PROJECTE DE SOFTWARE PER A LA GESTIO D'ENTITATS CULTURALS

- **Idea del projecte:**

Aquest projecte es basa en la necessitat d'un grup d'entitats culturals de Blanes de tenir un software de gestió integral d'entitats que faciliti el seu funcionament diari.

Lluny de buscar una rendibilitat econòmica en la seva activitat, el que volen és una gestió fàcil de la mateixa per tal de poder arribar a més i més gent alhora i per tant poder escampar la cultura a través del país i també a l'estranger.

El problema de moltes d'elles és que degut a que els càrrecs de la junta directiva són no-remunerats, ningú sembla estar interessat en assumir-ne la responsabilitat per un període de temps llarg, i per tant, molt sovint, aquestes entitats canvien de mans ràpidament i és molt difícil per la persona que acaba d'arribar fer una bona gestió.

El que s'intenta crear és una aplicació que pugui ser utilitzada per qualsevol entitat cultural, cosa que no sembla gens complicada a simple vista ja que totes elles tenen molts trets característics en comú.
Tant si estem parlant d'un esbart, com una coral o una colla gegantera, podem dir que tenim un conjunt de socis i col·laboradors que fan que l'entitat tiri endavant.
Totes elles, a més, fan una sèrie d'actuacions que son el motiu principal de la seva existència. En aquest cas tant és anomenar-ho actuació, ballada, exhibició o qualsevol altra cosa, el cas és que aquest esdeveniment tindrà un nom de poble, una data, una durada, etc.
Si miréssim una mica més a fons el funcionament de totes elles veuríem que hi ha poques coses que les diferenciïn a nivell de volum i tipus de dades que generen. Per tant, un disseny consensuat entre vàries d'elles hauria de ser suficient perquè totes elles poguessin utilitzar l'aplicació sense haver d'introduir-hi canvis significants.

Així doncs, i entrant en matèria, la idea inicial és desenvolupar un software amb les següents característiques:

- Gestió de socis: Persones que són el motor de l'entitat. Aquests socis segurament pagaran una quota, i generen una sèrie de dades associades com ara el nom, DNI, telèfon, ...
  És vital portar al dia una relació dels socis que una entitat té.
  Per una banda, ens interessa que els socis estiguin contents pel fet de formar part d'una entitat. Coses com la data de naixement ens poden servir per enviar-los una carta el dia del seu aniversari, o potser, simplement volem tenir-los informats de totes les activitats per als pròxims 3 mesos.
  En un principi, es preveia la possibilitat de que l'aplicació facilités la gestió de les quotes dels socis, però es va decidir no implementar-ho, ja que s'entén que aquesta informació seria redundant i podria arribar a ser confusa i complicada de mantenir. És a dir, la idea era mantenir una relació de quotes igual que guardem l'adreça, però a part d'aquests ingressos, les entitats solen tenir subvencions i altres ingressos, així que igualment hauríem de disposar d'aquesta informació en una altra aplicació. D'aquesta manera, altres possibles eines com ara una calculadora acoblada a l'aplicació, queden automàticament desestimades.

- Gestió d'actuacions: La raó de ser d'una entitat cultural. Cada cop que a l'entitat li ofereixen una actuació, és convenient afegir-la al sistema amb totes les dades que es puguin obtenir.
  Un cop l'activitat s'ha realitzat, s'hi pot afegir alguna dada extra o fins i tot una fotografia. Per exemple, si per el motiu que sigui es va haver de canviar l'ordre de les cançons que es van representar seria interessant actualitzar-ho en la informació que teníem abans de dur a

terme l'esdeveniment.
Més que per a usos purament estadístics, pot ser interessant si pensem que , per exemple, l'any vinent tornarem al mateix poble. En aquest cas, ens interessa saber exactament les peces que es van interpretar per no repetir-les.

- Gestió del repertori: Sembla interessant tenir com una mena de control de totes les "peces" que una entitat és capaç de interpretar, ja siguin balls, cançons, etc. Es poden guardar el nom, l'autor, l'any que l'entitat la va començar a interpretar, una imatge adjunta, i multitud de curiositats més.

- Gestió de participants: Els socis poden intervenir en l'activitat de l'entitat de moltes maneres, des de simplement pagar una quota a formar part de la junta directiva. Ara bé, hem considerat important gestionar de manera separada la informació d'aquells que participen duent a terme l'actuació; és a dir, castellers, dansaires, cantants, etc. D'aquestes persones, ens interessarà saber, per exemple, la data en que van decidir participar activament en aquelles activitats que són la raó de ser d'una entitat d'aquestes característiques, les actuacions.

- Galeria d'imatges: El projecte contempla la possibilitat de per exemple, poder guardar una imatge juntament amb una actuació quan aquesta ha finalitzat. En un moment donat, però, potser ens interessa, per el motiu que sigui, mirar imatges de totes les actuacions que hem realitzat; en aquest punt, doncs, és on ens cal la implementació d'una galeria d'imatges, que estarà classificada en socis, actuacions, etc.

- **Passos seguits:**

El primer que vaig fer va ser anar a parlar amb Xavier Frigole, professor de la Universitat de Girona, per exposar-li la meva idea de projecte.
Li vaig explicar que, un dia, parlant amb la gent de l'Esbart Joaquim Ruyra de Blanes va sortir el tema de crear una aplicació que els ajudés en la seva activitat diària. Ells van semblar molt interessats en la idea i jo ho vaig veure com a una bona oportunitat per a fer el projecte final de carrera.
Després d'això, i sense haver adoptat cap compromís amb l'entitat, vaig anar a parlar amb el professor Frigole per tal de que em donés la seva opinió. Vaig pensar amb ell de seguida principalment per dos motius:

1. Ell havia estat el professor d'una assignatura de bases de dades que jo havia fet amb anterioritat, i després de tot, això hi està molt relacionat.
2. Ell està dintre d'un esbart i pot aportar no només una idea com a tutor de projecte sinó també com a usuari de l'aplicació.

Ell es va mostrar interessat amb la idea i va acceptar ser-ne el tutor.
El primer que vaig fer, tot seguit, va ser pensar en els diagrames UML. Per realitzar aquesta part vaig usar l'eina Objecteering. El motiu va ser que ja havia utilitzat aquesta eina en una parell d'assignatures i estava familiaritzat amb ella.

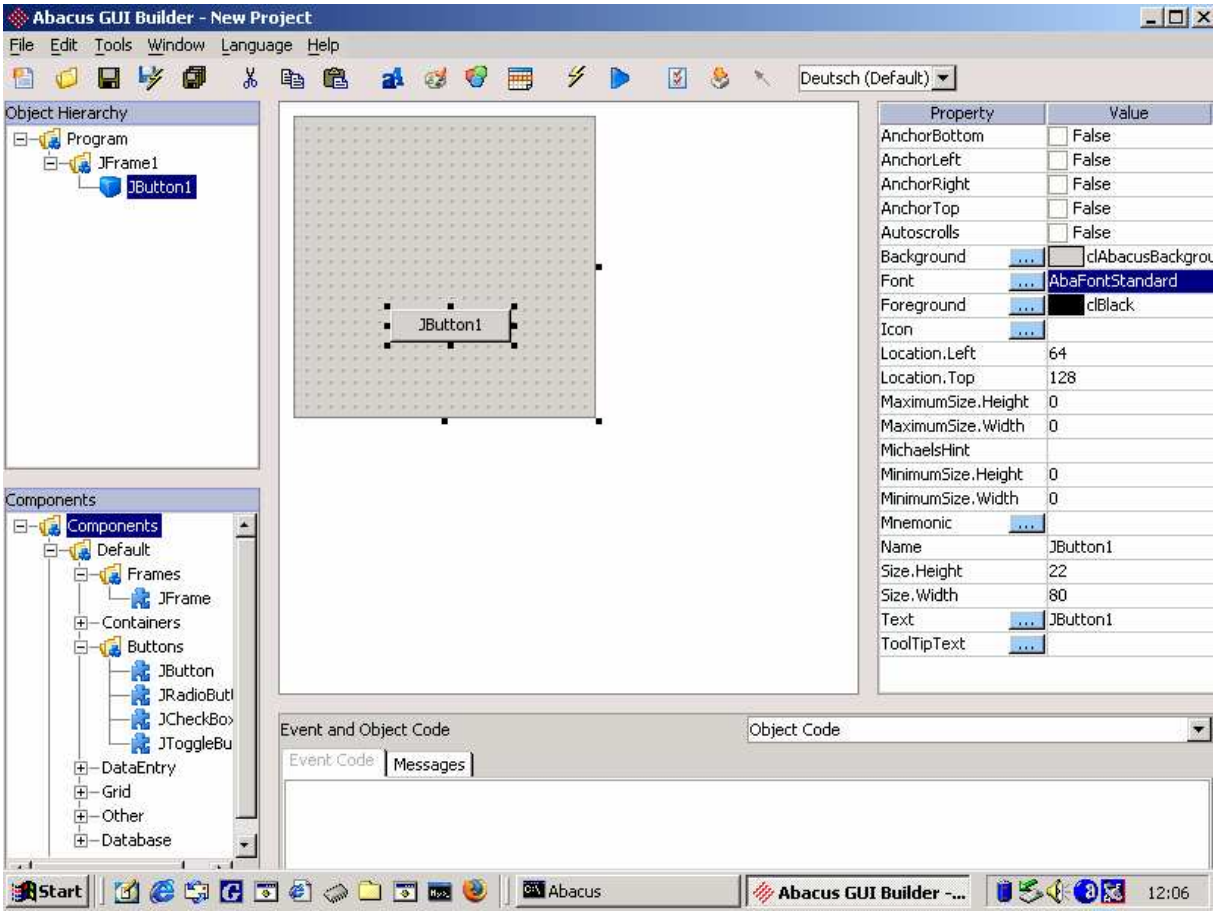
Mentre feia els diagrames UML de l'aplicació, vaig decidir que el més adient seria desenvolupar–la en dues fases separades. La primera seria la part de gestió de socis, participants, actuacions i repertori, que en essència venen a ser el mateix, així com també tota la part de la interfície gràfica.
La segona, quan la primera fos totalment funcional, seria integrar a l'aplicació altres funcionalitats, com ara la creació de llistats de socis o una galeria d'imatges.

Un cop fets els diagrames, vaig començar a redactar la present documentació i a desenvolupar el codi.
Una de les alternatives, per aprofitar el temps, era construir un prototip mentre s'estava enllestint la documentació, per tal de lliurar-la a diferents futurs usuaris de l'aplicació. D'aquesta manera, un cop finalitzada la documentació, al ja tenir diferents impressions de part dels usuaris, podria construir ràpidament el software.
Aquesta opció, però, va ser declinada, perquè com ja s'ha comentat, la idea es que l'aplicació sigui testejada per diferents entitats, per tant, es va creure que coordinar tot això enrederiria la redacció de la documentació i, per tant, no seria una gran ajuda, que seria més que res un factor de retard.

Un cop havia elaborat els diagrames i havia mig elaborat la documentació era el moment de decidir com implementar-ho, i sobretot, triar les eines per fer-ho. Tenia la intenció de desenvolupar el projecte en un entorn orientat a objectes i vaig creure molt interessant que l'aplicació es pogués executar en qualsevol plataforma, així que vaig decidir utilitzar Java. A més, necessitava escollir un sistema de base de dades per poder recolzar les dades manipulades per l'aplicació. Havia ja treballat amb sistemes com Oracle i em vaig fixar en un projecte de software lliure que s'estava expandint ràpidament, MySql, i vaig decidir utilitzar-lo.

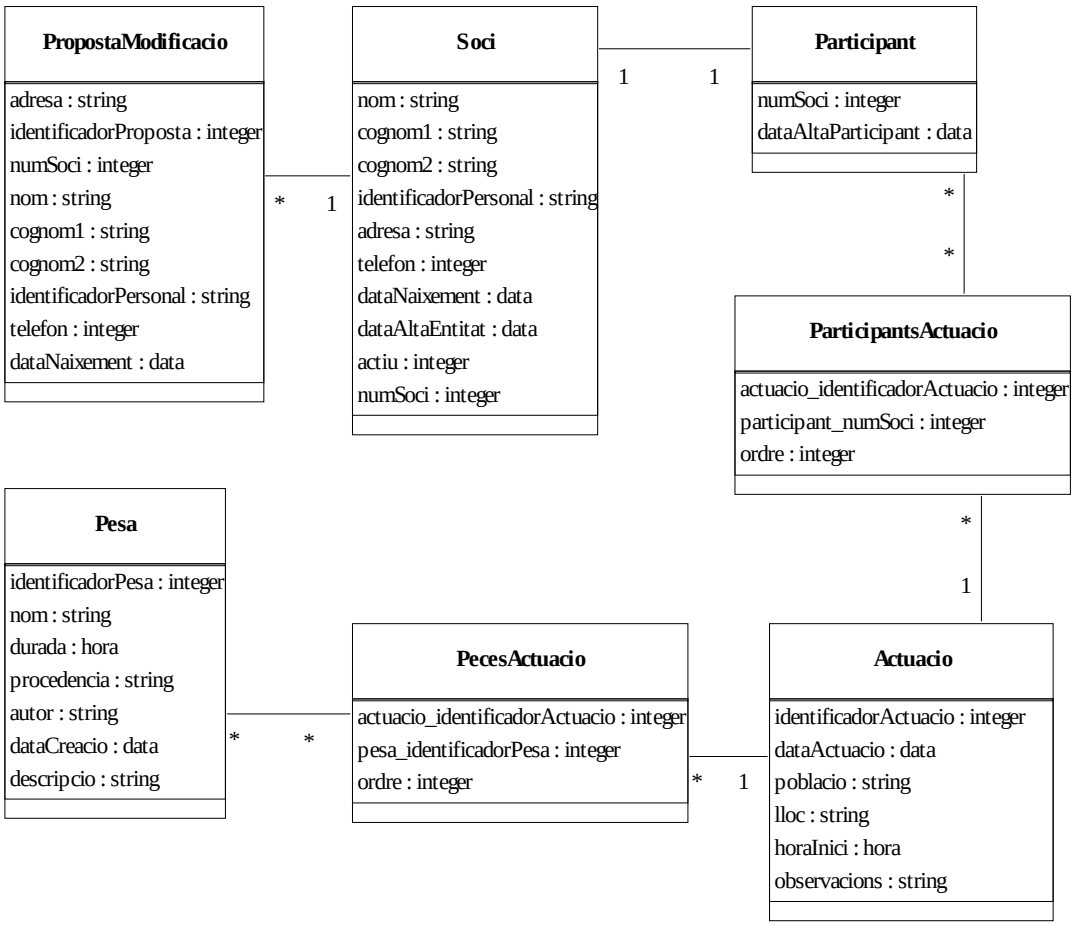
Així doncs, per començar la primera part vaig decidir que primer desenvoluparia les pantalles i a partir d'aquí elaboraria el codi que havia d'interactuar amb el menús i botons. Em vaig trobar amb diferents dificultats per dur a terme aquesta part. La veritat és que Java ofereix una

API excel·lent per poder elaborar tot tipus de finestres i manejar l'apartat gràfic en general, però vaig trobar més dificultat en fer que el que jo dissenyava es veiés més o menys igual, acceptant el canvi evident entre plataformes, en tots els sistemes operatius i arquitectures de computador. Em trobava, per exemple, que en un sistema amb Windows 2000 instal·lat podia posar un fons a un botó i en una distribució Linux, com ara Debian, no m'apareixia ni el botó. A més, notava una diferencia d'eficiència depenent de la plataforma. Vaig fer una recerca extensa per Internet i no vaig saber trobar una solució que em permetés, com a mínim, una certa homogeneïtat en les pantalles, fins que vaig trobar un altre projecte de software lliure anomenat Abacus, fet en Java, en el qual es donaven unes eines per poder elaborar la part gràfica d'una aplicació amb una facilitat realment sorprenent. Vaig elaborar un parell de finestres senzilles i vaig quedar més que satisfet, ja que no m'havia de tornar a preocupar de que si tal botó es veuria o no es veuria en tal plataforma.
Abacus és un conjunt de llibreries que no fa res més que invocar la API del propi Java, estalviant al programador la feina de la creació del codi de marcs, botons i altres elements; tot és fa des d'un entorn gràfic i el propi Abacus s'encarrega de generar el codi de les pantalles. En un principi imaginava que aquest pas entremig podria fer variar negativament la velocitat de càrrega de les pantalles, però ràpidament vaig veure que això no era així, així que vaig decidir adoptar aquest projecte com a part del meu. A més, Abacus em serviria com a entorn gràfic de programació, que en la majoria de casos facilita molt la feina del programador. Però això només m'ajudaria a crear les finestres, igualment hauria de programar tota la interacció d'aquestes amb la resta de l'aplicació.



El primer que vaig fer va ser elaborar un primer model de les pantalles amb els menús corresponents, d'aquesta manera podria ensenyar a les entitats que estiguessin interessades el primer model perquè poguessin valorar diferents conceptes com la facilitat d'ús. Aquí em vaig trobar amb una sorpresa, i era que vaig veure que ells tenien més aviat poc interès en col·laborar i que realment volien o bé una pàgina web de l'entitat o una galeria de fotografies que feien en els seus desplaçaments. Tot i això, van aportar algun comentari que vaig recollir i que em va servir per modificar alguns aspectes de disseny que segurament jo havia cregut trivials o prou bons però que no ho eren per una persona que ho veia per primer cop.

Tot seguit, vaig desenvolupar el model de la base de dades, amb tots els elements que acabo de comentar. A continuació podem veure un diagrama de l'estructura de les base de dades realitzat també amb l'aplicació Objecteering.



Ara tocava el torn al nucli de l'aplicació, és a dir, totes les classes de gestió. Per una banda, vaig decidir construir una classe o conjunt de classes que fos capaç de gestionar l'accés a la base de dades de forma que a la resta de l'aplicació li fos igual a quin sistema de base de dades accedia. Així doncs, vaig creure convenient que aquesta classe estes formada per mètodes genèrics per obtenir la informació. Les classes que interactuen amb la base de dades ni tan sols han de construir la sentencia SQL, sinó que és la pròpia classe de persistència la que ho fa. D'aquesta manera, les demés classes només necessiten fer una crida d'aquest tipus:

Persistencia.obteValor(taula_persones,columna_nom,nom_persona) i obtindran la informació de la persona amb el nom indicat a “nom_persona”.

Pel que fa a la resta del programa vaig decidir que realitzaria unes classes que fossin les de gestió, les que donen d’alta i baixa els socis, les actuacions, etc i unes altres que fossin de control, que fessin de pont entre tota la part gràfica i les de gestió. D’aquesta manera, quan, per exemple cliquéssim sobre el boto de donar d’alta un soci, la classe ControlAltaSoci seria cridada i aquesta sabria si primer ha de cridar la que busca si aquell soci ja existeix, després la que el crea, etc.

Només em faltava una cosa per decidir, les classes que havia de crear i les relacions entre elles. Com ja he comentat ja havia decidit en gran part com havia de ser l’estructura de classes. Per una banda hi havia tota l’estructura gràfica. Per una altra, tota una sèrie de classes de gestió. Finalment, unes classes “pont” que permetrien una estructura per capes molt senzilla de comprendre i que permetrien una alta modularització del software. Això era la idea general, però calia veure per exemple les relacions entre els 4 blocs que havíem decidit construir: socis, participants, actuacions i repertori. Vaig creure que a efectes de realització de codi, els participants havien de derivar dels socis, ja que per una banda, no es podia ser participant sense ser soci, així és com funcionen la majoria d’entitats culturals. Per altra banda, del participant volíem guardar la mateixa informació bàsica, afegint dades pròpies com la data d’alta com a participant. Pel que fa a la persistència de les dades vaig creure que amb una sola classe no podria assolir els objectius de reusabilitat, independència del sistema de base de dades i utilitat. Vaig creure que aquesta classe havia d’anar acompanyada d’altres que, per exemple, manipulessin els resultats. És a dir, la classe persistència seguiria pràcticament com l’havia imaginat inicialment però si per exemple volia buscar un soci per més d’un valor, no podia, necessitaria un objecte auxiliar. Per exemple, podria tenir un “ObjectePersistencia” que guardés la informació que vull buscar. La crida quedaria així.

```
ObjectePersistencia.afegeixCamp(“nom”,”Jordi”);
ObjectePersistencia.afegeixCamp(“cognom1”,”Moles”);
Persistencia.obte(“soci”,ObjectePersistencia);
```

D’aquesta manera podríem consultar socis a partir dels valors que l’ObjectePersistencia conté. A part, segurament necessitaríem un altre objecte que recollís els resultats i permetés treballar còmodament amb ells.

