladors en robòtica.

- *Híbrida*

  Combina aspectes de les dues arquitectures anteriors per tal de resoldre les seves limitacions individuals.

  Adopta una organització per capes distribuïdes en tres nivells d'abstracció. La capa reactiva, de baix nivell, està relacionada a la presa de decisions en temps real; la capa de coneixement, de nivell mig, que conté una representació simbòlica del l'entorn; la capa de comunicació, d'alt nivell, relacionada amb la sociabilitat de l'agent en l'entorn i amb l'intercanvi d'informació amb d'altres agents.

- *Basades en comportament*

  Son una extensió de l'arquitectura reactiva que s'aplica al disseny d'agent autònoms o robots. S'utilitza amb la finalitat de desenvolupar mètodes per controlar sistemes artificials.

2.2.4. Tipus d'agents

Segons la utilitat que se li pugui donar a un agent software, es poden classificar en diverses categories. A destacar es descriuen les següents:

- *Agents d'informació*

  Agents amb capacitat d'obtenir i produir informació, de o cap a diverses fonts, com Internet. Son capaços d'obtenir la informació de vàries fonts.

- *Agents mòbils*

  Agents amb capacitat per moure's per la xarxa electrònica. Poden ser programats en un ordinador client i realitzar la seva funcionalitat en un ordinador servidor.

- *Agents col·laboratius*

  Tenen facilitat per comunicar-se amb d'altres agents. Compleixen les característiques de proactivitat i sociabilitat, per tal d'executar negociacions amb d'altres agents. Comparteixen un mateix llenguatge de comunicació.

- *Agents recomanadors*

  Son capaços de fer un seguiment individual d'un comprador, aprenent dels seus gustos i preferències i poder suggerir-li altres compres apropiades.

2.2.5. Infohabitants com a agents recomanadors

Un **infohabitant** és una entitat, o agent, capacitada de processar informació. Pot interactuar i existir en societats amb més infohabitants, en un entorn d'informació global.

Els infohabitants maneguen molta informació descentralitzada - que es pot adquirir de varis orígens -. Es comporten de vàries maneres segons la seva finalitat: infohabitants que produeixen o consumeixen informació, infohabitants com a agents de borsa, etc. [Rolf, et. al, 06] [de la Rosa, et. al, 99]

Per el projecte, es dotarà als *infohabitants amb el rol de recomanador*. En aquest aspecte, un agent recomanador te informació centralitzada de les dades d'un comprador. Aplicant els concepte d'infohabitant i d'agent recomanador, es dissenyaran els infohabitants per executar tasques de recomanació mantenint informació descentralitzada dels compradors, és a dir, per les tasques de recomanació un infohabitant podrà aprendre de i recomanar a varis compradors. Un infohabitant aplicarà els dos tipus de recomanació - interacció entre agent i usuari - que es mencionaran en el següent apartat.

En el projecte s'aplicarà la definició d'infohabitant i el concepte d'agent software per realitzar experiments de recomanació on s'apliquen els conceptes de recomanació; interacció amb usuaris, tècniques i mesures d'avaluació, que es presentaran en el següent apartat.

2.3. SISTEMA DE RECOMANACIÓ

Un primer requeriment del laboratori és el treballar amb un **sistema recomanador** (**SR**), que permeti estudiar diverses tècniques de recomanació aplicades a diversos tipus de recomanació. Però, què es un SR?

Un Sistema Recomanador, SR, és un sistema que té com a principal tasca seleccionar certs tipus d'objectes d'acord amb els requeriments d'un usuari, donat que els objectes estan emmagatzemats i caracteritzats en base als seus atributs [Wang, 98]. Sorgeixen de la necessitat de proveir als usuaris informació rellevant i personalitzada [Ricci and Del Missier, 2004].

Per implementar el simulador d'un SR es defineix un **procés de recomanació** en el qual una entitat exposa informació acceptable a una altra entitat que cerca informació pels seus interessos durant un període de temps determinat. Aquesta informació és presentada segons els gustos del usuari a qui es recomana, essent valorada, prèviament, com a útil i adient per a qui és recomanat. I així repetidament al llarg de tot un històric de dades.

El SR a implementar en el projecte, doncs, executa aquests processos, creant i gestionant els elements que hi intervenen: creació de les entitats recomanadores o agents recomanadors,

informació històrica de compres o transaccions entre usuaris i productes, càlcul de valors que mesuren l'èxit o fracàs de cada recomanació, presa de decisions sobre què cal recomanar per un usuari concret, etc.

2.3.1. Tipus d'interacció entre agent i usuari

Per l'execució d'una recomanació és necessari saber amb quin usuari s'està treballant, per aprendre els seus gustos i poder aconsellar-li els productes que més apropiats. Cada execució o *procés de recomanació* es realitzarà a través d'un històric de compres, i es dividirà en varis **cicles de recomanació** - períodes de temps, configurables - compresos entre una data inicial i una data final, dins del període global de tot l'històric.

Amb aquestes consideracions, pel present projecte s'implementen dos tipus de recomanació o interacció entre agent i usuari:

1) Recomanació associativa.

A partir d'una data inicial, s'assigna a un infohabitant un usuari, i aquest realitzarà consultes i càlculs per mesurar les recomanacions *per al mateix usuari durant tots els cicles amb què s'hagi definit el procés*. Amb aquest tipus de recomanació, l'infohabitant sempre conté les dades d'un mateix usuari i una bona recomanació hauria de donar valors més bons d'èxit a mida que avança el procés.

2) Recomanació dissociativa.

També a partir d'una data inicial, un infohabitant realitzarà consultes i càlculs per mesurar les recomanacions, *per un usuari diferent per cada cicle amb què s'hagi definit el procés*. D'aquesta manera, l' infohabitant no contindrà informació de cap usuari mentre no executi cap recomanació, encara que això impliqui donar uns valors no tant bons d'èxit a mida que avança el procés, perquè en l'aprenentatge no sempre es coneixen els gustos d'un mateix usuari. Aquesta falta d'increment d'èxit d'una recomanació, al llarg del temps, ha de permetre garantir un alt nivell de privacitat de les dades d'un usuari [Delfin, et al., 06]. Per el projecte, s'implementen varis tipus de dissociació:

- *Rotació*

L'algorisme fa una rotació dels usuaris a través dels agents existents en el SR: el primer agent és assignat al primer usuari, després se li assigna el segon usuari, després el tercer, etc.

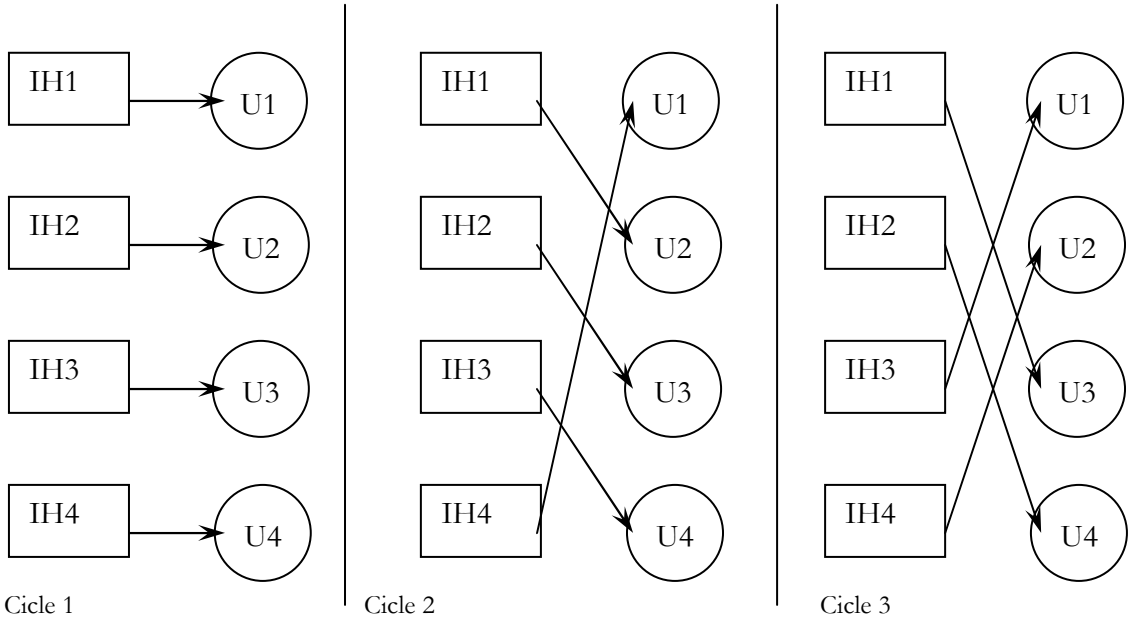


Figura 2.1.: Seqüència de rotació entre infohabitant, IHx, i usuaris, Ux

- *Intercanvi*

  En aquesta dissociació, cada agent intercanvia l'usuari que te assignat amb l'usuari d'un altre agent a cada cicle.

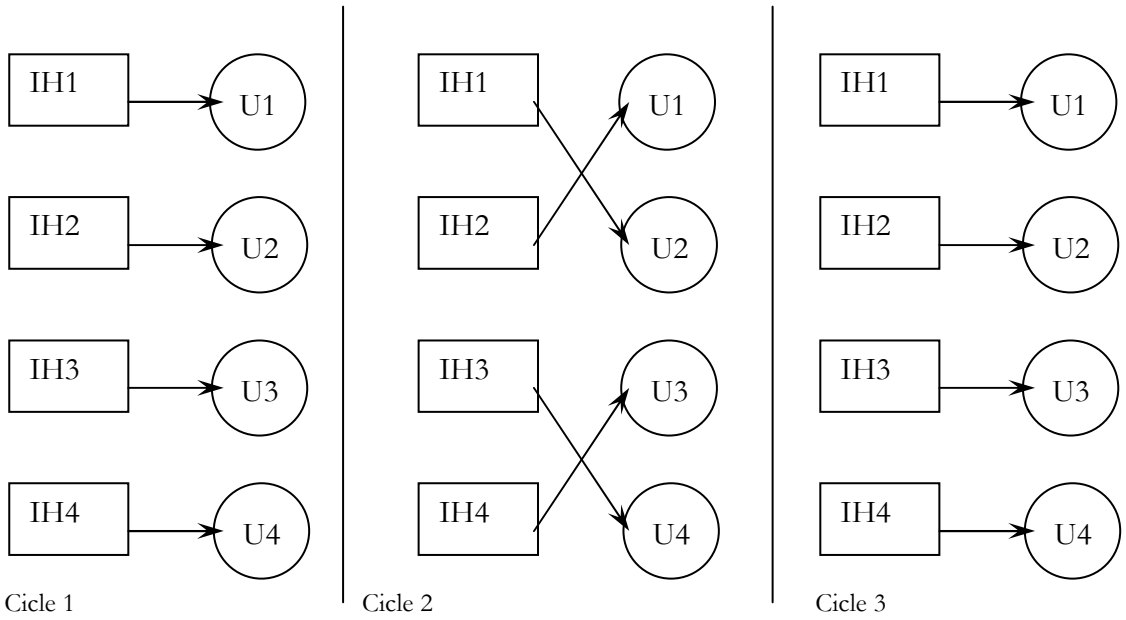


Figura 2.2.: Seqüència d'intercanvi entre infohabitant, IHx, i usuaris, Ux

- *Aleatòria*

L'algoritme permet que els agents canviïn d'usuari assignat sense cap ordre ni sentit; De manera aleatòria, es fa una assignació d'usuari a agent per cada cicle del procés de recomanació, al atzar, segons quin usuari està disponible per ser recomanat

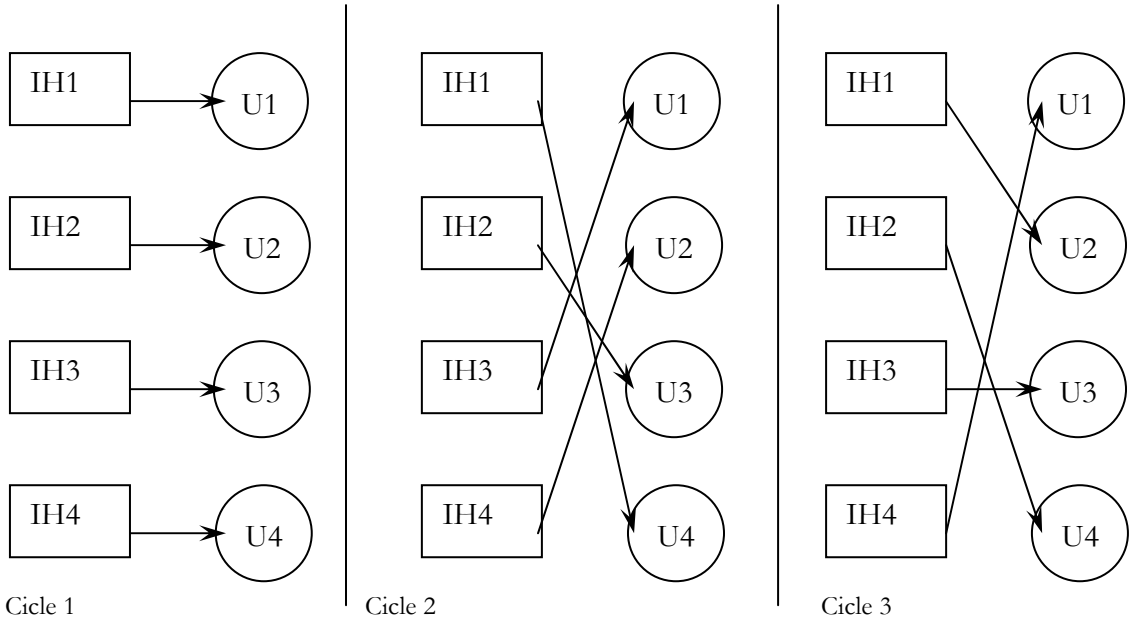


Figura 2.3.: Seqüència aleatoria d'interacció entre infohabitant, IHx, i usuaris, Ux

2.3.2. Tècniques de recomanació

No només d'aprenentatge de gustos dels usuaris es realitzen recomanacions, si no que cal aplicar certes tècniques de recomanació per trobar d'altres productes que puguin agradar a un usuari o comprador determinat. De les diverses tècniques existents, a continuació es descriuen tres que seran les que s'apliquen per la implementació de l'actual projecte.

1) Filtrat basat en contingut

Segons el filtrat basat en contingut, cada producte te una descripció o uns atributs que ajuden a la seva catalogació o classificació.

El filtrat basat en contingut selecciona productes que concordin amb la mateixa descripció que els productes més afins a un usuari/comprador determinat i siguin, també, els més afins o més valorats per la resta d'usuaris. És a dir, se seleccionen els productes més similars de tots els usuaris/compradors [Adomavicius, 05] [Balabanovic and Shoham, 97] [Xu, et al., 05].

2) Filtrat Col·laboratiu

El filtrat col·laboratiu te en compte els gustos de compra dels usuaris més propers o del mateix segment que un usuari determinat.

Comença, doncs, amb una selecció d'usuaris més similars en gustos de compra que l'usuari actual: els usuaris més propers en gustos de compra seran els escollits per fer la recomanació. D'aquests usuaris, se seleccionen els productes més valorats, sense tenir en compte la descripció dels mateixos. Se seleccionen els productes més valorats dels usuaris més similars en gustos de compra [Yang, et al., 04] [Herloker, 00] [Linden, 03] [Billsus, 98]

3) Filtrat Híbrid

Amb aquest filtrat híbrid, es busca un major èxit en la recomanació donat que es barreja la selecció d'usuaris més similars i es recomanen els productes, més afins a aquests usuaris, que concordin amb la mateixa descripció que els productes afins a l'usuari actual.

Per tant, aquest filtrat híbrid, selecciona els productes més valorats i més similars dels usuaris més similars a l'usuari a qui se li fa la recomanació [Burke, 02] [Smyth and Cotter, 00].

Aquesta tècnica aplica primer un filtrat col·laboratiu entre compradors, i dels més similars se'ls aplica un filtrat basat en contingut, per seleccionar els productes a recomanar.

Per executar els experiments de recomanació, *una primera tasca que ha de fer el laboratori és obtenir una llista ordenada d'usuaris o compradors*. La llista de compradors servirà per executar els tipus i tècniques de recomanació, abans descrites, sobre un major volum de dades que el volum de dades que es pugui fer servir en una experimentació manual.

Aquesta llista serà obtinguda amb l'aplicació d'un algorisme de segmentació conegut com *Recency, Frequency, Monetary, RFM (Recència, Freqüència, Despesa)* [Hughes, 00], que agrupa els compradors per la recència de la última compra, per la freqüència de compra i per la despesa realitzada en les últimes compres. Els valors d'aquests atributs de la segmentació estaran compresos entre **1..5**. Serviran per poder filtrar les llistes de compradors, delimitant segments més específics segons el valor que es doni als atributs R, F i M.

2.3.3. Mesures de recomanació

S'ha parlat de la possibilitat que amb recomanacions de tipus dissociatives es perdi o es millori l'èxit de les recomanacions. Existeixen vàries mesures [Salton and McGill, 83] per avaluar una recomanació de les quals, en el projecte, s'apliquen les següents [Montaner, 03]:

- *Precisió*

  És el percentatge d'elements seleccionats que son rellevants per la informació que un usuari necessita, és a dir, representa la probabilitat que un element sigui rellevant. Es calcula de la següent manera:

  $$P = (e/n) * 100$$

  on *e* és el nombre d'encerts d'una recomanació i *n* és el nombre total d'elements a recomanar. El resultat és un valor percentual comprés entre el rang 0..100.

- *Recall*

  És el percentatge d'elements seleccionats com a rellevants que s'han classificat correctament com a rellevants, és a dir, representa la probabilitat que un element sigui seleccionat com a rellevant. Es calcula de la següent manera:

  $$R = (t/n) * 100$$

  on *t* és el nombre de possibles elements a recomanar i *n* és el nombre total d'elements a recomanar. El resultat és un valor percentual comprés entre el rang 0..100.

- *Fallout*

  És el percentatge d'elements no seleccionats com a rellevants, és a dir, aquest valor avalua el percentatge de les recomanacions fallides. Es calcula de la següent manera:

  $$F = (u/n) * 100$$

  on *u* és el nombre d'elements no seleccionats o no recomanats i *n* és el nombre total d'elements a recomanar. El resultat és un valor percentual comprés entre el rang 0..100.

Aquestes mesures serviran perquè un usuari de SIRA pugui avaluar els processos de recomanació realitzats.

2.4. FUNCIONALITAT DEL LABORATORI

Esmentat anteriorment, pel present projecte s'implementarà un laboratori de recomanacions per tal de realitzar experiments de recomanació utilitzant varis tipus d'interacció i tècniques de recomanació, creant agents software que s'encarreguin de dur a terme aquests experiments.

Aquests experiments no son res més que simulacions d'un procés de recomanació, en massa, on a cada experiment hi intervenen molts usuaris i molts agents recomanadors. Aquests

agents seran creats dins d'un mateix contenidor virtual; pel present projecte, no caldrà implementar cap mena de mobilitat, ni canvis en la programació del seu codi: la implementació d'aquests es desenvoluparà amb la finalitat de fer recomanacions, sense establir cap col·laboració entre agents recomanadors.

El laboratori s'enfocarà amb la finalitat de que varis usuaris finals pugin llançar peticions a l'hàbitat dels agents. Els agents recomanadors viuran en una màquina servidor.

Es farà ús una arquitectura client/servidor per la implementació del projecte i, òbviament, s'implementarà un mòdul per comunicar el servidor amb les interfícies que faran servir els usuaris finals.

Cada experiment o procés de recomanació estaran dividits per cicles - internament s'anomenaran **estats** pel fet que, de cada cicle, la memòria de l'infohabitant aprèn noves dades - i s'han de realitzar sobre molts usuaris/compradors recomanats per molts agents.

SIRA ha de permetre:

- *Seleccionar segments o conjunt d'usuaris/comprador, als quals es faran les recomanacions*
- *Seleccionar la longitud de cada cicle, a partir d'una data inicial*
- *Seleccionar el tipus i tècnica de recomanació a estudiar*
- *Definir la quantitat d'agents recomanadors a crear*
- *Monitoritzar els experiments de recomanació, veient les mesures d'avaluació*
- *Analitzar resultats d'experiments previs*

Un usuari de SIRA podrà estudiar els resultats obtinguts de cada experiment de recomanació. Finalitzada una recomanació - sigui per cada cicle o en la finalització del procés global - cada infohabitant enviarà una resposta a l'usuari de SIRA, perquè aquest pugui analitzar cada experiment. Des de la aplicació gràfica, doncs, haurà de poder guardar els resultats obtinguts i mostrar-los per pantalla. També haurà permetre la creació de gràfiques per veure la tendència de cada recomanació, a partir dels càlculs de valors que ajudin a determinar l'èxit o fracàs d'una recomanació.

Addicionalment, es programarà un mòdul de gestió de dades. Una primera aproximació d'aquesta implementació serà la de carregar la base de dades del laboratori amb dades de

compradors, productes i transaccions de compra des de fitxers de dades, desenvolupant una gestió remota per aprofitar l'arquitectura proposada per fer laboratori.

3. ANÀLISI DE FUNCIONALITATS

Amb l'ajut de diagrames de cas d'ús i diagrames d'activitat, els següents apartats aniran destinats a elaborar un anàlisi de com ha de ser l'aplicació, i quines funcionalitats es duran a terme per executar els experiments de recomanació.

Es començarà l'anàlisi des d'un punt de vista global, exposant la funcionalitat completa de SIRA distingint, però, les parts de l'aplicació que hi actuen: part client i part servidor. D'aquí, i vist que s'ha de desenvolupar un sistema recomanador, SR, s'analitzarà la part servidor com un sistema a part. D'aquesta manera es farà un anàlisi més exhaustiu del funcionament del servidor, ja que ha de contenir un entorn on habitin els diferents agents a desenvolupar.

3.1. MÒDULS GENERALS

Segons la definició de funcionalitats presentada anteriorment, en aquest apartat es mostraran els mòduls a implementar pel present projecte. S'acompanyaran d'una breu descripció per cadascun d'ells.

Com s'observa en el següent diagrama, les funcionalitats de monitorització de les mesures van incloses en el mòdul *llançar experiment recomanació*. Aquest tracta la seqüència d'execució de consulta de segments d'usuaris, per poder triar els usuaris a recomanar i la seqüència d'execució d'un procés de recomanació.

Tota la funcionalitat que fa referència a la configuració d'un experiment, selecció de tècniques, selecció de períodes de temps, selecció de quantitat d'agents recomanadors ... va inclòs en el mòdul de *configuració dades d'operació*. A partir d'ara, s'identificaran com a **dades d'operació** tots aquells paràmetres que regeixen el comportament d'execució d'un experiment o procés de recomanació.

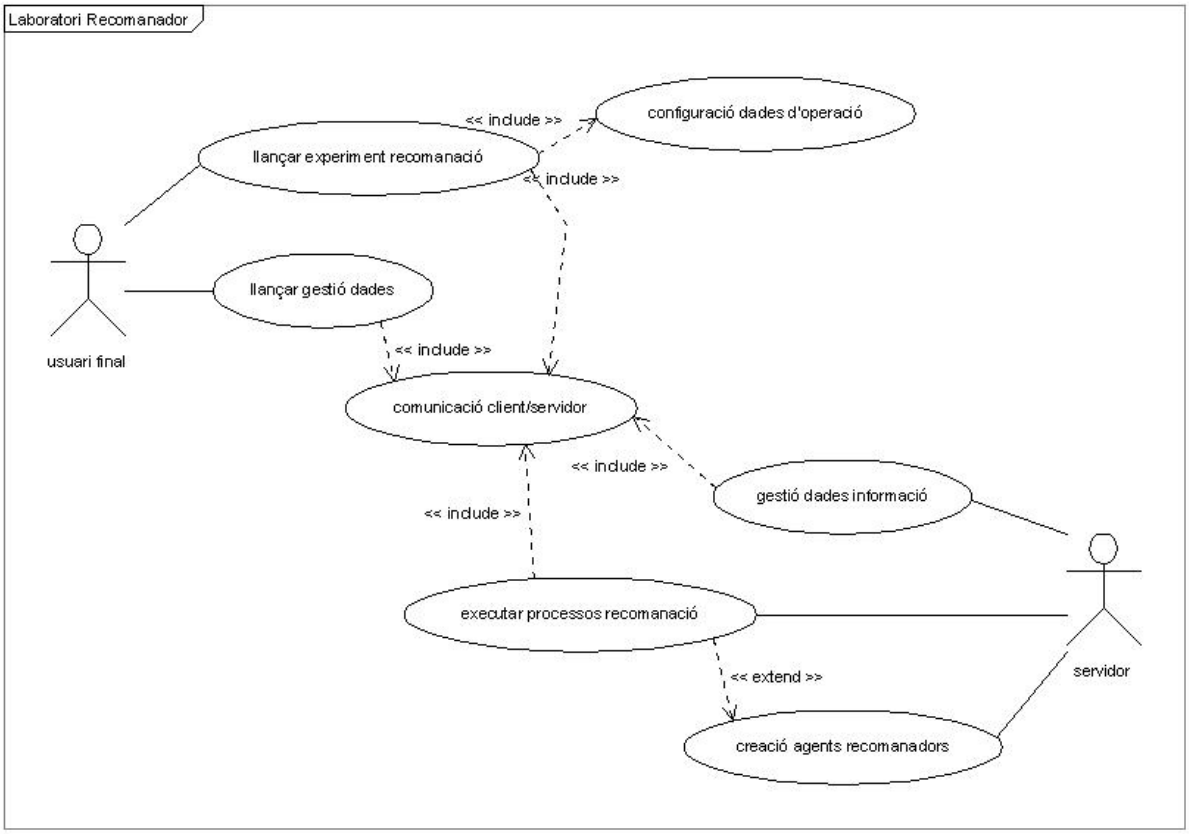


Figura 3.1.: Diagrama de cas d'ús, representació de la funcionalitat global del laboratori.

El mòdul de comunicació neix del requeriment de desenvolupar el laboratori aplicant l'arquitectura client/servidor. Cada mòdul que requereixi una interacció dels dos actors presents al gràfic, inclou la funcionalitat del mòdul de *comunicació client/servidor*.

Entenen com a **dades d'informació** aquelles que fan referència a usuaris, productes i transaccions de compra - base d'estudi del projecte - els mòduls *llançar gestió dades* i *gestió dades d'informació* són els que descriuen aquesta funcionalitat.

Les dades d'operació es fan servir en el mòdul *executar processos recomanació*, a fi de determinar com es realitzaran els experiments. La seqüència d'un experiment ve descrita per aquest mòdul. Abans de l'execució d'un procés de recomanació, no sempre, aquest últim mòdul necessita utilitzar el mòdul de *creació agents recomanadors*, que és el que descriu la funcionalitat de crear els infohabitants necessaris per cada procés. Així poder gestionar l'escalabilitat del sistema recomanador.

3.1.1. Identificació d'actors

Com a actors s'identifica a una entitat física o lògica que interacciona amb el sistema o aplicació, sigui una persona, un dispositiu, el temps, un subsistema, etc.

Es distingeixen dos actors diferents, que intervenen en el funcionament de l'aplicació, per poder separar les accions que pot realitzar l'usuari de SIRA, **usuari final**, i les funcionalitats que es desenvolupen com a gestió de l'hàbitat dels infohabitants, **servidor**. Dintre d'aquest últim actor, s'identifiquen i s'analitzen les funcionalitats que tindran els diferents agents software que viuran dins l'entorn a desenvolupar.

3.1.2. Descripció dels mòduls

Vista una breu descripció de les funcionalitats generals i els actors que hi participen, tot seguit s'exposa una explicació ordenada i estructurada de cada mòdul o cas d'ús.

CAS D'ÚS	**LLANÇAR EXPERIMENT RECOMANACIÓ**
DESCRIPCIÓ	Presentació de formulari per poder seleccionar segments d'usuaris i portar un control de l'execució d'un experiment de recomanació.
PRE-CONDICIÓ	Dades d'operació configurades.
FLUX PRINCIPAL	1. Carregar els valors de les dades d'operació. 2. Seleccionar usuaris d'un segment, com a compradors a recomanar. 3. Enviar una petició de recomanació sobre els compradors seleccionats. 3.1. Rebre resposta amb dades i mesures de cada recomanació. 4. Navegar i veure per les dades resultants obtingudes. 4.1. Guardar els resultats
POST-CONDICIÓ	
OBSERVACIONS	Es rebrà una resposta de recomanació per cada cicle i infohabitant, a fi de fer un seguiment complet de l'experiment.

CAS D'ÚS	**LLANÇAR GESTIÓ DADES / GESTIÓ DADES D'INFORMACIÓ**
DESCRIPCIÓ	Funcionalitats de l'aplicació que permeten modificar les dades del magatzem de dades i consultar les dades existents.
PRE-CONDICIÓ	Disposar de gran volum de dades, en fitxers.
FLUX PRINCIPAL	1. Creació de taules. 2. Llegir les dades d'informació dels fitxers. 3. Enviar petició de gestió de dades 3.1. Si és una petició d'importació de dades 3.1.1. aquestes es guardaran al magatzem de dades 3.1.2. Enviar resposta com a confirmació de la gestió. 3.2. Si és una petició de consulta de dades 3.2.1. Llegir dades del magatzem, segons els valors de les dades d'operació 3.2.2. Enviar petició amb les dades d'informació consultades.
POST-CONDICIÓ	El contingut del magatzem de dades ha estat modificat o consultat
OBSERVACIONS	Hi haurà dades d'operació que regiran les gestions de consulta.