Vaig realitzar la primera part del projecte sense més entrebancs que els que ja he comentat. Abans de passar a la segona fase, però, vaig creure molt oportú testejar l’aplicació en diferents plataformes, i no només per el problemes inicials que vaig tenir amb l’aspecte gràfic, sinó també per altres aspectes com ara l’eficiència. Al final del document “manual.pdf” que s’inclou en aquest projecte podem trobar les captures de pantalla realitzades en aquest punt. El resultat va ser satisfactori així que vaig passar immediatament a desenvolupar la segona fase. En aquest cas, el que havia de formar part d’aquesta fase encara estava per decidir. Fent la gestió de socis vaig veure com a eina imprescindible poder generar llistats de socis ja definits, on només haguéssim de facilitar alguna dada. Aquests són alguns exemples: Socis nascuts a l’any X, socis actius actualment, socis que es van donar d’alta l’any X.

Un altre punt important que em va venir al cap mentre desenvolupava la primera part era el tema de l’accés a la informació. Tal i com es mostra en el diagrama de cas d’ús que podem trobar en pagines posteriors, l’aplicació disposa de diferents usuaris, que tenen una visió diferent del sistema, poden realitzar tasques diferents. Això ho vaig implementar amb un objecte persistent en memòria mentre el programa s’està executant i que en qualsevol moment

només pot tenir un rol d'un usuari. Cada cop que intentem entrar a una secció, es verifica que en aquell moment, l'objecte tingui un rol que li permeti accedir. En cas contrari, a l'usuari se li demana un usuari i una contrasenya; però això era la primera fase. El cas és que se'm va plantejar el dubte de si per exemple seria interessant que els diferents socis, participants i demés de l'entitat, poguessin tenir l'aplicació instal·lada en el seu ordenador i poguessin accedir en qualsevol moment a una base de dades centralitzada i evidentment actualitzada. Vaig descartar la possibilitat de refer l'aplicació i dissenyar un model client/servidor. El que vaig fer va ser modificar la classe Persistencia perquè pogués acceptar connexions a qualsevol servidor i qualsevol port d'aquest. A més, vaig afegir en el menú principal una opció per poder modificar aquestes dades i també l'usuari i contrasenya d'accés a la base de dades. D'aquesta manera, podríem tenir un servidor Mysql instal·lat en qualsevol maquina amb connexió a Internet i engegat les 24 hores del dia amb totes les dades de l'entitat, i llavors tothom que ho volgués, configurant l'aplicació correctament, podria accedir a tota la informació des de casa seva.
Sense deixar aquest tema, vaig implementar, com a mesura que creia necessària si es permetia aquest accés remot, un sistema de registre de totes les connexions al sistema de base de dades. D'aquesta manera, cada cop que un usuari s'identifiqués per poder accedir a qualsevol part del sistema reservada a certs rols, com per exemple modificar un soci, dades com l'usuari i l'hora de connexió quedarien enregistrades.
Per acabar ja amb aquest tema, vaig considerar oportú fer una petita pantalla on usuaris prèviament identificats poguessin donar d'alta nous usuaris amb diferents rols associats.

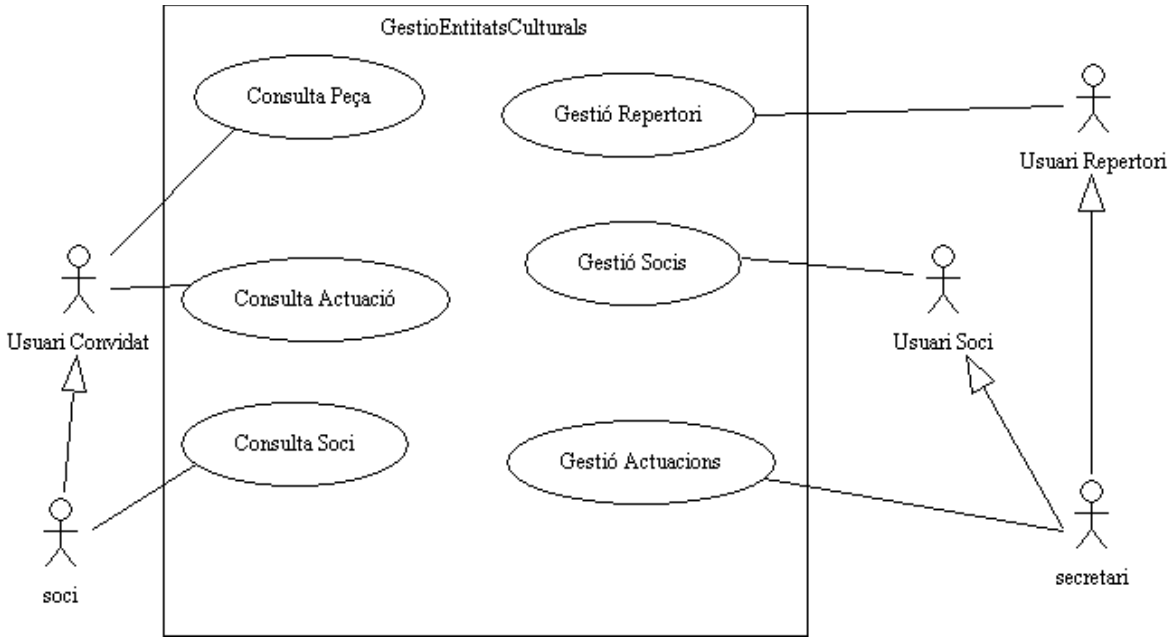
Un altre tema important era l'elaboració d'una galeria d'imatges on els usuaris poguessin consultar, per exemple, totes les imatges de socis. Quan en donem un d'alta, podem assignar-li una imatge i després podem consultar-la o modificar-la. Ara bé, la idea era poder disposar d'una galeria on pogués simvisualitzar totes les imatges associades a socis o actuacions. Això havia estat una reclamació de les entitats, que tenien el desig de crear en les respectives pàgines web una secció d'imatges. Així doncs, vaig pensar que al executar aquesta opció en el menú, enlloc d'una nova finestra de l'aplicació s'hauria d'obrir una finestra del navegador per defecte del sistema. A més, vaig creure necessari dissenyar un sistema amb el que es pogués integrar fàcilment a una pàgina web tal i com ells desitjaven. Vaig decidir elaborar la galeria en Java de totes maneres, però executant-la com un applet dins d'una pàgina html. Els applets, per motius de seguretat, no són capaços d'escriure fitxers, només llegir-ne, i ni tan sols poden realitzar connexions a bases de dades. De totes maneres, això no representa un problema ja que podem mantenir un fitxer amb totes les imatges associades a socis i que l'aplet elabori la galeria a partir d'ell, tal i com fa més d'una aplicació comercial. Una alternativa podia haver estat desenvolupar una galeria en un llenguatge dinàmic com ASP o PHP, però això hauria significat un requeriment extra a l'aplicació i desitjava mantenir com a únic requeriment el servidor Mysql.
Així doncs, vaig crear una carpeta separada dins l'estructura de l'aplicació on es trobarien totes les imatges i els fitxers amb la relació d'imatges, així com els .html. Aquesta carpeta es podria copiar a qualsevol servidor web i des de la nostra pàgina web podríem crear un link als fitxers .html corresponents.

Per últim, i per poder complir amb la llei de protecció de dades, vaig afegir una opció al menú que serveix per poder eliminar les dades d'un soci que hem donat de baixa si aquest ho desitja. Quan un soci demana la baixa, aquest no s'elimina per raons estadístiques però també d'integritat de la base de dades. Si ell ho sol·licita, però, hem d'eliminar les dades. Amb aquesta eina el que fem es seleccionar aquells camps que volem eliminar i el sistema posarà zeros o asteriscs en el seu lloc. Si per exemple un soci demana que el donem de baixa i eliminem totes les seves dades, podem seleccionar totes les caselles d'aquesta eina; tots els camps numèrics seran reemplaçats per zeros i els de text per asteriscs.

- **Diagrames UML:**

  - ✓ **Cas d'ús general:**

Diagrama de cas d'ús general de l'aplicació un cop superada la primera fase.
El diagrama conté 6 usuaris amb diferents permisos cadascú.



Usuari Convidat: És aquell que no necessita identificar-se per accedir al sistema. Té un nombre restringit d'operacions permeses i està pensat per aquella gent externa a l'entitat que vol consultar alguna informació sobre l'entitat. Com que no n'és membre, només està autoritzat a consultar el repertori de peces que l'entitat pot interpretar així com també les activitats que aquesta realitza.
Si pensem, per exemple, que aquesta persona està interessada en contractar l'entitat per a una actuació i implementem un sistema web, aquesta persona, des de casa podrà veure, per exemple, de quines peces l'entitat disposa.

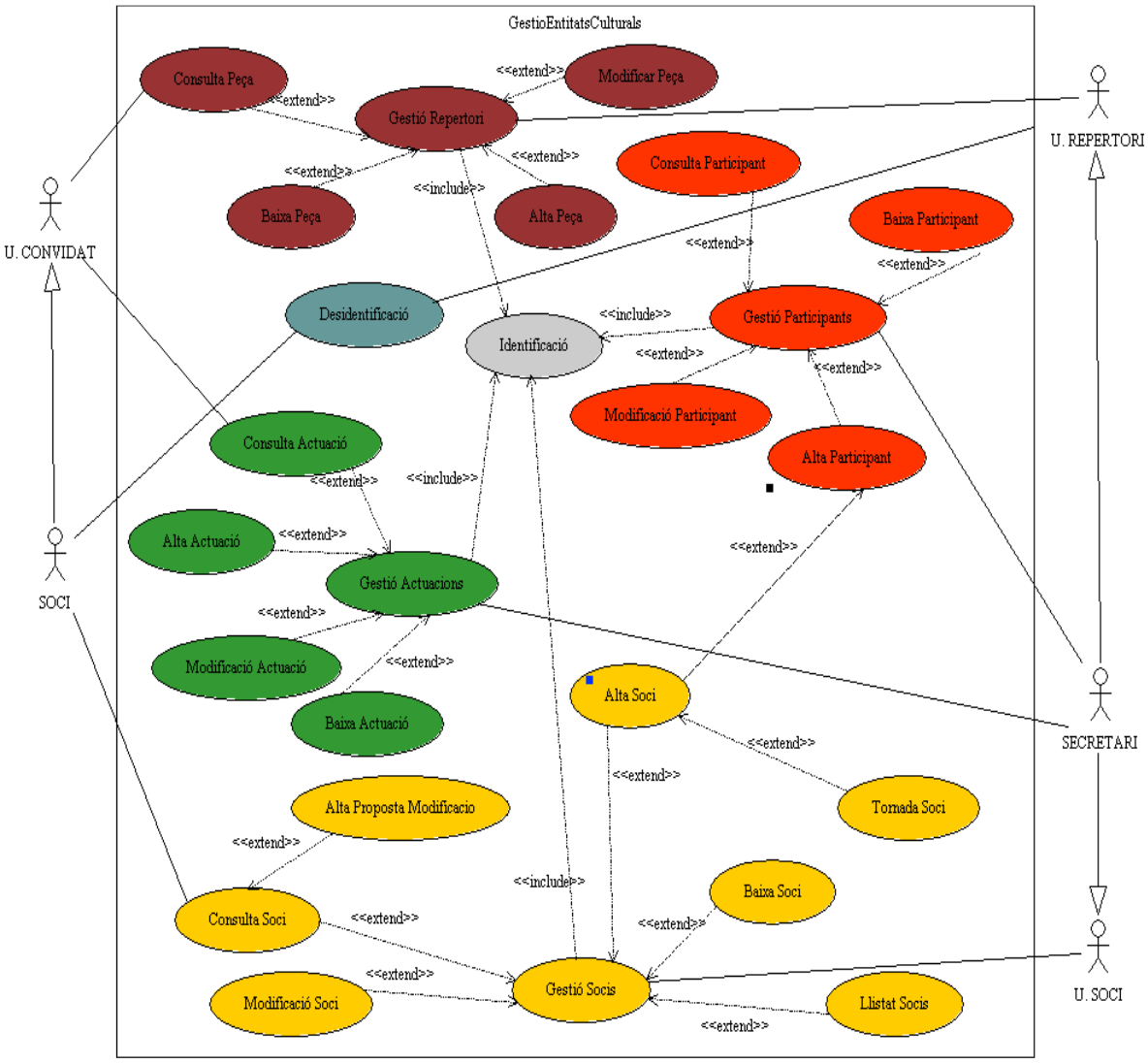
Soci: Cada soci és proveït amb una clau per identificar-se en el sistema. Aquesta clau li permet, a part de les operacions permeses a l'usuari convidat, consultar les dades que l'entitat guarda d'ell.
Com a funcionalitat addicional, el soci pot demanar una modificació de les seves dades un cop dins de la zona restringida, però no modificar-les directament.

Usuari Repertori: La idea d'aquest usuari (igual que l'Usuari Soci) és la de tenir alguna persona que ajudi en l'activitat assignada al secretari.
D'entrada, s'autoritza aquest usuari a portar la gestió del repertori. Aquesta tasca, per una banda, és molt laboriosa, perquè de fet, qualsevol dada que es pugui aportar és valida per a ser guardada en aquest apartat (per tant, és una permanent recerca); d'altra banda, pot ser assignada a qualsevol persona ja que no és una part fonamental del funcionament de l'entitat, o en altres paraules, no compromet ni la privacitat de les dades del soci ni altres dades que no és convenient que estiguin a disposició de qualsevol.

Usuari Soci: És el mateix plantejament que per a l'Usuari Repertori. Aquest punt, però, afecta a la gestió de dades dels socis i, aquests, com a persones, tenen el dret a la privacitat i a la protecció de les seves dades, per tant, en el moment que es creï un usuari capaç de manipular dades privades dels socis s'ha d'anar amb compte i fer el possible per assegurar que no se'n fa un mal ús.

Secretari: És l'usuari encarregat de portar al dia totes les dades que genera l'entitat. Simplement és l'usuari que té accés a totes les parts del programa.

- ✓ **Cas d'ús 1ª fase(Diagrama detallat):**



Aquest és el diagrama de casos d'ús extens de la primera fase de desenvolupament del software.
En ell s'hi representen, amb el màxim nivell de detall possible els casos d'ús que s'implementaran en aquesta primera etapa.

La idea és que hi hagi una instància d'un objecte Usuari Actual que estigui present mentre l'aplicació està en marxa.
Quan l'aplicació s'inicia, aquest objecte obté el rol de Usuari Convidat. La persona que està executant aquest rol només pot consultar les actuacions programades de l'entitat i les peces del repertori.

Si l'usuari desitja consultar/manipular altres tipus de dades, seleccionarà, per exemple, Gestió Repertori per tal de fer consultes, altes, baixes,... d'alguna peça del repertori. En aquest moment, el sistema validarà que el rol d'Usuari Convidat sigui el corresponent per poder accedir a la secció restringida. Com que no ho és, el sistema demanarà a l'usuari que s'identifiqui amb un nom d'usuari i contrasenya. Un cop validada aquesta informació, el rol de l'objecte canviarà i la persona podrà circular per aquelles zones que el nou rol li permet accedir.

Quan s'accedeixi a una zona no autoritzada per al rol actual, el sistema demana de nou a l'usuari que s'identifiqui.
Un cop l'usuari de l'aplicació hagi acabat amb la feina que tenia pendent, hauria de sortir de la sessió, i aquí és on té lloc el cas d'ús Desidentificació. Aquest cas d'ús permet que l'objecte Usuari Actual adopti el rol de Usuari Convidat, i per tant, que la pròxima persona que utilitzi el sistema hagi d'introduir el seu nom d'usuari i contrasenya si vol accedir a una zona restringida.

A part dels casos d'ús de identificació/desidentificació, podem veure en el diagrama 4 blocs clarament diferenciats amb colors.

El color groc representa la gestió de socis. Com s'ha explicat amb anterioritat, aquesta part del sistema està controlada per el secretari o bé per un usuari autoritzat, que s'encarregaran de donar d'alta i baixa els socis. També en modificaran les dades quan sigui necessàries o fins i tot faran simples consultes de la informació que l'entitat guarda d'aquesta persona.
En aquest cas s'ha cregut oportú afegir-hi un cas d'us anomenat Alta Proposta de Modificació, que només podrà ser executat per el soci. L'objectiu és que l'usuari pugui demanar una actualització de les seves dades, donat el cas, per exemple, que canviï de casa. El soci no té permís per modificar cap de les seves dades, però sí que pot usar el sistema per fer una petició de modificació de les seves dades. Aquestes petició serà revisada per el secretari i a partir d'això, les dades del soci seran actualitzades en el sistema. També permet a l'usuari generar un llistat amb paràmetres predefinits, a partir del cas d'ús Llistat Socis.

El color vermell clar representa la gestió dels participants. Qualsevol participant ha de ser prèviament soci. Les operacions implementades seran les de alta, baixa,consulta i modificació de dades. Així doncs, quan es doni d'alta un participant amb unes dades que no són al sistema, també s'agregarà el corresponent soci al sistema. En el sentit contrari, quan un participant es dona de baixa, no té perquè significar la baixa del soci, doncs aquest pot seguir formant part de l'entitat. Per tant, si també volem donar de baixa el soci haurem d'utilitzar el cas d'ús corresponent.

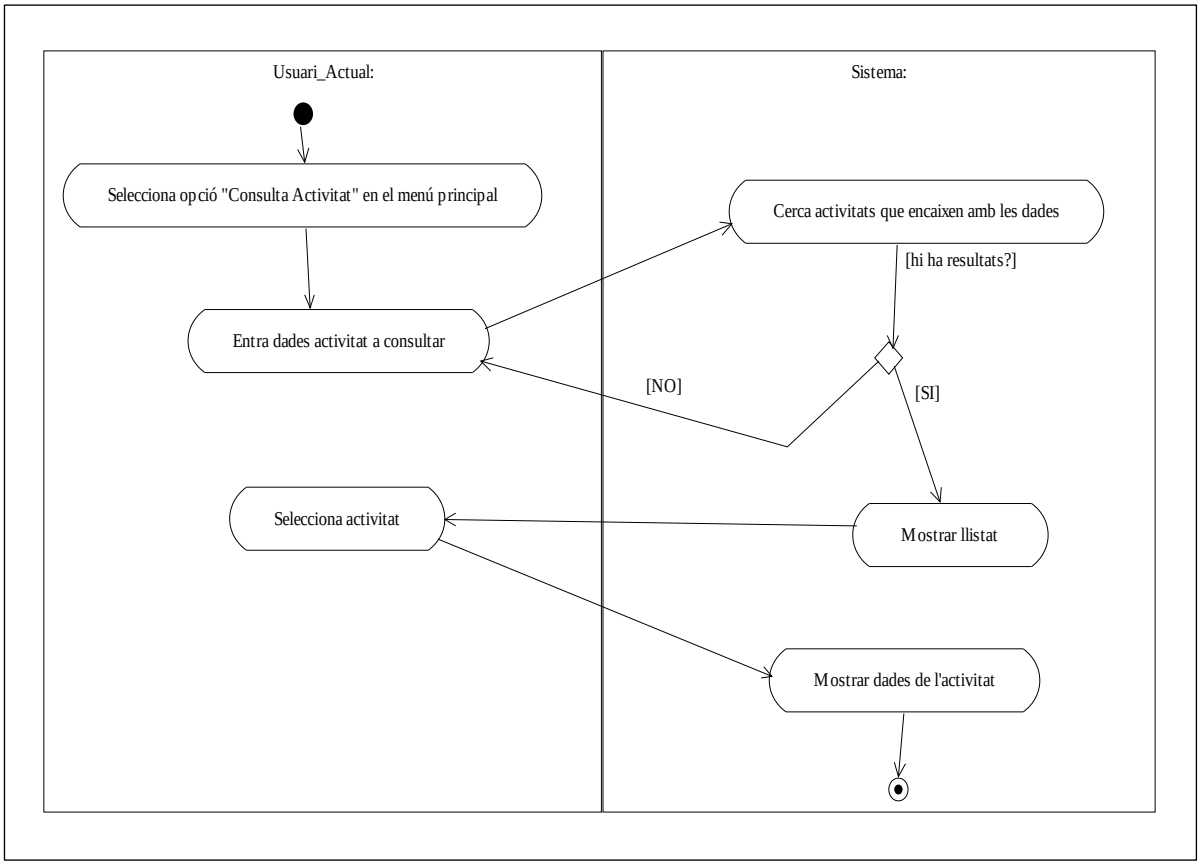
El color verd representa la gestió d'actuacions. Com ja s'ha comentat, en principi és una secció restringida al secretari, però no seria mala idea ampliar els permisos per tal de permetre a un nou usuari autoritzat accedir-hi. Això no representa un gran canvi, solament s'hauria de canviar el procés de confirmació dels permisos del rol actual de l'objecte Usuari Actual.
En aquesta part del sistema es poden crear i eliminar actuacions, així com també modificar-les o simplement consultar-ne alguna dada.

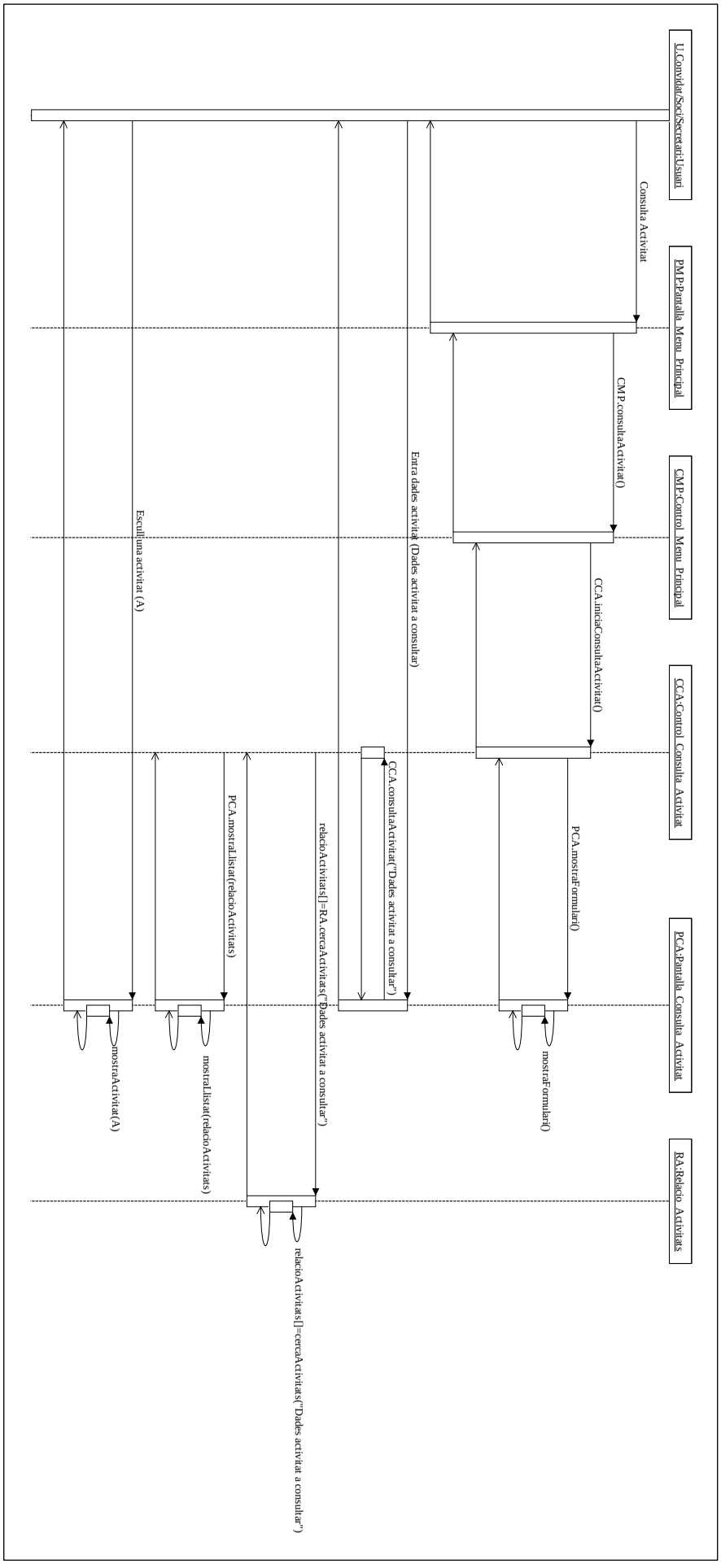
Finalment, la última part del sistema prevista en aquesta primera fase és la de gestió de peces del repertori, diferenciada de les altres amb el color vermell fosc.
La persona autoritzada a accedir a aquesta secció podrà fer altes i baixes del repertori, així com també modificar-ne dades o consultar-ne els detalls.

- **Casos d'ús:**

  - ✓ **Consulta Actuació:**

Cas d'ús: Consulta Actuació
Objectiu: Consultar les dades d'una determinada actuació
Actors: Usuari Convidat, Soci, Secretari
Precondició: -
Flux bàsic: 1- Usuari: Entra una o més dades de l'actuació 2- Sistema: Valida la lògica de les dades entrades i cerca en el sistema totes aquelles actuacions que hi coincideixen. 3- Sistema: Si n'hi ha alguna, es mostra una llista. 4- Usuari: Selecciona una actuació. Si només n'hi ha una, la mostra directament. 5- Sistema: Mostra totes les dades disponibles de l'actuació seleccionada
Postcondició: S'ha mostrat tota la informació disponible sobre l'actuació si alguna reuneix les condicions
Flux alternatiu (extensions): 2.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: un '1' en un nom de poble) es tornen a demanar fins que siguin vàlides. 3.a- Sistema: En cas de no trobar cap actuació, s'informa a l'usuari i es torna al punt inicial, on l'usuari haurà de tornar a seleccionar l'opció corresponent en el menú principal.

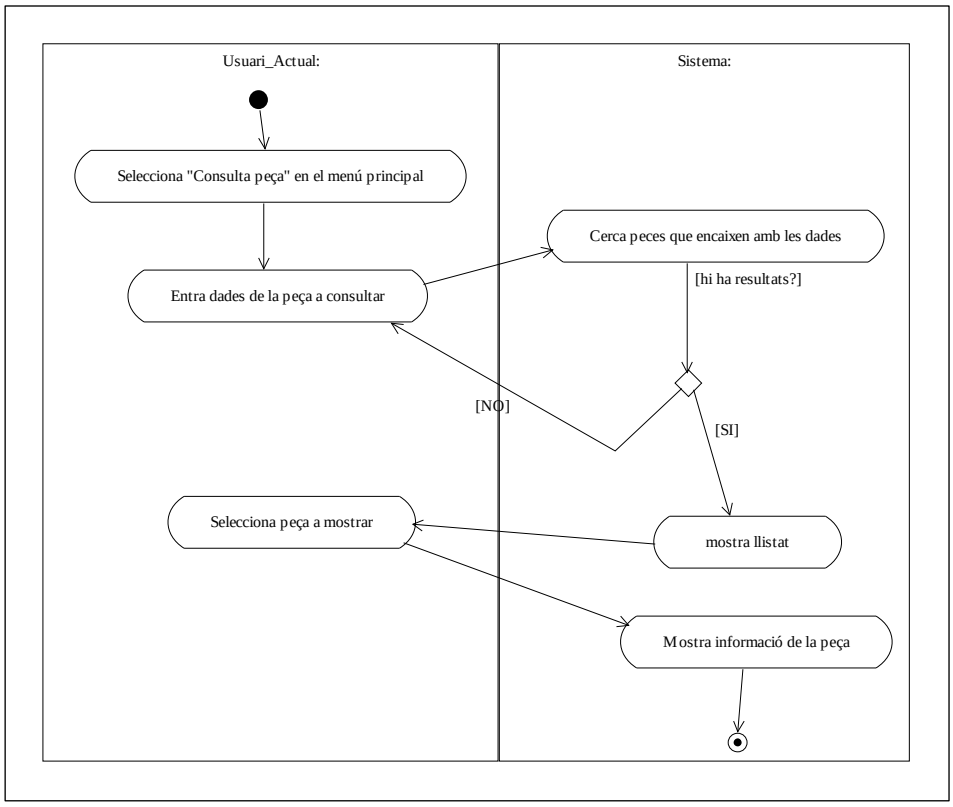


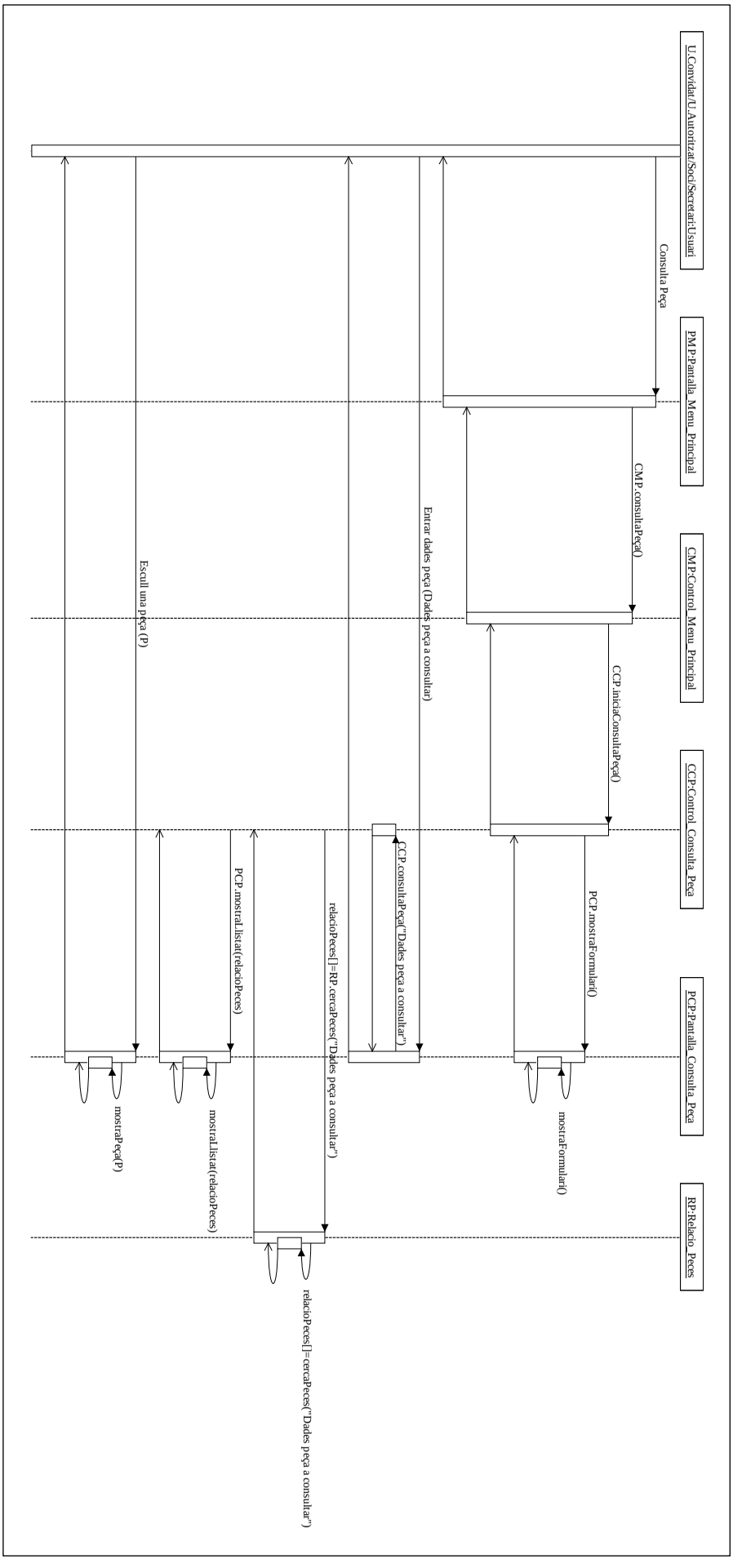

U:Convidat/Soci/Secretari/Usuari
PMP:Pantalla Menu Principal
CMP:Control Menu Principal
CCA:Control Consulta Activitat
PCA:Pantalla Consulta Activitat
RA:Relacio Activitats
Consulta Activitat
CMP.consultaActivitat()
CCA.iniciaConsultaActivitat()
PCA.mostraFormulari()
mostraFormulari()
Entra dades activitat (Dades activitat a consultar)
CCA.consultaActivitat("Dades activitat a consultar")
relacioActivitats[]=RA.cercaActivitats("Dades activitat a consultar")
relacioActivitats[]=cercaActivitats("Dades activitat a consultar")
Escull una activitat (A)
PCA.mostraLlistat(relacioActivitats)
mostraLlistat(relacioActivitats)
mostraActivitat(A)

Consultar Actuació (o activitat) permet a l'usuari de l'aplicació consultar totes les dades que l'entitat té d'una determinada actuació. Quan l'usuari clica sobre aquesta opció en el menú, el sistema li demana alguna dada sobre l'actuació (Ex: mes de l'any en que tindrà lloc). El sistema llavors busca totes les actuacions, les dades de les quals "encaixen" amb les dades aportades per l'usuari de l'aplicació, i en mostra un llistat. La persona selecciona una actuació de la llista i el sistema en mostra totes les dades disponibles: lloc, hora, programa, etc.

- ✓ **Consulta Peça:**

Cas d'ús: Consulta Peça
Objectiu: Consultar les dades d'una determinada peça
Actors: Usuari Convidat, Soci, Usuari Repertori, Secretari
Precondició: -
Flux bàsic: • Usuari: Entra una o més dades de la peça • Sistema: Valida la lògica de les dades i cerca en el sistema totes aquelles peces que hi coincideixen. • Sistema: Si n'hi ha alguna, es mostra una llista. • Usuari: Selecciona una peça. Si només n'hi ha una, es mostra directament. • Sistema: Mostra totes les dades disponibles de la peça seleccionada
Postcondició: S'ha mostrat tota la informació disponible sobre la peça si alguna reuneix les condicions
Flux alternatiu (extensions): 2.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: un '1' en el nom de la peça) es tornen a demanar fins que siguin vàlides. 3.a- Sistema: En cas de no trobar cap peça, s'informa a l'usuari i es torna al punt inicial, on l'usuari haurà de tornar a seleccionar l'opció corresponent en el menú principal.

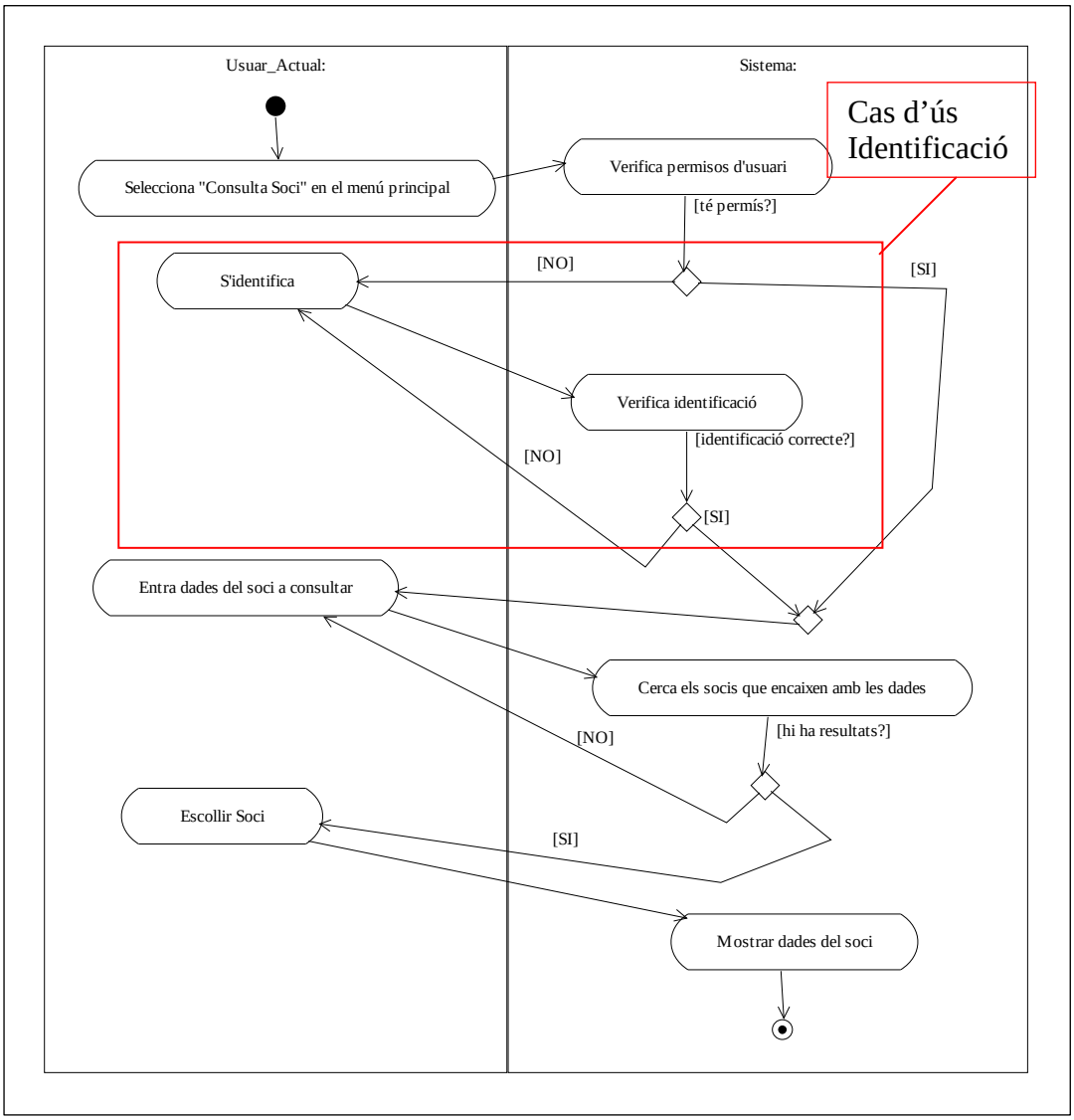



U:Convidat/U:Autoritzat/Soci/Secretari:Usuari
PM:P:Pantalla_Menu_Principal
CM:P:Control_Menu_Principal
CCP:Control_Consulta_Peça
PCP:Pantalla_Consulta_Peça
RP:Relació_Peces
Consulta Peça
CMP.consultaPeça()
CCP.iniciaConsultaPeça()
PCP.mostraFormulari()
mostraFormulari()
Entrar dades peça (Dades peça a consultar)
CCP.consultaPeça("Dades peça a consultar")
relacioPeces[]=RP.cercaPeces("Dades peça a consultar")
relacioPeces[]=cercaPeces("Dades peça a consultar")
Escull una peça (P)
PCP.mostraLlista(relacioPeces)
mostraLlista(relacioPeces)
mostraPeça(P)