CAS D'ÚS	**CONFIGURACIÓ DADES D'OPERACIÓ**
DESCRIPCIÓ	Per poder dur a terme les consultes de dades de comprador i les posteriors recomanacions sobre aquests, cal donar valor a un seguit de dades que regiran el mode d'operació de cada experiment de recomanació.
PRE-CONDICIÓ	
FLUX PRINCIPAL	1. Donar valors als paràmetres de recomanació. 2. Donar valors als paràmetres de consulta. 3. Guardar els valors.
POST-CONDICIÓ	Les dades d'operació tenen valor.
OBSERVACIONS	Aquest cas d'ús té sentit incloent-lo en la funcionalitat de selecció de dades. Cada cop que es modifiquen les dades d'operació, un experiment de recomanació varia respecte els altres.

CAS D'ÚS	**COMUNICACIÓ CLIENT/SERVIDOR**
DESCRIPCIÓ	Controla la comunicació i el pas de missatges entre la part client i la part servidor del laboratori.
PRE-CONDICIÓ	Ha d'existir un canal de comunicació
FLUX PRINCIPAL	1. Llegir un missatge d'una cua de peticions. 2. Enviar petició pel canal de comunicació. 3. Identificar petició i agent qui ha de tractar la petició. 4. Enviar resposta pel canal de comunicació. 5. Desar missatge resposta a una cua de respostes
POST-CONDICIÓ	El client ha rebut un missatge com a resposta, del servidor
OBSERVACIONS	La comunicació sempre comença per la part client, que és la que sol·licita informació de la part servidor. Aquesta última s'ha de mantenir passiva, preparada per proporcionar informació sol·licitada. En una primera versió del laboratori, la gestió de comunicació sempre tractarà les peticions una darrera l'altra; una única connexió.

CAS D'ÚS	**CREACIÓ AGENTS RECOMANADORS**
DESCRIPCIÓ	Segons sigui el tipus de petició rebuda, el servidor ha de fer néixer, o activar els agents software encarregats de realitzar les tasques de recomanació.
PRE-CONDICIÓ	Existeixen varis o cap infohabitants.
FLUX PRINCIPAL	1. Crear agents recomanadors. 2. Propagar peticions de recomanació a cada agent. 3. Recollir els resultats calculats per cada agent.
POST-CONDICIÓ	Hi ha agents vius esperant una petició de recomanació.
OBSERVACIONS	La funcionalitat de creació d'aquest agents ve lligada amb el tractament de peticions. Es crea aquest cas d'ús per identificar una part important del projecte.

CAS D'ÚS	EXECUTAR PROCESSOS RECOMANACIÓ
DESCRIPCIÓ	Aquest mòdul engloba tota la funcionalitat referent a la realització d'un experiment: tractament de peticions i de consultes de recomanació, així com el càlcul de mesures de cada recomanació.
PRE-CONDICIÓ	S'ha rebut una petició de recomanació.
FLUX PRINCIPAL	1. Extreure dades d'operació de la petició. 2. Assignació d'usuari/comprador amb un infohabitant. 3. Fer consultes, condicionades per les dades d'operació. 3.1. Calcular les mesures per cada recomanació. 3.2. Guardar resultat de recomanació. 4. Crear resposta amb els resultats obtingut.
POST-CONDICIÓ	S'ha creat una resposta per cada cicle de recomanació.
OBSERVACIONS	El guardar els resultats d'una recomanació forma part de l'aprenentatge que fan els agents recomanadors de cada cicle.

3.2. ACTIVITAT GENERAL

Les tasques del laboratori es poden representar amb diferent activitats. Encara que en la seva representació s'hagi de distingir un principi i un final, degut als requeriments a seguir per dibuixar el diagrama, cada execució que faci l'actor usuari final es podrà fer, o s'ha de poder fer, independentment de la resta. L'aplicació gràfica, doncs, ha de permetre realitzar anàlisi de resultats, per exemple, sense necessitat d'executar prèviament una recomanació, ja que els resultats s'han de poder emmagatzemar, sigui en fitxers, sigui en una taula de la base de dades.

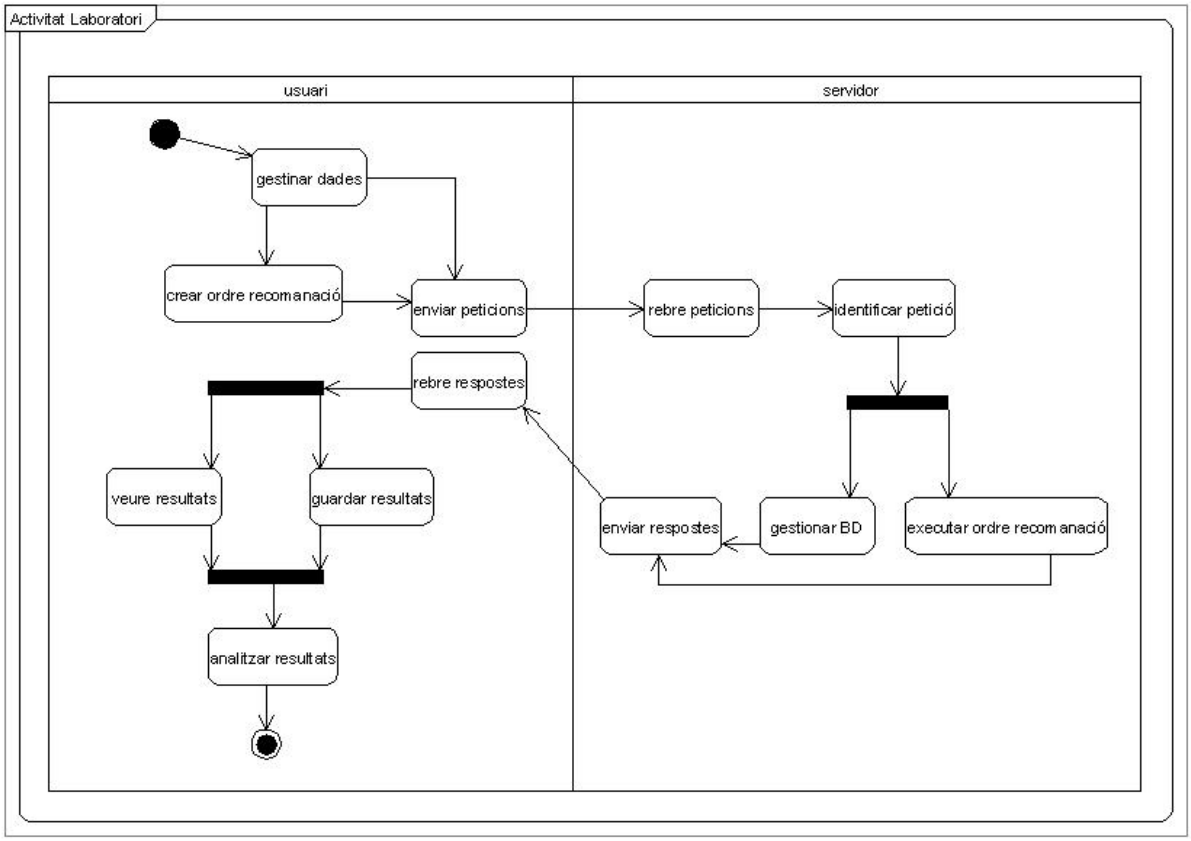


Figura 3.2.: Diagrama d'activitat, representant el flux de funcionalitat global de l'aplicació

La part client del laboratori, conté activitats de presentació de dades i configuració. Permet configurar els valors de les diverses dades d'operació. I també permet veure informació del magatzem de dades que sigui necessària per l'execució dels experiments, a més de veure els resultats d'aquests obtinguts per aquests experiments.

La part servidor coneix els valors dels paràmetres i reconeix quines peticions son enviades des de la part client, realitzant dues tasques principals: gestió i consulta de les dades emmagatzemades i executar diferents processos de recomanació.

3.3. ARQUITECTURA DEL LABORATORI

En la identificació de casos d'ús i activitats a realitzar per el laboratori, s'ha parlat de dues parts. La part client és una *aplicació gràfica* amb què l'usuari final podrà gestionar tota la funcionalitat presentada. Mentre que la part servidor conté el codi necessari per gestionar els diversos agents software i el magatzem de dades. Serà l'*hàbitat dels agents*.

També s'analitzarà el tipus d'implementació que ha de tenir la part servidor. Com s'ha citat amb anterioritat, no és un únic actor qui actua amb el codi servidor, si no que seran varis

agents software els qui controlaran la seva funcionalitat. Per tant, es farà anàlisi més exhaustiu d'aquesta part, identificant les funcionalitats concretes i quines activitats han de complir per controlar l'hàbitat.

3.3.1. Client/Servidor

Per elaborar el laboratori de recomanacions, s'utilitza un paradigma d'aplicació en xarxa anomenat model client- servidor. Aquest model assigna diferents rols als dos processos que col·laboren entre ells. El servidor, proveeix un servei - resultats de càlculs o resultats d'una consulta a un BD -, esperant de manera passiva l'arribada de peticions. Mentre que el client crea peticions al servidor i espera les respostes del servidor, per ser presentades a la persona que interactua amb l'aplicació.

Perquè existeixi comunicació, perquè col·laborin les dues parts de SIRA, ha d'existir un pas de missatges entre les dues parts. És necessari, doncs, un protocol per especificar les regles que han de compartir el client i servidor durant un servei, seguint amb un patró de petició i resposta. Les dues parts han de poder interpretar el llenguatge amb què es realitzarà el pas de missatges.

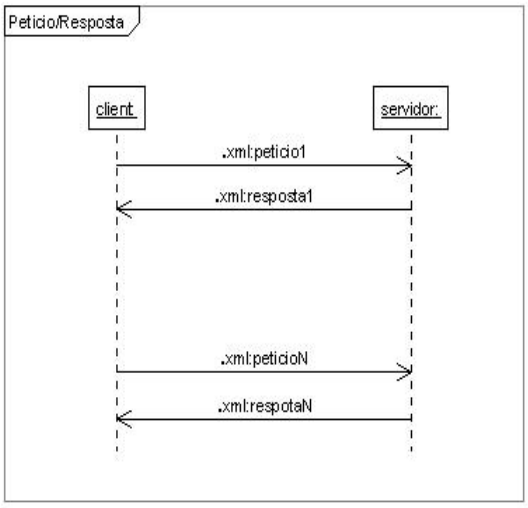


Figura 3.3.: Patró de petició- resposta de l'aplicació. S'hi mostra el llenguatge a emprar per la comunicació.

I per dur a terme el pas de missatges, hi ha d'haver un mecanisme que permeti que el procés client localitzi al procés servidor. De manera simple, la localització es realitza estàticament amb una adreça d'identificació formada per el nom de la màquina - o adreça IP física - i un número de port del protocol assignat al procés servidor.

3.3.2. Implementació del client

Sorgeix la necessitat de programar una aplicació que presenti les dades i resultats calculats en el servidor. La part client està pensada per mostrar una interfície al usuari final, el qual pugui canviar els valors a les dades d'operació i analitzar, veure, les dades d'informació que sol·licita.

També cal tractar les dades de manera modular, amb objectes que continguin atributs per fer una abstracció d'aquestes dades i mètodes que permetin la visualització d'aquestes i puguin ser codificades per ser comunicades del client al servidor, i viceversa.

Per el desenvolupament del projecte, en aquesta primera fase, es crearà una aplicació escriptori, així que cal un llenguatge que permeti la programació de pantalles, tractament d'accions sol·licitades per l'usuari final i que organitzi les dades a mostrar i les accions a raó d'objectes que encapsulin les dades i mètodes. Dels llenguatges de programació gràfica existents, **C#** supleix la necessitat d'una programació gràfica, facilitant la creació de pantalles i components. Així com també està orientat a programar amb objectes. C# permet la gestió d'accions llançades per un usuari final, quan aquest sol·licita que mostrin les dades per pantalla.

Gràcies a projectes de desenvolupament que circulen en l'actualitat, una aplicació en C# pot ser desenvolupada i executada sobre vàries plataformes o sistemes operatius.

3.3.3. Implementació del servidor

Per la implementació de les funcionalitats reals de l'habitat dels agents es farà ús d'objectes, que permeten una abstracció de dades reals i funcionalitats del servidor. Els mètodes i funcions que han de regir el l'execució de les funcionalitats, s'associen als objectes per proporcionar-los un comportament autònom. Disposar de les dades estructurades jeràrquicament permet la reutilització de mètodes i dades definides com a més generals en la jerarquia. També cal dissenyar i implementar objectes que estiguin relacionats entre sí per tal de gestionar la funcionalitat de cada mòdul del servidor, descrits més endavant.

Per això apareix la necessitat de treballar amb una Programació Orientada a Objectes, POO, i fer servir un llenguatge que permeti treballar amb aquest tipus de programació.

La programació del codi de servidor es realitzarà amb **JAVA**. Aquest llenguatge proporciona independència de la plataforma o sistema operatiu. És un llenguatge fortament tipificat: totes les variables i atributs que es defineixen tenen un tipus declarat i controlat per totes

les operacions i expressions. Està pensat per suplir les necessitats de la POO: la seva utilització facilita la programació perquè està completament orientat a programar amb objectes.

3.3.4. Gestor de dades

Amb la finalitat de realitzar les proves de recomanació per assolir l'objectiu del projecte, cal disposar d'un **Sistema Gestor de Bases de Dades**, **SGBD**, que permeti gestionar un magatzem de dades.

Per el projecte es disposarà de registres de compra o transaccions, organitzades de manera consistent, per donar lloc a un aprenentatge a partir d'elles, però no caldrà aprofundir massa en tenir les dades emmagatzemades en una arquitectura distribuïda pròpia d'un magatzem de dades. És a dir, no caldrà emmagatzemar les dades en varies bases de dades repartides en varis servidors. El *SGBD* ha de tenir la capacitat de controlar el seguiment de comportaments de compra d'usuaris al llarg del temps, és a dir, ha d'emmagatzemar una col·lecció de dades que sigui:

- *Orientada al subjecte*

  Les dades han d'estar organitzades a partir dels subjectes a analitzar, així com compradors, productes, transaccions...
- *Variant en el temps*

  Les dades son vàlides durant un interval de temps durant el qual son necessàries per l'anàlisi.
- *No volàtil*

  Les dades no son actualitzades dia a dia. La base de dades sempre ha d'estar adquirint noves dades, afegides a les dades emmagatzemades amb anterioritat o reemplaçant-les.

PostgresSQL és un gestor de dades de codi lliure, que pot ser instal·lat i usat sobre qualsevol plataforma. És un sistema gestor de dades relacionals que suporta la integritat referencial de dades per assegurar la validesa d'aquestes.

Permet la creació i/o declaració de sentències SQL, complint les especificacions SQL99 sobre la creació d'instruccions SQL. També fa possible la declaració d'operadors i funcions definides per l'usuari, fent ús de mètodes d'accés i tipus de dades, a part de proporcionar una extensa llibreria de funcions que pot ser utilitzada per la majoria de llenguatges de programació actuals, essent JAVA un d'aquests.

Com que l'aplicació a desenvolupar ha de permetre el treball de varis usuaris a la vegada, amb una connexió per cadascun, hi ha un parell de característiques que han fet decantar l'elecció cap a aquest SGBD. La primera és el control de concurrència, o *Multi-Version Concurrency Control*, que no bloqueja l'accés per la lectura de les dades mentre altres connexions hi accedeixen per permetre modificar el contingut de la base de dades. La segona és l'existència d'un procés mestre que habilita múltiples connexions per cada aplicació o usuari que s'hi intenti connectar. A més, conté un sistema d'enregistrament de canvis a la base de dades, que deixa constància de cada canvi abans que sigui executat. Així permet assegurar que en cas de fallida en l'execució d'alguna transacció, hi ha un registre de transaccions, de canvis, des del qual es pot restaurar les dades i l'esquema de la base de dades.

3.4. SERVIDOR AMB AGENTS SOFTWARE

Com s'ha esmentant en la descripció dels mòduls generals, i un cop presentada l'arquitectura que seguirà l'aplicació en la part servidor, es distingeixen varis actors o agents software que regeixen el comportament del codi de servidor.

3.4.1. Mòduls del servidor

Un dels requeriments del projecte és la implementació d'agents software que executin els processos de recomanació necessaris per les proves dels tipus de recomanació. Pensant en una Programació Orientada a Agents (POA), cal pensar que els actors que actuen en el comportament del servidor son agents software, cadascun dels quals ha d'encarregar-se d'una funcionalitat específica.

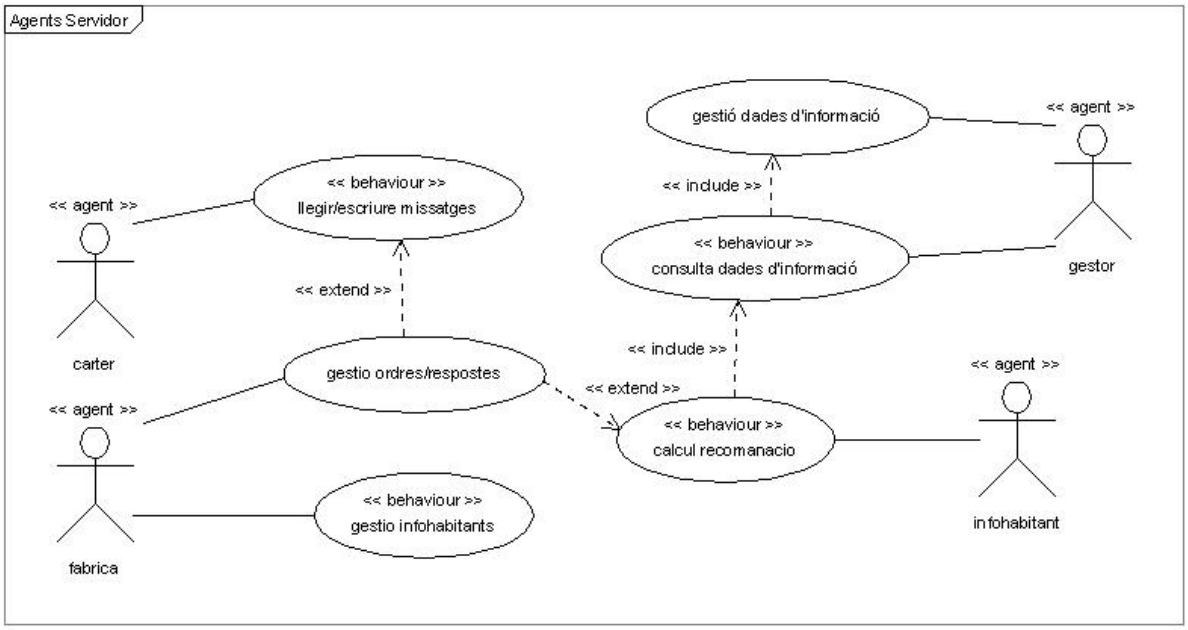


Figura 3.4.: Diagrama de cas d'ús, representant la funcionalitat del servidor.

3.4.2. Identificació d'actors

Els actors que intervenen en el servidor son agents. Cada agent té és una implementació de codi, amb unes característiques pròpies i executen el seu propi comportament o algorismes. Un agent, per si sol, pot controlar tot el funcionament del servidor però es creen varis agents per desacoblar les tasques principals:

- Un agent s'encarrega de gestionar les connexions a xarxa, les connexions de clients al servidor així com identificar les peticions rebudes i controlar l'enviament de respostes a servir. En endavant s'anomenarà *agent connector* o **carter**.
- Un segon agent, anomenat *agent consultor* o **gestor**, és l'encarregat de gestionar la connexió al SGBD i gestionar les dades que s'hi emmagatzemen.
- S'anomenarà *agent fabricador* o **fàbrica** a un tercer agent encarregat de la gestió dinàmica dels agents recomanadors. A partir de peticions de recomanació, ha de localitzar infohabitants vius o crear-ne de nous i distribuir-los les ordres de recomanació.
- Les peticions de recomanació rebudes son la base de treball d'un *agent recomanador* o **infohabitant**. Aquests realitzen les consultes i càlculs de cada experiment de recomanació.

3.4.3. Descripció dels mòduls

Amb les taules següents es descriuen breument les funcionalitast generals de cada mòdul.

CAS D'ÚS	**LLEGIR/ESCRIURE MISSATGES**
DESCRIPCIÓ	Controla el flux de dades de xarxa cap a la lògica interna del servidor, i a la inversa.
PRE-CONDICIÓ	
FLUX PRINCIPAL	1. Crear canal de comunicació. 2. Acceptar connexió entrant. 2.1. Llegir missatge de xarxa. 2.2. Distribuir missatge a la lògica interna. 3. Recepció de missatge de resposta de la lògica interna. 3.1. Escriure missatge a xarxa. 4. Deixar el canal de comunicació obert.
POST-CONDICIÓ	
OBSERVACIONS	

CAS D'ÚS	**CONSULTA/GESTIÓ DADES D'INFORMACIÓ**
DESCRIPCIÓ	Es descriuen els dos casos d'us a la vegada. Un s'encarrega de servir les peticions de consulta directa de la BD, mentre que el segon gestiona les dades emmagatzemades.
PRE-CONDICIÓ	Rebre una petició de consulta o modificació de les dades.
FLUX PRINCIPAL	1. Llegir dades de configuració de la petició. 2. Crear instrucció per accedir a les dades. 2.1. Crear resposta confirmació. 2.2. Crear resposta amb les dades consultades. 3. Passar missatge a l'agent comunicador.
POST-CONDICIÓ	La base de dades ha estat modificada o consultada.
OBSERVACIONS	

CAS D'ÚS	**GESTIÓ INFOHABITANTS/ GESTIÓ ORDRES- RESPOSTES**
DESCRIPCIÓ	Segons un valor dels paràmetres de configuració de les recomanacions, es creen tants agents recomanadors com indiqui el valor. Se'ls propaga les peticions de recomanació perquè les executin, i es recullen tots els càlculs que els recomanadors realitzen.
PRE-CONDICIÓ	No hi ha infohabitants registrats o localitzats
FLUX PRINCIPAL	1. Llegir dades de configuració, de la petició. 1.1. Llegir número d'infohabitants a crear. 2. Localitzar infohabitants vius. 3. Crear infohabitants.
FLUX ALTERNATIU	1. Localitzar infohabitants vius. 2. Distribuir a cada infohabitant la mateixa petició de recomanació. 3. Llegir de cada infohabitant un missatge de resposta.
POST-CONDICIÓ	L' agent fàbrica sap trobar i comunicar-se amb els agents recomanadors.
OBSERVACIONS	Si el número d'infohabitants que s'ha llegit de la petició és menor a la quantitat de vius, només caldrà localitzar-los. El flux alternatiu s'executa just després de crear o localitzar els infohabitants.

CAS D'ÚS	**CÀLCUL RECOMANACIÓ**
DESCRIPCIÓ	Executa els processos d'assignació d'usuari per a cada infohabitant creat, i executa els algorismes de recomanació que s'indiquen en la configuració, obtenint valors que mesuren l'èxit de cada recomanació.
PRE-CONDICIÓ	Els agents recomanadors estan localitzats i s'hi pot establir comunicació amb ells.
FLUX PRINCIPAL	1. Llegir dades de configuració, de la petició. 1.1. Llegir la tècnica i el tipus de recomanació a emprar. 2. Si hi ha usuaris per recomanar. 2.1. Assignar un usuari a un agent, segons tipus de recomanació. 2.2. Seleccionar dades a recomanar. 2.3. Calcular mesures de la recomanació feta. 2.4. Trencar assignació amb l'usuari, segons tipus de recomanació. 3. Crear missatge de resposta. 4. Passar el missatge a l'agent comunicador.
POST-CONDICIÓ	L'agent aprèn dades d'informació de l'usuari acabat de recomanar.
OBSERVACIONS	

3.4.4. Activitats del servidor

Seguidament es presenta, amb més detall, la interacció d'activitats que representen les funcionalitats del codi. Un cop s'engega el servidor, els agents queden en estat passiu mentre no hi ha cap aplicació client connectat. El diagrama, doncs, mostra la seqüència a seguir quan es connecta una aplicació client i es llegeix una petició.

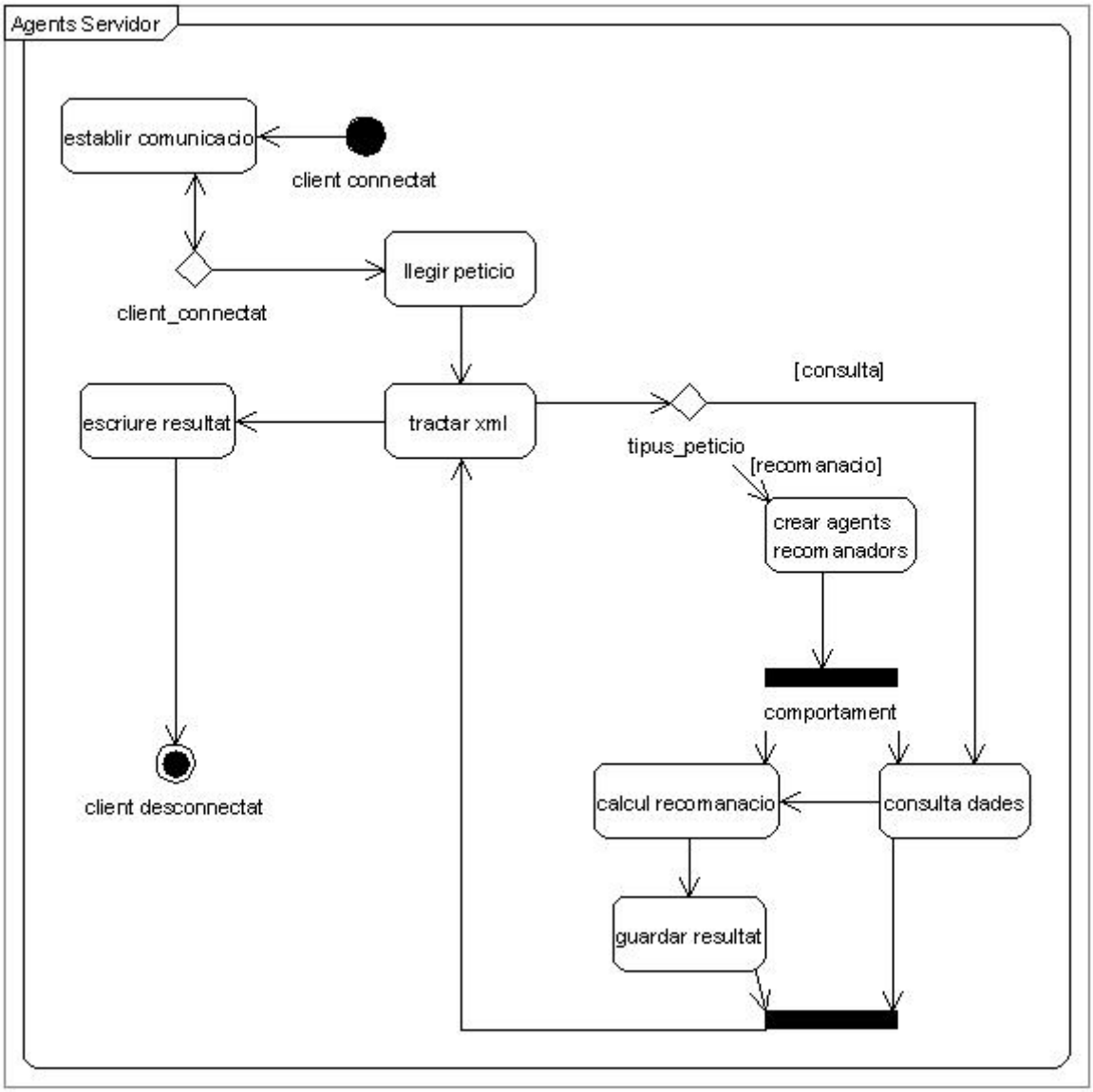


Figura 3.5.: Diagrama d'activitat del servidor. Funcionalitat general dels agents software.

S'observa en la Figura 3.5. que, segons el tipus de petició rebuda, se segueixen dues tasques distintes. El tipus de petició pot implicar una ordre de consulta, *consulta dades*, que serà tractada per el gestor, o una ordre de recomanació, que serà gestionada primer per l'agent fabricador, *crear agent recomanadors*, i finalment tractada per aquests, *consulta dades* i *càlcul recomanació,* per tal de servir els resultats calculats de recomanació a les pantalles de client.

4. DISSENY DE L'ARQUITECTURA

El disseny anirà acompanyat d'explicacions teòriques del tipus d'arquitectura analitzada, tant per la part client com per la part servidor, i dels patrons de disseny i del tipus d'implementació que s'aplicarà per l'elaboració del projecte.

Donat que el desenvolupament de varis objectes i la utilització de molts mètodes i atributs per cada objecte pot ser costós d'entendre si es mostra tot el conjunt d'aquests en un mateix diagrama, es partirà el disseny per mòduls, descrivint per cadascun d'ells les definicions de classe, objectes i crida de mètodes o missatges per resoldre la seva funcionalitat. També s'exposarà l'estructura i format dels missatges de comunicació que s'empraran per establir la comunicació entre les dues parts de l'aplicació.

A cada apartat es descriuran els objectes a fer servir i les crides als mètodes, de cada objecte, que cal implementar. Amb l'ajuda de diagrames de disseny s'explicarà la implementació de cada mòdul: *diagrama de seqüència*, per identificar interaccions entre objectes a través del temps i descriure una possible seqüència d'aquestes interaccions quan un cas d'ús és executat; *diagrama de col·laboració,* s'utilitzen per veure l'estructura dels objectes i les seves relacions, revelant els requeriments estructurals per completar una tasca i identificant els objectes que hi participen; *diagrama de classes*, per descriure un model de definicions dels recursos essencials (atributs i mètodes) per la correcte implementació del sistema. Aquests diagrames son l'origen per la generació de codi, o el destí per l'enginyeria inversa.[1]

[1] Com que el disseny i la implementació de l'aplicació han seguit camins paral·lels, és a dir, aquestes dues tasques s'han realitzat seguint una filosofia de programació extrema: s'ha realitzat la implementació al mateix temps que es dissenya la funcionalitat, part de la definició de les classes s'ha realitzat per la generació de codi, però contrastant i millorant la definició aplicant enginyeria inversa.

4.1. REPRESENTACIÓ DE L'ARQUITECTURA

És bo veure, abans d'entrar en detalls de disseny i implementació de les parts del laboratori, una representació general de l'arquitectura a implementar, *Figura 4.1*.

La part servidor conté un Sistema Multi Agent (SMA). En aquest SMA existeix una hàbitat d'agents que s'intercanvien informació per assolir els objectius globals de la funcionalitat descrita per el servidor. Els agents d'aquest SMA estan organitzats en dues subsistemes: subsistema de control, amb tres agents (amb tasques específiques cadascun) que controlen el funcionament del servidor; subsistema de simulació, amb infohabitants, on s'hi troba el Sistema Recomanador, SR.

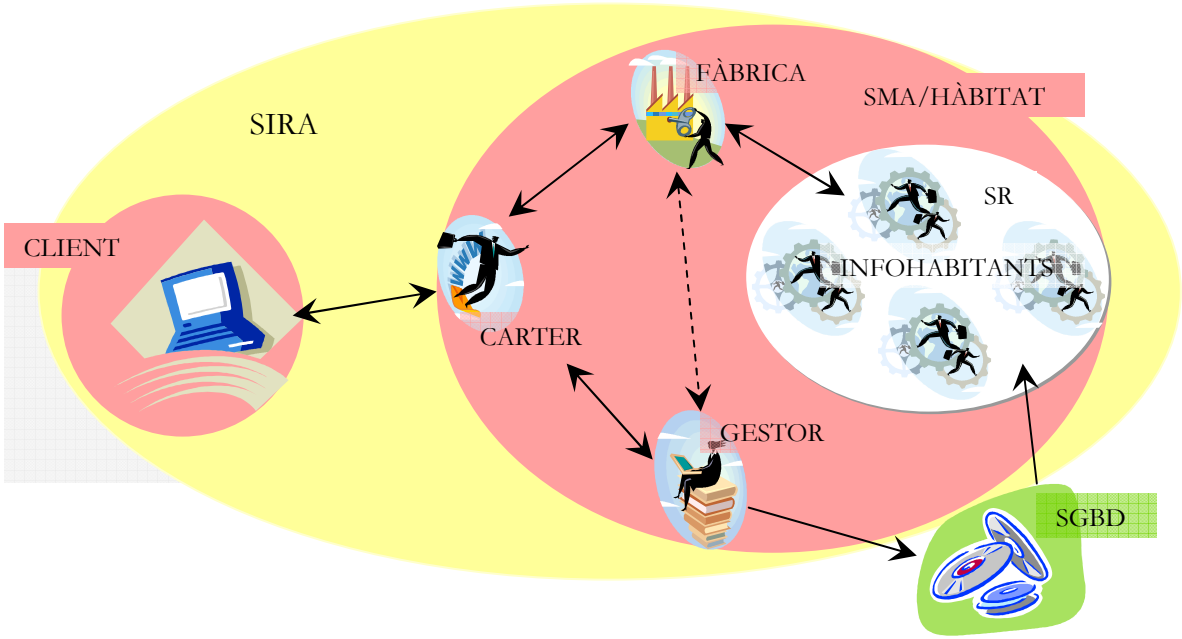


Figura 4.1.: Representació gràfica de l'arquitectura, amb l'aplicació gràfica, CLIENT, i l'hàbitat d'agents o SMA, SERVIDOR.

Una única aplicació client per el projecte serà suficient per experimentar amb l'hàbitat dels agents.

L'hàbitat està gestionat pels tres agents de control; agent carter que gestiona el flux de missatges entre client i servidor; agent gestor per alimentar el SGBD amb dades; agent fàbrica, per crear els infohabitants i recollir els resultats que aquests vagin generant. El SGBD no estaria dins de l'hàbitat però si estaria encabit en el mateix servidor. (Les funcionalitats són molt bàsiques i concretes per cada agent, per assolir l'objectiu final del projecte. Com es veurà en el treball futur, veure *apartat 6.1.*, aquesta arquitectura està oberta a noves implementacions o noves funcionalitats.)

En els següents apartats s'entrarà en detall en el disseny de la funcionalitat de cada part de l'arquitectura, donant èmfasi en el disseny dels agents software i com s'implementarà el codi del servidor, així com una descripció dels missatges que es comuniquen tant la part client i servidor (a fi de testejar els experiments) com els agents entre ells (a fi d'executar les funcionalitats descrites en l'apartat *3. Anàlisi de funcionalitats*).

4.2. CLIENT: MVC

Existeix la necessitat de mantenir una o múltiples vistes sobre les mateixes dades d'informació en el disseny de la part client del laboratori. Aquestes dades han de poder ser llegides per l'usuari final.

S'ha de desenvolupar una interfície d'usuari composta de formularis i components gràfics per veure i controlar els resultats de consultes i recomanacions.

En el desenvolupament d'aquesta part se segueix el patró de disseny Model-Vista-Controlador, MVC, que separa per capes els objectes que formen el codi de l'aplicació gràfica. Amb aquesta separació per capes tenim els objectes agrupats per funcionalitats o responsabilitats.

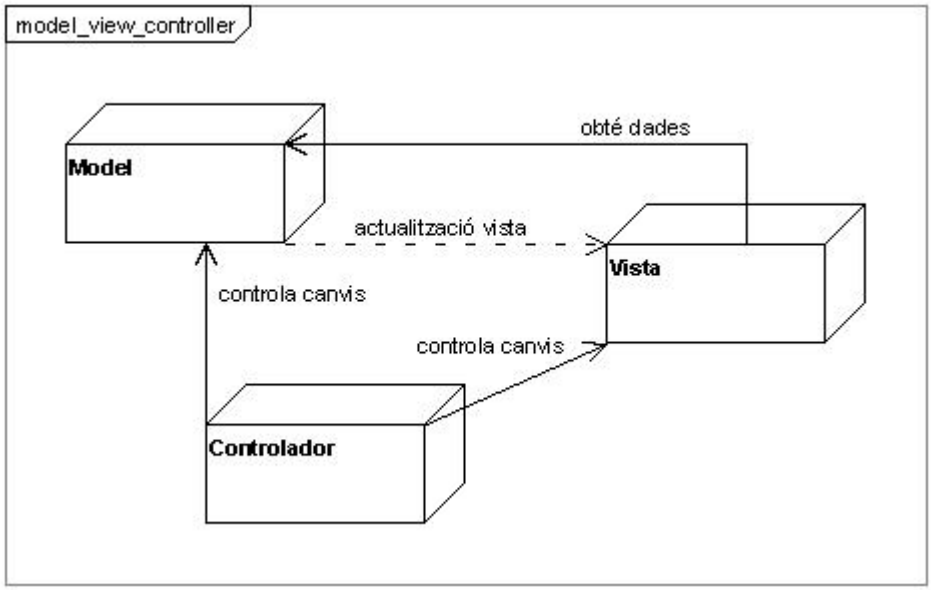


Figura 4.1. Diagrama representatiu del MVC

Els objectes de la capa de *model* son una representació de les dades amb què tractar. Els objectes de la capa de *vista* fan servir components gràfics per visualitzar les dades; la seva

funcionalitat no ha de ser més que mostrar les dades apropiadament - mostrar l'estat dels objectes del model - i gestionar la funcionalitat de components gràfics: botons, graelles, etc. Els objectes de la capa *controlador* recullen aquestes ordres i amb l'ajuda dels objectes del model, passen les ordres, en el cas d'aquest projecte, perquè s'executin en la part servidor. Els objectes controladors gestionen el flux d'informació que s'envia i es rep del servidor, modifiquen el valor dels objectes del model, guardant-hi la informació rebuda de manera pugui ser mostrada per els respectius objectes de la vista.

No hi ha hagut cap especificació concreta de com havia de ser la interfície gràfica de SIRA, per tant l'elecció d'aquesta solució i la seva implementació s'han basat en els coneixements propis sobre l'enginyeria del software, adquirits al llarg de l'etapa d'aprenentatge.

4.3. DISSENY DE MÒDULS: CLIENT

Per dissenyar la funcionalitat dels mòduls de la part client, cal identificar els objectes i definir els tipus o classes de què seran instàncies aquests objectes.

El disseny de classes i objectes se separa per cada mòdul descrit en els diagrames de cas d'ús. Encara que la informació pugui semblar redundant, d'aquesta manera en el disseny de la implementació dels mòduls s'hi distingiran només aquelles classes que hi intervinguin. Amb aquesta mesura s'intentarà aportar una millor lectura i entesa del disseny i evitar crear diagrames amb massa informació, cosa que pot dificultar la lectura d'aquests.

4.3.1. Gestió de paràmetres de configuració

Els paràmetres de configuració son les dades d'operació de les quals s'alimenta l'hàbitat d'agents per dur a terme els processos de recomanació. Però no només es descriu la funcionalitat del mòdul per guardar aquestes dades, si no que també és aplicable per les dades de connexió necessàries per connectar les parts client i servidor del laboratori.

1. Col·laboració entre objectes

Tant les dades de configuració de procés com les dades de connexió, han d'estar disponibles per ser configurades per un usuari final. Aquestes es guardaran en fitxers i s'aprofita que l'enviament de dades entre client i servidor es fa en format XML, per guardar les dades en el mateix format i homogeneïtzar la lectura i escriptura d'aquestes.

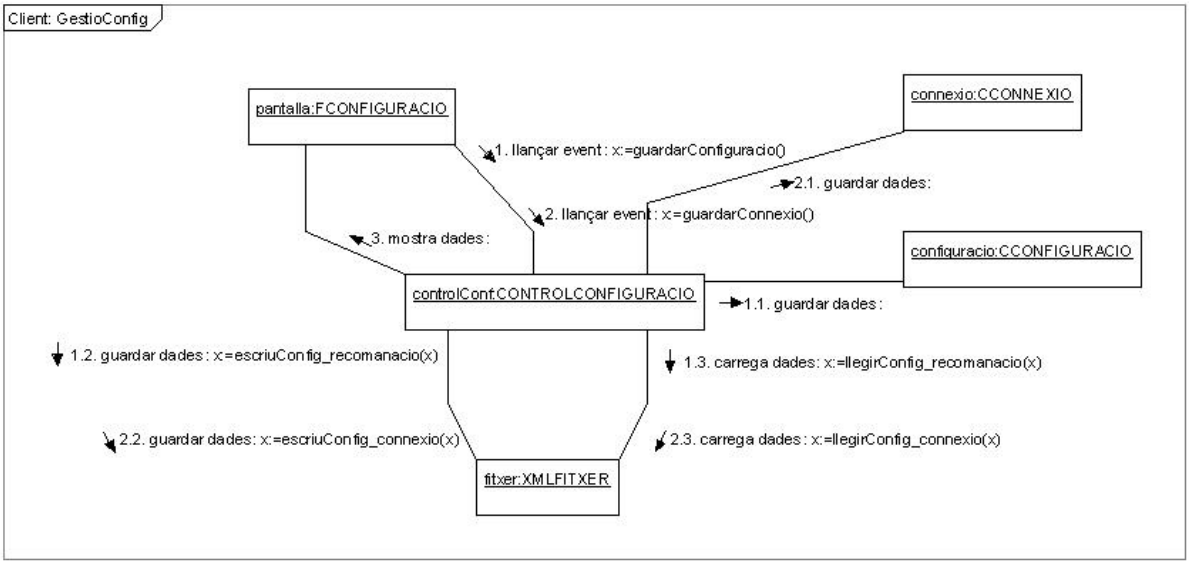


Figura 4.8.: Diagrama de col·laboració: Tractament de dades de configuració de recomanació i dades de connexió, lectura i escriptura d'aquestes.

Com s'aprecia en el diagrama anterior, hi ha un objecte **XMLFITXER** que s'encarrega d'accedir als fitxers on es guarda la informació de configuració de procés i de connexió. Aquest objecte ha de saber com llegir i escriure dades en format *XML* i poder gestionar aquests fitxers.

Un segon objecte a destacar és el **CONTROLCONFIGURACIÓ**, (sorgeix de l'aplicació del patró de disseny *Controller*2) que gestiona tots els missatges entre objectes, manté les dades en memòria, amb els objectes tipus **CCONNEXIO** i **CCONFIGURACIO** i permet al formulari accedir a aquestes dades, guardades en memòria, i mostrar-les per pantalla. El formulari ha de disposar de components gràfics que permetin:

- *Modificar les dades de configuració o connexió,*
- *Guardar les dades de configuració o connexió,*
- *Carregar les dades de configuració o connexió*

[2] Patró de disseny de classes per dissenyar objectes que controlin tota o part de la funcionalitat d'una aplicació a fi de desacoblar aquests mètodes d'un objecte interfície o d'un objecte de definició de dades.

2. Seqüència de crides

Es dissenya la crida d'interaccions entre objectes per les dades de configuració de procés, però la seqüència és igualment vàlida per les dades de connexió.

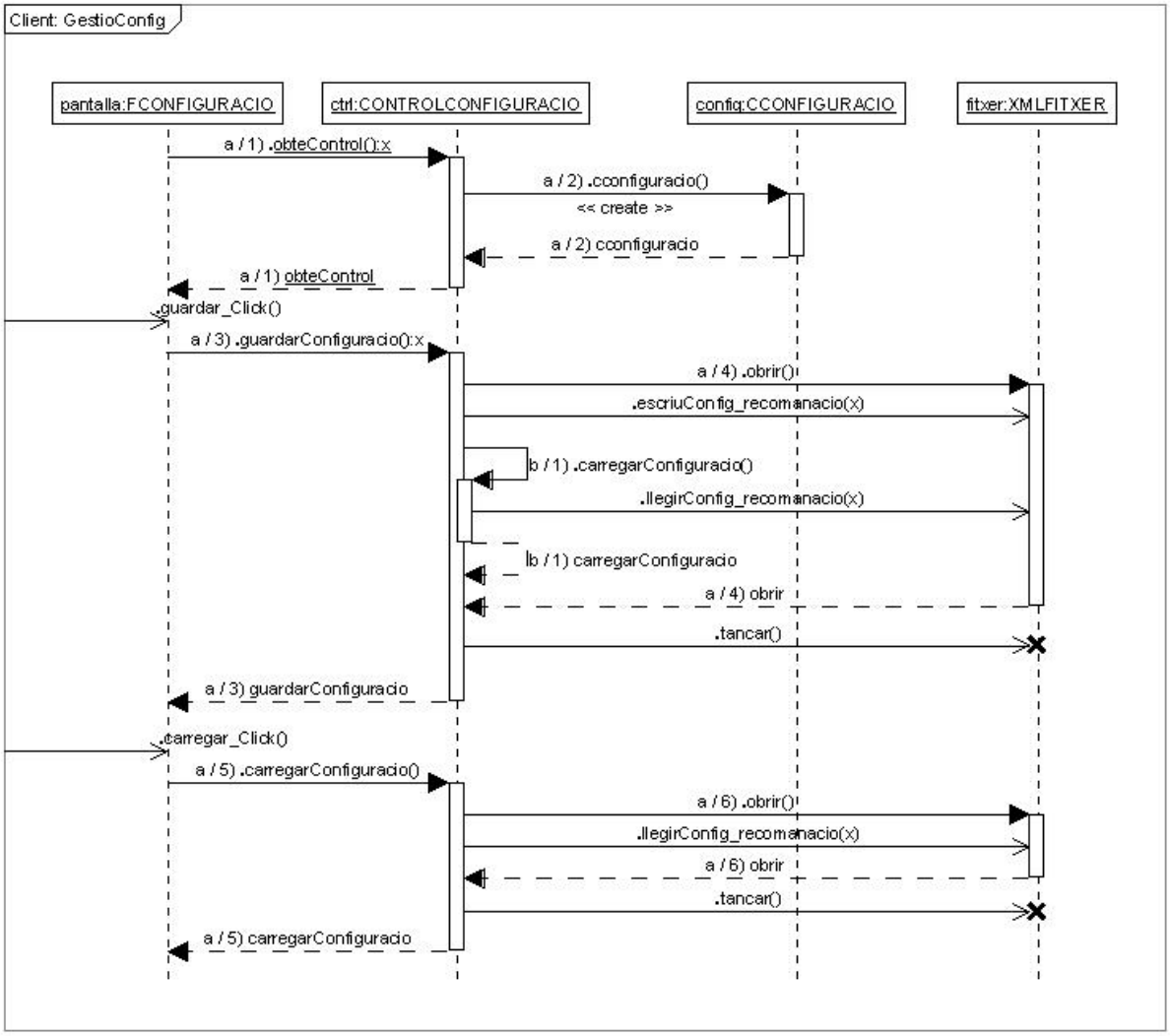


Figura 4.9.: Diagrama de seqüència: Lectura i escriptura de dades de configuració de recomanació.

3. Definició de classes

Vistos els objectes necessaris per representar la implementació i la crida de mètodes que cal seguir per controlar la funcionalitat d'aquest mòdul, tot seguit es presenta l'estructura de cada objecte, amb els mètodes i atributs a fer servir.

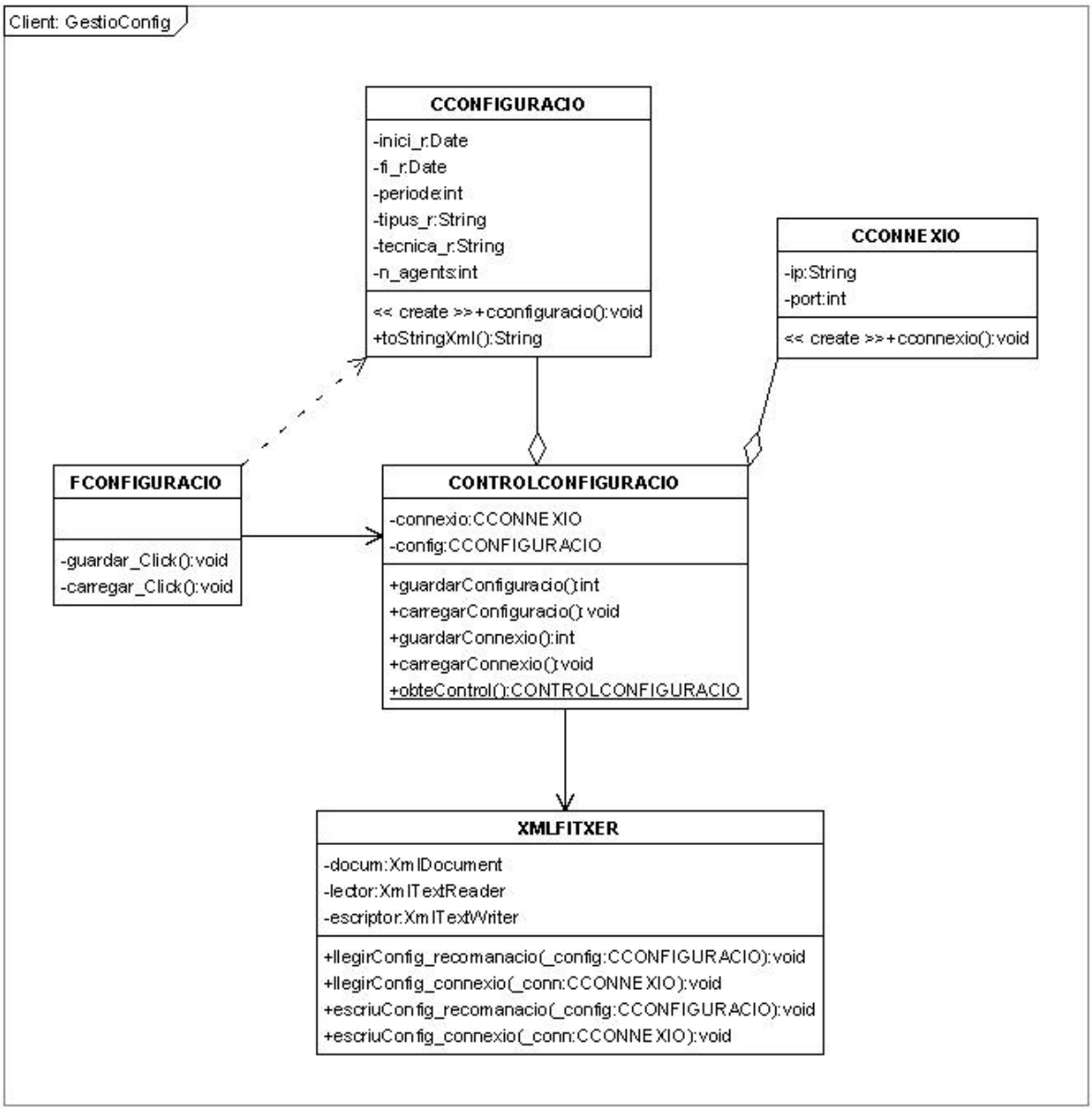


Figura 4.10.: Diagrama de classes. Definició dels atributs i mètodes a fer servir per la implementació del mòdul.

4.3.2. Comunicació Client/Servidor

1. Col·laboració entre objectes

Amb el següent diagrama es mostra la funcionalitat per establir connexió cap a l'hàbitat. Els passos descriuen la seqüència de col·laboració entre els objectes, per comprovar que l'establiment de la connexió s'ha fet correctament. L'esquema, però, és aplicable a l'enviament de qualsevol petició; proporcionarà un patró d'implementació per crear i enviar qualsevol petició.

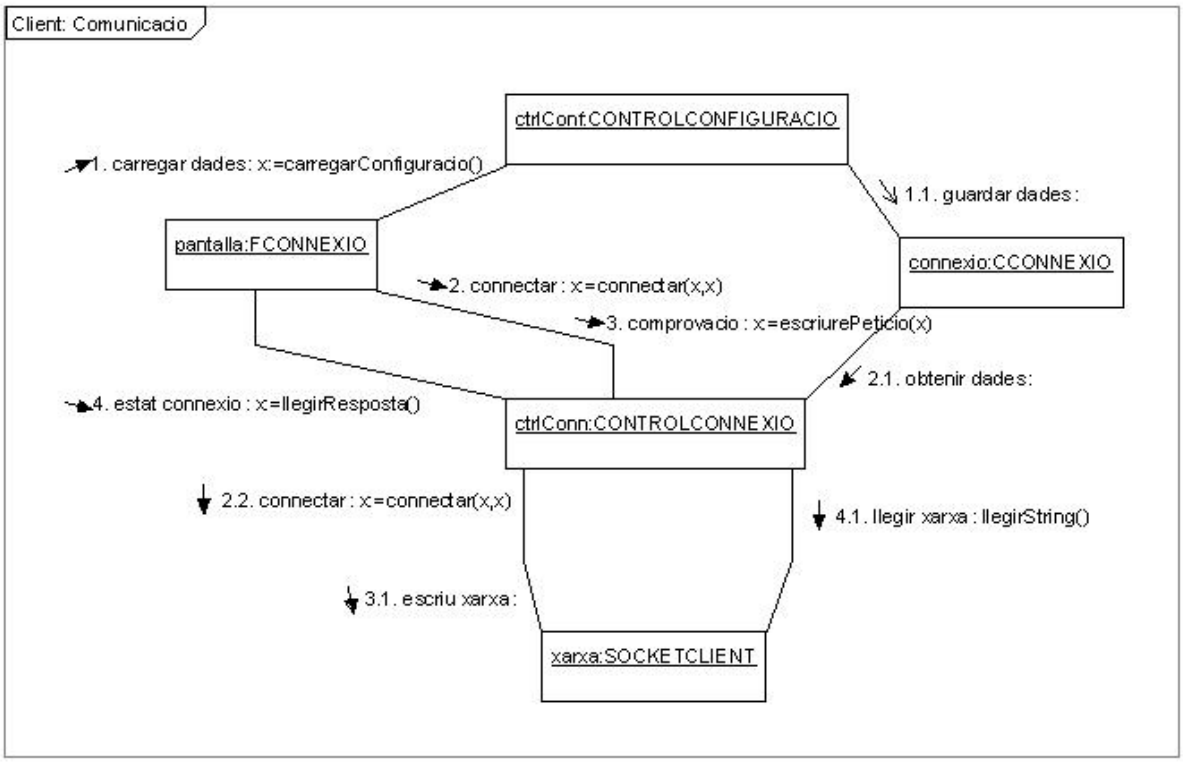


Figura 4.11.: Diagrama de col·laboració: Càrrega de valors de connexió per comprovar el seu establiment.

Si els paràmetres de connexió no estan disponibles en l'objecte *CCONNEXIO*, l'objecte de control carregarà aquestes dades des del fitxer corresponent - estímuls número 1.x -. Aquestes dades s'empraran per crear un objecte **SOCKETCLIENT** que serà el que gestionarà el pas de missatges de la xarxa i cap la xarxa, i que aquest romangui disponible mentre l'aplicació gràfica està encesa - estímuls número 2.x -.

La funcionalitat que es pretén descriure amb al diagrama és que la classe **CONTROLCONNEXIO** implementi una cua de missatges entrants i una cua de missatges sortints, i que aquests missatges s'enviïn i es recullin sense que la interfície quedi bloquejada a l'espera de la recepció de tots els missatges - estímuls número 3.x i 4.x -. Aquesta funcionalitat també serveix per l'enviament de qualsevol tipus de petició XML i per la recepció de les moltes respostes que es puguin rebre de l'hàbitat servidor.

2. Seqüència de crides a mètodes

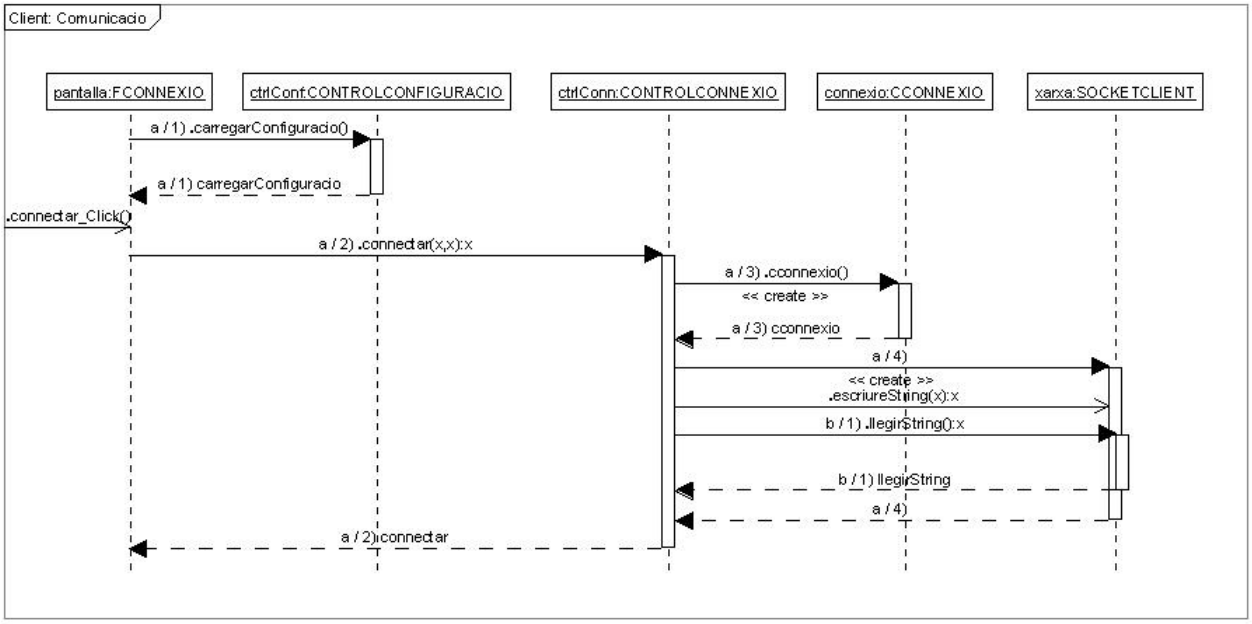


Figura 4.12.: Diagrama de seqüència: Creació i comprovació de la connexió establerta.

Un cop s'invocarà al mètode connectar(ip, port) només quan es connecti la interfície gràfica amb la xarxa. L'objecte de control es guardarà instància d'aquest i només farà servir els mètodes d'escriptura i lectura per controlar el pas de missatges. Aquest esquema d'enviament servirà per qualsevol petició.

3. Definició de classes

Aquí es mostra en detall els mètodes més importants a fer servir per l'enviament de peticions i la recepció de respostes XML.