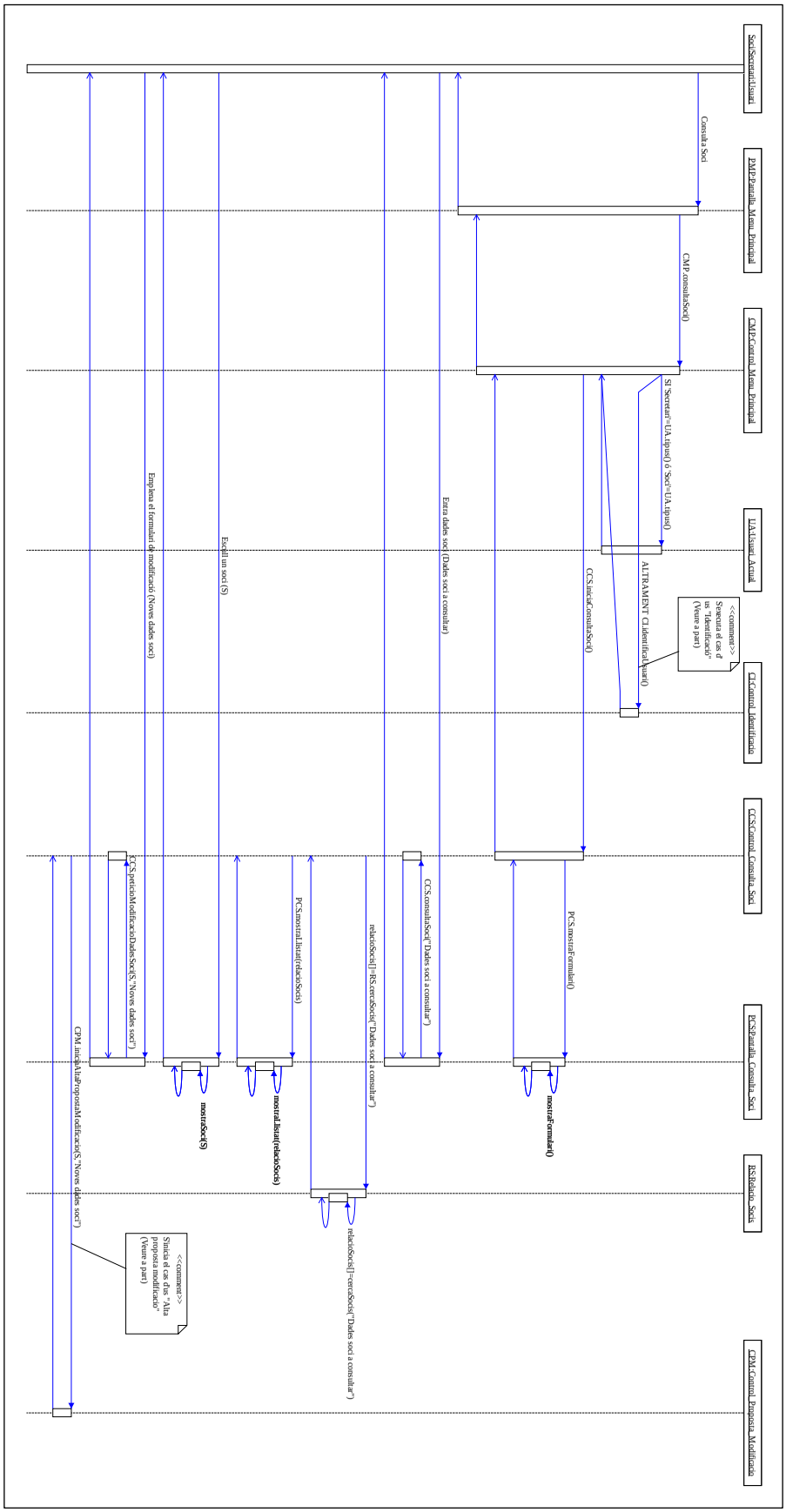
Consultar Peça permet a l'usuari de l'aplicació consultar totes les dades que l'entitat té d'una determinada peça del repertori. Quan l'usuari clica sobre aquesta opció en el menú, el sistema li demana alguna dada sobre la peça (Ex: lloc d'on es creu que procedeix). El sistema llavors busca totes les peces, les dades de les quals "encaixen" amb les dades aportades, i en mostra un llistat. La persona selecciona una peça de la llista i el sistema en mostra totes les dades disponibles: lloc de procedència, nom, durada, etc.

- ✓ **Consulta Soci:**

Cas d'ús: Consulta soci
Objectiu: Conèixer les dades d'un soci
Actors: Soci, Usuari Soci, Secretari
Precondició: -
Flux bàsic: 1- Sistema: El sistema verifica que el rol de l'usuari actual sigui el de Soci o Secretari 2- Usuari: Entra una o vàries dades del soci 3- Sistema: Valida lògicament les dades entrades per l'usuari i cerca aquells socis que hi coincideixen. 4- Sistema: Si en troba algun, mostra una llista. 5- Usuari: Selecciona el soci del que vol llegir la informació. Si només en troba un, es mostra automàticament. 6- Usuari: Si l'usuari vol modificar alguna de les dades del soci, emplena el formulari adjunt 7- Sistema: Si l'usuari ha completat el formulari del pas anterior, es valida la lògica de les dades entrades i s'executa el cas d'us "Alta proposta modificació".
Postcondició: S'ha mostrat les propietats del soci seleccionat si algun coincideix amb les paràmetres introduïts.
Flux alternatiu (extensions): 1.a- Sistema: Si el rol de l'usuari actual no és el que correspon a la zona a la que es vol accedir, s'executa el cas d'ús Identificació. 3.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: un '1' en el nom del soci) es tornen a demanar fins que siguin vàlides. 4.a- Sistema: Si no es troba cap soci, es mostra un avís i es torna al punt de partida, on l'usuari ha de tornar a seleccionar l'opció corresponent en el menú. 7.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: un '1' en el nom del soci) es tornen a demanar fins que siguin vàlides.

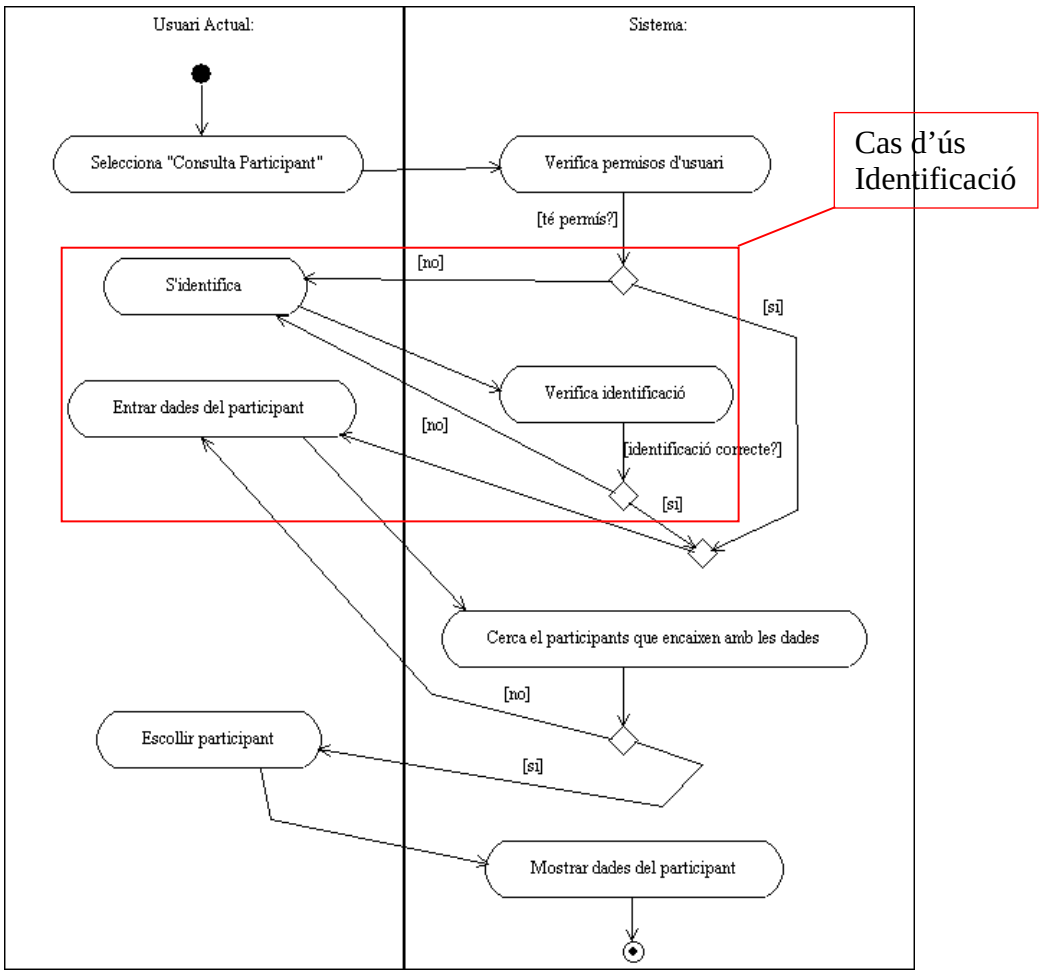


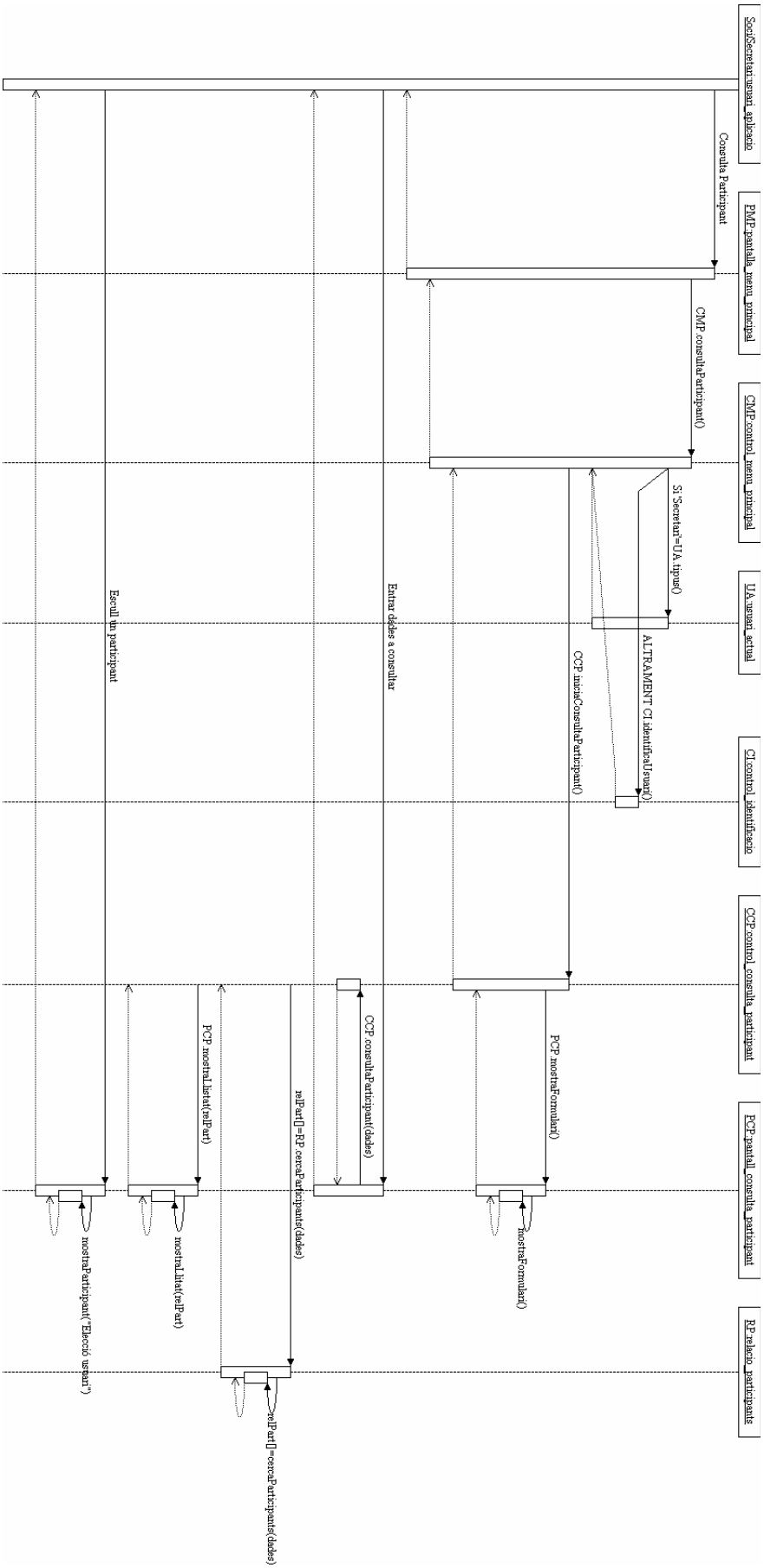
Consultar Soci permet a l'usuari de l'aplicació consultar totes les dades que l'entitat té d'un determinat soci. Quan l'usuari clica sobre aquesta opció en el menú, el sistema comprova que l'usuari estigui autoritzat per fer aquesta operació. Un cop comprovat, li demana alguna dada sobre el soci (Ex: cognom). El sistema llavors busca tots els socis que les dades dels quals "encaixen" amb les dades aportades, i en mostra un llistat. La persona selecciona un soci de la llista i el sistema en mostra totes les dades disponibles: nom, cognoms, etc. A part, el sistema mostra a l'usuari l'opció de demanar la modificació d'una o més dades que són incorrectes.


SocSecretariUsuari
PMP:Pantalla_Menu_Principal
CMP:Control_Menu_Principal
UA:Usuari_Actual
CI:Control_Identificacio
CCS:Control_Consulta_Socs
PCS:Pantalla_Consulta_Socs
RS:Relacio_Socis
CPM:Control_Proposta_Modificacio
Consulta Socs
CMP.consultaSoc()
SI "Secretari"=UA.tipus() ó "Soci"=UA.tipus()
ALTRAMENT CI.identificarUsuari()
<<comment>>
Simulem el cas d'identificació
(Veure a part)
CCS.iniciaConsultaSoc()
Entra dades soci (Dades socis a consultar)
CCS.consultaSoc("Dades socis a consultar")
PCS.mostraFormulari()
mostraFormulari()
relacióSocis[]=RS.cercaSocis("Dades socis a consultar")
relacióSocis[]=cercaSocis("Dades socis a consultar")
PCS.mostrarLlistat(relacióSocis)
mostrarLlistat(relacióSocis)
Escolli un soci (S)
mostraSoci(S)
Emplena el formulari de modificació (Noves dades soci)
PCS.getAmbModificacioDadesSoci(S,"Noves dades soci")
CPM.iniciaAltaPropostaModificacio(S,"Noves dades soci")
<<comment>>
Simula el cas d'us "Alta proposta modificació"
(Veure a part)

✓ Consulta Participant

Cas d'ús: Consulta participant
Objectiu: Conèixer les dades d'un participant
Actors: Secretari
Precondició: -
Flux bàsic: 1- Sistema: El sistema verifica que el rol de l'usuari actual sigui el de Secretari 2- Usuari: Entra una o vàries dades del participant 3- Sistema: Valida lògicament les dades entrades per l'usuari i cerca aquells participants que hi coincideixen. 4- Sistema: Si en troba algun, mostra una llista. 5- Usuari: Selecciona el participant del que vol llegir la informació. Si només en troba un, es mostra automàticament. 6- Sistema: Mostra tota la informació disponible.
Postcondició: S'ha mostrat les propietats del participant seleccionat si algun coincideix amb les paràmetres introduïts.
Flux alternatiu (extensions): 1.a- Sistema: Si el rol de l'usuari actual no és el que correspon a la zona a la que es vol accedir, s'executa el cas d'ús Identificació. 3.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: un '1' en el nom del participant) es tornen a demanar fins que siguin vàlides. 4.a- Sistema: Si no es troba cap participant, es mostra un avís i es torna al punt de partida, on l'usuari ha de tornar a seleccionar l'opció corresponent en el menú.

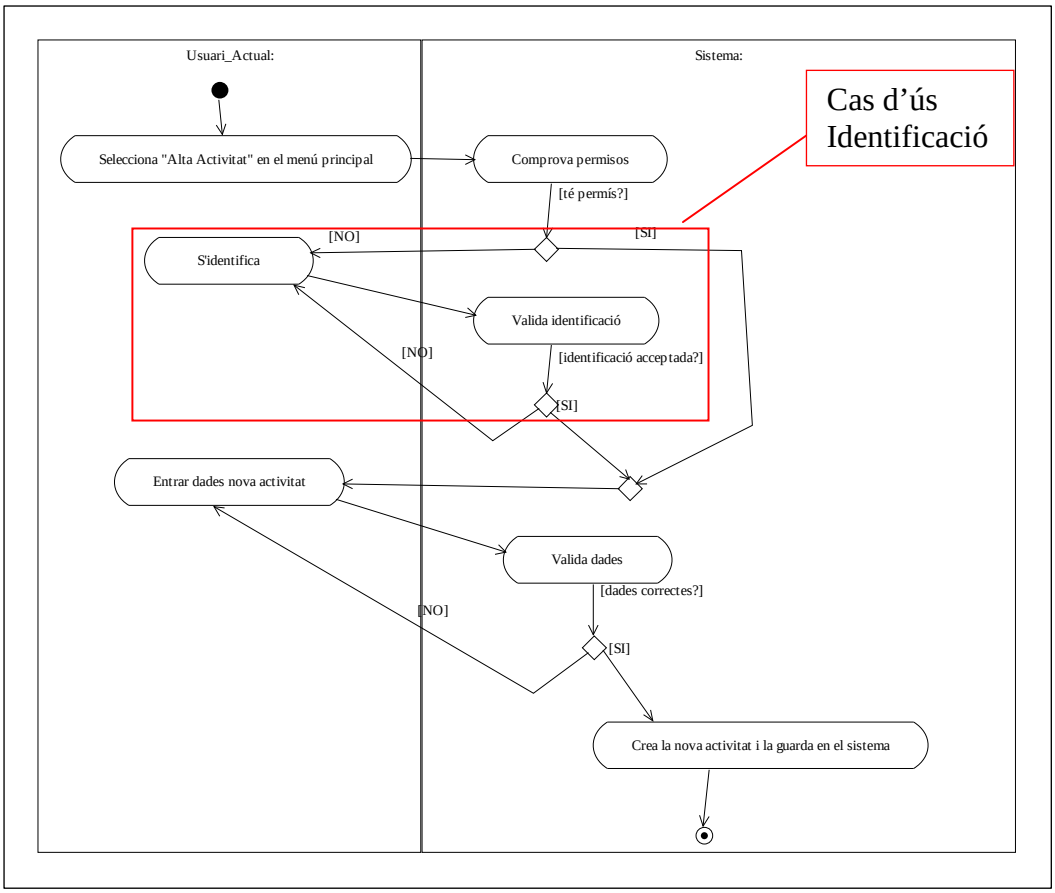


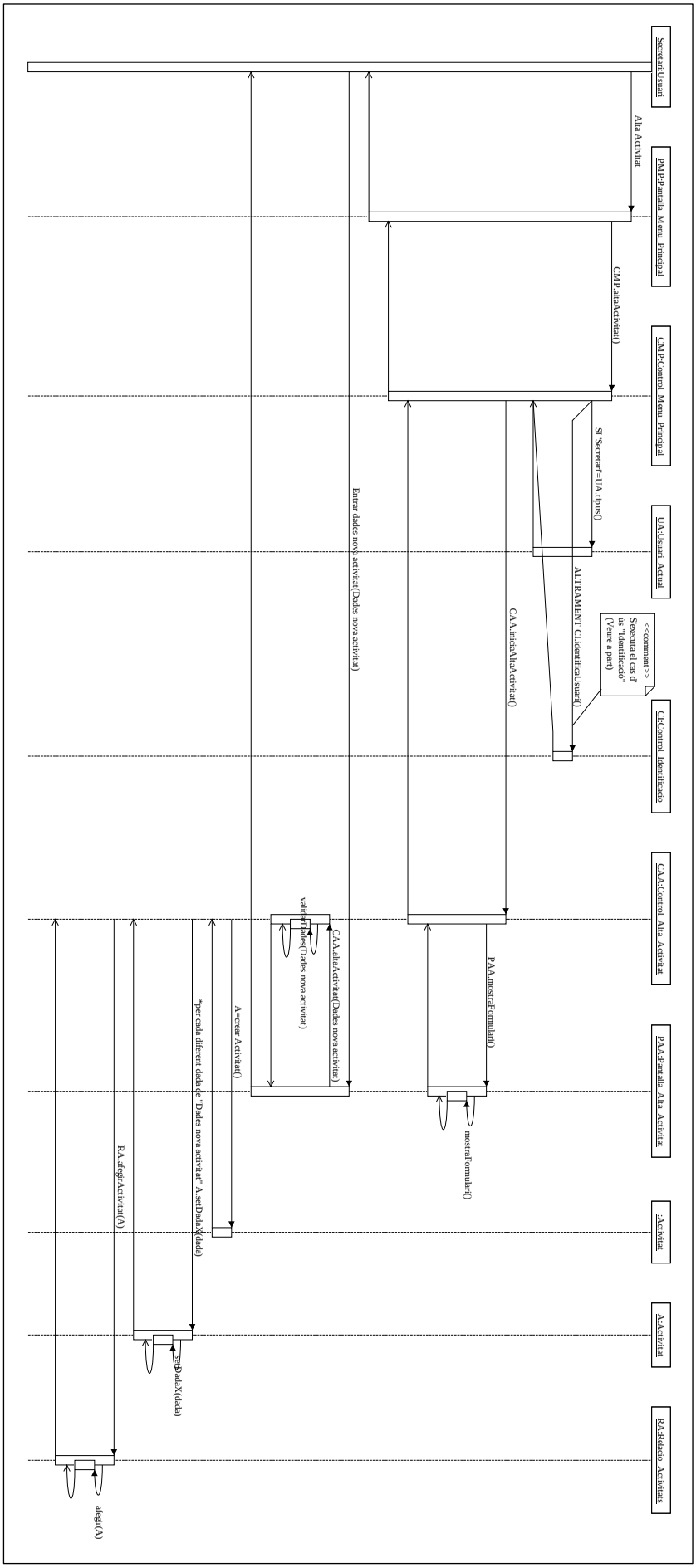

SocisSecretari usuari aplicacio
PMP pantalla menu principal
CMP control menu principal
UA usuari actual
CI control identificacio
CCP control consulta participant
PCP pantall consulta participant
RP relacio participants
Consulta Participant
CMP consultaParticipant()
Si 'Secretari'=UA.tipus()
ALTRAMENT CI.identificaUsuari()
CCP iniciaConsultaParticipant()
Entrar dades a consultar
Escull un participant
PCP mostraFormulari()
mostraFormulari()
CCP consultaParticipant(dades)
relPart[]=RP.cercaParticipants(dades)
PCP mostraLlistat(relPart)
mostraLlistat(relPart)
mostraParticipant("Elecció usuari")
relPart[]=cercaParticipants(dades)

Consultar Participant permet a l'usuari de l'aplicació consultar totes les dades que l'entitat té d'un determinat participant. Quan l'usuari clica sobre aquesta opció en el menú, el sistema comprova que l'usuari estigui autoritzat per fer aquesta operació. Un cop comprovat, li demana alguna dada sobre el participant (Ex: cognom). El sistema llavors busca tots els participants, les dades dels quals "encaixen" amb les dades aportades, i en mostra un llistat. La persona en selecciona un de la llista i el sistema en mostra totes les dades disponibles: nom, cognoms, etc.