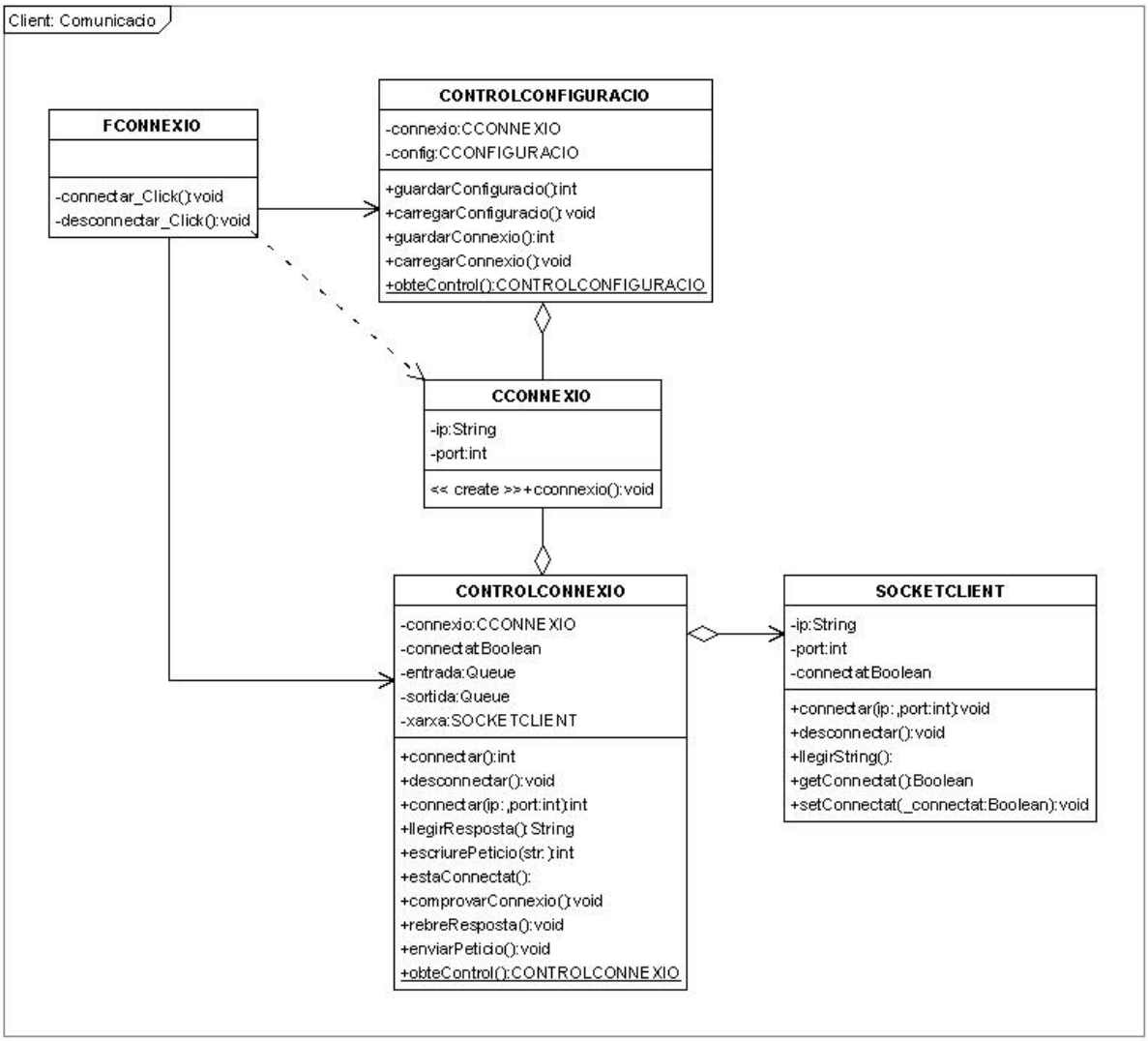


Figura 4.13.: Diagrama de classes. Definició dels atributs i mètodes a fer servir per la implementació del mòdul.

La classe *CONTROLCONNEXIO* implementa dues cues que contindran els missatges XML de petició i resposta que s'enviïn o es rebin de client a servidor i de servidor a client. La funcionalitat es descriurà més endavant.

4.3.3. Experiment de recomanació

En la descripció d'aquesta funcionalitat intervenen les dades d'operació (o paràmetres de configuració d'experiment) juntament amb dades de compradors que formen part d'un segment segons els valors *RFM*. Per tant, es descriuran les classes i seqüències per crear una petició de recomanació i la recepció d'una resposta d'un infohabitant en un cicle de recomanació.

1. Col·laboració entre objectes

Com a primer pas per aquesta funcionalitat, cal disposar dels paràmetres de configuració de l'experiment a realitzar, les dades d'operació corresponents, i el segment de compradors sobre el que es vol experimentar: la pantalla **FEXPERIMENT** mostrarà aquest segment, a raó d'una llista, i amb *CONTROLCONFIGURACIO* es carregaran les dades d'operació - estímul 1.x -.

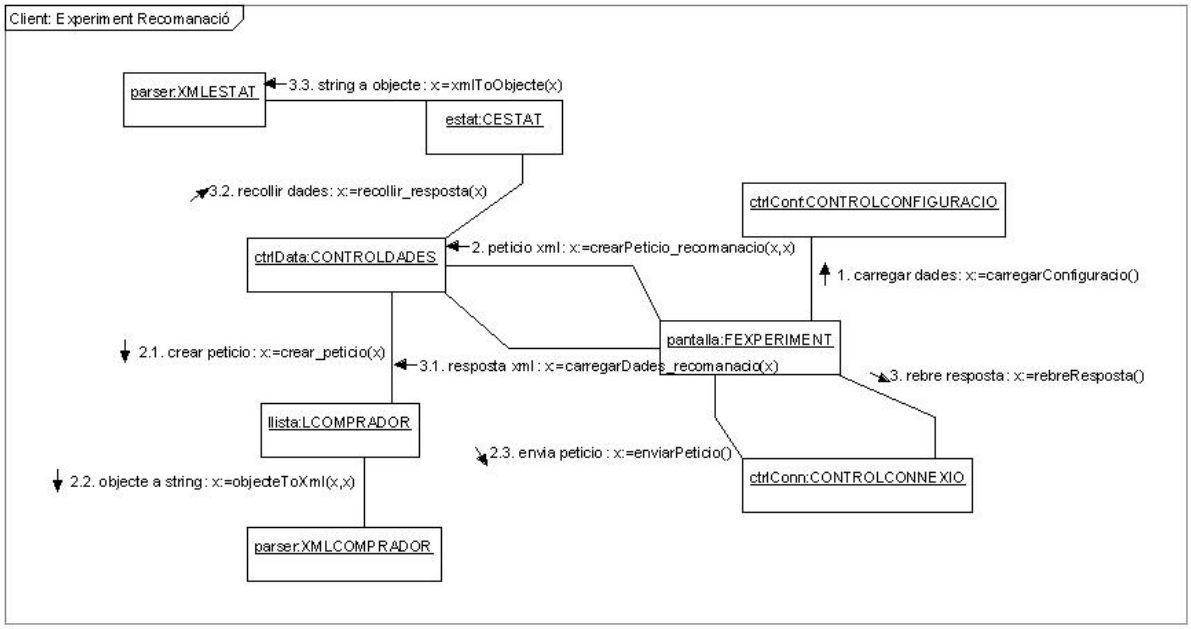


Figura 4.14.: En el diagrama es mostren tres seqüències distintes; passos a seguir per veure resultats d'un cicle de recomanació.

Amb l'ajut de l'objecte **CONTROLDADES** es crea la cadena de caràcters que contindrà les dades de la petició de recomanació a enviar a l'hàbitat dels agents recomanadors - estímul 2.x - . Aquesta cadena XML es guardarà en una cua de sortida de *CONTOLCONNEXIO* per notificar l'enviament d'aquest XML.

L'enviament i recepció de peticions i respostes XML es controlarà amb un fil d'execució per cada bústia - bústia d'entrada i bústia de sortida - implementades en aquest objecte de control.

Cada missatge de resposta que es rep conté informació d'un cicle de recomanació o estat. Les dades de cada resposta es guardaran en memòria, amb l'estructura que marcarà l'objecte **CESTAT** a partir de l'objecte encarregat de tractar aquestes dades d' XML a objecte, **XMLESTAT**, i es mostraran per pantalla - estímul 3.x -.

2. Seqüència de crides a mètodes

Les dues funcionalitats anteriors s'utilitzaran també per el tractament d'una petició XML. Vistes, doncs, les descripcions d'aquestes funcionalitats, en el següent diagrama només es farà referència a les crides a mètodes i la seqüència que intervinguin en el tractament mencionat.

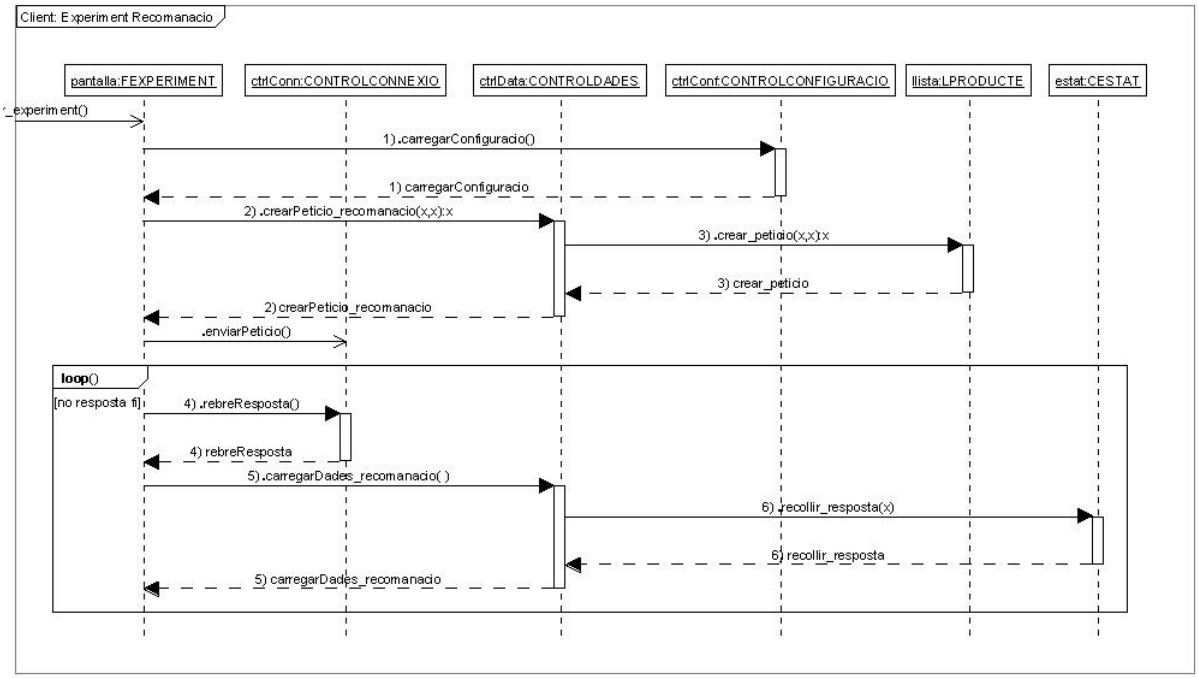


Figura 4.15.: Seqüència de mètodes a executar per engegar un procés de recomanació i anar rebent respostes de cada cicle de recomanació.

Un cop tractades les dades d'operació i la llista de compradors, es passa la petició XML a la cua/bústia de sortida de *CONTROLCONNEXIO*. L'enviament de la petició s'ha de fer sense que la pantalla quedi a l'espera de resposta.

Tenint en compte que els agents recomanadors enviaran vàries respostes cadascun, la recepció d'aquestes s'ha de fer cíclicament, mentre no es rebi una resposta final que indiqui que l'experiment ha finalitzat. Un cop llegida una resposta de la cua/bústia d'entrada, aquesta es passa a l'objecte *CESTAT* per ser tractada i posteriorment mostrada per pantalla.

3. Definició de classes

Amb el diagrama següent es mostra les relacions entre classes que intervenen en la implementació d'aquest mòdul.

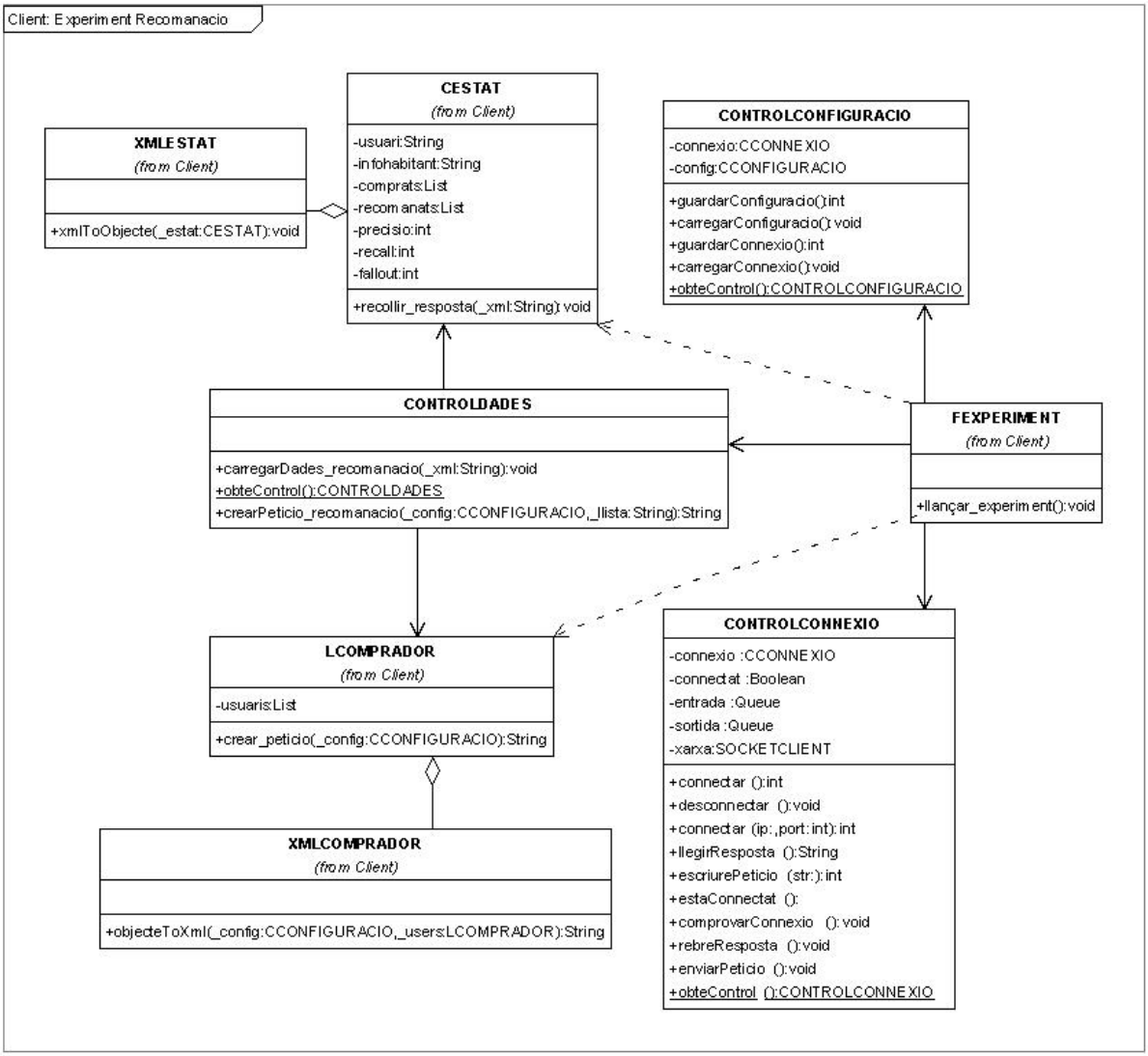


Figura 4.13.: Atributs i mètodes a fer servir per implementar la funcionalitat del mòdul.

Cada classe de control segueix el patró de disseny *Singleton*[3] a partir del qual s'implementa un únic objecte instància de cada classe, és a dir, una única instància de cada classe de control a utilitzar per la implementació de la interfície gràfica. S'utilitza aquest patró per la necessitat de disposar en memòria dades que poden ser visualitzades per el conjunt de pantalles.

4.4. COMUNICACIÓ CLIENT/SERVIDOR I LLENGUATGE

Per la comunicació entre les dues parts se segueix un patró petició - resposta. Aquest patró s'implementarà usant el llenguatge **XML**.

XML proporciona conjunt de regles o pautes usades per dissenyar formats de text i permetre estructurar dades o informació. La creació de missatges amb aquestes regles és senzilla: s'organitzen les dades ben estructurades jeràrquicament i sense ambigüitats.

Un document XML utilitza etiquetes o *tags* i atributs, no per donar un significat de programació al contingut, com en *HTML*, si no per guardar informació. XML guarda les dades en format text, però la presència de *tags* el fa de difícil lectura, encara que si que és comprensible a simple vista permetent verificar o corregir qualsevol possible errada en les dades o mala distribució d'aquestes.[4]

Amb aquesta breu definició, es tria XML per codificar els missatges de petició i resposta que s'han de passar entre les parts client i servidor per establir comunicació entre aquestes. Aquest format és fàcil de codificar i interpretar en les dues parts de SIRA . Encara que és un format de text, en ser enviat per xarxa un missatge XML és codificat com a binari, amb les avantatges que aquest tipus de codificació comporta respecte una codificació en text: el pas per xarxa dels paquets que formen el missatge XML és més ràpid, per exemple.

S'aprofita la organització estructurada que proporciona XML per poder codificar i interpretar cada missatge de petició i/o resposta.

---

[3] Patró de disseny de classes el qual s'aplica per restringir la creació d'objectes a partir d'una classe. Permet un accés global a l'únic objecte creat de la classe. Per el projecte, les dades que controla la classe de control han de ser accessibles per varis objecte.

[4] Resum de la definició d'XML que es pot trobar a la pàgina web de *World Wide Web Consortium* http://www.w3.org/XML/1999/XML-in-10-points

A continuació es mostrarà el disseny d'aquests missatges, a raó de patró de comunicació entre parts de l'aplicació.

4.4.1. Creació de perfils

Per realitzar els experiments de recomanació l'usuari final ha de poder consultar perfils de compradors i seleccionar sobre quin segment d'aquests vol experimentar les recomanacions. S'envien les dades de configuració, indicant a quin període es consulten els compradors - es delimita el número de compradors, els que compren entre unes dates definides, per agilitar l'execució de les consultes sobre al BD - i el tipus de segmentació i criteris de consulta:

```
<peticio tipus="consulta">
      <configuracio>
      <data ini_epoch=" " inici_r=" " fi_r=" " />
      <periode unitat=" " valor=" " epoca=" " />
      <recomanacio tipus=" ">
            <tecnica candidats=" ">" "</tecnica>
            <classificacio comparacio=" " tall=" ">" "</classificacio>
            <agents tants=" " percent=" " />
      </recomanacio>
      <segmentacio>
            <data inici_seg=" " fi_seg=" " />
            <perfil ordre_u=" " tants=" ">" "</perfil>
      </segmentacio>
      </configuracio>
</peticio>
```

Per simplificar la comunicació s'envien totes les dades de configuració. L'objecte que s'encarrega de tractar aquesta informació recopila tots els paràmetres i només tracta els que son necessaris en cada cas.

Com a resposta per una llista d'usuaris segmentada segons els paràmetres establerts per la tècnica RFM emprada en el projecte:

```
<resposta>
      <set_usuaris>
      <usuari codi=" " nom=" " r="" f="" m="" />
      ...
      </set_usuaris>
</resposta>
```

4.4.2. Gestió de dades

En el cas pràctic del projecte, hi ha tres tipus de dades d'informació a emmagatzemar a la base de dades. I la gestió a realitzar ha de ser una actualització del magatzem de dades, a raó d'afegir noves dades d'informació a les existents o d'actualitzar la informació del magatzem, suprimint les existents per noves.

a) Dades d'usuaris:

```
<peticio tipus="importacio">
      <taula nom=" " eliminaDades=" " />
      <set_usuaris>
      <usuari codi=" " nom=" " />
      ...
      </set_usuaris>
</peticio>
```

b) Dades de producte:

```
<peticio tipus="importacio">
      <taula nom=" " eliminaDades=" " />
      <set_productes>
      <producte grup=" " linia=" " marca=" " nom=" " clau=" " />
      ...
      </set_ productes>
</peticio>
```

c) Dades de transaccions:

```
<peticio tipus="importacio">
      <taula nom=" " eliminaDades=" " />
      <set_transaccions>
      <transaccio data=" " usuari=" " producte=" " despesa=" " />
      ...
      </set_transaccions>
</peticio>
```

L'etiqueta d'actualització regeix el tipus de gestió a fer, si cal afegir noves dades a les existents o afegir-les de nou.

Com a resposta es rep un simple missatge informatiu informant de l'èxit o fracàs de la operació de gestió:

```
<resposta>
      <fi>"missatge informatiu"</fi>
</resposta>
```

4.4.3. Ordre de recomanació

S'executarà després d'una petició de consulta. Inclou dades de configuració amb les dades de comprador seleccionades:

```
<peticio tipus="recomanacio">
      <configuracio>
      <data ini_epoch=" " inici_r=" " fi_r=" " />
      <periode unitat=" " valor=" " epoca=" " />
      <recomanacio tipus=" ">
            <tecnica candidats=" ">" "</tecnica>
            <classificacio comparacio=" " tall=" ">" "</classificacio>
            <agents tants=" " percent=" " />
      </recomanacio>

      <segmentacio>
            <data inici_seg=" " fi_seg=" " />
            <perfil ordre_u=" " tants=" ">" "</perfil>
      </segmentacio>
      </configuracio>
      <set_usuaris>
            <usuari codi=" " nom=" " />
            ...
      </set_usuaris>
</peticio>
```

Cada *infohabitant* enviarà un missatge resposta amb el resultat de cada cicle o estat de recomanació:

```
<resposta>
      <estat id=" " n_encert=" " precisio=" " recall=" " fallout=" ">
      <data inici= " " fi= " " />
      <infohabitant trust= " " >"nom de l'infohabitant"</infohabitant>
      <usuari codi=" " nom=" " />
      <comprat n_comprats=" ">
            <set_productes>
            <producte grup=" " linia=" " marca=" " nom=" " clau=" " />
            ...
            </set_productes>
      </comprat>
      <recomanat n_recomanats=" ">
            <set_productes>
            <producte grup=" " linia=" " marca=" " nom=" " clau=" " />
            ...
            </set_ productes >
      </recomanat>
      <fi>"SI"/"NO"</fi>
      </estat>
</resposta>
```

La marca de *<fi>* indica si hi ha més dades d'estat per enviar o si tots els infohabitants han acabat els seus respectius processos.

Els missatges estudiats i dissenyats formaran la part principal de la comunicació entre el client i el servidor del laboratori. Pot existir algun patró més, com el de verificació de connexió al servidor per part d'un client, però no és una qüestió important del projecte per tant no s'hi entrarà en més detall.

Com s'ha pogut observar, en els últims patrons XML s'envia informació del usuari/comprador. Aquestes dades formen part de cada experiment, per tal que l'usuari final del laboratori pugui analitzar millor cada experiment. Cal tenir clar que l'estudi de la privacitat de dades de comprador només s'aplica per protegir la informació d'usuari continguda en un infohabitant

4.5. SERVIDOR: POA

El codi a dissenyar i implementar ha d'estar pensat per treballar amb una **Programació Orientada a Agents** (**POA**), segons els requeriments i anàlisi vistos amb anterioritat. Aquest tipus de programació, és una extensió de la **Programació Orientada a Objectes (POO)**, la qual es basa en la programació d'objectes agents

Per aquest tipus de programació serà necessari fer servir un llenguatge o eina de programació que permeti dissenyar i organitzar les dades i els mètodes entorn dels agents, i permetrà entendre el disseny del codi del servidor.

4.5.1. Implementació de POA amb JADE

Per la creació dels agents és necessita una infraestructura o eina de programació que faci possible implementar: piles de protocols en els agents, mecanismes d'enviament i recepció de missatges, localització dels agents, identificació i cicle de vida dels agents (creació, destrucció, supervisió i replicació).

JADE és una eina de programació o *framework*, completament desenvolupada en llenguatge JAVA, que serveix per desenvolupar sistemes basats en agents, complint en l'estàndard FIPA[5].

JADE ofereix serveis de noms i pàgines grogues i transport de missatges, per facilitar la comunicació inter-agent. Implementa, també, un entorn d'execució per agents, a més de proporcionar una llibreria de classes JAVA funcionalitats llestes per ser usades i tipus de dades o *interfaces* per crear els propis mètodes i classes, i un paquet d'eines gràfiques per l'administració i monitorització dels agents.

L'entorn d'execució JADE crea una plataforma distribuïda que s'executa sobre la màquina virtual de JAVA de cada computador on hi estigui instal·lat. Cada màquina virtual és un contenidors d'agents, que proporcionen un entorn per l'execució d'aquests i permeten que s'executin concurrentment, tractant cada agent com un únic fil d'execució, *thread* de JAVA, o com

[5] Veure la pàgina web de *Foundation for Intelligent Platform Agents, http://www.fipa.org*

a varis fils d'execució, segons les tasques a realitzar. Per la gestió de la plataforma, JADE inclou varis agents:

- *AMS (Agent Management Service)*
  Controla l'accés a cada plataforma, monitoritza la resta d'agents creats a la plataforma i manté un directori d'identificadors i estats dels agents registrats.
- *DF (Directory Facilitator)*
  Crea a un servei de pàgines grogues amb les adreces de cada agent registrat al AMS, facilitant la cooperació i comunicació inter-agent.
- *ACC (Agent Comunication Channel)*
  Controla tot l'intercanvi de missatges entre agents d'una mateix plataforma o de plataformes remotes.

JADE introdueix el concepte de *Behaviour* o comportament per suportar activitats paral·leles que pugui executar un agent - varis fils d'execució -. Cada comportament és un gestor d'esdeveniments que descriu, amb la programació de codi, com l'agent ha de reaccionar segons certs estímuls: per exemple, la recepció d'un missatge. Per tant, les tasques d'un agent venen especificades per un o varis comportaments. JADE implementa una agenda per activar els comportaments. L'agenda segueix un algorisme de *Round-Robin*[6] per gestionar l'execució de cada comportament. L'activitat, execució de comportaments, de cada agent consisteix en una alternança de fases actives que succeeixen quan l'agent executa una acció i envia un missatge, amb fases passives: quan l'agent espera una resposta en forma de missatge.

Per establir la comunicació entre agents, JADE proporciona un sistema de cua o bústia privada on s'hi guarden els missatges enviats per altres agents. Cada cop que un missatge arriba, l'agent receptor és notificat i actua segons el codi implementant en el comportament de cadascun.

[6] **Round-Robin** és un algorisme de selecció i planificació equitativa i ordenada, que tracta tots els processos amb la mateix prioritat, començant pel primer fins a arribar a l'últim i tornant a començar pel primer. Assigna a cada procés el mateix temps d'execució.

Els missatge en JADE compleixen l'estàndard FIPA de comunicació, **ACL** (**A**gent **C**omunication **L**anguage). Un missatge ACL conté:

- *L'emissor del missatge*
- *Un llista de receptors*
- *La intenció comunicativa del missatge: utilitza els tipus de missatge definits per FIPA.*
- *El contingut del missatge*
- *El llenguatge del contingut.*
- *Un nom d'ontologia*
- *Un identificador de conversa.*

Encara que es tinguin en compte aquestes característiques per identificar un missatge, JADE també proporciona patrons, **Message Templates**, i mètodes de recepció de missatges, que reben per paràmetre aquests patrons, per filtrar els missatges de varis agents o distingir varis tipus de missatge.

4.5.2. Aplicar JADE per desenvolupar la POA amb POO

Gràcies a la eina JADE, vista en el capítol d'anàlisi, es pot realitzar la programació dels agents com si es tractés d'una programació orientada a objectes corrent:

- Un agent pot *encapsular* informació, com a atributs, i mètodes que permetin canviar la informació o accedir a aquests atributs. Conté *informació amagada* o privada com a estat intern, encara que defineix quines funcions estan disponibles perquè d'altres agents coneguin les dades que necessita o vol compartir.

- De la mateixa manera que un objecte és la representació en temps d'execució d'una definició de codi o *classe*, un **agent és una instància d'una classe**, podent ser creat o destruït en temps d'execució: JADE proporciona una classe de definició, **jade.core.Agent**, que defineix els atributs i mètodes bàsics dels quals es crearà la definició dels agents a implementar pel projecte:

```
public class Jane extends Agent
{
      //codi definició de la classe agent
}
```

Amb el tractament d'objecte donat per *JADE*, també és possible la creació de *classes abstractes* o *definició de tipus - interfaces* - com una definició de tipus d'agents i programar agents que es basin en aquests tipus, així com programar polimorfisme: Les classes que defineixen les característiques d'un agent es poden organitzar jeràrquicament; classes més específiques hereten els atributs i mètodes de la classe superior i/o creen els seus propis mètodes per substituir la definició dels de la súper classe - *JADE* proporciona una definició abstracta d'una classe agent que serveix com a tipus i definició general per cada agent que es vulgui crear -:

```
public interface IAgent
{
	public void registrarAgentDF() throws FIPAException;
	public void crearRegistreLog() throws IOException;
}
public class InfoHabitant extends Agent implements IAgent
{
	public void registrarAgentDF() throws FIPAException
	{
		//codi del mètode
	}
	public void crearRegistreLog() throws IOException
	{
		//codi del mètode
	}
}
```

- No existeix una relació directa entre agents que permeti invocar des d'un els mètodes de l'altre. Tampoc existeix una delegació de responsabilitats directe, ja que cada agent és programat amb les seves pròpies funcionalitats i no es pot accedir al comportament d'un altre. La **relació** i la **delegació** de la programació orientada a objectes adquireix una definició distinta amb la orientació a agents. Cal implementar una comunicació entre agents perquè es passin ordres i informació d'un a l'altre, amb una localització prèvia de l'adreça de cadascun; no serveix una invocació directa.

- El comportament d'un agent està compost per un o varis algorismes complets, i no només un mètode que accedeix a un atributs. Un comportament pot ser interpretat com un estat al qual s'hi ha arribat amb el compliment o no d'alguna condició i que deixa l'agent en un estat diferent amb què es trobava amb anterioritat.

JADE proporciona un altra classe de definició general, **jade.core.behaviour.Behaviour**. que es comporta com un fil d'execució. Heretant els mètodes i atributs de la definició, es crearan els comportaments de cada agent, els quals responen a comportament que només s'executen una vegada, **SingleBehaviour**; repetides vegades, **CyclicBehaviour**; o com a combinacions de varis comportaments que formen part d'un comportament compost, **CompositeBehaviour**:

```
public class JaneBehaviour extends Behaviour
{
	...
	public void action()
	{
		//algorisme del comportament
	}
	public boolean done()
	{
		//condició de fi de comportament
	}
	...
}

public class Jane extends Agent
{
	...
	public void setup()
	{
		addBehaviour(new JaneBehaviour());
	}
	...
}
```

Una de les avantatges d'utilitzar aquesta *POA*, amb JADE, és la possibilitat d'implementar una programació concurrent i/o paral·lela sense la necessitat de programar les funcionalitats de control de fils d'execució manualment i tota la dificultat del seu que els seu ús comporta. Els objectes *Behaviour* simplifiquen aquesta programació, amb mètodes i atributs ideats per programar només les tasques que ha d'executar un agent; el control del fil d'execució es realitza internament.

Vistes aquestes dues últimes característiques, a continuació s'entra en més detall en el disseny de les entitats software que han d'executar la funcionalitat del servidor.

4.5.3. Creació dels agents

El disseny dels agents es durà a terme amb l'ajut de diagrames d'estat. En ells es distingiran les condicions que porten d'un estat d'execució a un altre i serviràn per encarar millor la programació final de cada agent software.

1. Agent consultor: Gestor

Aquest agent segueix el concepte d'arquitectura reactiva: està pendent de rebre algun estímul, en format de missatge, per executar el seu comportament. Aquest, és un comportament bàsic que s'executa cíclicament, del qual se'n distingeixen pocs estats. Realitza simples accessos a la base de dades, però si gestiona la lògica de connexió al SGBD que sense la qual els infohabitants recomanadors no tindrien accés a la informació emmagatzemada.

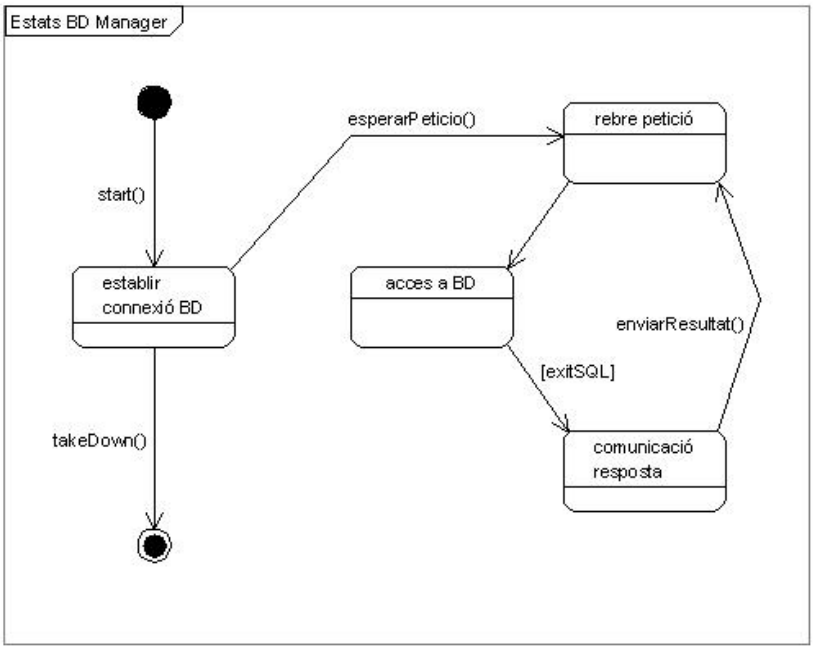


Figura 4.14.: Estats i condicions de l'agent per tractar una gestió de dades.

Un cop nascut, l'agent sempre estarà a l'espera d'una petició de consulta o de gestió de dades, realitzarà l'accés corresponent i crearà un missatge de confirmació.

S'implementa la gestió per afegir noves dades i esborrar-les per unes taules ja creades i amb mètodes específics per el tractament de les dades. Per tractar les dades amb la funcionalitat completa referent a la gestió de dades, hagués calgut un estudi amb més profunditat i amb unes especificacions de format de les dades ben definides.

2. Agent connector: Carter

Al igual que l'anterior, de manera cíclica amb un *CyclicBehaviour* que proporciona l'eina JADE, s'aconsegueix un disseny vàlid per el comportament de l'agent.

També segueix el concepte d'arquitectura reactiva, però també actua segons el seu comportament, ja que estarà programat per executar-se en llegir una petició per la xarxa.

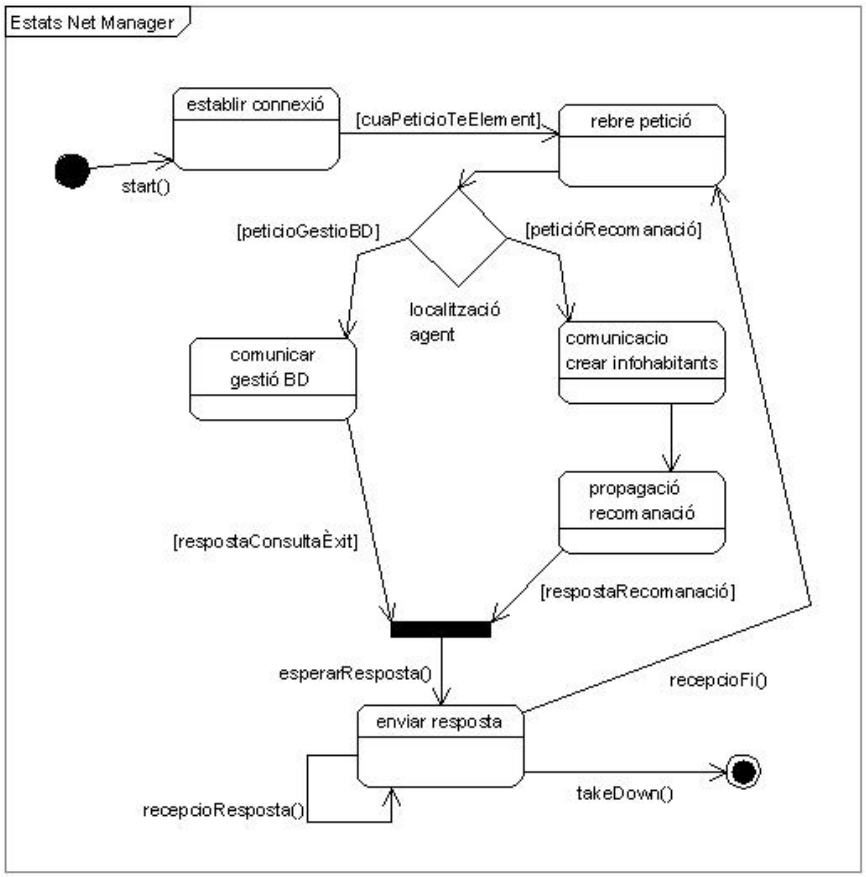


Figura 4.15.: Estats i condicions que segueix l'agent per canalitzar el flux d'informació de la lògica del servidor cap al client i al revés.

L'agent està sempre actiu un cop ha establert un canal de comunicació a través del qual s'hi han de connectar els clients.

Depenent del tipus de missatge petició que rebi, es comunica amb un agent o un altre informant de les dades rebudes a través d'aquest missatge. A continuació, queda a l'espera la resposta d'aquests un cop hagin finalitzat el seu procés. Es torna a repetir el pas per l'estat

d'enviar resposta perquè els infohabitants recomanadors contínuament estan realitzant càlculs i obtenint dades que han de ser disponibles per l'usuari final.

No s'arribarà a l'estat final a no ser que es produeixi algun error en temps d'execució.

3. Agent fabricador: Fàbrica

Es crea un agent intermediari entre l'agent connector i l'infohabitant, que tingui la responsabilitat de crear tants infohabitants com l'usuari final per cada experiment de recomanació.

Aquest disposa de dos comportaments cíclics: un que s'encarrega de crear agents recomanadors i l'altre per propagar a cadascun dels recomanadors la mateixa petició de recomanació, i proporcionar les dades d'usuaris a recomanar. Reacciona, també, segons els missatges que rebi: segons el tipus de missatge executarà el comportament de creació o de distribució.

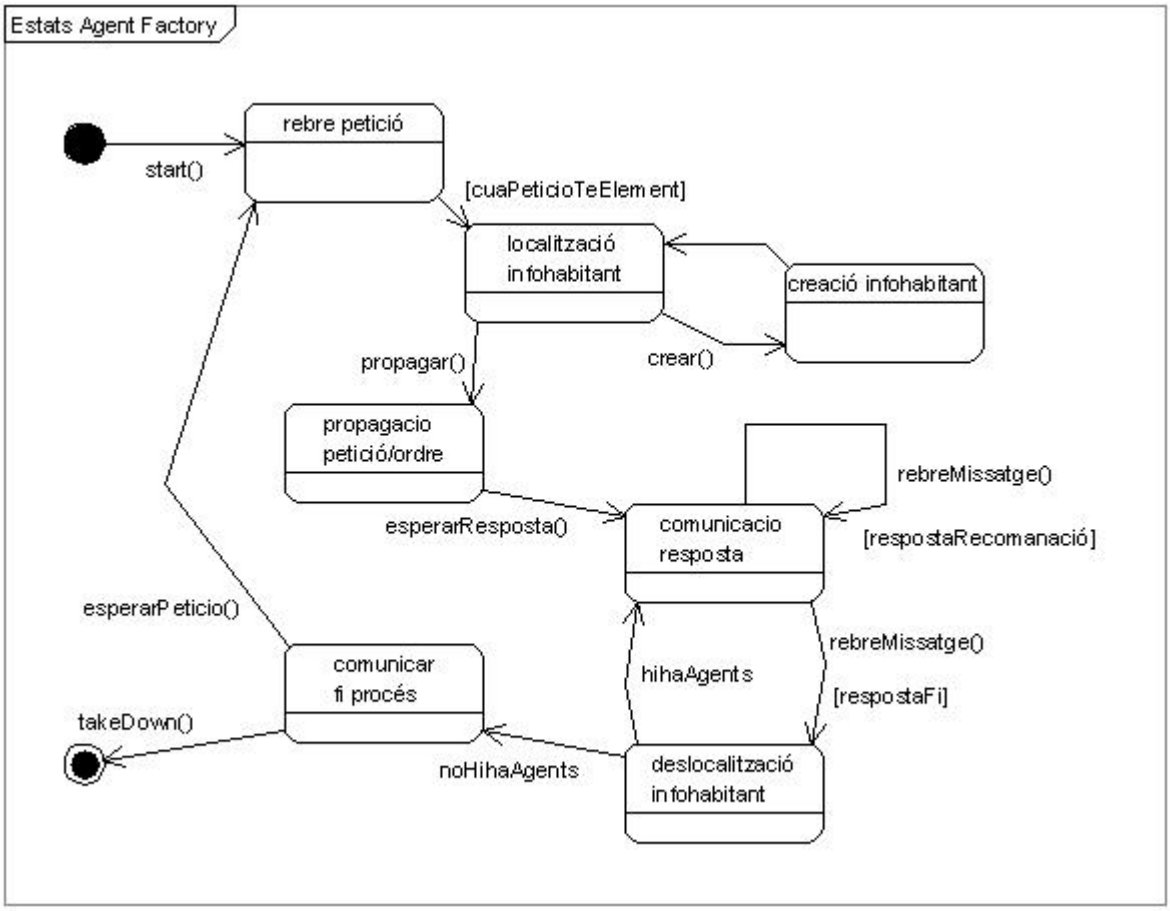


Figura 4.16.: Estats i condicions que segueix la lògica de creació i propagació de peticions.

Els estats *localització infohabitant* i *creació infohabitant* corresponen a un primer comportament que serà el primer a executar-se en rebre una petició de recomanació: sempre es realitza una localització prèvia per comprovar que no hi ha cap infohabitant en actiu.

Els estats de *propagació ordre*, *comunicació resposta* i *deslocalització infohabitant*, al segon comportament. Aquest darrer comportament, no només controla el flux d'entrada de dades d'operació al agents recomanador, si no que rep la informació creada per aquests i la passa a l'agent connector perquè sigui transferida a la part client.

Tampoc s'ha d'arribar a l'estat final, mentre el codi servidor romangui actiu.

4. Agent recomanador: Infohabitant

Donats els dos tipus d'interacció entre infohabitant i usuari que s'estudien en el projecte, *associació* i *dissociació*, l'infohabitant segueix dos cicles de vida diferents degut a que l'associació amb un usuari no compleix el mateix patró. El disseny d'aquest agent està realitzat com a primera solució per l'estudi de recomanacions en massa.

L'arquitectura d'aquest agent segueix el concepte de l'arquitectura BDI. Les creences, serien les dades a consultar de l'històric de transaccions de compra, així com les dades que aprèn el mateix agent de cada recomanació feta a un usuari, *BELIEVES*. Els desitjos seguirien la funcionalitat de executar els càlculs per mesurar les recomanacions i proporcionar els resultats de cada cicle, *DESIRES*. Amb els diagrames següents es descriuen les intencions, que no serien altra cosa que els processos que ha d'executar l'agent per assolir els objectius, segons el que sap, *INTENTIONS*.

Per una recomanació associada, en la qual l'agent recomana al mateix usuari per tots els cicles de tot el procés:

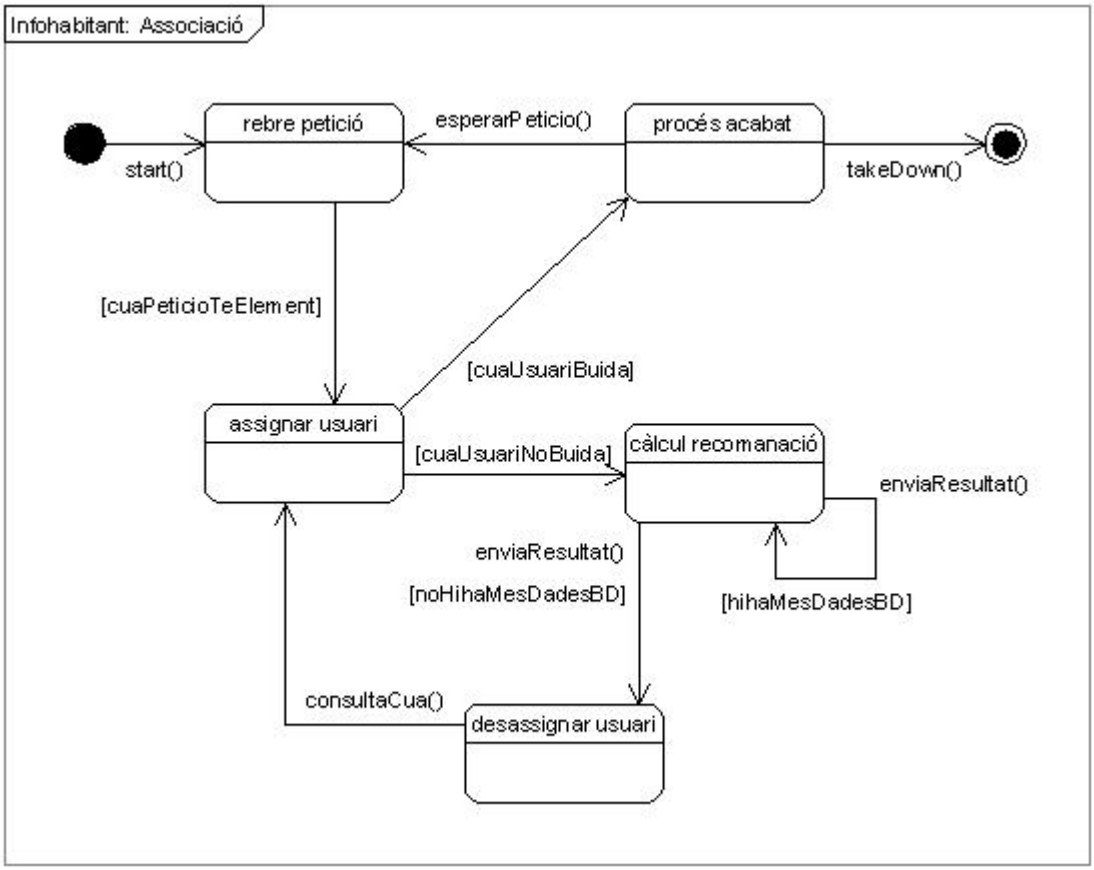


Figura 4.17.: Estats i condicions que segueix l'agent per la recomanació per associació.

Per aquest agent, cada estat representa un comportament simple, és a dir, representa un algorisme diferent, amb lògica i funcionalitat pròpia. Per la implementació es fa ús del concepte de màquina d'estats, definit i implementat per la classe de comportament compost, **FSMBehaviour**, que també proporciona *JADE* (més detalls en l'apartat 4.5.3.).

Al tractar-se del mateix agent, la definició d'estats és la mateixa per tots dos casos. Canvia la ruta a seguir d'un cas a un altra, per tal de complir l'especificació de cada tipus de recomanació.

Per una recomanació dissociada, en la qual l'agent recomana a un usuari diferent per cada cicle del procés:

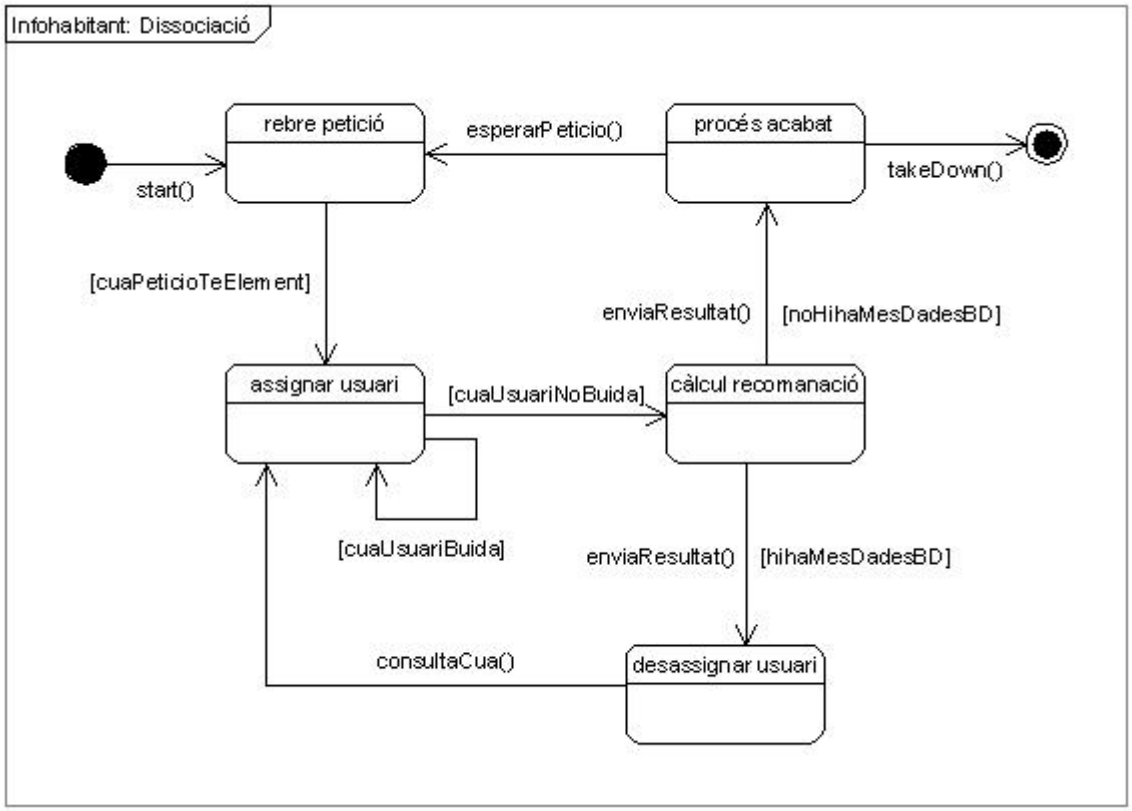


Figura 4.18.: Estats i condicions que segueix l'agent per una recomanació dissociada.

Els estats inicial i final (representats per els cercles negres) indiquen el naixement i mort de l'infohabitant - com que aquests agents poden romandre vius durant un temps indefinit pot ser que no s'hi arribi mai a l'estat final -.

En associació, s'assigna un usuari a un infohabitant si la cua d'usuaris conté algun que encara no ha estat recomanat. El procés de recomanació o càlcul s'executa mentre hi ha dades per consultar en l'històric emmagatzemant al SGBD. Un cop s'ha acabat el procés, l'assignació es trenca i l'infohabitant torna a accedir a la cua per assignar-se un nou usuari. Si la cua està buida, volent dir que tots els usuaris han estat recomanats, l'agent ha acabat la seva execució i roman a l'espera de noves peticions.

En dissociació, en canvi, mentre la cua està buida l'infohabitant espera a que hi hagi un usuari disponible a recomanar. Executa el procés de recomanació per un únic usuari i trenca l'assignació amb aquest, deixant-lo de nou a la cua. Si encara hi ha dades per consultar torna a executar un càlcul per un nou cicle. En cas d'arribar al final de l'històric de dades del SGBD, l'agent finalitza la seva execució i roman a l'espera de noves peticions.

4.5.4. Comunicació interagent

Una de les característiques de la *POA* és la de programar missatges per la comunicació entre agents, necessària perquè aquests es comuniquin i estableixin una col·laboració de responsabilitats per gestionar la funcionalitat del nucli del laboratori.

Amb la utilització del llenguatge *ACL* es defineixen uns patrons i/o regles d'estructuració d'informació per crear una semàntica i vocabulari entès per tots els agents implicats en la gestió de l'habitat dels agents i per l'execució dels experiments de recomanació. Per la interacció dels agents és necessari dissenyar una **ontologia**, o descripció estructurada de conceptes i relacions, que permeti gestionar cada funcionalitat del projecte i poder crear un hàbitat col·laboratiu entre agents, cadascun amb les seves pròpies tasques i responsabilitats, a fi de dur a terme les tasques de gestió de dades i d'execució d'experiments de recomanació.

A continuació es presenta el tipus de missatge que es passen els agents, la descripció de cadascun, i la seqüència que segueixen els missatges per implementar la lògica de les funcionalitats més importants.

1. Comunicació per consulta i/o gestió dades.

Per resoldre les peticions de consulta i importació, la comunicació comença amb el requeriment d'una acció i la resposta serà enviada amb un missatge informatiu: *REQUEST* de la consulta o gestió i *INFORM* amb les dades consultades o missatge d'èxit o error.

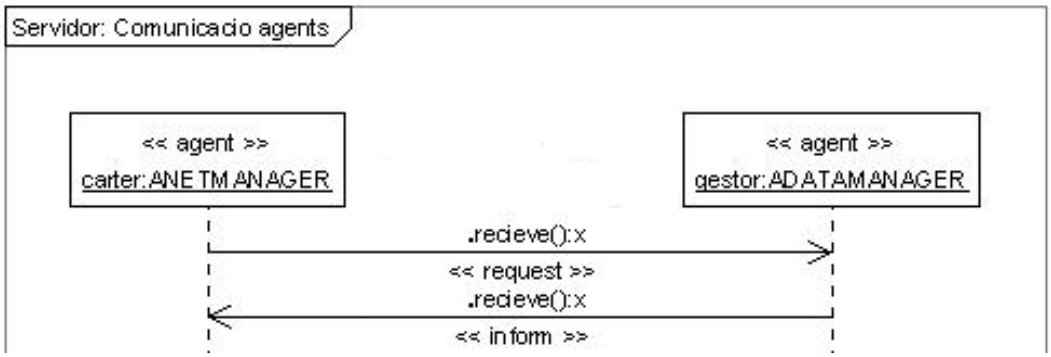


Figura 4.19.: Seqüència de consulta i/o gestió de dades. Cada fletxa indica el nom de l'acció que es dur a terme i el tipus de missatge que es rep amb cada acció.

I el contingut dels missatges per aquesta comunicació és el següent:

```
REQUEST(Ac; Ag; id: gestió; CONSULTA);
INFORM(Ag; Ac; id: gestió; RESPOSTA_CONSULTA);
```

2. Comunicació per recomanació.

Per les peticions de recomanació es dissenya una ontologia separada en dos patrons, un seguit de l'altre.

El primer patró es fa servir per implementar la comunicació per la creació dels *infohabitants*: un missatge de tipus **PROPOSE** de l'agent connector, **Ac,** a l'agent fabricador, **Af,** amb un identificador del missatge indicatiu de la funcionalitat a realitzar i les dades d'operació, **CONFIG_R**, com a contingut del missatge - d'aquestes només es tindrà en compte el valor de la quantitat d'agents recomanadors a crear -. **USER_LIST** fa referència al total d'usuaris per els quals es realitzaran els experiments de recomanació:

```
PROPOSE(Ac; Af; id: creació; CONFIG_R/USER_LIST);
      QUERY_IF(Af; Ag; id: SGBD; CONNECTAT_SGBD);
      INFORM_IF(Ag; Af; id: SGBD; NOM_SGBD);
ACCEPT_PROPOSAL(Af; Ac; id: creació);
```

El missatge **QUERY_IF** és creat per preguntar a l'agent gestor, **Ag**, amb quin SGBD es guarden les dades d'informació. Amb la comprovació de la connexió amb l'SGBD, el nom d'aquest és passat com a contingut del missatge **INFORM_IF**.

El nom del SGBD és usat per cada infohabitant per tal d'establir connexió al mateix i poder realitzar les consultes necessàries per els experiments.

Un cop creats els infohabitants, l'agent fàbrica ho notifica a l'agent connector, amb **ACCEPT_PROPOSAL**, i es prepara per rebre un nou missatge, com a ordre de recomanació.

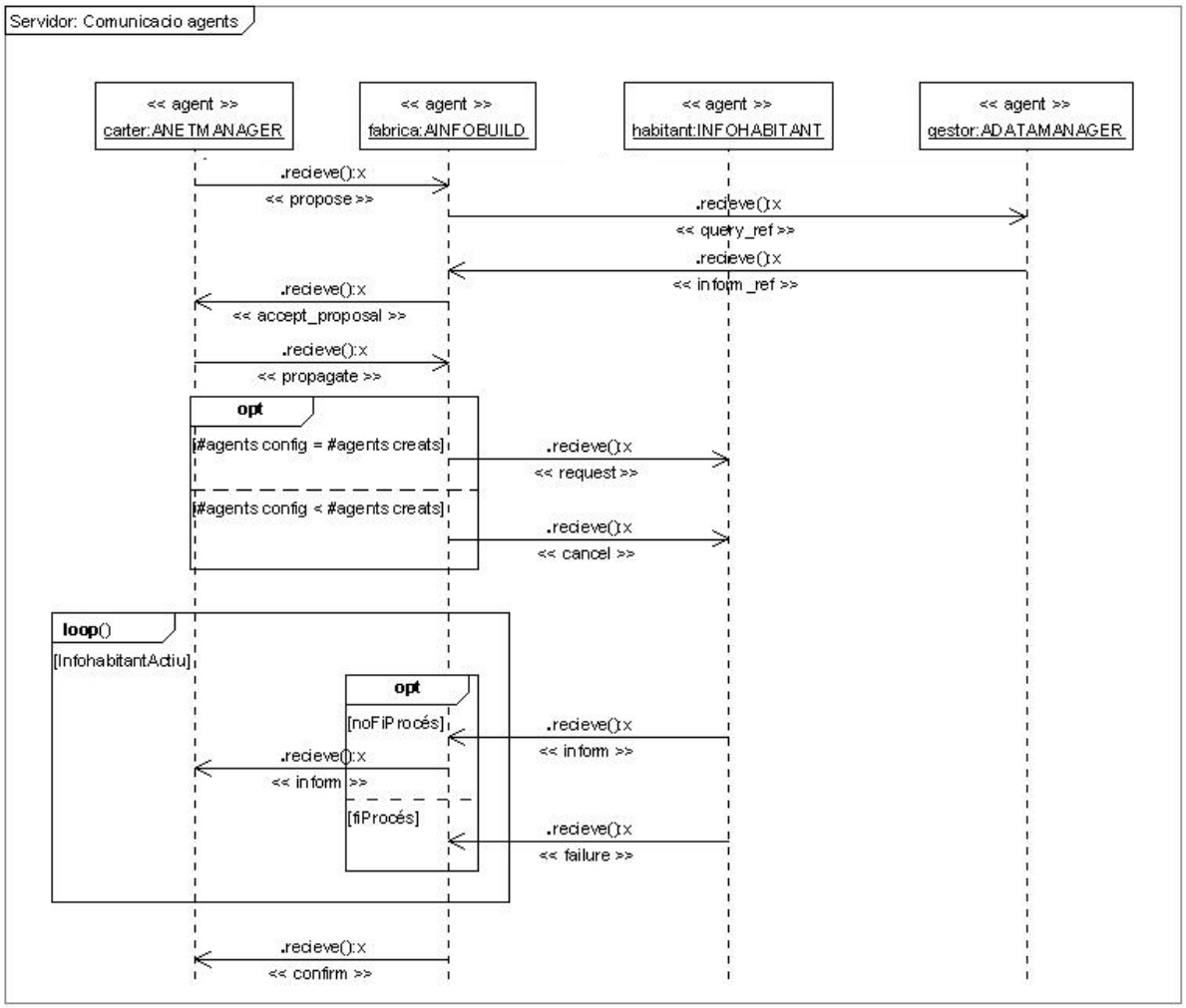


Figura 4.20.: Seqüència de recomanació. Cada fletxa indica el nom de l'acció que es dur a terme i el tipus de missatge que es rep amb cada acció.