- ✓ **Alta Actuació:**

Cas d'ús: Alta Actuació
Objectiu: Crear un nova actuació (o activitat) dins el sistema
Actors: Secretari
Precondició: -
Flux bàsic: 1- Sistema: El sistema verifica que el rol de l'usuari actual sigui el de Secretari 2- Usuari: Entra dades de l'actuació (dia, hora, lloc,...) 3- Sistema: Recull les dades i en valida la seva lògica. 4- Sistema: S'instancia un nou objecte actuació, s'omple amb les dades aportades per l'usuari i s'afegeix al sistema
Postcondició: L'actuació queda accessible dins el sistema
Flux alternatiu (extensions): 1.a- Sistema: Si el rol de l'usuari actual no és el que correspon a la zona a la que es vol accedir, s'executa el cas d'ús Identificació. 3.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: un '1' en un nom de poble) es tornen a demanar fins que siguin vàlides.

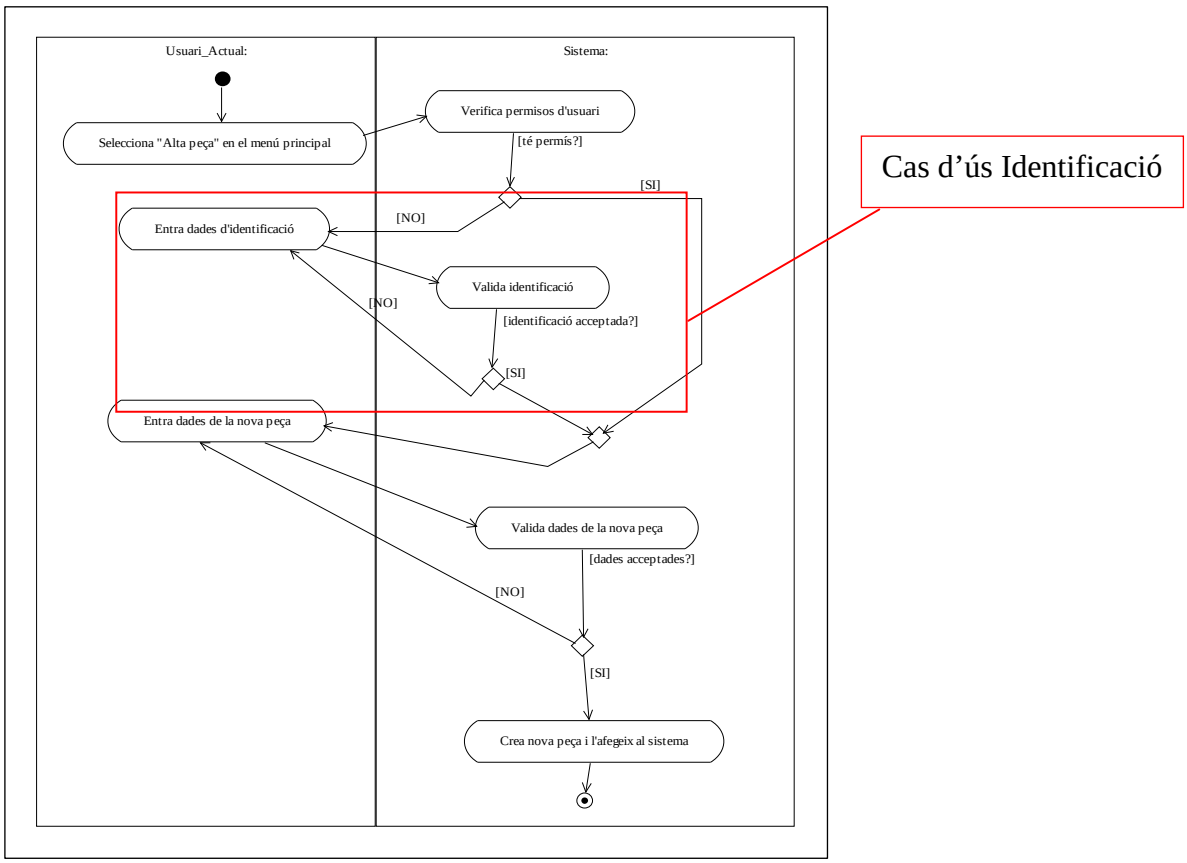


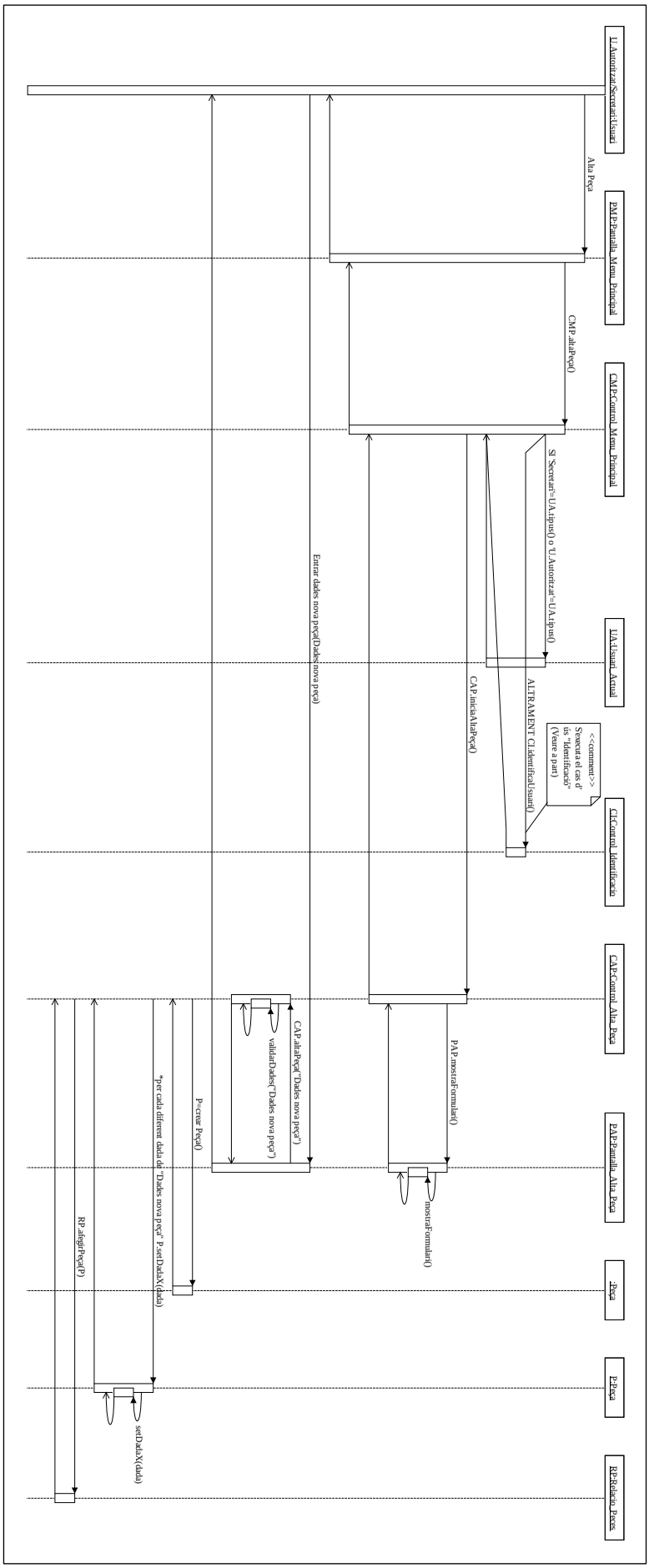

Secretari:Usuari
PMP:Pantalla Menu Principal
CMP:Control Menu Principal
UA:Usuari Actual
CIC:Control Identificacio
CAA:Control Alta Activitat
PAA:Pantalla Alta Activitat
A:Activitat
A:Activitat
RA:Rebecio Activitats
Alta Activitat
CMP.altaActivitat()
Si 'Secretari'=UA.tipus()
<<comment>>
S'executa el cas d'
ús "Identificació"
(Veure a part)
ALTRAMENT CI.identificaUsuari()
Entrar dades nova activitat(Dades nova activitat)
CAA.iniciaAltaActivitat()
PAA.mostrarFormulari()
mostrarFormulari()
validarDades(Dades nova activitat)
CAA.altaActivitat(Dades nova activitat)
A=crear Activitat()
*per cada diferent dada de "Dades nova activitat" A.setDadaX(dada)
setDadaX(dada)
RA.afegir(Activitat(A)
afegir(A)

Alta Actuació permet a l'usuari de l'aplicació donar d'alta una actuació en el sistema. El sistema comprova primer de tot que el rol de l'objecte Usuari Actual sigui el correcte per a executar aquest cas d'ús. Un cop feta aquesta comprovació, demana a l'usuari les dades de l'actuació i tot seguit en valida lògicament les dades, que vol dir que comprova que el valor donat per a cada camp estigui dins d'uns paràmetres establerts (Ex: un nom de poble no pot ser Barc99ona). Un cop fet això, es crea una nova actuació i s'afegeix al sistema.

- ✓ **Alta Peça:**

Cas d'ús: Alta Peça
Objectiu: Crear un nova peça dins el sistema
Actors: Usuari Repertori, Secretari
Precondició: -
Flux bàsic: 1- Sistema: El sistema verifica que el rol de l'usuari actual sigui el de Usuari Autoritzat o Secretari 2- Usuari: Entra dades de la peça (durada, nom,...) 3- Sistema: Recull les dades i en valida la seva lògica. 4- Sistema: S'instancia un nou objecte peça, s'omple amb les dades aportades per l'usuari i s'afegeix finalment al sistema.
Postcondició: La peça queda accessible dins el sistema
Flux alternatiu (extensions): 1.a- Sistema: Si el rol de l'usuari actual no és el que correspon a la zona a la que es vol accedir, s'executa el cas d'ús Identificació. 3.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: una 'a' en la durada d'una peça) es tornen a demanar fins que siguin vàlides.



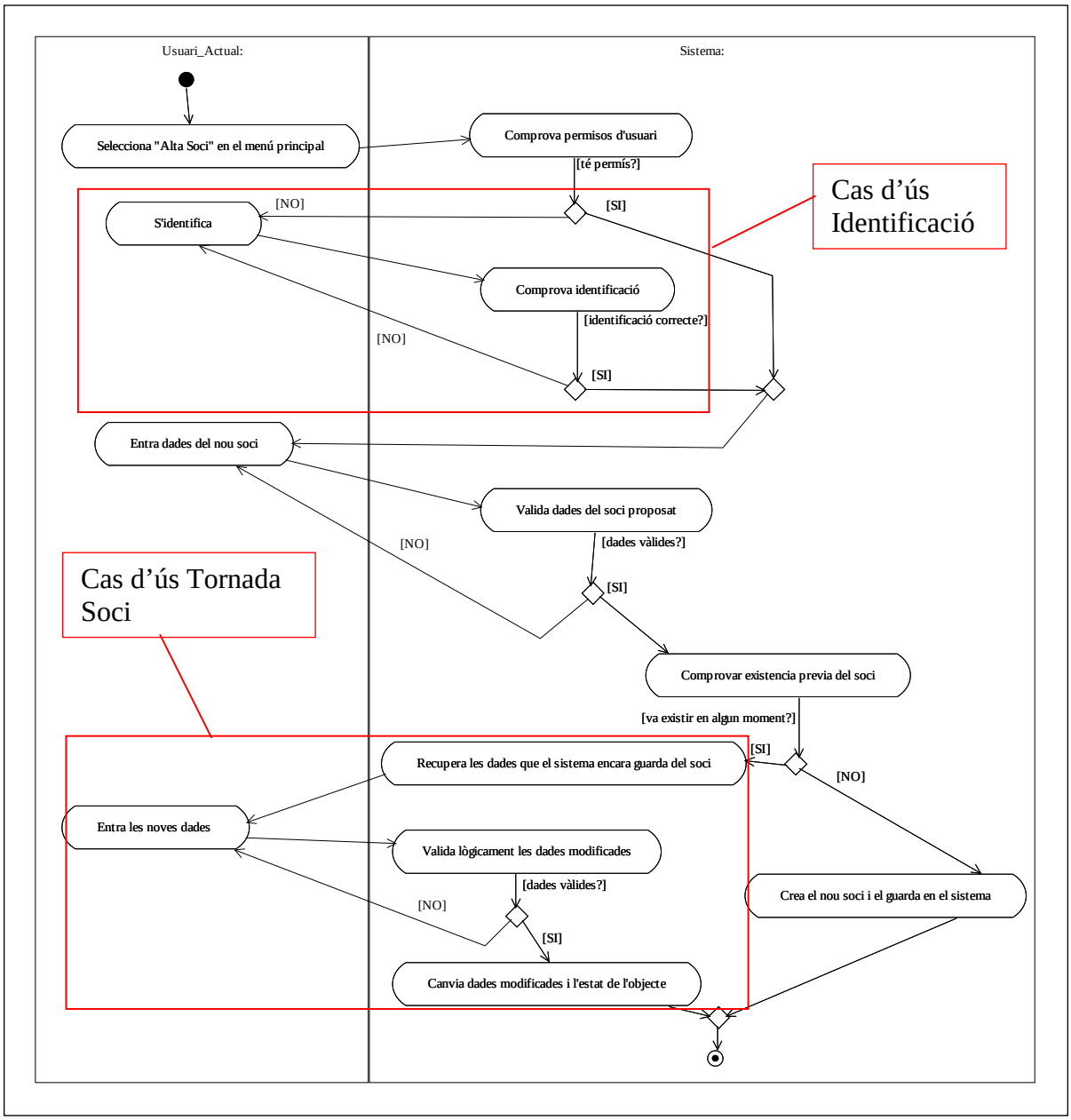

U:Autoritzar/Secretari/Usuari
PM:P.Pantalla_Menu_Principal
CMP:Control_Menu_Principal
UA:Usuari_Actual
CI:Control_Identificacio
CAP:Control_Alta_Peca
PAP:Pantalla_Alta_Peca
:Peca
P:Peca
BP:Robotic_Pieces
Alta Peça
CMP.altaPeça()
Si 'Secretari'=UA.tipus() o 'U.Autoritzar'=UA.tipus()
<<comment>>
S'executa el cas d'
ús "Identificació"
(Veure a part)
ALTRAMENT CI.identificar(Usuari)
Entrar dades nova peça(Dades nova peça)
CAP.iniciaAltaPeça()
CAP.altaPeça("Dades nova peça")
validarDades("Dades nova peça")
PAP.mostrarFormulari()
mostrarFormulari()
P=crear Peça()
"per cada diferent dada de "Dades nova peça" P.setDadaX(dada)
setDadaX(dada)
RP.afegirPeça(P)

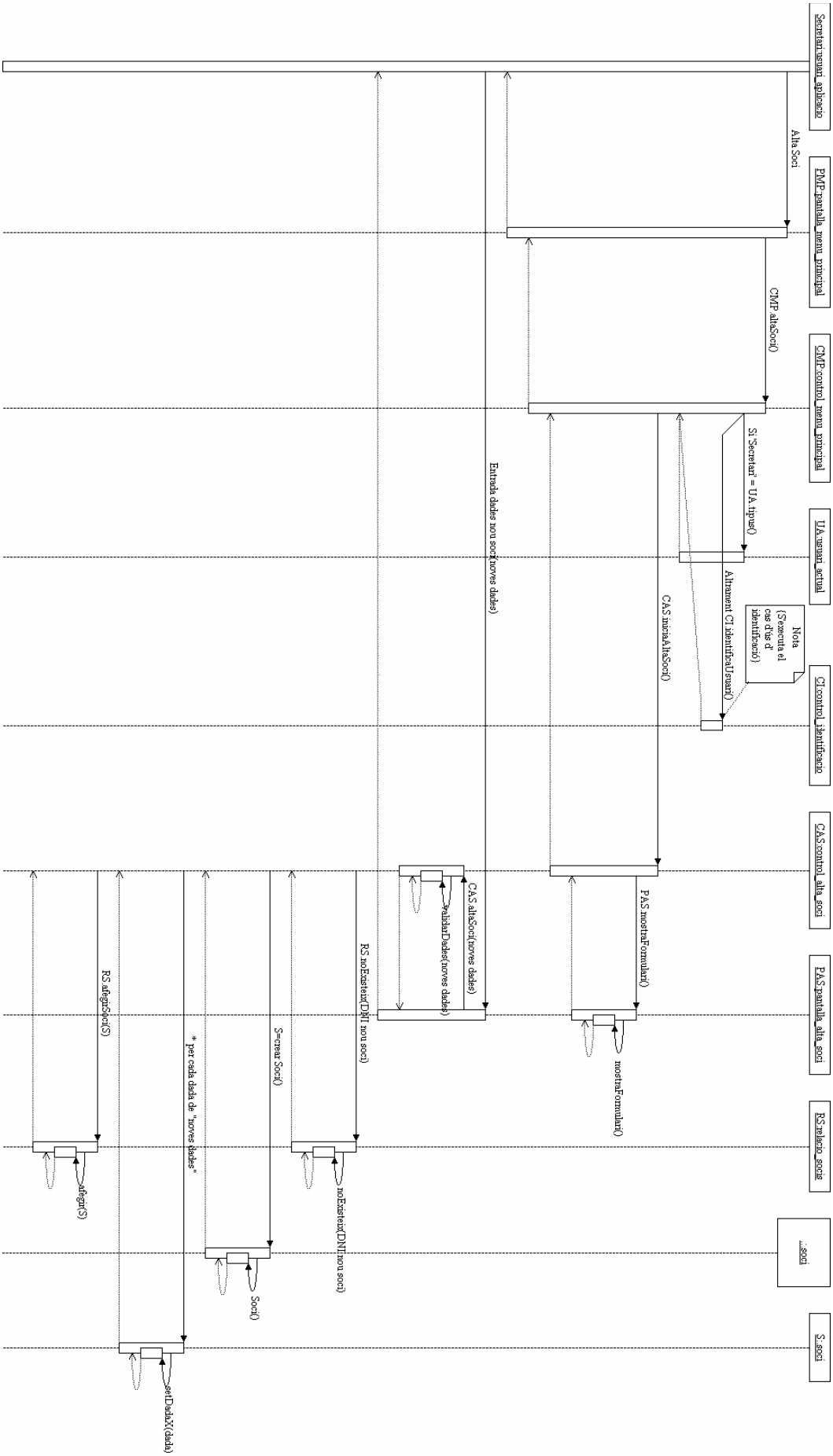
Alta Peça permet a l'usuari de l'aplicació donar d'alta una peça en el sistema. El sistema comprova primer de tot que el rol de l'objecte Usuari Actual sigui el correcte per a executar aquest cas d'ús. Un cop feta aquesta comprovació, demana a l'usuari les dades de la peça i tot seguit en valida lògicament les dades, que vol dir que comprova que el valor donat per a cada camp estigui dins d'uns paràmetres establerts (Ex: un nom de peça no pot ser e777). Un cop fet això, es crea una nova peça i s'afegeix al sistema.