Cada infohabitant creat roman a l'espera d'una petició de recomanació. Un cop creats, l'agent connector enviarà un missatge **PROPAGATE** a l'agent fàbrica perquè aquest distribueixi les peticions de recomanació als *n* infohabitants creats, amb missatges de tipus **REQUEST**, i contenint les dades d'operació, *CONFIG_R*, referents a la funcionalitat d'un procés de recomanació.

Com que per un experiment es poden sol·licitar un nombre d'agents recomanadors menor al nombre dels mateixos agents vius en l'habitat, s'envia un missatge **CANCEL** a aquells que no participin en un experiment.

```
PROPAGATE(Ac; Af; id: distribució; CONFIG_R);
REQUEST(Af; H1..Hn; id: recomanació; CONFIG_R);
      CANCEL(Af; Hi..Hj; id: recomanació);
```

```
INFORM(H1..Hn; Af; id: recomanació; ESTAT_R);
      INFORM(Af; Ac; id: recomanació; ESTAT_R, send: Hx);
FAILURE(H1..Hn; Af, id: recomanació; FI_R)
CONFIRM(Af; Ac, id: recomanació; FI_R);
```

Cada *infohabitant* enviarà un missatge **INFORM** a l'agent fàbrica indicant les dades i els càlculs de cada cicle del procés de recomanació, **ESTAT_R** (*Esquema 4.3.*). Cada resposta és enviada al agent connector, mitjançant un nou missatge *INFORM* amb les dades del cicle de recomanació i el nom de l'infohabitant que enviar la resposta.

Un cop un infohabitant finalitza un procés de recomanació, envia un missatge **FAILURE** amb una trama XML que indica que s'ha arribat al final del procés. Quan tots els infohabitants han enviat el seu missatge de fi, l'agent fabrica comunica una confirmació de fi de l'experiment, **CONFIRM**.

Les dades d'operació, *CONFIG_R* i les dades d'un cicle de recomanació, *ESTAT_R*, estan escrites en format XML seguint el disseny dels missatges entre client i servidor indicats en l'apartat anterior.

4.6. DISSENY DE MÒDULS: SERVIDOR

Per la implementació dels agents software també es realitzarà un disseny de classes i objectes, que permetran identificar els atributs i mètodes que definiran els comportaments de cada agent. Aquest disseny es farà des d'un punt de vista orientat a objectes, per tal d'identificar els mètodes i crides a aquests mètodes, com a base de la POA.

S'identifiquen els objectes agent amb l'estereotip **<<agent>>** i els objectes de comportament amb l'estereotip de **<<behaviour>>**. Amb la conjunció dels objectes comportament i objectes agent es mostraran els mètodes i atributs per implementar els estats amb què s'han definit cada agent.

4.6.1. Comunicació Servidor/Client

En aquest apartat es defineix la implementació dels mòduls o casos d'ús *llegir/escriure missatges* i *gestió ordres/respostes.*

1. Col·laboració entre objectes.

L'agent *carter* ve definit per l'objecte **ANETMANAGER**. Aquest s'ajuda d'un objecte de control, **CTRLCONNEXIO** per tal de crear un pont de comunicació o socket per llegir i/o escriure missatges a la xarxa.

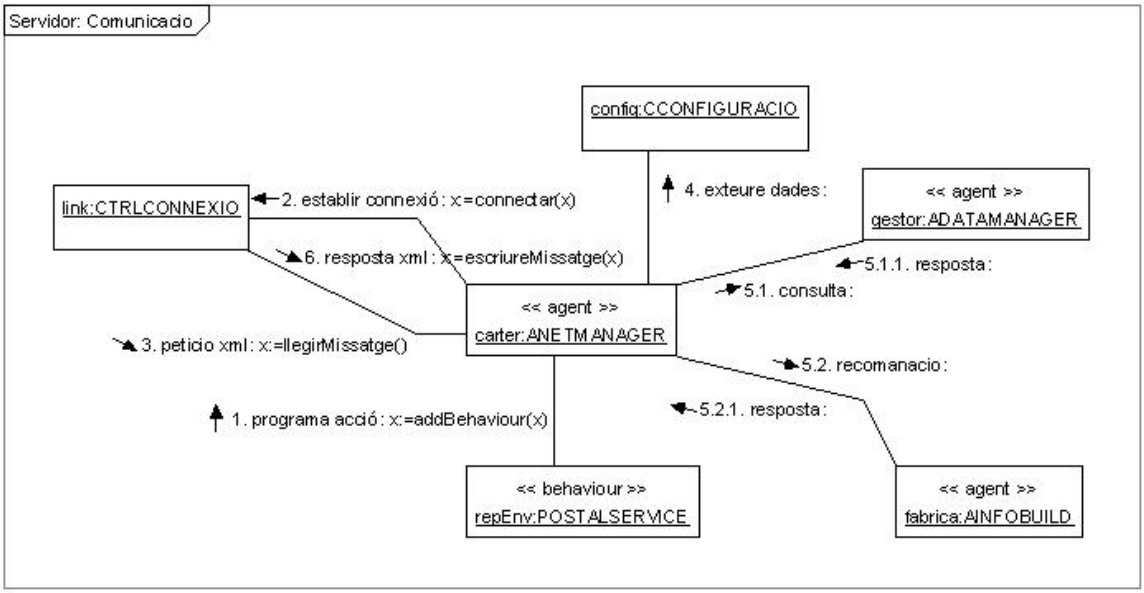


Figura 4.21.: Diagrama de col·laboració: lectura de peticions i escriptura de respostes, a xarxa.

Depenent de quin tipus de petició es rep, l'agent *carter* distribuirà la petició a l'agent *fàbrica* o a l'agent *gestor*.

Tot i que la col·laboració és entre els objectes <<*agent*>>, la implementació de les crides sempre s'executaran des de la classe de comportament, en aquest cas **POSTALSERVICE**. En els diagrames de seqüència es veu el disseny final de les crides de cada mètode.

2. Seqüència de crides a mètodes.

La primera crida que ha de fer un agent és la de crear i assignar-se els comportaments definits - en la *Figura 4.22.* veure la crida al mètode *addBehaviour*, com a creació de l'objecte - per tal d'activar les seves funcionalitats.

Amb el següent diagrama es descriu la distribució de peticions cap als objectes agent pertinents. La seqüència de escriptura de respostes s'indicaria des del *gestor* o *fàbrica* cap al *carter*.

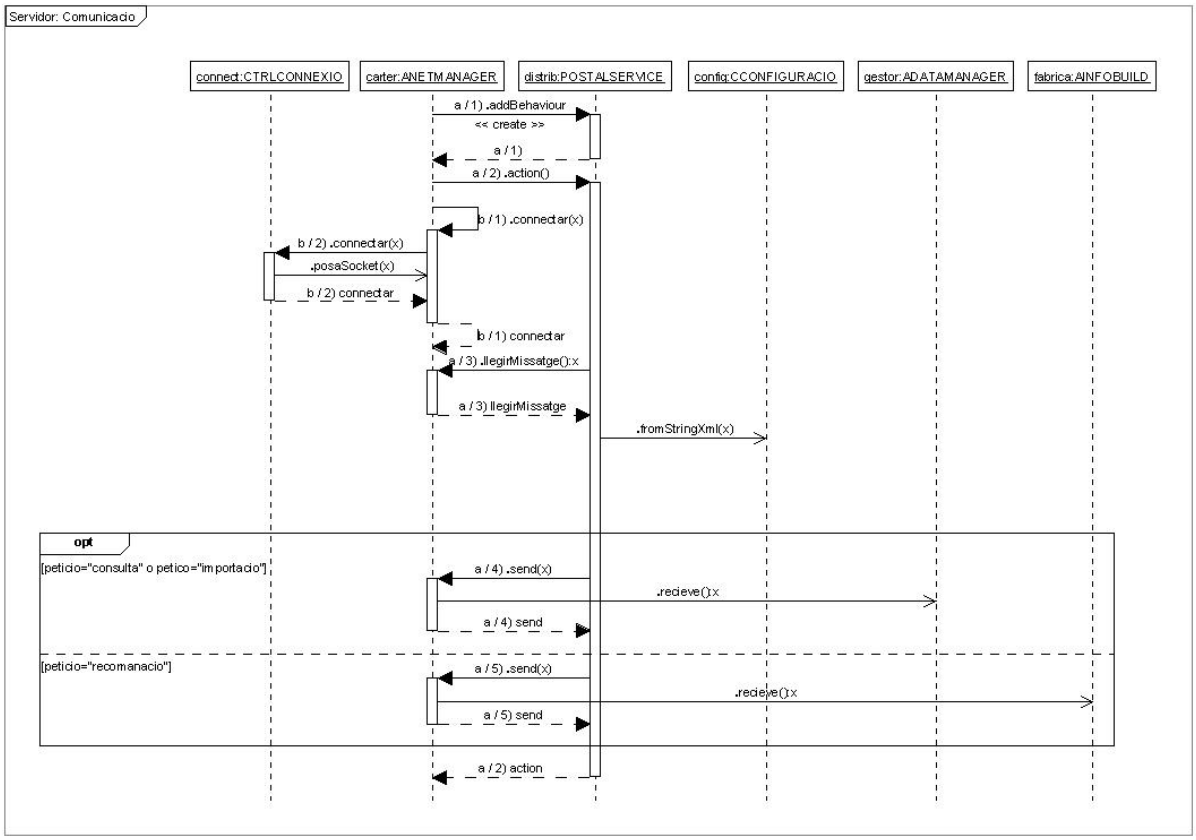


Figura 4.22. Seqüència: lectura i distribució de peticions.

L'execució del comportament començarà amb la creació del canal de comunicació a través del qual es llegiran i s'escriuran missatges. Com que no és un objectiu important del projecte, l'habitat o servidor no treballa amb diverses connexions en paral·lel si no que cada petició és rebuda en acabar el tractament d'una petició anterior.

Es controlarà la seqüència del tractament de petició/resposta amb un atribut de la classe **POSTALSERVICE** (veure *Figura 4.23*.). Es llegeix una petició del *socket,* que té assignat l'agent connector, i segons el tipus de petició, aquest agent la comunica a l'agent corresponent - crides 4 i 5 segons és una petició de consulta o importació o una petició de recomanació -.

3. Definició de classes.

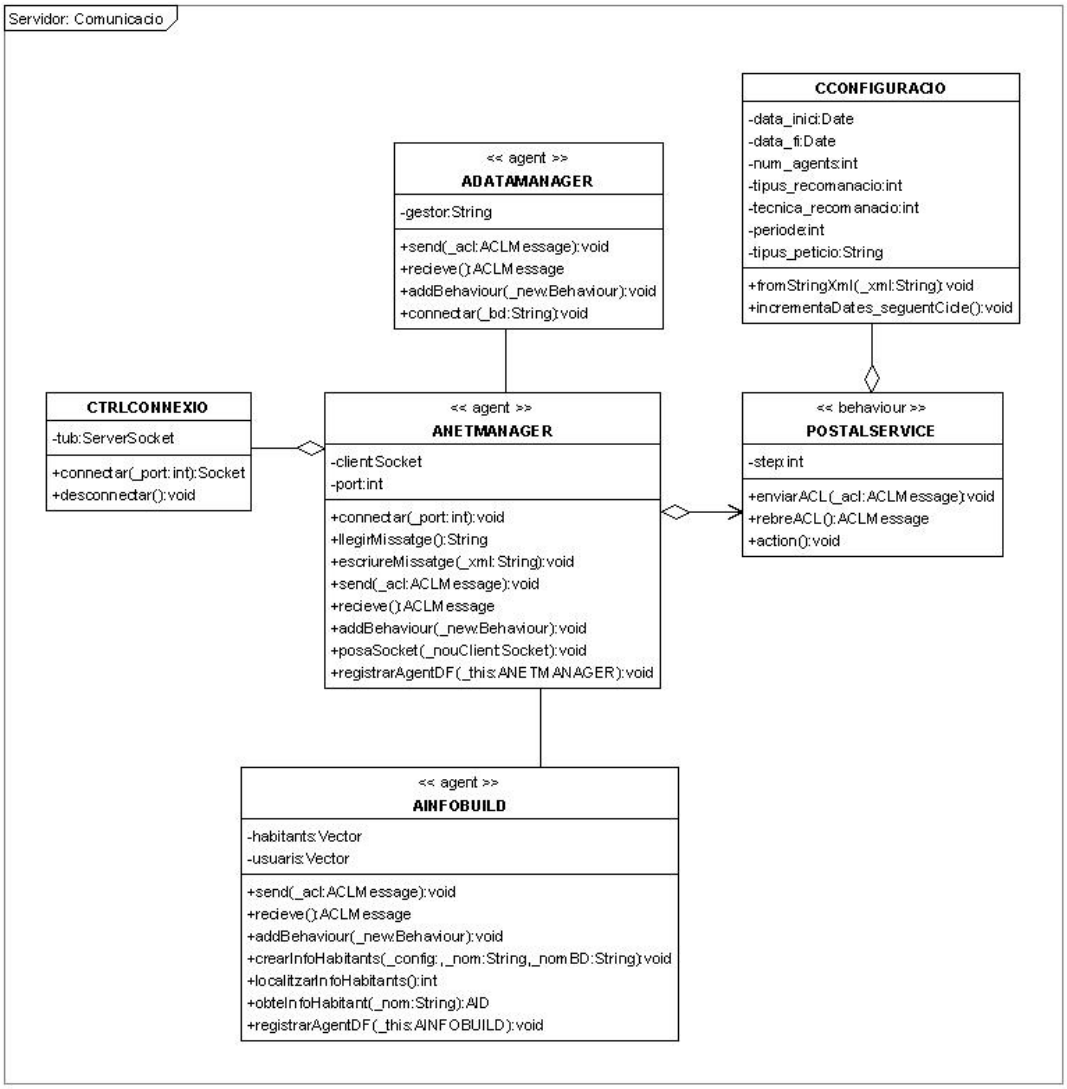


Figura 4.23. Classes que defineixen els mètodes i atributs de cada agent.

Intervindrà una classe **CCONFIGURACIO** per recollir les dades d'operació que es llegeixen d'una petició. L'agent connector presenta atributs i mètodes per gestionar el flux d'informació que ve de i va cap al client: els mètodes seran cridats dintre del mètode *action()* de la classe comportament *POSTALSERVICE*. Aquesta classe serà una extensió de la classe *CyclicBehaviour* de JADE; quan aquest agent hagi escrit totes les respostes d'un procés de recomanació, el comportament estarà en disposició de tornar a llegir una nova petició.

En el diagrama també es presenta la definició de la classe de l'agent fàbrica, **AINFOBUILD**. En ella s'hi descriuen mètodes propis d'un agent - segons la definició de *JADE*

- i d'altres mètodes que gestionaran la creació d'habitants, als quals s'hi entrarà en detall en el següent apartat.

També s'hi descriu una petita definició de l'agent consultor, a destacar un atribut per saber el nom del SGBD amb què es treballa i un mètode per establir connexió amb l'SGBD. Aquest agent, ha de ser implementat per tal de poder treballar amb altres SGBD.

4.6.2. Creació d'Infohabitants

Una de les parts importants del present projecte és la creació i localització d'agents recomanadors, que s'encarregaran de dur a terme els experiments de recomanació.

No s'entrarà en detalls d'una gestió general d'aquests. El focus d'atenció d'aquesta primera implementació és la d'analitzar el rendiment dels tipus de recomanació a implementar, per tant, per alguns experiments hi haurà infohabitants que poden no intervenir-hi, depenent de la quantitat d'agents que es sol·licitin crear.

Aquest apartat descriu una funcionalitat del mòdul *gestió infohabitants*.

1. Col·laboració entre objectes.

El primer que cal implementar és la recepció de la petició, en aquest cas petició de recomanació i guardar les dades d'operació corresponents amb *CCONFIGURACIO*.

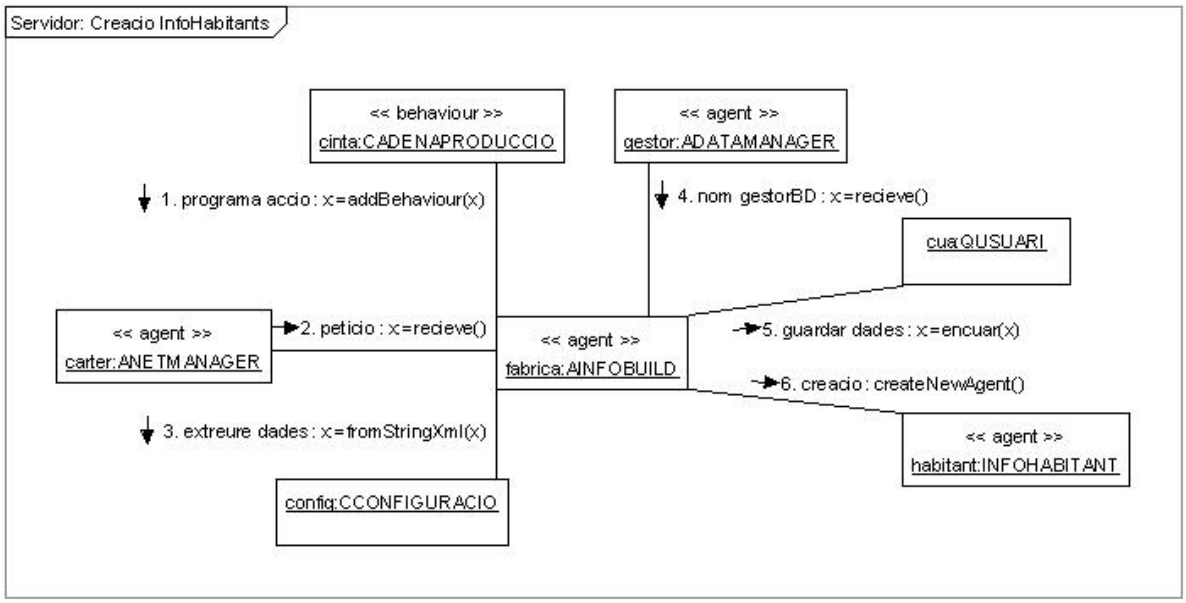


Figura 4.24.: Objectes a fer servir per la creació dels infohabitants.

Com s'ha exposat anteriorment, la petició de recomanació ve acompanyada d'una llista d'usuaris comprador sobre els quals es realitzaran les recomanacions. Es crea un nou objecte, una

cua d'usuaris en el qual s'hi disposaran les dades dels usuaris a qui recomanar. Aquesta estructura es farà servir per la implementació dels tipus de recomanació a estudiar.

L'agent *fàbrica* obtindrà el nom del SGBD de l'agent consultor, informació necessària perquè l'infohabitant consulti les dades escaients per elaborar les recomanacions.

2. Seqüència de crides a mètodes.

La funcionalitat s'executa des de la classe de comportament, però fent servir mètodes de la classe agent.

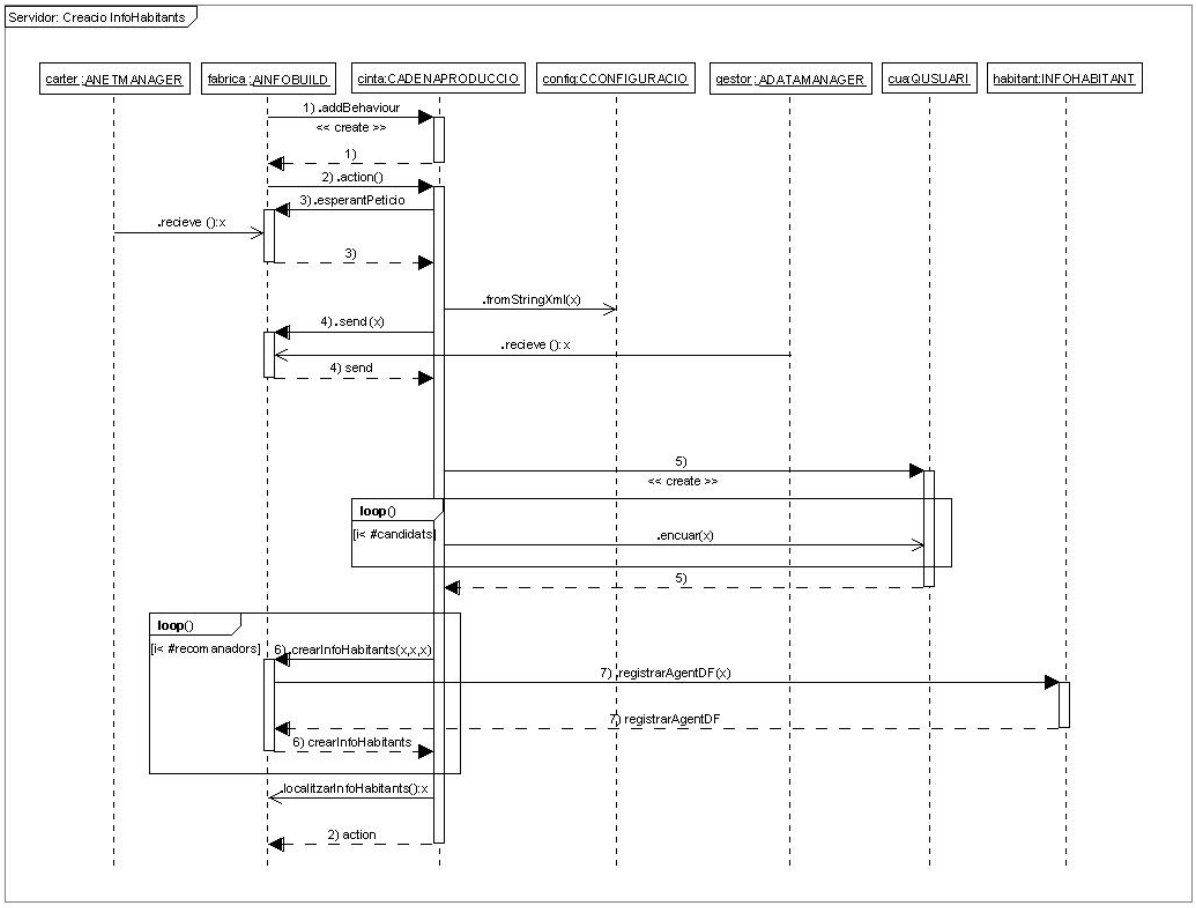


Figura 4.25.: Diagrama de seqüència descriptiu de les crides per preparar un experiment.

S'extreuen dades de la petició usant un objecte *CCONFIGURACIO* i s'envia un missatge a l'agent consultor per obtenir el nom del SGBD - seqüència 4 -. Seguidament es crea l'objecte **QUSUARI** amb les dades a fer servir per l'experiment. Una última seqüència a realitzar és la creació dels infohabitants - seqüència 6 i 7 -. L'agent *fàbrica* haurà d'esperar a que cada infohabitant es registri al *DF* i poder identificar quins s'han creat. Aquesta localització és

necessària per el procés de recomanació, ja que totes les respostes que generin els agents recomanadors passaran per aquest agent fàbrica. S'explicarà en més detall en el següent apartat.

3. Definició de classes.

Aquí també intervindrà la classe *CCONFIGURACIO*: cada agent, o cada comportament d'un agent farà servir aquesta classe per saber les dades d'operació a utilitzar per implementar la funcionalitat que li toca. En aquest cas, es faran servir les dades per la creació dels *infohabitants*.

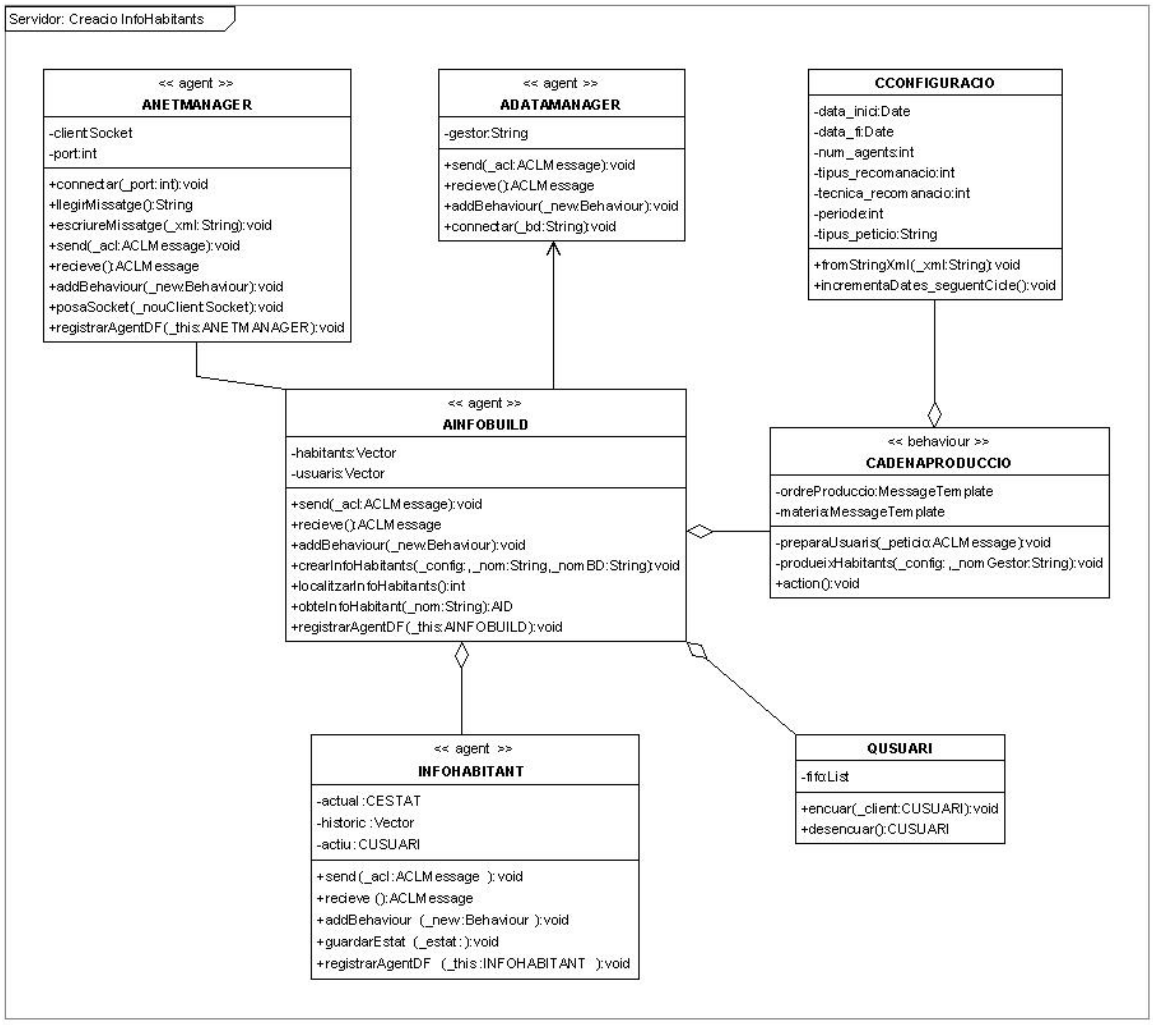


Figura 4.26.: En el diagrama es descriuen els les classes INFOHABITANT i QUSUARI per preparar un procés de recomanació.

L'objecte **CADENAPRODUCCIO** te definits uns patrons de missatge per identificar els que siguin peticions de recomanació i els que sigui respostes dels infohabitants, i és l'objecte comportament que te la responsabilitat de crear els habitants. Aquesta classe també serà una extensió de *CyclicBehaviour*.

El fabricador es guardarà, en una estructura simple, els identificadors de cada *infohabitant*. A mida que aquests vagin acabant els seus processos de recomanació, el fabricador els traurà d'aquesta estructura. D'aquesta manera es podrà notificar a la part client que l'experiment de recomanació ha acabat. També guardarà, en una segona estructura simple, els identificadors dels usuaris comprador utilitzats per realitzar els experiments de recomanació.