- ✓ **Alta Soci:**

Cas d'ús: Alta Soci
Objectiu: Crear un nou soci dins el sistema
Actors: Usuari Repertori, Secretari
Precondició: -
Flux bàsic: 1- Sistema: El sistema verifica que el rol de l'usuari actual sigui el de Secretari 2- Usuari: Entra dades del soci (dni,nom,cognoms,...) 3- Sistema: Recull les dades i en valida la seva lògica. 4- Sistema: Es validen les dades amb els altres socis que ja té el sistema. Si aquestes no creen conflicte amb algun element ja present es crea el soci amb les dades entrades i s'afegeix al sistema.
Postcondició: El soci queda accessible dins el sistema
Flux alternatiu (extensions): 1.a- Sistema: Si el rol de l'usuari actual no és el que correspon a la zona a la que es vol accedir, s'executa el cas d'ús Identificació. 3.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: una 'a' en un telèfon) es tornen a demanar fins que siguin vàlides. 4.a- Sistema: En cas que l'identificador ja existís en el sistema, s'informaria a l'usuari que ja existeix un soci amb aquest valor de DNI i s'avortaria l'operació. Si es dona el cas que el soci que es vol donar d'alta ja ho va ser en un altre moment, s'executa el cas d'us Tornada Soci.

Alta Soci permet a l'usuari de l'aplicació donar d'alta un soci en el sistema. El sistema comprova primer de tot que el rol de l'objecte Usuari Actual sigui el correcte per a executar aquest cas d'ús. Un cop feta aquesta comprovació, demana a l'usuari les dades del soci i tot seguit en valida lògicament les dades, que vol dir que comprova que el valor donat per a cada camp estigui dins d'uns paràmetres establerts (Ex: un telèfon no pot ser 972 34k 34k). Tot seguit, es comprova que el soci no estigui ja en el sistema, normalment, comprovant l'existència de l'identificador personal del nou soci. Un cop fet això, es crea un nou soci i s'afegeix al sistema. Es podria donar el cas que les dades introduïdes coincideixin amb les d'un soci que vam donar de baixa. En aquest cas, passarem al cas d'ús de Tornada Soci.


Usuari_Actual:
Sistema:
Selecciona "Alta Soci" en el menú principal
Comprova permisos d'usuari
[té permís?]
[NO]
[SI]
S'identifica
Comprova identificació
[identificació correcte?]
[NO]
[SI]
Cas d'ús Identificació
Entra dades del nou soci
Valida dades del soci proposat
[dades vàlides?]
[NO]
[SI]
Cas d'ús Tornada Soci
Comprovar existencia previa del soci
[va existir en algun moment?]
[SI]
[NO]
Recupera les dades que el sistema encara guarda del soci
Entra les noves dades
Valida lògicament les dades modificades
[dades vàlides?]
[NO]
[SI]
Canvia dades modificades i l'estat de l'objecte
Crea el nou soci i el guarda en el sistema

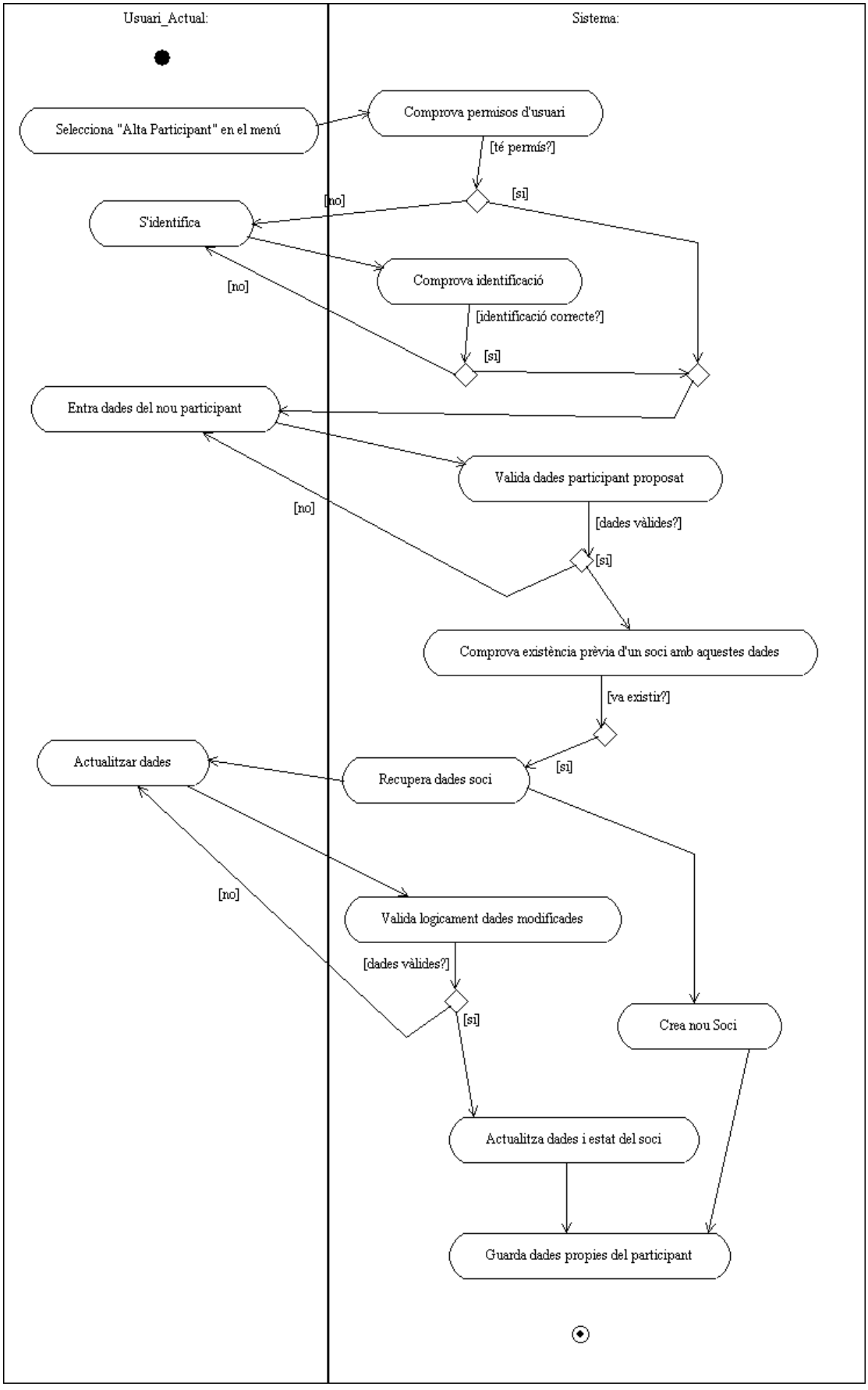

Secretari:usuari_aplicacio
PMP:pantalla_menu_principal
CMP:control_menu_principal
UA:usuari_actual
CI:control_identificacio
CAS:control_alta_socis
PAS:pantalla_alta_socis
RS:relacio_socis
:socis
S:socis
:Alta Soci
CMP:altaSoci()
Si 'Secretari' = UA.tipus()
Nota
(S'executa el cas d'ús d'identificació)
Altrament CI: identificaUsuari()
Entrada dades nou soci(noves dades)
CAS:iniciaAltaSoci()
CAS:altaSoci(noves dades)
validarDades(noves dades)
PAS:mostraFormulari()
mostraFormulari()
RS:noExisteix(DNI nou soci)
noExisteix(DNI:nou soci)
S=crear Soci()
Soci()
* per cada dada de "noves dades"
setDadaX(dada)
RS:afegirSoci(S)
afegir(S)

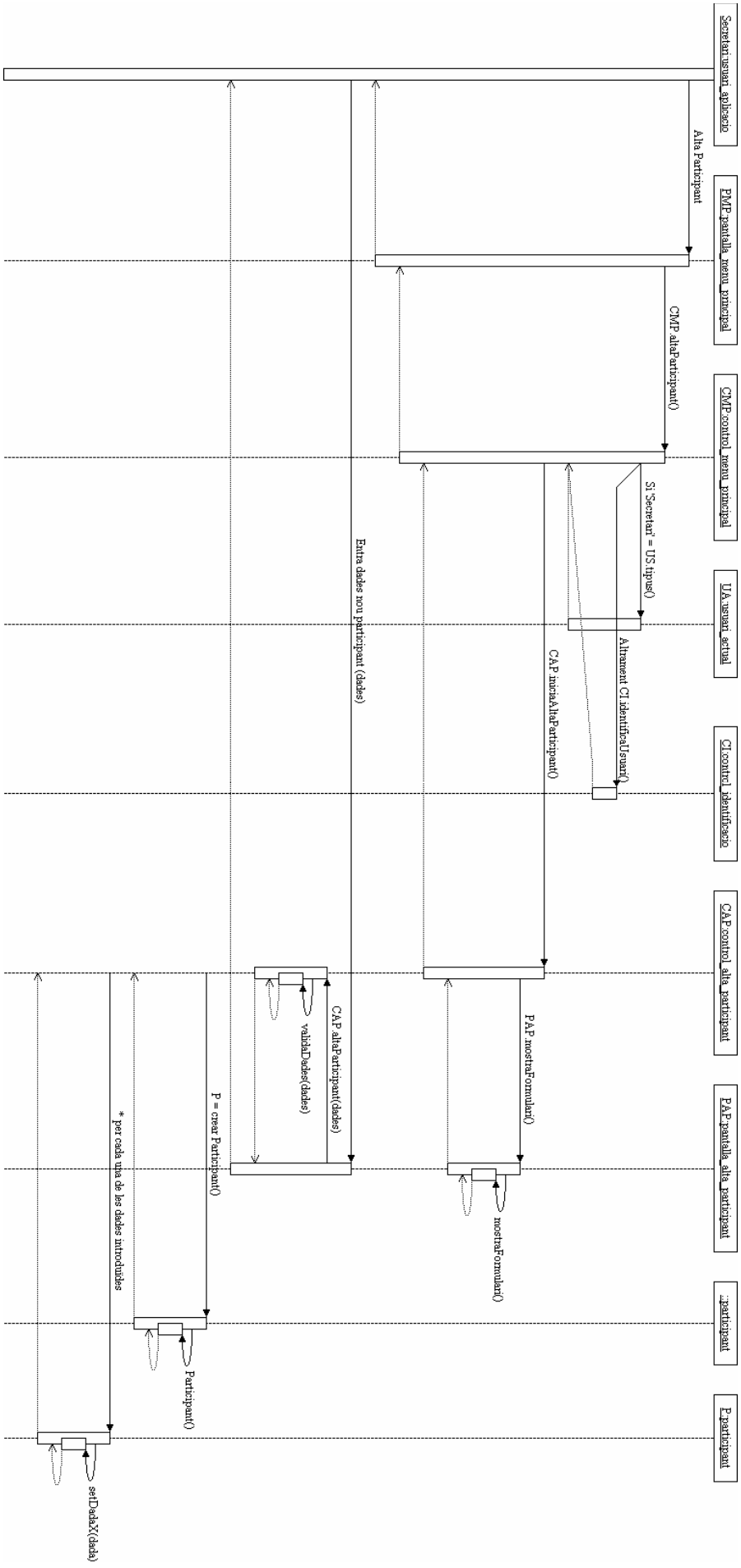
- ✓ **Alta Participant**

Cas d'ús: Alta Participant
Objectiu: Crear un nou participant dins el sistema
Actors: Secretari
Precondició: -
Flux bàsic: 1- Sistema: El sistema verifica que el rol de l'usuari actual sigui el de Secretari 2- Usuari: Entra dades del participant (dni,nom,cognoms,...) 3- Sistema: Recull les dades i en valida la seva lògica. 4- Sistema: Es validen les dades amb els altres participants que ja té el sistema. A més, en aquest cas, com que un participant ha de ser també soci de l'entitat, comprovarem si hi ha algun soci amb les dades introduïdes. Si ja tenim el soci al sistema, simplement crearem l'objecte participant amb les seves dades úniques, com ara la data d'alta com a participant. Si no hi és, haurem de passar al cas d'ús de "Alta Soci".
Postcondició: El participant queda accessible dins el sistema
Flux alternatiu (extensions): 1.a- Sistema: Si el rol de l'usuari actual no és el que correspon a la zona a la que es vol accedir, s'executa el cas d'ús Identificació. 3.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: una 'a' en un telèfon) es tornen a demanar fins que siguin vàlides. 4.a- Sistema: En cas que l'identificador ja existís en el sistema, s'informaria a l'usuari que ja existeix un participant amb aquestes dades. Si es dona el cas que les dades del nou participant coincideixen amb les d'un soci donat de baixa, s'executa el cas d'us Tornada Soci.

Alta Participant permet a l'usuari de l'aplicació donar d'alta un participant en el sistema. El sistema comprova primer de tot que el rol de l'objecte Usuari Actual sigui el correcte per a executar aquest cas d'ús. Un cop feta aquesta comprovació, demana a l'usuari les dades del participant i tot seguit en valida lògicament les dades, que vol dir que comprova que el valor donat per a cada camp estigui dins d'uns paràmetres establerts (Ex: un telèfon no pot ser 972 34k 34k). Tot seguit, es comprova que les dades introduïdes no corresponguin a la d'un soci ja existent, ja que tot participant està relacionat a un soci, ningú pot ser participant sense ser soci. Un cop fet això, es crea un nou soci i s'afegeix al sistema. Tot seguit es guarden les dades corresponents al participant, com ara la data d'alta com a tal.

Es podria donar el cas que les dades introduïdes coincideixin amb les d'un soci que vam donar de baixa. En aquest cas, passarem al cas d'ús de Tornada Soci.

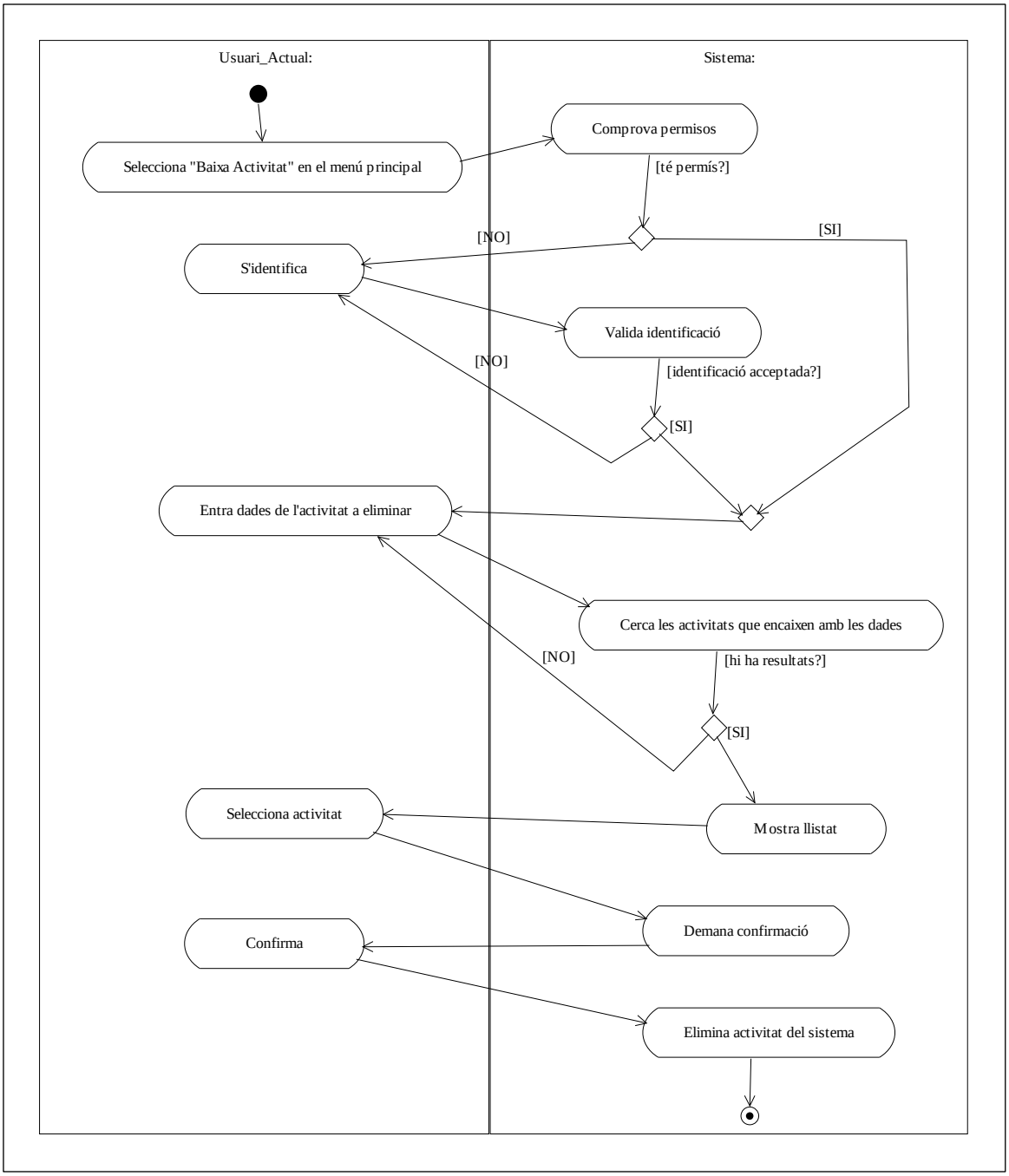

Usuari_Actual:
Sistema:
Selecciona "Alta Participant" en el menú
Comprova permisos d'usuari
[té permís?]
[no]
[si]
S'identifica
Comprova identificació
[no]
[identificació correcte?]
[si]
Entra dades del nou participant
Valida dades participant proposat
[no]
[dades vàlides?]
[si]
Comprova existència prèvia d'un soci amb aquestes dades
[va existir?]
Actualitzar dades
Recupera dades soci
[si]
[no]
Valida logicament dades modificades
[dades vàlides?]
[si]
Crea nou Soci
Actualitza dades i estat del soci
Guarda dades propies del participant

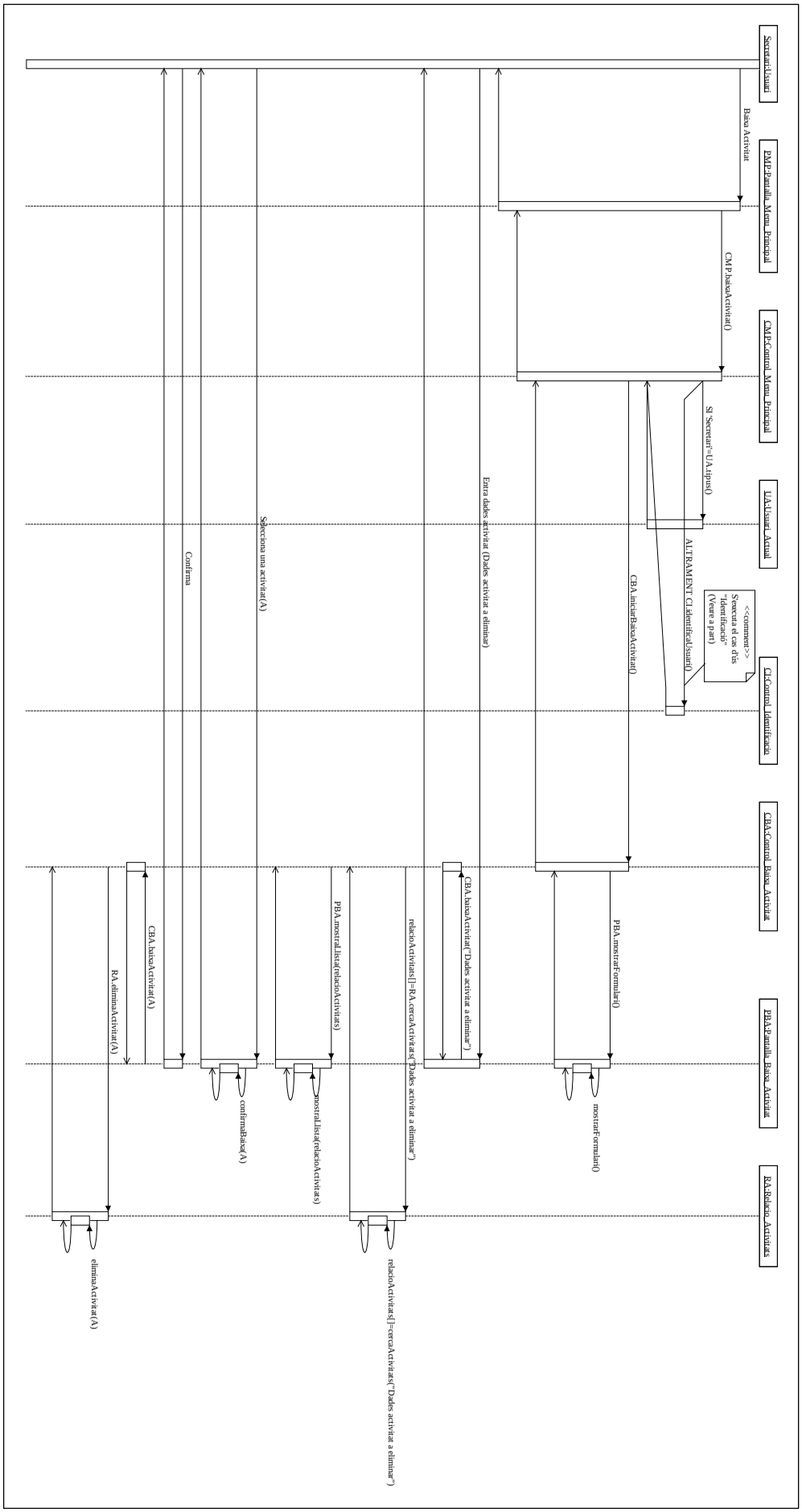

Secretari:usuari_aplicacio
PMP:pantalla_menu_principal
CMP:control_menu_principal
UA:usuari_actual
CI:control_identificacio
CAP:control_alta_participant
PAP:pantalla_alta_participant
:participant
P:participant
Alta Participant
CMP.altaParticipant()
Si 'Secretari' = US.tipus()
Altrament CI.identificaUsuari()
CAP.iniciaAltaParticipant()
Entra dades nou participant (dades)
PAP.mostraFormulari()
mostraFormulari()
CAP.altaParticipant(dades)
validaDades(dades)
P = crear Participant()
* per cada una de les dades introduides
Participant()
setDadaX(dada)

- ✓ **Baixa Actuació:**

Cas d'ús: Baixa Actuació
Objectiu: Eliminar una actuació del sistema
Actors: Secretari
Precondició: -
Flux bàsic: 1- Sistema: El sistema verifica que el rol de l'usuari actual sigui el de Secretari 2- Usuari: Entra una o més dades de l'actuació 3- Sistema: Valida les dades entrades i cerca totes aquelles actuacions que hi coincideixen. 4- Sistema: Si se'n troba alguna, es mostra una llista. 5- Usuari: Selecciona aquella que vol eliminar. Si només se'n troba una, es mostra directament. 6- Sistema : Demana confirmació de baixa. 7- Usuari: Confirma baixa. 8- Sistema: Elimina l'actuació del sistema de forma permanent.
Postcondició: L'actuació s'ha eliminat del sistema correctament
Flux alternatiu (extensions): 1.a- Sistema: Si el rol de l'usuari actual no és el que correspon a la zona a la que es vol accedir, s'executa el cas d'ús Identificació. 3.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: un '1' en un nom de poble) es tornen a demanar fins que siguin vàlides. 4.a- Sistema: En cas de no trobar cap actuació, s'informa a l'usuari i es torna al punt inicial, on l'usuari ha de tornar a seleccionar l'opció corresponent en el menú principal.

Baixa Actuació ens permet eliminar una actuació del sistema de forma definitiva. La idea és que si l'entitat programa una actuació i resulta que aquesta no es du a terme per el motiu que sigui, no té cap sentit guardar-ne detalls.
El primer que es fa és comprovar que l'Usuari Actual tingui els permisos corresponents. Tot seguit, l'usuari entra alguna dada de l'actuació a eliminar, el sistema busca les actuacions que coincideixen amb les dades entrades i en mostra una llista. Acte seguit, l'usuari en selecciona una i el sistema demana confirmació per eliminar-la. Així que l'usuari confirma l'eliminació de l'objecte, el sistema l'elimina de manera definitiva.

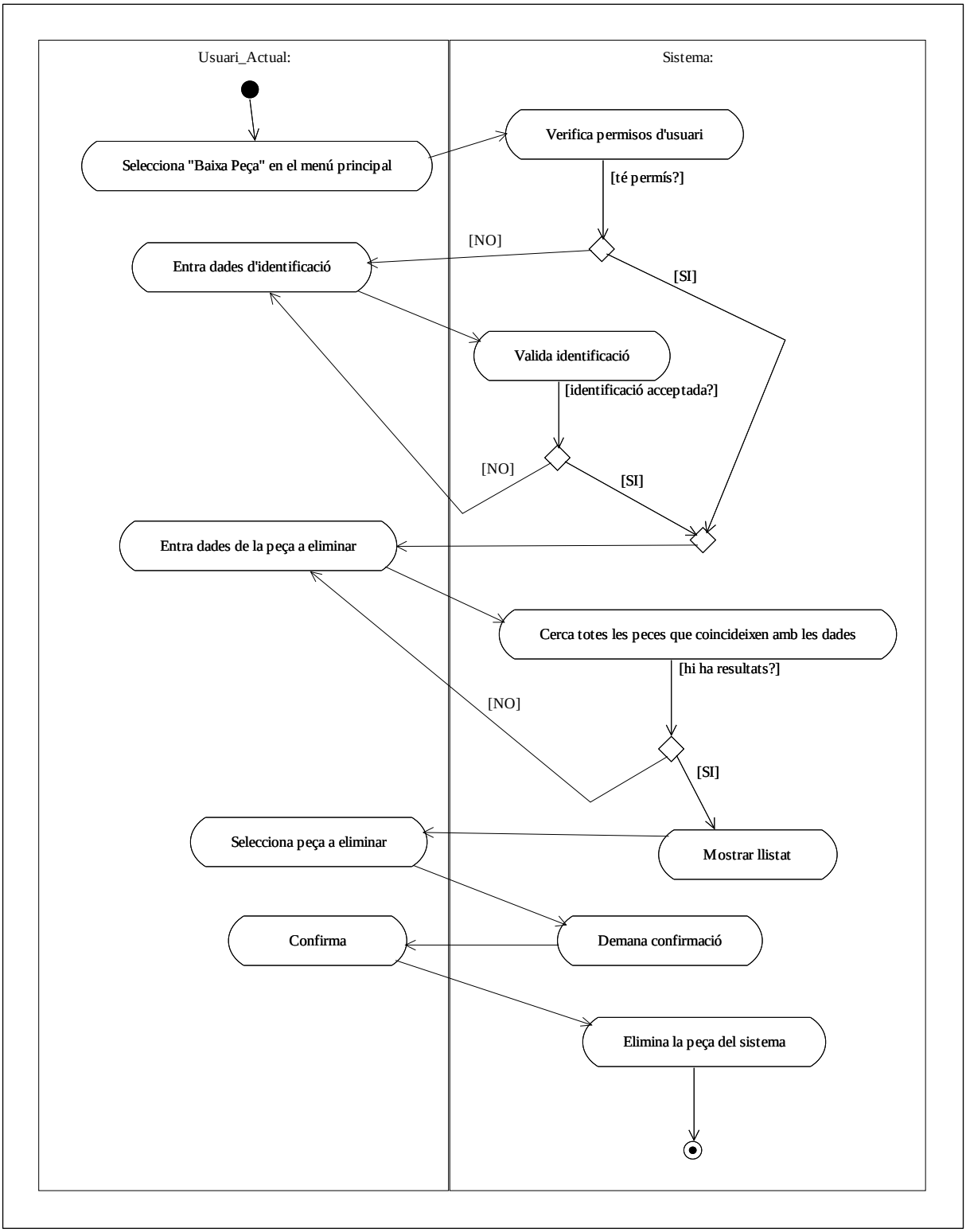

Usuari_Actual:
Sistema:
Selecciona "Baixa Activitat" en el menú principal
Comprova permisos
[té permís?]
[NO]
[SI]
S'identifica
Valida identificació
[NO]
[identificació acceptada?]
[SI]
Entra dades de l'activitat a eliminar
Cerca les activitats que encaixen amb les dades
[NO]
[hi ha resultats?]
[SI]
Selecciona activitat
Mostra llistat
Demana confirmació
Confirma
Elimina activitat del sistema

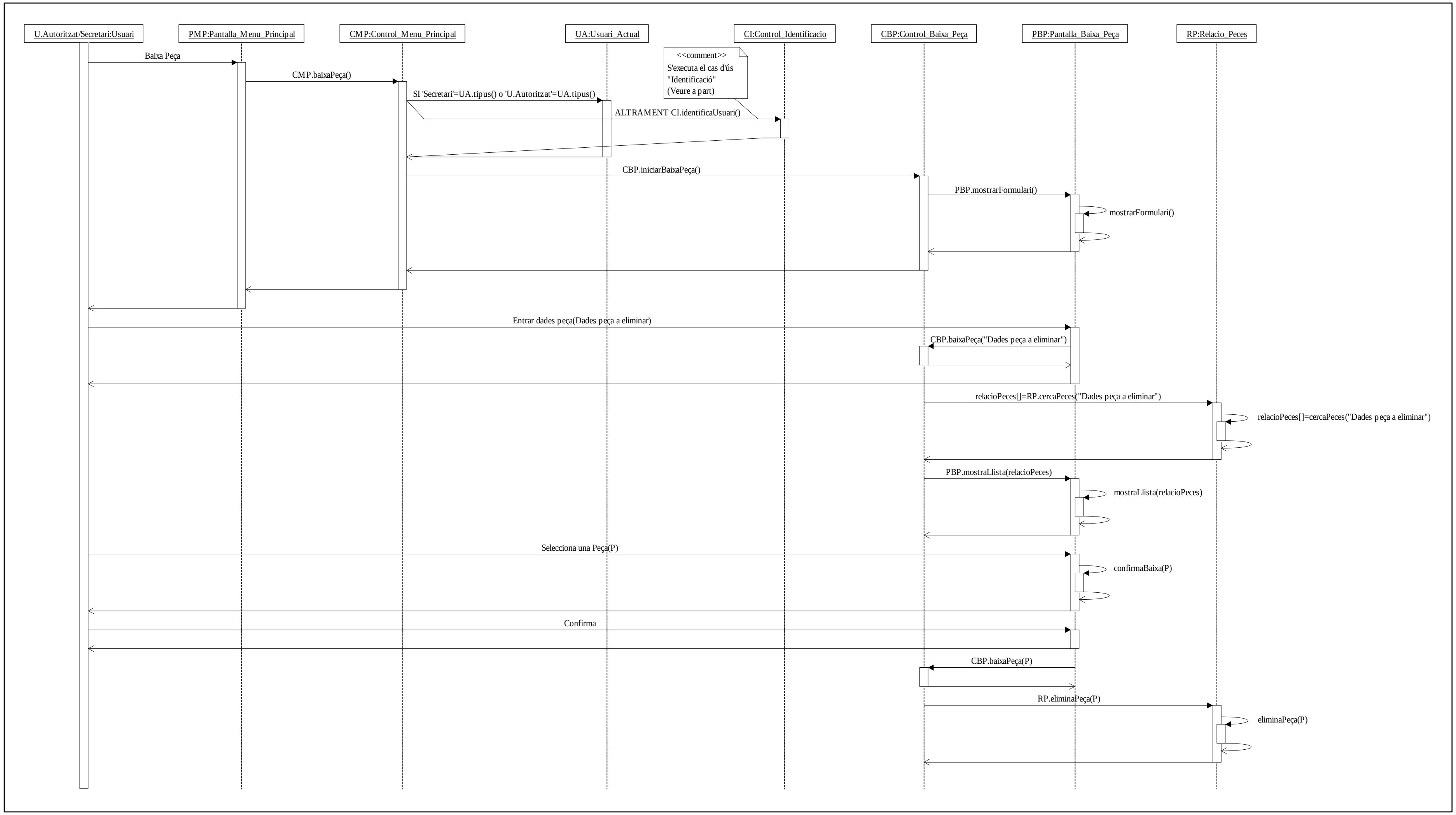

Secretari:Usuari
PMP:Pantalla_Menu_Principal
CMP:Control_Menu_Principal
UA:Usuari_Actual
CI:Control_Identificacio
CBA:Control_Baixa_Activitat
PBA:Pantalla_Baixa_Activitat
RA:Relacio_Activitats
Baixa Activitat
CMP.baixaActivitat()
Si 'Secretari'=UA.tipus()
ALTRAMENT CI.identifical(usuari)
<<comment>>
Sinocurra el cas d'ús "Identificació" (Veure a part)
CBA.iniciarBaixaActivitat()
PBA.mostrarFormulari()
mostrarFormulari()
Entra dades activitat (Dades activitat a eliminar)
CBA.baixaActivitat("Dades activitat a eliminar")
Selecciona una activitat(A)
PBA.mostral.llista(relacioActivitats)
mostral.llista(relacioActivitats)
relacioActivitats[]=RA.cercaActivitats("Dades activitat a eliminar")
relacioActivitats[]=cercaActivitats("Dades activitat a eliminar")
Confirma
CBA.baixaActivitat(A)
confirmaBaixa(A)
RA.eliminaActivitat(A)
eliminaActivitat(A)

- ✓ **Baixa Peça:**

Cas d'ús: Baixa Peça
Objectiu: Eliminar una peça del sistema
Actors: Usuari Repertori, Secretari
Precondició: -
Flux bàsic: 1- Sistema: El sistema verifica que el rol de l'usuari actual sigui el de Usuari Autoritzat o Secretari 2- Usuari: Entra una o més dades de la peça 3- Sistema: Valida les dades entrades i cerca totes aquelles peces que hi coincideixen. 4- Sistema: Si se'n troba alguna, es mostra una llista. 5- Usuari: Selecciona aquella que vol eliminar. Si només se'n troba una, es mostra directament. 6- Sistema : Demana confirmació de baixa. 7- Usuari: Confirma baixa. 8- Sistema: Elimina la peça del sistema de forma permanent.
Postcondició: La peça s'ha eliminat del sistema correctament
Flux alternatiu (extensions): 1.a- Sistema: Si el rol de l'usuari actual no és el que correspon a la zona a la que es vol accedir, s'executa el cas d'ús Identificació. 3.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: un '1' en un nom de poble) es tornen a demanar fins que siguin vàlides 4.a- Sistema: En cas de no trobar cap peça, s'informa a l'usuari i es torna al punt inicial, on l'usuari ha de seleccionar l'opció corresponent en el menú principal.

Baixa Peça ens permet eliminar una peça del sistema de forma definitiva.
El primer que es fa és comprovar que l'Usuari Actual tingui els permisos corresponents. Tot seguit, l'usuari entra alguna dada de la peça a eliminar, el sistema busca les peces que coincideixen amb les dades entrades i en mostra una llista. Acte seguit, l'usuari en selecciona una i el sistema demana confirmació per eliminar-la. Així que l'usuari confirma l'eliminació de l'objecte, el sistema l'elimina de manera definitiva.


Usuari_Actual:
Sistema:
Selecciona "Baixa Peça" en el menú principal
Verifica permisos d'usuari
[té permís?]
[NO]
[SI]
Entra dades d'identificació
Valida identificació
[identificació acceptada?]
[NO]
[SI]
Entra dades de la peça a eliminar
Cerca totes les peces que coincideixen amb les dades
[hi ha resultats?]
[NO]
[SI]
Selecciona peça a eliminar
Mostrar llistat
Confirma
Demana confirmació
Elimina la peça del sistema

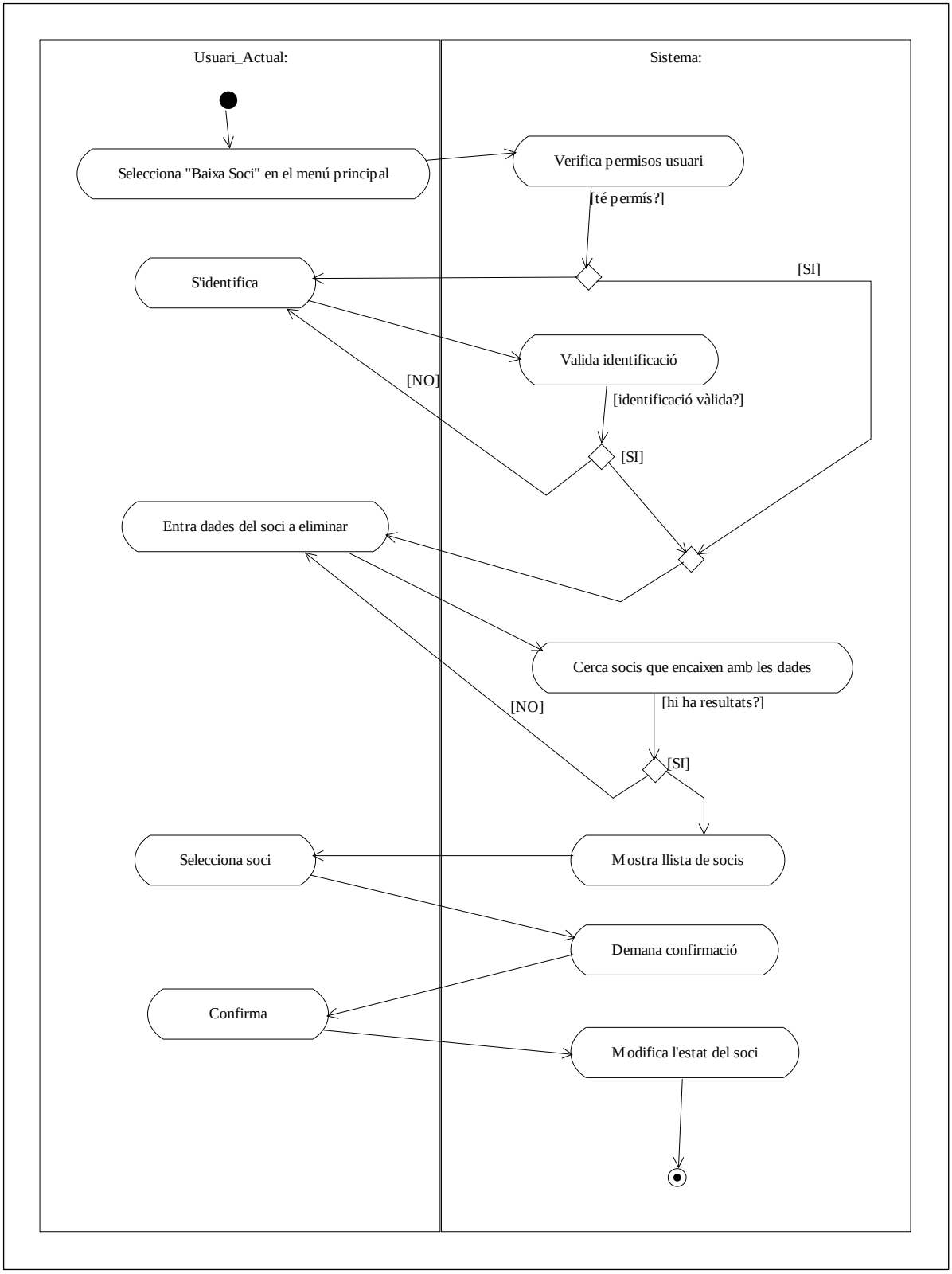

U.Autoritzat/Secretari:Usuari
PMP:Pantalla_Menu_Principal
CMP:Control_Menu_Principal
UA:Usuari_Actual
CI:Control_Identificacio
CBP:Control_Baixa_Peça
PBP:Pantalla_Baixa_Peça
RP:Relacio_Peces
<<comment>>
S'executa el cas d'ús
"Identificació"
(Veure a part)
Baixa Peça
CMP.baixaPeça()
SI 'Secretari'=UA.tipus() o 'U.Autoritzat'=UA.tipus()
ALTRAMENT CI.identificaUsuari()
CBP.iniciarBaixaPeça()
PBP.mostrarFormulari()
mostrarFormulari()
Entrar dades peça(Dades peça a eliminar)
CBP.baixaPeça("Dades peça a eliminar")
relacioPeces[]=RP.cercaPeces("Dades peça a eliminar")
relacioPeces[]=cercaPeces("Dades peça a eliminar")
PBP.mostraLlista(relacioPeces)
mostraLlista(relacioPeces)
Selecciona una Peça(P)
confirmaBaixa(P)
Confirma
CBP.baixaPeça(P)
RP.eliminaPeça(P)
eliminaPeça(P)