4.6.3. Procés de recomanació

1. Col·laboració entre objectes.

Apareix un objecte comportament nou per l'agent fàbrica. **CADENADISTRIBUICIÓ** és un comportament encarregat de enviar les dades de recomanació als infohabitants creats i recollir les respostes que generen per cada cicle del procés de recomanació que executen.

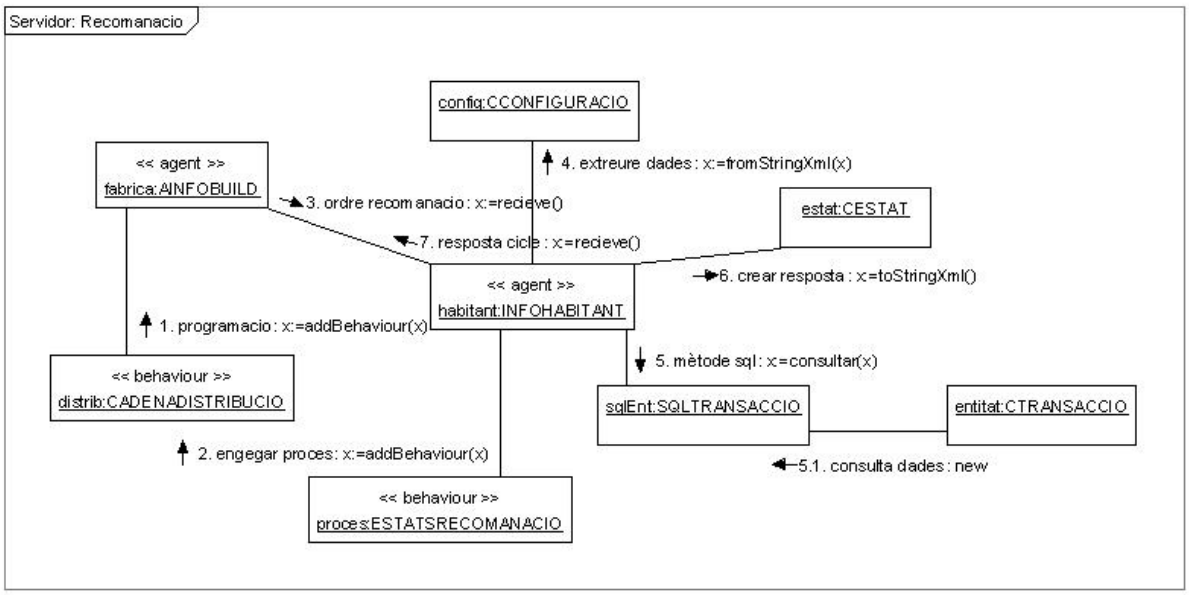


Figura 4.27.: Col·laboració entre objectes. Es mostren les crides més generals per donar una resposta d'un cicle de recomanació.

Com que les dades de tipus i tècnica de recomanació formen part dels paràmetres de configuració o dades d'operació d'un experiment, l'objecte **INFOHABITANT** també col·labora amb un objecte *CCONFIGURACIO* propi.

Les recomanacions es consulten a partir de les dades de les transaccions de compra que hi ha a la BD. L'objecte **SQLTRANSACCIO** es farà servir per consultar aquestes dades i els resultats de les consultes, així com les mesures de cada cicle de recomanació es guardaran en objectes *CESTAT*, que es guardaran en la memòria de cada *infohabitant*.

2. Seqüència de crides a mètodes.

La iteració presentada descriu la consulta de dades per la recomanació i el càlcul de les mesures durant els cicles de recomanació. En la lectura del diagrama se suposarà que l'infohabitant te comprador assignat.

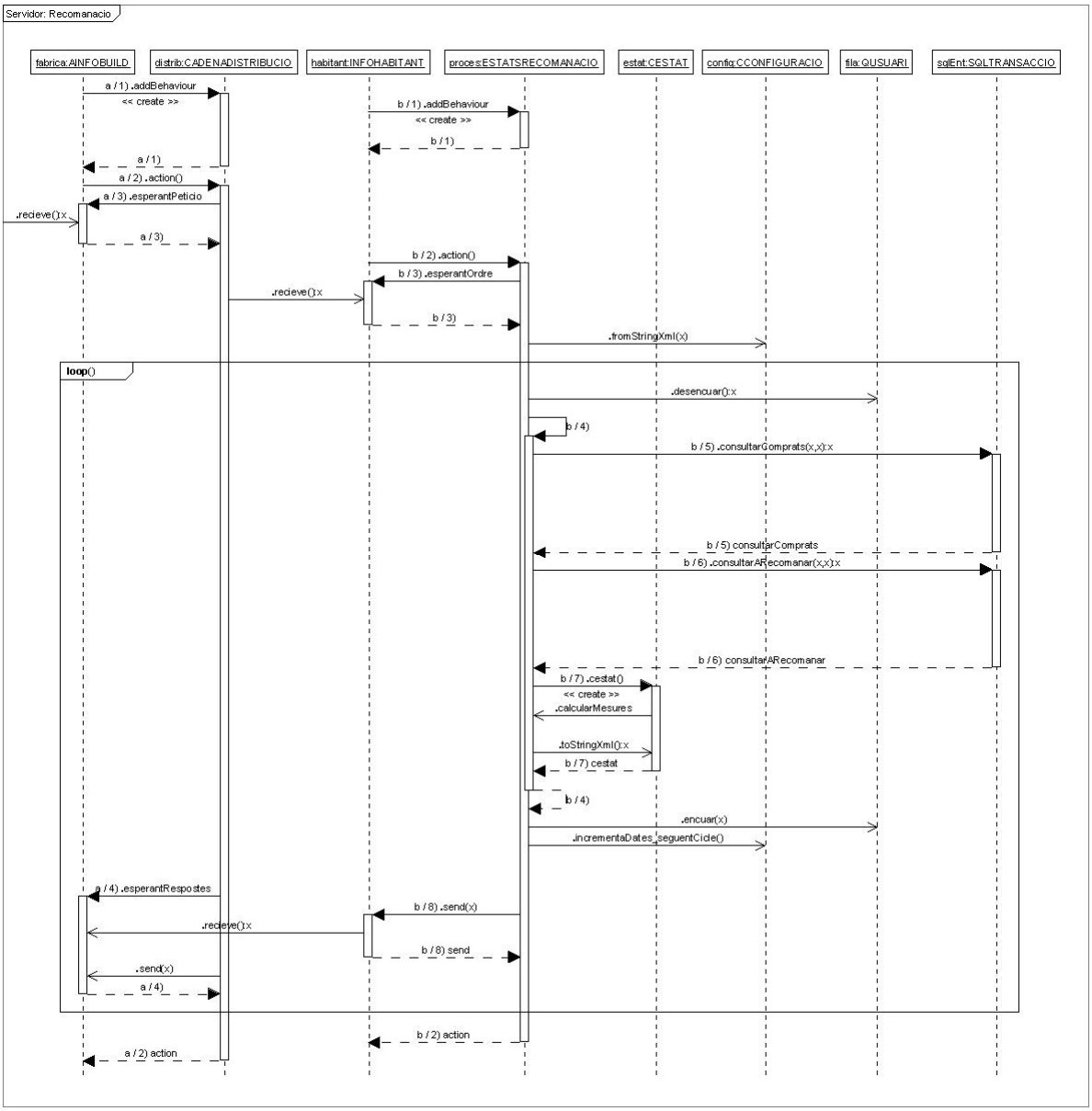


Figura 4.28.: Execució del procés de recomanació per un infohabitant.

Totes les seqüències mostrades dintre de la iteració, *loop*, es repeteixen per cada cicle del procés de recomanació.

Com a primer pas de cada recomanació, l'infohabitant s'assigna un usuari a qui recomanar, *desencuar():x*. A partir de les dades consultades es crea un estat amb aquestes dades i les

mesures de cada recomanació. En finalitzar la consulta i càlculs per la recomanació, l'infohabitant trenca l'assignació amb l'objecte usuari, *encuar(x)*, s'incrementen les dates entre les quals estarà comprés el següent cicle i s'envia la resposta creada amb les dades del cicle anterior.

3. Definició de classes.

CADENADISTRIBUCIÓ necessita saber les dades d'operació referents al número d'infohabitants, per facilitar la tasca de repartiment d'ordres de recomanació i la recollida de respostes que generin aquests.

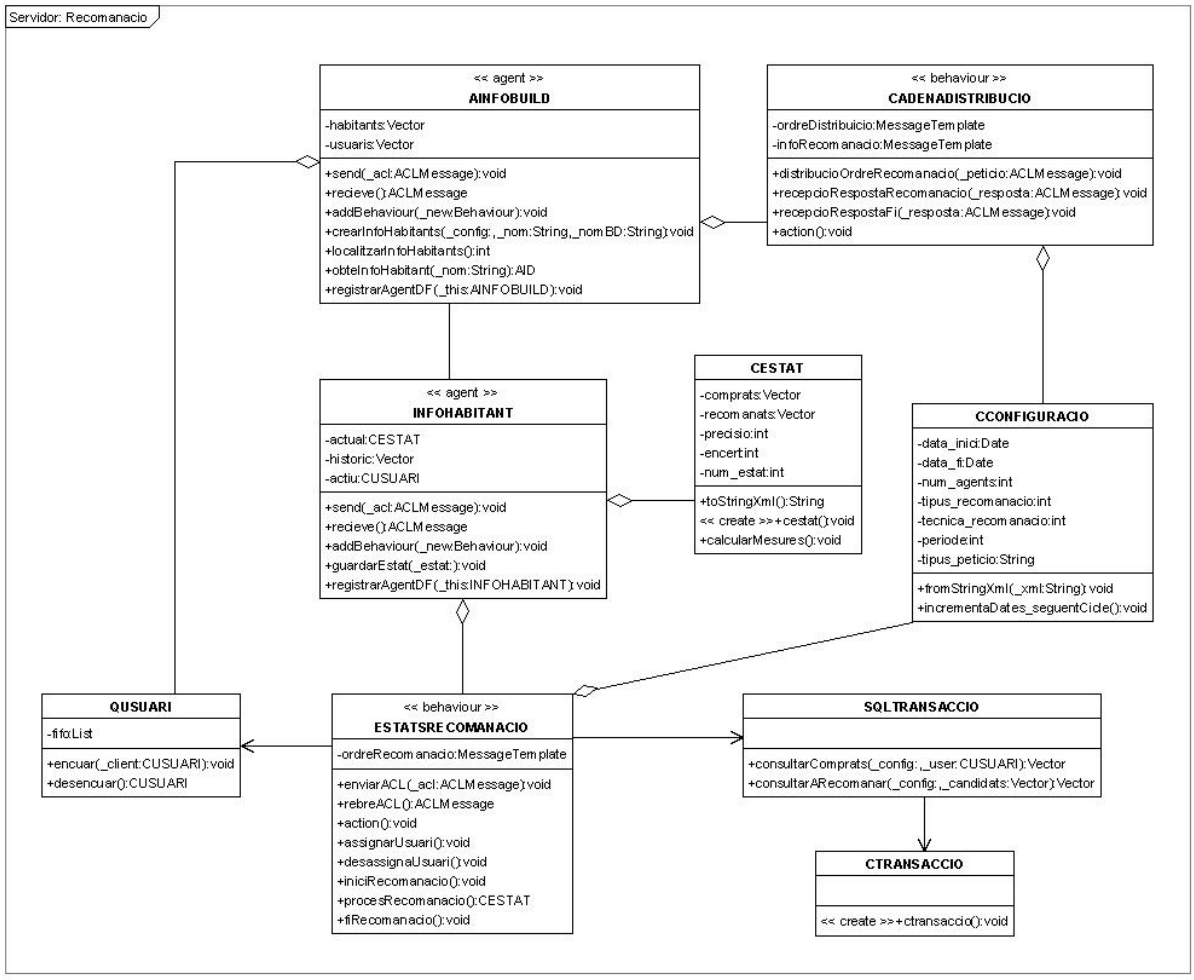


Figura 4.29.: Definició de mètodes i atributs a fer servir per el procés de recomanació.

La classe *INFOHABITANT* coneix la classe *CESTAT*, ja que ha de tractar les seves dades per enviar-les com a resposta de cada cicle de recomanació. A partir del seu comportament, la classe *INFOHABITANT* també coneix les dades d'operació per saber quin tipus i tècniques de recomanació ha de fer servir, i entre quines dates ha d'executar les tècniques.

INFOHABITANT també farà us de la classe *QUSUARI* per gestionar les assignacions i desassignacions de comprador/usuari. Cal recordar que per cada cicle pot ser que l'infohabitant tingui assignat el mateix comprador/usuari o tingui assignat un comprador/usuari diferent; aquesta cua servirà per treure o deixar els compradors de l'experiment.

Com a agent recomanador, l'infohabitant ha d'aprendre dels gustos de compra de cada usuari a qui recomana. Amb l'atribut *històric*, l'infohabitant te representades les seves creences seguint una estructura de vector. En aquests vector només hi guardarà els gustos dels usuaris, però sense tenir-los identificats, vàlid per una interacció associativa o dissociativa amb els usuaris a qui ha de recomanar. Per la darrera interacció, hi haurà un número limitat de productes a memoritzar.

En diagrama presentat a continuació es descriu el comportament **ESTATSRECOMANACIO**. Només per simplificar la lectura del diagrama, es mostren com a mètodes *assignarUsuari(), desassignarUsuari(), iniciRecomanacio(), procesRecomanacio()* i *fiRecomanacio()* els diferents comportaments senzills que formen els estats d'un infohabitant. Amb el diagrama següent es mostra amb més detall els comportaments que regeixen l'execució de l'infohabitant, que han d'implementar els diversos estats d'execució presentats en el disseny de l'agent:

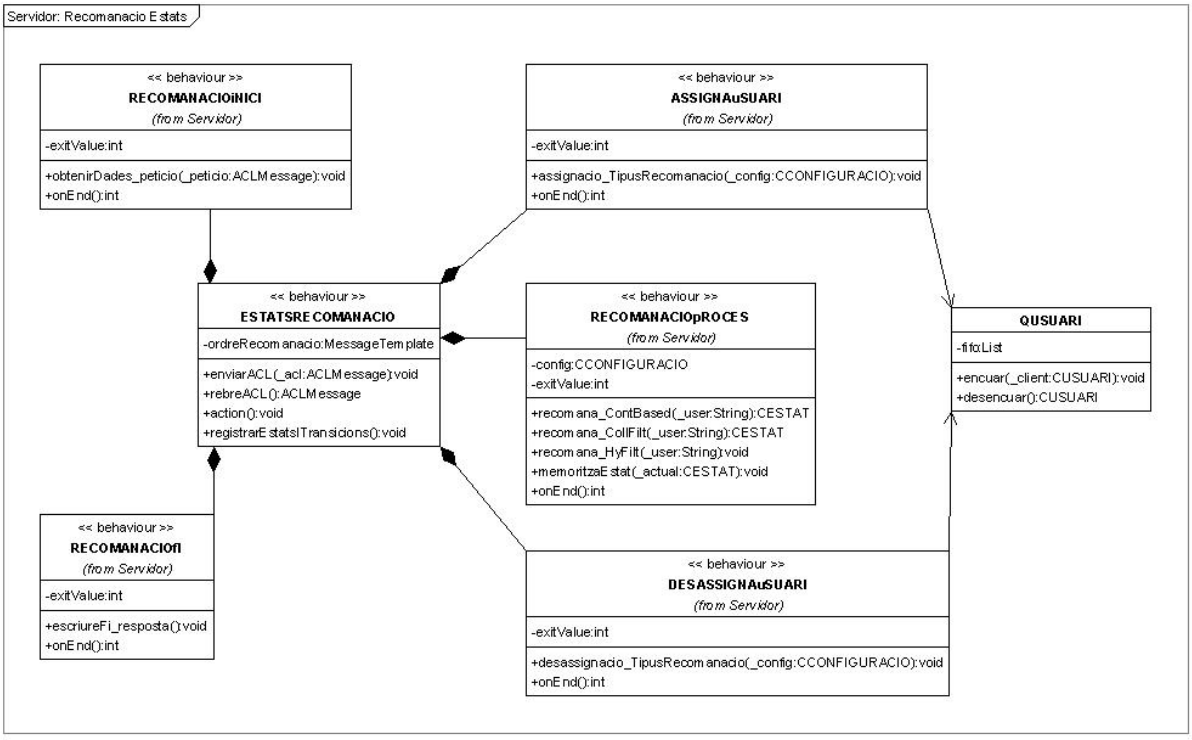


Figura 4.30.: Definició dels objectes comportament per dissenyar la funcionalitat dels estats d'un infohabitant.

Aquest serà el disseny final per la implementació de la funcionalitat de recomanació. Essent *ESTATSRECOMANACIO* una classe de tipus *FSMBehaviour* caldrà registrar els estats del comportament i definir transicions d'estat a estat per controlar l'execució o la traça d'execució de l'algorisme de recomanació. Cada comportament descrit a la figura serà un estat. L'atribut *exitValue* de cada comportament es farà servir per controlar les transicions de comportament a comportament, depenent del seu valor. Els comportaments simples seran del tipus **SimpleBehaviour**.

5. MANUAL D'USUARI

5.1. POSADA EN MARXA

El laboratori necessita que l'hàbitat dels agents estigui en funcionament i connectat a un gestor de dades, per poder executar els experiments de recomanació. Cal:

- Instal·lació prèvia del SGBD de codi lliure *PostgreSQL* en la seva versió 8.2. Es pot descarregar de la pàgina web ***http://www.postgresql.org/download/***
- Instal·lació prèvia de la màquina virtual de JAVA, versió mínima de la *JVM*: J2SE jre1.5.0_07. Es pot descarregar de ***http://java.sun.com/javase/downloads/?intcmp=1281*** - actualment ja està disponible la versió 1.6.0_06 -.

Per altra banda, cal fer córrer l'aplicació gràfica de control i gestió de l'hàbitat en un ordinador diferent d'on s'ha posat en marxa l'hàbitat. Per poder instal·lar l'aplicació gràfica:

- Cal instal·lar prèviament el framework d'execució, *.NET 2.0.*, per el control de codi realitzat en C#. Està disponible en la pàgina de Microsoft ***http://www.microsoft.com/downloads.***

5.2. APLICACIÓ DEL SIMULADOR

S'ha dissenyat una aplicació d'escriptori que contindrà i permetrà la gestió dels diversos formularis o pantalles per fer funcionar el simulador.

Aquest disseny i implementació és una primera proposta per gestionar i controlar l'experimentació amb l'habitat dels agents recomanadors. Tot i que no és una solució definitiva, ja que és un disseny específic per entorns Windows, d'aquí es poden treure idees per futures millores o canvis d'implementació en la interfície gràfica.

5.2.1. Pantalla principal i menús

Tots els formularis de l'aplicació estan continguts en una pantalla principal. A cada formulari s'hi accedirà a partir dels menús de què disposa aquesta pantalla, i que es descriuran a continuació.

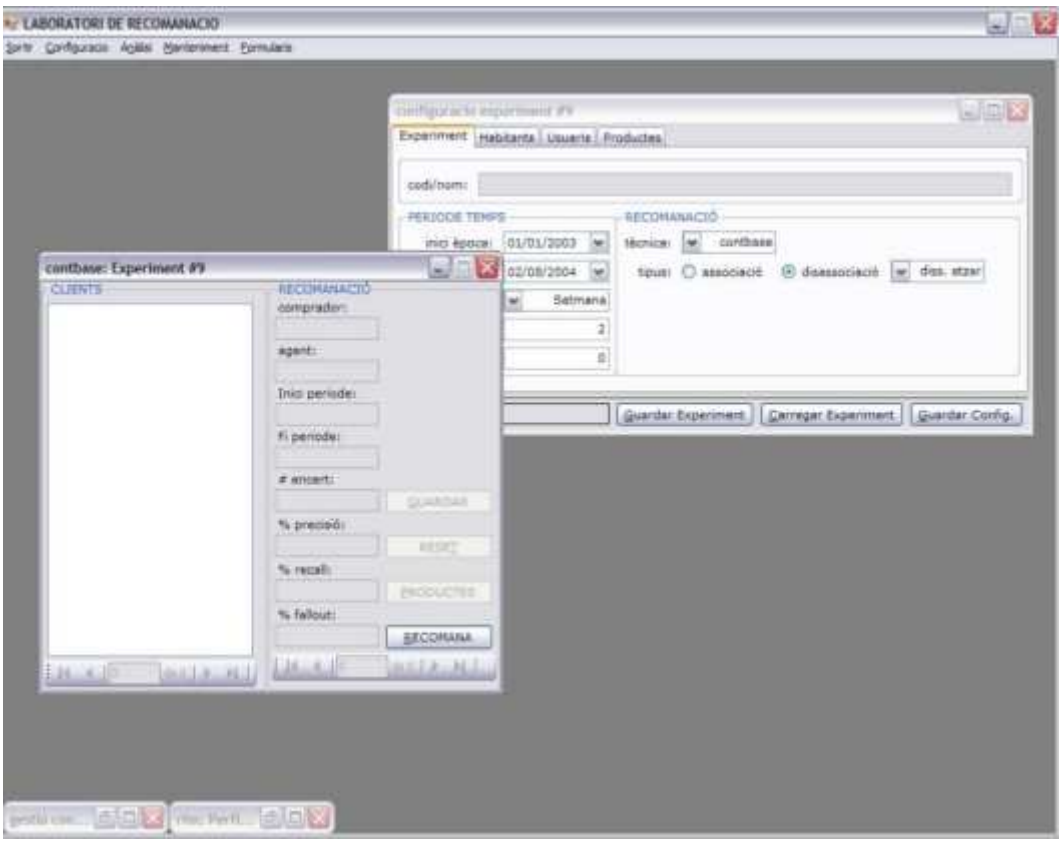


Figura 5.1.: Vista de l'escriptori per la gestió del laboratori.

- *Menú Configuració*

Figura 5.2.: Menús secundaris per la gestió de les dades d'operació.

Per accedir als formularis de gestió dels paràmetres de, *Experiment*, i al formulari de gestió de les dades de connexió, *Connexió*.

- *Menú Anàlisi*

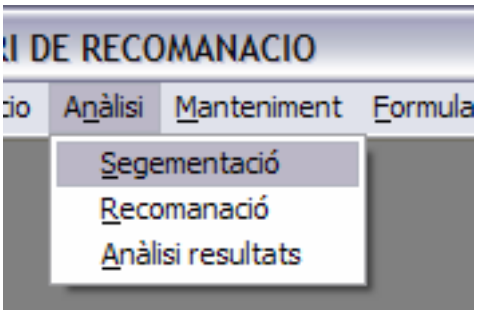


Figura 5.2.: Menús secundaris per la gestió d'experiments i anàlisi de recomanacions.

S'accedeix als formularis per calcular perfils de compra dels compradors, *Segmentació*, formularis de gestió i monitorització dels experiments, *Recomanació*, i formulari d'anàlisi d'experiments, *Anàlisi resultats*.

- *Menú Manteniment*

Figura 5.3.: Menús secundaris per la gestió de dades d'informació.

Obre els formularis per crear les taules del magatzem de dades, *Crear Taules*, a on s'afegiran les dades carregades des de fitxers de text, amb el formulari *Importació Dades*.

- *Menú Formularis*

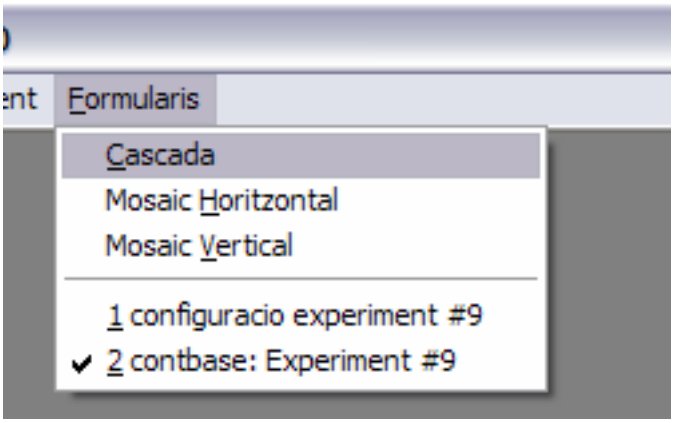


Figura 5.4.: Menú per controlar la visibilitat dels formularis oberts en la pantalla principal.

Amb aquest menú, s'ha volgut estudiar la funcionalitat que ofereixen els formularis tipus **MDI**, que proporciona *.NET* amb la programació de formularis. Permet distribuir els formularis un a sobre de l'altra o ordenats un al costat de l'altra dins de la pantalla principal, *Cascada* i *Mosaic Horitzontal* o *Mosaic Vertical*.
Apareix el nom de cada formulari obert, per poder activar un formulari específic en primer terme, per sobre dels demés.

5.2.2. Gestió de dades d'operació.

Com a primer pas per executar un experiment de recomanació, cal configurar els valors de certs paràmetres. El formulari presentat a continuació és l'encarregat de gestionar aquests paràmetres:

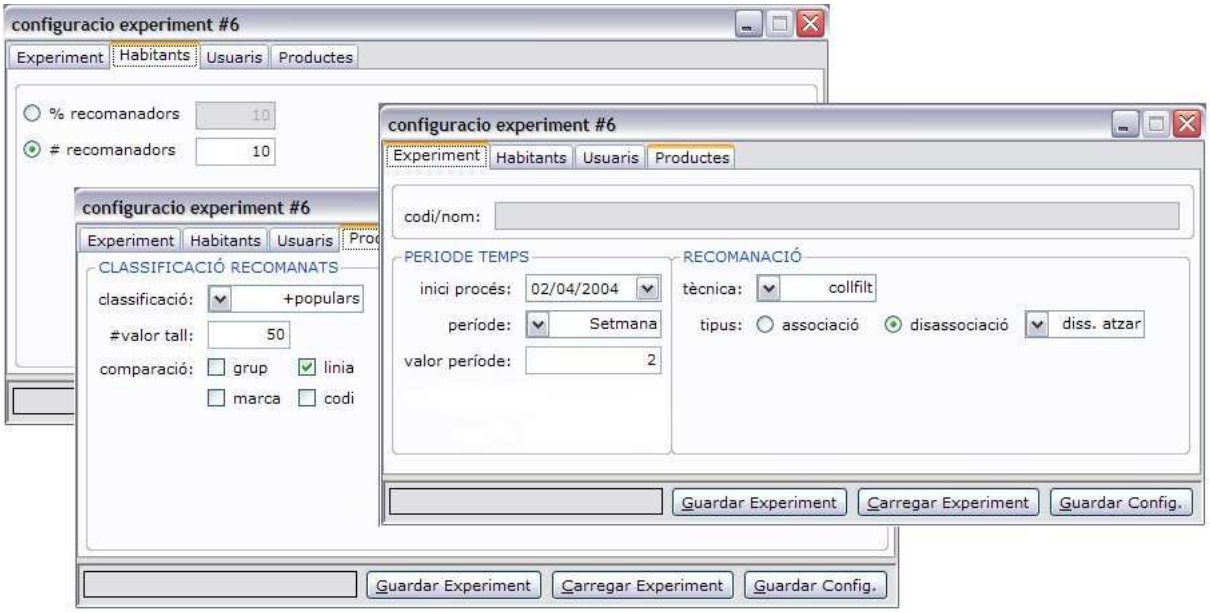


Figura 5.5.: Vàries vistes del formulari de configuració

El formulari conté varies pestanyes, agrupant els paràmetres en vàries categories. En la pestanya *Experiment* s'hi troben les dades d'operació, o paràmetres, principals per l'execució d'un experiment o procés de recomanació:

- *Data inicial per un procés*
- *La longitud de cada cicle, definint el valor i la unitat del període.*

  La unitat compren els valors: *mes; setmana.*
- *La tècnica de recomanació*

  A escollir entre: *contbase* - per la tècnica de filtrat basat en contingut -, *collfilt* - per el filtrat col·laboratiu - o *hyfilt* - per per el filtrat híbrid -.
- *El tipus d'interacció amb els usuaris.*

  A escollir entre: *diss. atzar* - per la dissociació aleatòria -, *diss. rotació* - per la dissociació en rotació -, *diss. intercanvi* - per la dissociació d'itercanvi d'usuaris entre infohabitants.

En la pestanya *Producte*, hom pot escollir uns atributs a tenir en compte per la selecció de productes a recomanar:

- *La classificació de productes:*

  Estableix una ordenació dels productes seleccionats: *+populars* - els que més agraden o es compren van primer - o *-populars* - al revés -.

- *Valor de tall, per seleccionar els primers segons la classificació anterior.*
- *Un camp comparatiu, atribut de producte a tenir en compte en les tècniques de Content Based o Hybrid Filtering a la hora de seleccionar els similars als gustos del comprador a estudiar.*

Per la pestanya Habitants només hi ha una dada d'operació a configurar, però està pensada per afegir-hi més atributs, a fi de controlar possibles comportaments de recomanació futurs.

- *Número d'infohabitants, a crear per cada experiment de recomanació.*

5.2.3. Connexió amb l'hàbitat

Aquest formulari permet establir connexió amb el servidor o hàbitat dels agents. Només caldrà especificar una adreça *ip* i un número de port, per crear un canal de comunicació. Es pot connectar i desconnectar manualment.

Si per alguna raó el servidor s'atura, l'aplicació per el canal de connexió i es veuria reflectit en aquest formulari.

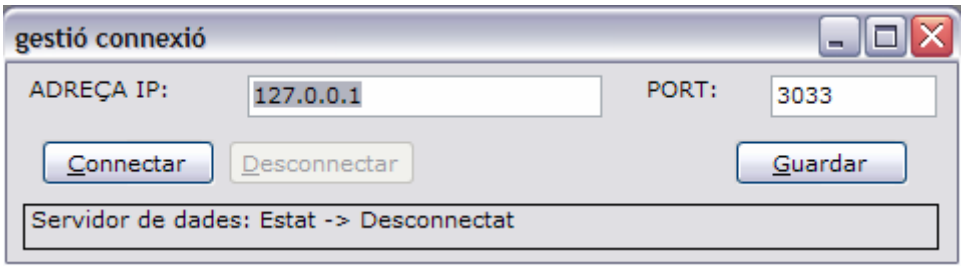


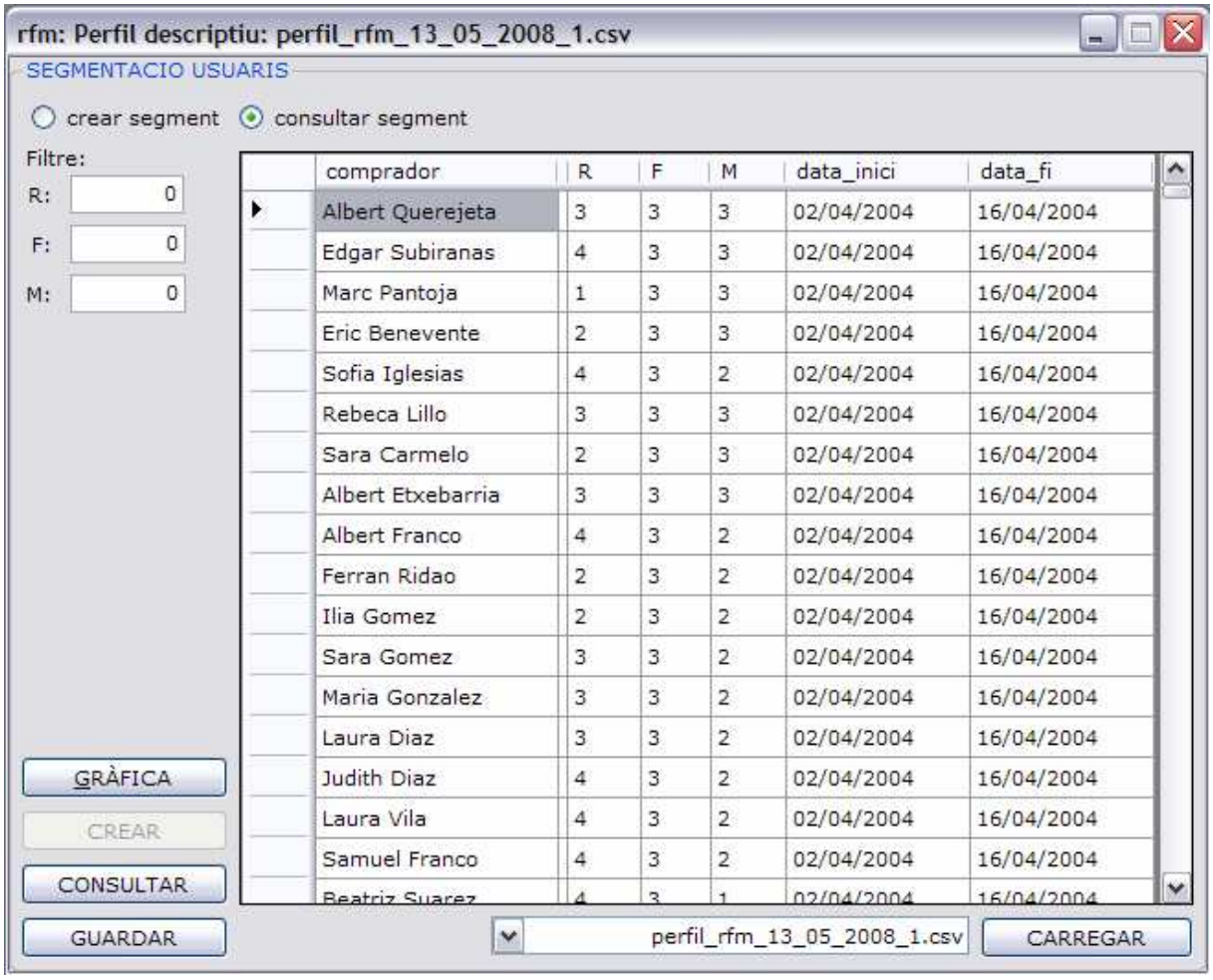
Figura 5.6.: Vista del formulari de connexió amb el servidor.

A la part inferior del formulari, apareix un missatge informatiu de l'estat de la connexió.

5.2.4. Segmentació d'usuaris

Un requeriment imprescindible per l'execució d'experiments de recomanació és la de saber a quins usuaris es fan les recomanacions.

Amb aquest formulari permet la creació de segments de compra dels usuaris, o consulta de segments ja creats, amb els quals es poden classificar els usuaris/compradors de què es disposa a la base de dades.



comprador	R	F	M	data_inici	data_fi
Albert Querejeta	3	3	3	02/04/2004	16/04/2004
Edgar Subiranas	4	3	3	02/04/2004	16/04/2004
Marc Pantoja	1	3	3	02/04/2004	16/04/2004
Eric Benevente	2	3	3	02/04/2004	16/04/2004
Sofia Iglesias	4	3	2	02/04/2004	16/04/2004
Rebeca Lillo	3	3	3	02/04/2004	16/04/2004
Sara Carmelo	2	3	3	02/04/2004	16/04/2004
Albert Etxebarria	3	3	3	02/04/2004	16/04/2004
Albert Franco	4	3	2	02/04/2004	16/04/2004
Ferran Ridao	2	3	2	02/04/2004	16/04/2004
Ilia Gomez	2	3	2	02/04/2004	16/04/2004
Sara Gomez	3	3	2	02/04/2004	16/04/2004
Maria Gonzalez	3	3	2	02/04/2004	16/04/2004
Laura Diaz	3	3	2	02/04/2004	16/04/2004
Judith Diaz	4	3	2	02/04/2004	16/04/2004
Laura Vila	4	3	2	02/04/2004	16/04/2004
Samuel Franco	4	3	2	02/04/2004	16/04/2004
Beatriz Suarez	4	3	1	02/04/2004	16/04/2004

Figura 5.7.: Vista del formulari de segmentació.

Existeixen dues opcions a escollir:

- *Crear segment, per calcular nous segments de compradors.*
- *Consultar segment, per consultar segments ja creats.*

Els segments es creen per dates. Per tenir una execució més ràpida de l'algorisme de segmentació, la creació de segments es crea per els usuaris/compradors que compren en un determinat període de l'històric de transaccions. Concretament, aquest correspon al període anterior al primer cicle d'un experiment de recomanació.

En el formulari també hi ha la opció de filtrar les dades consultades, a partir dels valors R, F i M calculats. De cada valor d'R, F i M s'ha de triar un número entre 1..5, a fi de concretar millor els usuaris sobre els quals es faran els experiments de recomanació.

5.2.5. Monitor de procés de recomanació

Un cop s'han configurat les dades d'operació i s'ha escollit el segment, filtrat o no, de compradors a qui recomanar, es pot accedir al formulari de monitorització dels processos de recomanació.

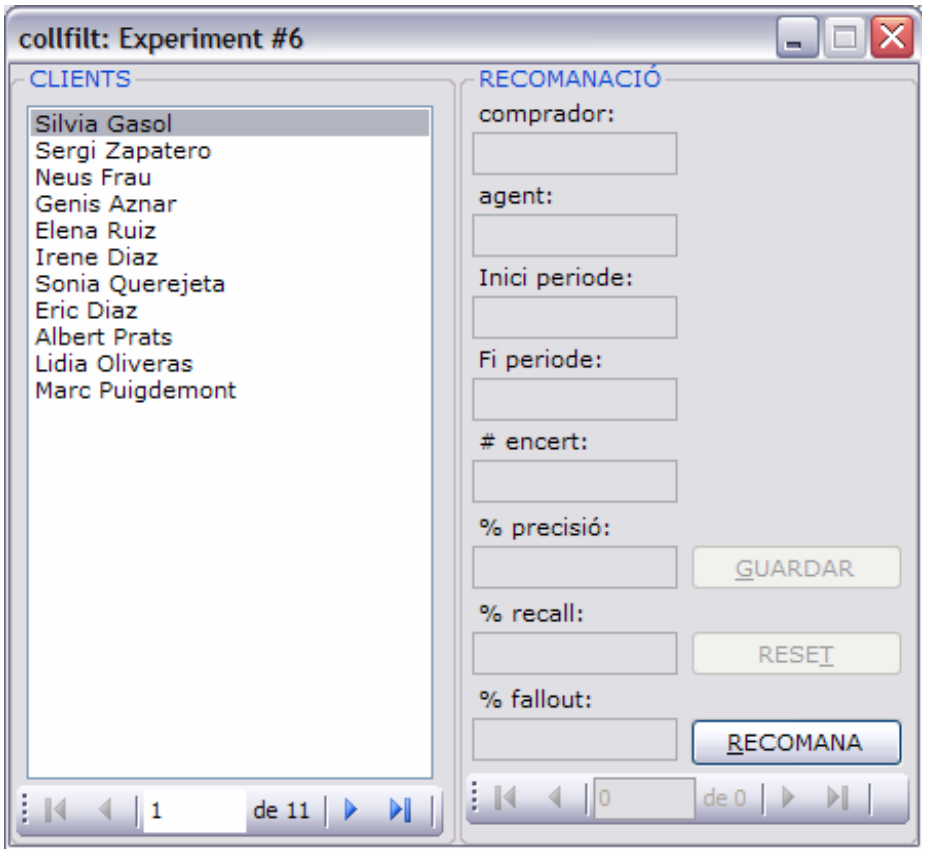


Figura 5.8.: Vista del formulari de monitorització, abans d'executar un experiment.

La funcionalitat del formulari és senzilla. Amb el botó **RECOMANA** s'envia la petició corresponent, cap al servidor, per començar un procés de recomanació.

A mida que els infohabitants enviïn les respostes dels càlculs de cada cicle, es poden anar consultant amb el navegador de dades que hi ha a la part inferior dreta del formulari.

Un cop un procés ha acabat, el botó **GUARDAR** s'activa per donar la opció de guardar els resultats en un fitxer de text, en la mateixa màquina del client, a fi de poder analitzar aquests resultats a posteriori.

5.2.6. Anàlisi d'experiments

Sense necessitat d'estar connectat amb el servidor, els experiments guardats com fitxers de text es poden mostrar en aquest formulari, la graella continguda en el mateix permet veure i llegir de manera entenedora els resultats de cada experiment.

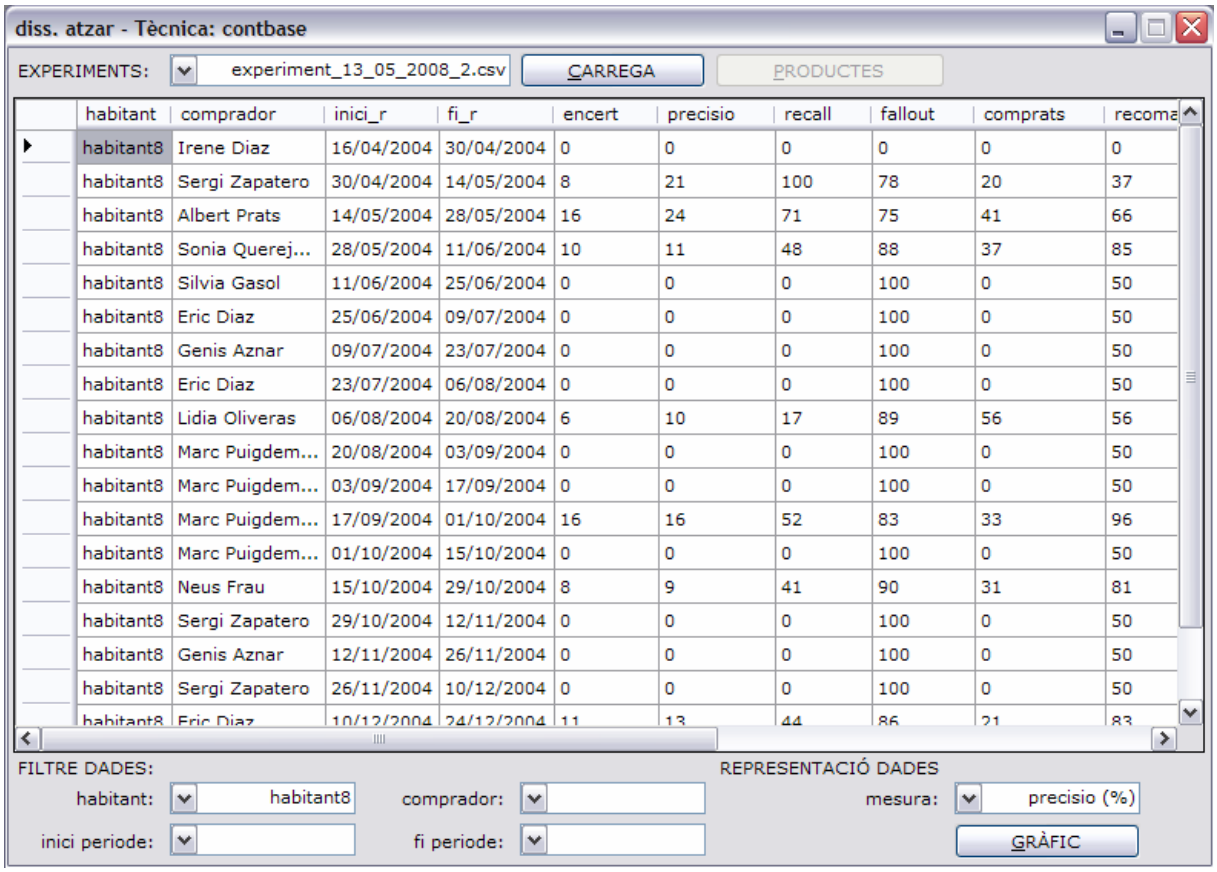


Figura 5.9.: Vista de les dades d'un experiment ja guardat en fitxer, en el formulari d'anàlisi.

La graella mostra tots els resultats rebuts de tots els infohabitants que han participat en un experiment. Podent ser un volum elevat de dades, hi ha la possibilitat de filtrar les dades per varis camps. Aquests es troben a la part inferior del formulari:

- *Nom de l'infohabitant.*
- *Nom del comprador.*
- *Data inicial o final d'un cicle.*

I perquè un usuari final pugui veure una representació visual de les mesures calculades per cada recomanació, seleccionant la mesura primer i prement el botó GRÀFIC després, s'obre un segon formulari amb una representació lineal de les mesures calculades.

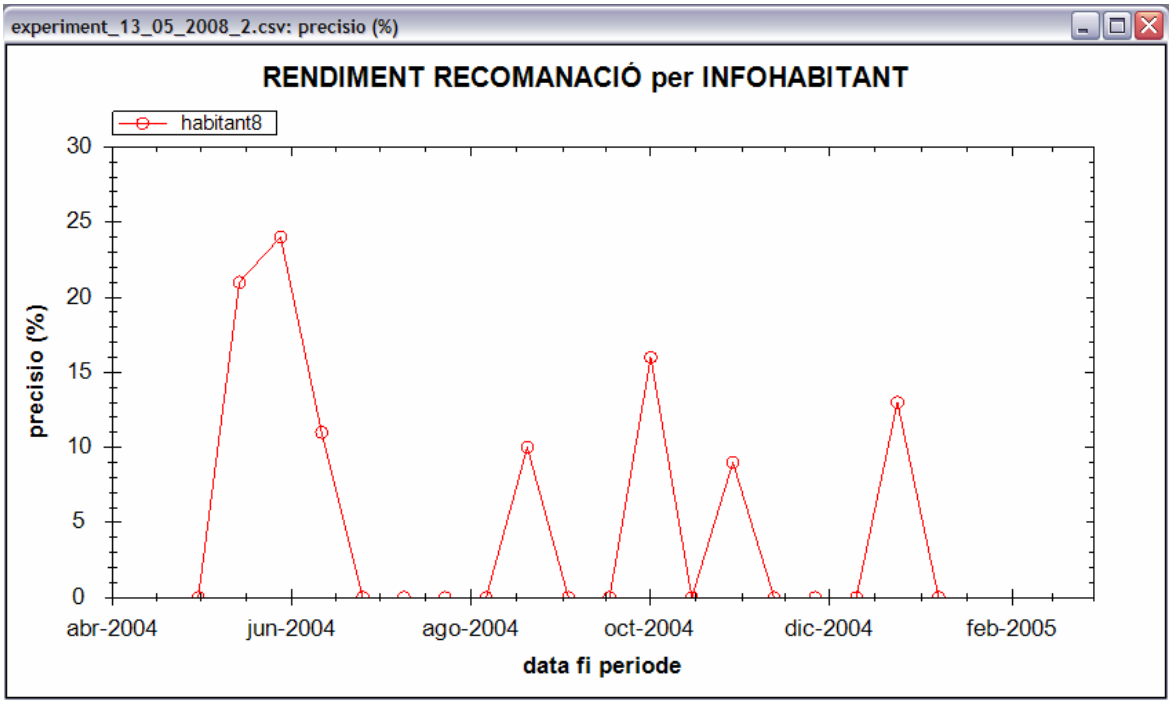


Figura 5.10.: Gràfica de valors de precisió d'un experiment, per l'infohabitant "habitant8", aplicant dissociació.

En aquest cas, la gràfica mostra l'evolució de la mesura de precisió que ha calculat l'infohabitant "habitan8" en un experiment realitzat. Aquesta és una representació dels valors de precisió calculats per un infohabitant durant l'execució d'un procés de recomanació: els punts representen la precisió d'una recomanació feta per cada usuari amb qui ha interactuat a cada cicle del procés.

5.2.7. Importació de dades

Aquest formulari s'encarrega de llegir les dades dels fitxers d'origen de dades i crea una petició per enviar les dades al servidor, a fi de ser emmagatzemades a la base de dades.

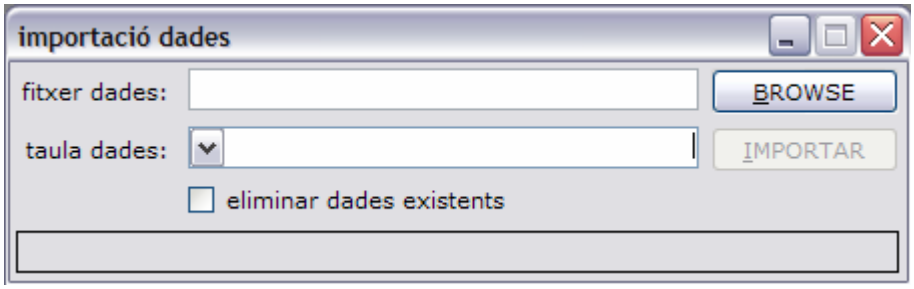


Figura 5.11.: Vista del formulari d'importació de dades.

S'ha de seleccionar el nom de la taula i el fitxer origen d'on llegir les dades. En acabat, només caldrà prémer el botó IMPORTAR per enviar les dades al servidor, a fi de ser emmagatzemades.

6. CONCLUSIONS

Per la elaboració del present projecte calia aprendre a programar amb agents software. Un cop aprés el concepte teòric d'agent, s'havia d'escollir una eina o llenguatge que permetés aplicar els coneixements. Per això, gràcies a l'ús de JADE, s'ha pogut conèixer d'una manera pràctica com enfocar una programació amb agents, a més d'aplicar conceptes teòrics en la programació dels agents.

Sobre l'objectiu de trobar nous algorismes, només ha calgut desenvolupar una primera implementació d'aquests. La investigació per trobar-los partia d'un treball prèvia (com s'explica en el capítol *2.1. Identificació de Requeriments*). Per aquesta raó, no ha calgut realitzar cap estudi d'investigació, si no que s'ha limitat el projecte a implementar-los.

Dels requeriments d'escalabilitat, sistema obert i estable (esmentats en el capítol *2.1. Identificació de Requeriments*) el disseny de l'hàbitat si compleix els dos primers però s'hauria de controlar millor l'estabilitat de tot l'hàbitat, ja que en presència d'alguna errada d'execució dels agents de gestió de l'habitat, carter, fabrica i gestor, aquests moren i el sistema queda inestable - aquesta gestió d'errors es deixa per futures versions. S'ha limitat l'avast del projecte en la implementació d'una funcionalitat mínima dels agents -. La implementació, però, del sistema recomanador i dels infohabitants si està ben controlada: la fallida d'un d'ells no altera l'execució global d'un procés de recomanació, aquest mor i es podria crear un de nou per un nou experiment.

Amb l'aplicació gràfica es disposa una eina per executar experiments de recomanació amb vàries variants i també serveix per analitzar cada experiment realitzat, a part de complir el requeriment d'usabilitat i permetre testejar el funcionament del sistema recomanador.

També calia aprendre nocions referents a la realització de recomanacions. S'ha estudiat el concepte de recomanació, des d'un punt de vista general, algunes tècniques que es fan servir i algunes mesures d'avaluació de recomanació. Com també s'han aprés, i posat a la pràctica, diversos tipus d'interacció entre agent recomanador i usuari, que es donen en qualsevol recomanació.

Amb aquest marc teòric, i aplicant el concepte d'infohabitant com a agent que treballa amb informació descentralitzada, s'havia de crear un sistema que permetés el control de simulacions de recomanació. Als infohabitants, com a agents recomanadors, se'ls ha programat

varis tipus de dissociació amb usuaris. Aquesta interacció dissociativa ha de servir per estudiar el concepte de privacitat de dades d'usuari que un agent pot contenir en si. Per el projecte no s'ha hagut de demostrar que la privacitat funciona o no, però si que s'ha aconseguit programar aquest d'interacció i realitzar experiments de recomanació on s'ha recomanat a varis usuaris durant tot un procés.

Amb totes aquests consideracions, es pot dir que s'ha implementat una primera versió d'aquest laboratori, amb èxit, i està a disposició d'una eina per realitzar experiments de recomanació. Partint de zero en nocions de recomanació i de programació d'agents, ha estat possible implementar agents software utilitzant JADE, i els experiments de recomanació s'han pogut provar satisfactòriament. Per tant, està a disposició una eina per poder realitzar simulacions de recomanació amb la finalitat d'estudiar, de manera més real, l'efectivitat d'aquests tipus i tècniques de recomanació.

6.1. TREBALL FUTUR

Però, dels objectius assolits, hi ha aspectes a millorar i a ampliar, a fi de tenir un laboratori de simulacions més complet.

Dintre de l'estudi de recomanacions, es podrien implementar nous agents, amb finalitat maligne, que ataquessin als infohabitants recomanadors per modificar els perfils d'usuari, o els gustos de compra que disposen en la seva memòria. I veure com es comporta una recomanació quan un infohabitant recomanador te dades modificades.

També per l'estudi de recomanacions, seria bo establir una col·laboració entre infohabitants recomanadors perquè puguin seleccionar productes afins a un usuari, no només de consultes a un magatzem de dades sinó també de productes que tinguin guardats altres infohabitants.

Un altre aspecte interessant, a fi de disposar un laboratori més complet, seria el d'ampliar el domini de dades amb què es treballa, i estudiar les recomanacions en dominis diferents, permetent que l'usuari del laboratori pugui escollir amb quines dades vol treballar.

Un altre punt a tenir en compte, i en el qual s'està treballant, és en la possibilitat que l'habitat dels agents pugui treballar per tal de suplir vàries peticions de recomanació a la vegada, per part de varis clients connectats al mateix temps. En aquest aspecte, es pretén treballar en la creació d'altres tipus d'aplicacions client, com pot ser una aplicació web. S'està estudiant també

amb la possibilitat de permetre la gestió dels infohabitants, crear-ne de nous, eliminar-ne, segons un valor de confiança donat a cada infohabitant, o segons si han patit atacs. Una bona opció d'assolir aquest objectiu seria la integració de la Plataforma JADE/Agent-0 [Palahí, 06].

7. BIBLIOGRAFIA

A. Mas. *Agentes Software y Sistemas Multi-Agente*. Pearson Education S.A., Madrid, 2005

T. Pender. *UML Bible*. Wiley Publishing Inc., 2003

F. Bellifemine, G. Caire, T. Trucco, G. Rimassa, *JADE Administrator's Guide*, JADE v3.5. CSELT, TILAB, 2007.

F. Bellifemine, G. Caire, T. Trucco, G. Rimassa, *JADE Programmer's Guide*, JADE v3.5. CSELT, TILAB, 2007.

7.1. Referències

[Adomavicius, 05] Adomavicius, G. (2005). *Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions*. IEEE Transactions on Knowledge and Data Engineering, 17(6):734-749.

[Balabanovic and Shoham, 97] M Balabanovic and Y. Shoham(1997). *Content-based, collaborative recommendation*. Comm. ACM, 40(3): 66-72.

[Billsus, 98] D. Billsus and M. Pazzani (1998). *Learning collaborative information filters*. In Proc. of the 15-th International Conference on Machine Learning, Wisconsin, pages 46-54.

[Burke, 02] R. Burke (2002). *Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction*, 12(4): 331-370.

[de la Rosa, et. al., 99] J.Ll. de la Rosa, I. Muñoz, B. Innocenti, A. Figueras, M. Montaner, J. A. Ramon. *Preliminary Studies of Dynamics of Physical Agents Ecosystems*. Lecture Notes in Computer Science (Artificial Intelligence) ISSN 0302-9743. Springer-Verlag LNAI pag. 373-378. 1999

[Delfin, et. al., 06] S. Delfín, C. Carrillo, E. Muntaner, A. Moreno, S. Ibarra and J.L. de la Rosa (2006). *Improving privacy of recommender agents by means of full dissociation*. Artificial Intelligence Research and Development, pages 308-315.

[Delfin, 07] S. Delfin. Improving users' privacy and attack resistance in recommender agents: Full dissociation approach. Projecte d'investigació, Departament d'Electrónica, Informàtica i Automàtica, Universitat de Girona, 2007.

[Herloker, 00] J. Herloker, J. Konstan and J. Riedl (2000). *Explaining collaborative filtering recommendations*. ACM Conf. Computer Supported Cooperative Work, pages 241- 250.

[Hughes, 00] Hughes, Arthur M. (2000), *Strategic Database Marketing: The Masterplan for Starting and Managing a Profitable Customer-Based Marketing Program*, 2nd edition, McGraw-Hill, New York

[Linden, 03] G. Linden, B. Smith and J. York (2003). *Amazon.com recommendations: Item-to-item collaborative filtering*. IEEE Internet Computing, 2003.

[Montaner, 03] M. Montaner. *Collaborative Recomender Agents Base on Case-Based Reasoning and Trust*. PhD Tesis, Universitat de Girona, 2003.

[Montaner, et. al, 03] M. Montaner, B. López, E. del Acebo, S. Aciar and I. Cuevas. *IRES: On the Integration of Restaurant Services*. Agent Technology Exhibition and Competition, AgentCities Third Information Day (iD3), Barcelona. February, 2003.

[Palahí, 06] M. Palahí. *Extensió i millora de les funcionalitats de la plataform mixta JADE/Agent-0*. Projecte fi de carrera, 2006.

[Rao and Georgeff, 91] S. A. Rao and M. P. Georgeff. (1991): *Modeling Rational Agents within a BDI-Architecture*. Second International Conference on Priciples of Knowledge Representation and Reasoning (KR91). San Mateo, C. A. 1991.

[Ricci and Del Missier, 2004] Ricci, F., and Del Missier, F. (2004). *Supporting Travel Decision Making through Personalized Recommendation.* In Clare-Marie Karat, Jan Blom, and John Karat (eds.), Designing Personalized User Experiences for eCommerce. 221-251. Kluwer Academic Publisher.

[Rolf, et. al, 06] Rolf H. van Lengen, Paul Marrow, Thies Bähr, Hans Hagen, Erwin Bonsma, Cefn Hoile. *Component Based Visualization of DIET Applications. In Scientific Visualization: The Visual Extraction of Knowledge from Data*, G.P. Bonneau, T. Ertl, G.M. Nielson (Editors), Reihe Mathematics and Visualization Springer, 367-385, 2006

[Salton and McGill, 83] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Publishing Company, New York, NY, 1983.

[Smyth and Cotter, 00] B. Smyth and P. Cotter (2000). *A personalized television listings service: Communications of the ACM*. Knowledge-Based Systems 13: 53-59.

[Wang, 98] Wang, P. 1998. *Why recommendation is special? Papers from the 1998 Workshop on Recommender Systems, part of the 15 th National Conference on Artificial Intelligence* (AAAI-98,Madison,Wisconsin, EUA), 111-113.

[Xu, et al., 05] B. Xu, M. Zhang, Z. Pan, H. Yang, O. Gervasi, M. Gavrilova, V. Kumar, A. Lagana, H. Lee, Y. Mun, D. Tanjar and C. Tan (2005). *Content-based recommendation in e-commerce. Lecture notes in computer science* (Lect. notes comput. sci.) ISSN 0302-9743.

[Yang, et al., 04] W. Yang, Z. Wang and M. You (2004). *An improved collaborative filtering method for recommendations' generation*. In Proc. of IEEE International Conference on Systems, Man and Cybernetics.