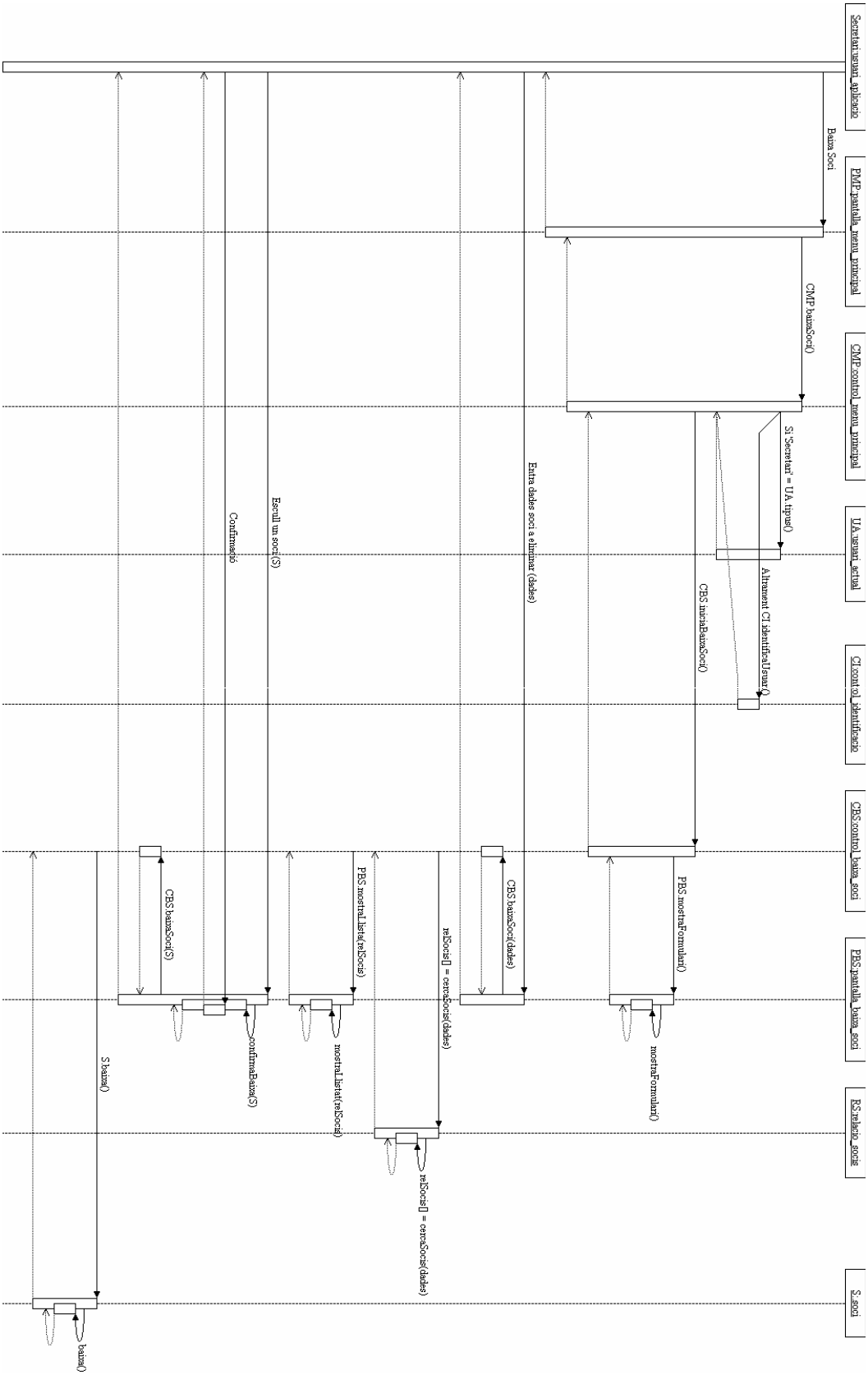
- ✓ **Baixa Soci:**

Cas d'ús: Baixa soci
Objectiu: Canviar l'estat d'un soci a Ex_soci
Actors: Usuari Soci, Secretari
Precondició: -
Flux bàsic: 1- Sistema: El sistema verifica que el rol de l'usuari actual sigui el de Secretari 2- Usuari: Entra una o vàries dades del soci 3- Sistema: Valida les dades entrades i cerca en el sistema aquells socis que hi coincideixen. 4- Sistema: Si en troba algun, mostra una llista. 5- Usuari: Selecciona el que es desitja donar de baixar com a soci. Si només n'hi ha un, es mostra automàticament. 6- Sistema : Demana confirmació de baixa. 7- Usuari: Confirma baixa. 8- Sistema: Actualitza l'estat del soci i l'assigna com a No Actiu.
Postcondició: El soci ha passat a ser ex_soci
Flux alternatiu (extensions): 1.a- Sistema: Si el rol de l'usuari actual no és el que correspon a la zona a la que es vol accedir, s'executa el cas d'ús Identificació. 3.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: un '1' en un cognom) es tornen a demanar fins que siguin vàlides 4.a- Sistema: En cas de no trobar cap soci, s'informa a l'usuari i es torna al punt inicial, on l'usuari ha de seleccionar l'opció corresponent en el menú principal.

Baixa Soci ens permet donar de baixa un soci del sistema. A diferencia dels altres casos, l'objecte no s'elimina de manera definitiva, sinó que se'n canvia l'estat, passa a ser No Actiu. Els motius poden ser diversos i variar depenent de l'entitat, però podem pensar, per exemple, que la seva supressió definitiva podria provocar una inconsistència en el sistema. Un altre exemple per a dur a terme aquesta acció seria que, tal vegada, la persona que ara desitja deixar l'entitat podria tornar a formar-ne part en el futur. En aquest possible escenari, no tindria cap sentit crear un nou soci, que possiblement tindria un nou numero de soci (o el que sigui que li assignem per identificar-lo de forma única) i de fet, seria la mateixa persona.

El primer que es fa és comprovar que l'Usuari Actual tingui els permisos corresponents. Tot seguit, l'usuari entra alguna dada del soci a eliminar, el sistema busca els socis que coincideixen amb les dades entrades i en mostra una llista. Acte seguit, l'usuari selecciona un soci i el sistema demana confirmació per eliminar-lo. Així que l'usuari confirma l'eliminació de l'objecte, el sistema en canvia l'estat.


Usuari_Actual:
Sistema:
Selecciona "Baixa Soci" en el menú principal
Verifica permisos usuari
[té permís?]
S'identifica
[SI]
Valida identificació
[NO]
[identificació vàlida?]
[SI]
Entra dades del soci a eliminar
Cerca socis que encaixen amb les dades
[NO]
[hi ha resultats?]
[SI]
Selecciona soci
Mostra llista de socis
Demana confirmació
Confirma
Modifica l'estat del soci

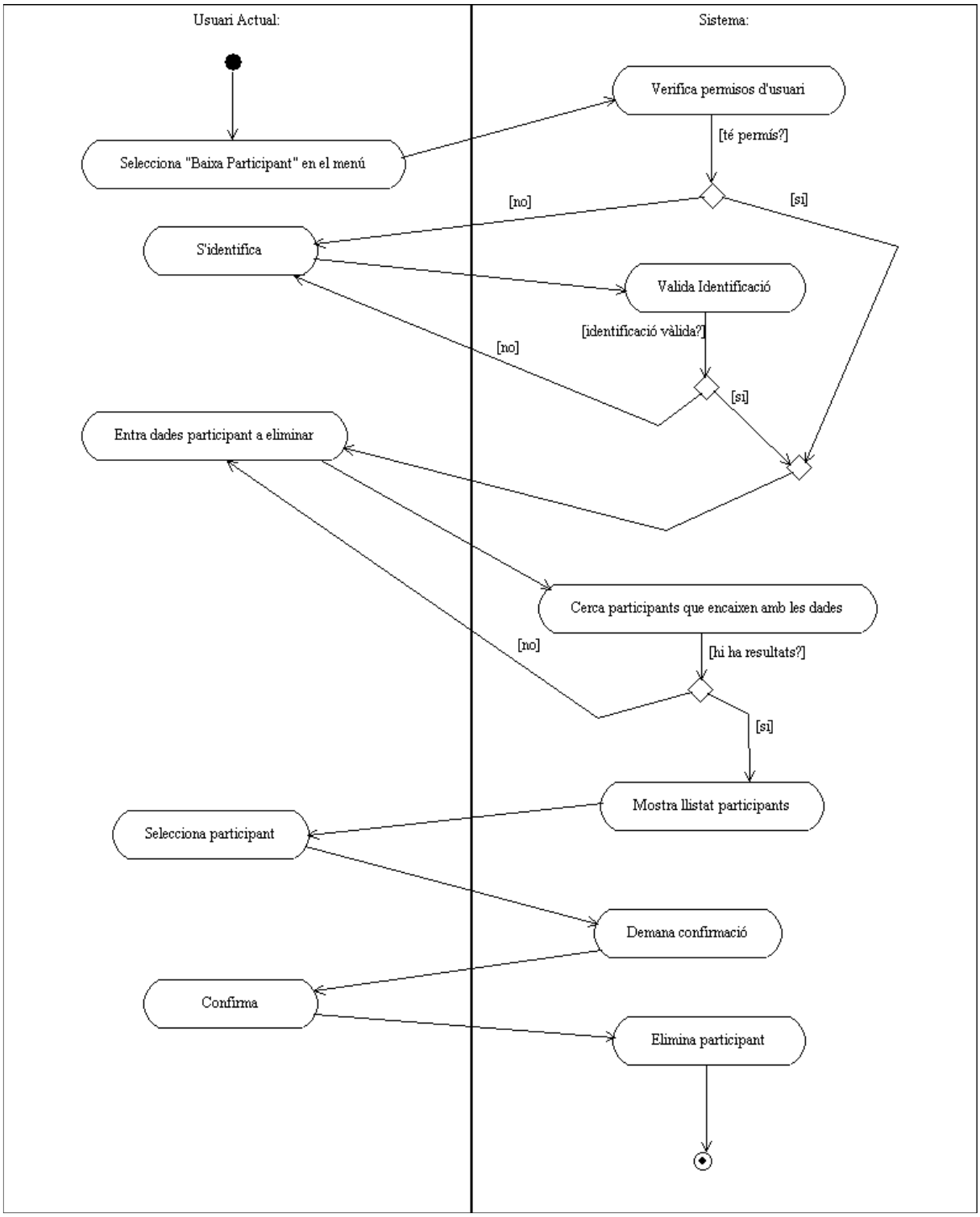

Secretari usuari_aplicacio
PMP:pantalla_menu_principal
CMP:control_menu_principal
UA:usuari_actual
CI:control_identificacio
CBS:control_baixa_socis
PBS:pantalla_baixa_socis
ES:arbovio_socis
S:_socis
Baixa Socis
CMP baixaSocis()
Si 'Secretari' = UA.tipus()
A.tipusent CI.identificaUsuari()
CBS iniciaBaixaSocis()
PBS mostraFormulari()
mostraFormulari()
Entra dades socis a eliminar (dades)
CBS baixaSocis(dades)
reSocis[] = cercaSocis(dades)
reSocis[] = cercaSocis(dades)
PBS mostraListat(reSocis)
mostraListat(reSocis)
Escull un soci(S)
Confirmació
CBS baixaSocis(S)
confirmaBaixa(S)
S'baixa()
baixa()

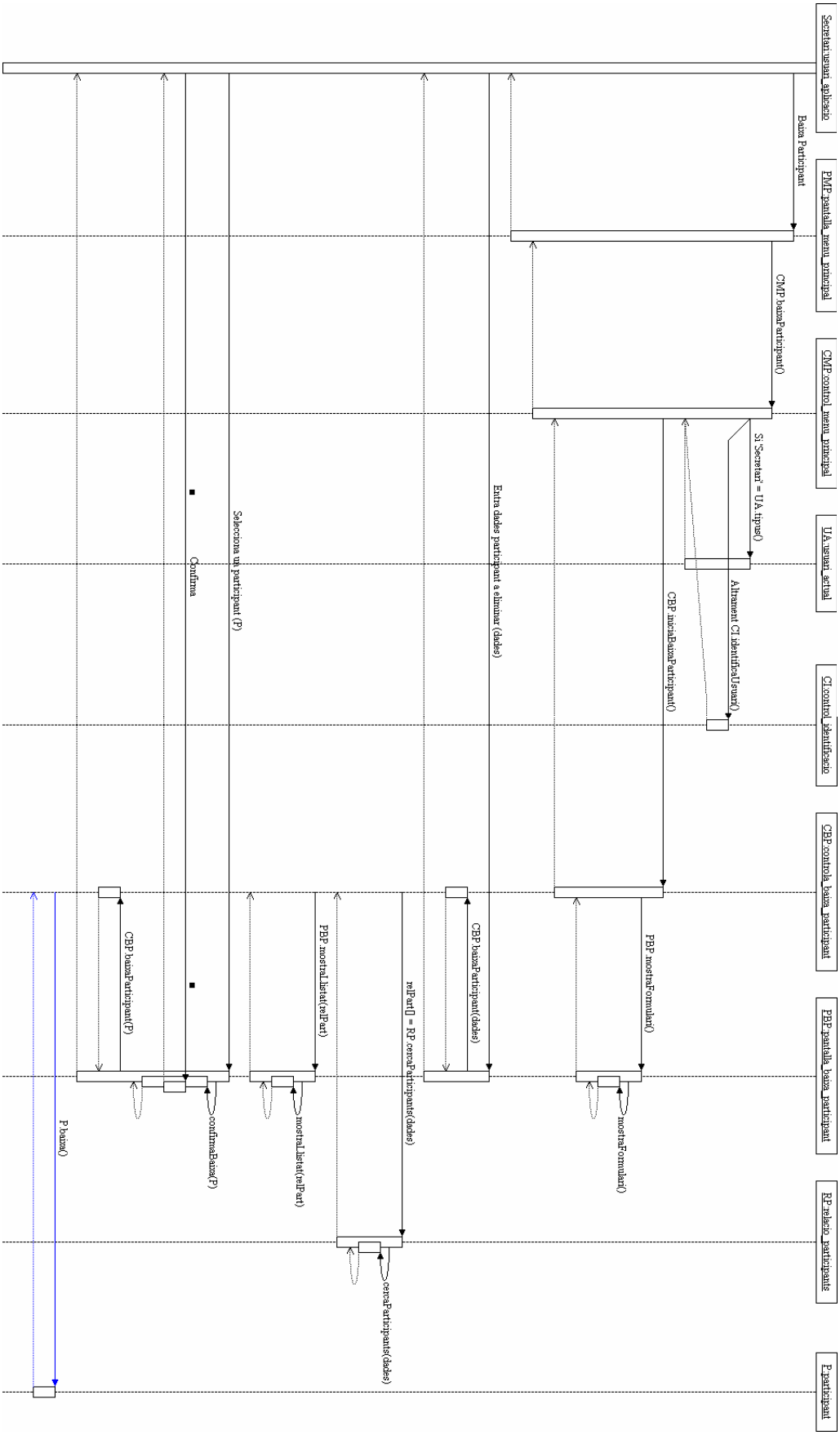
- ✓ **Baixa Participant:**

Cas d'ús: Baixa Participant
Objectiu: Donar de baixa un participant
Actors: Secretari
Precondició: -
Flux bàsic: 1- Sistema: El sistema verifica que el rol de l'usuari actual sigui el de Secretari 2- Usuari: Entra una o vàries dades del participant 3- Sistema: Valida les dades entrades i cerca en el sistema aquells participants que hi coincideixen. 4- Sistema: Si en troba algun, mostra una llista. Si només en troba un, el mostra directament. 5- Usuari: Selecciona el que es desitja donar de baixar com a participant. Si només n'hi ha un, es mostra automàticament. 6- Sistema : Demana confirmació de baixa. 7- Usuari: Confirma baixa. 8- Sistema: Elimina el participant.
Postcondició: El participant ha estat eliminat del sistema, però el soci relacionat amb ell continua actiu.
Flux alternatiu (extensions): 1.a- Sistema: Si el rol de l'usuari actual no és el que correspon a la zona a la que es vol accedir, s'executa el cas d'ús Identificació. 3.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: un '1' en un cognom) es tornen a demanar fins que siguin vàlides 4.a- Sistema: En cas de no trobar cap participant, s'informa a l'usuari i es torna al punt inicial, on l'usuari ha de seleccionar l'opció corresponent en el menú principal. **Observació:** Si també es desitja donar de baixar el soci relacionat amb aquest participant, haurem d'utilitzar el cas d'ús corresponent.

Baixa Participant ens permet donar de baixa un participant del sistema. Quan el donem d'alta, creem un soci que hi està relacionat, ja que, en realitat, el participant deriva del soci, només en guarda informació extra, com ara la data d'alta com a participant. En canvi, quan el volem donar de baixa, no tenim perquè eliminar el soci, ja que una determinada persona pot seguir formant part de l'entitat com a soci.

El primer que es fa és comprovar que l'Usuari Actual tingui els permisos corresponents. Tot seguit, l'usuari entra alguna dada del participant a eliminar, el sistema busca els que coincideixen amb les dades entrades i en mostra una llista. Acte seguit, l'usuari selecciona un participant i el sistema demana confirmació per eliminar-lo. Així que l'usuari confirma l'eliminació de l'objecte, el sistema l'elimina.


Usuari Actual:
Sistema:
Verifica permisos d'usuari
[té permís?]
Selecciona "Baixa Participant" en el menú
[no]
[si]
S'identifica
Valida Identificació
[identificació vàlida?]
[no]
[si]
Entra dades participant a eliminar
Cerca participants que encaixen amb les dades
[no]
[hi ha resultats?]
[si]
Mostra llistat participants
Selecciona participant
Demana confirmació
Confirma
Elimina participant

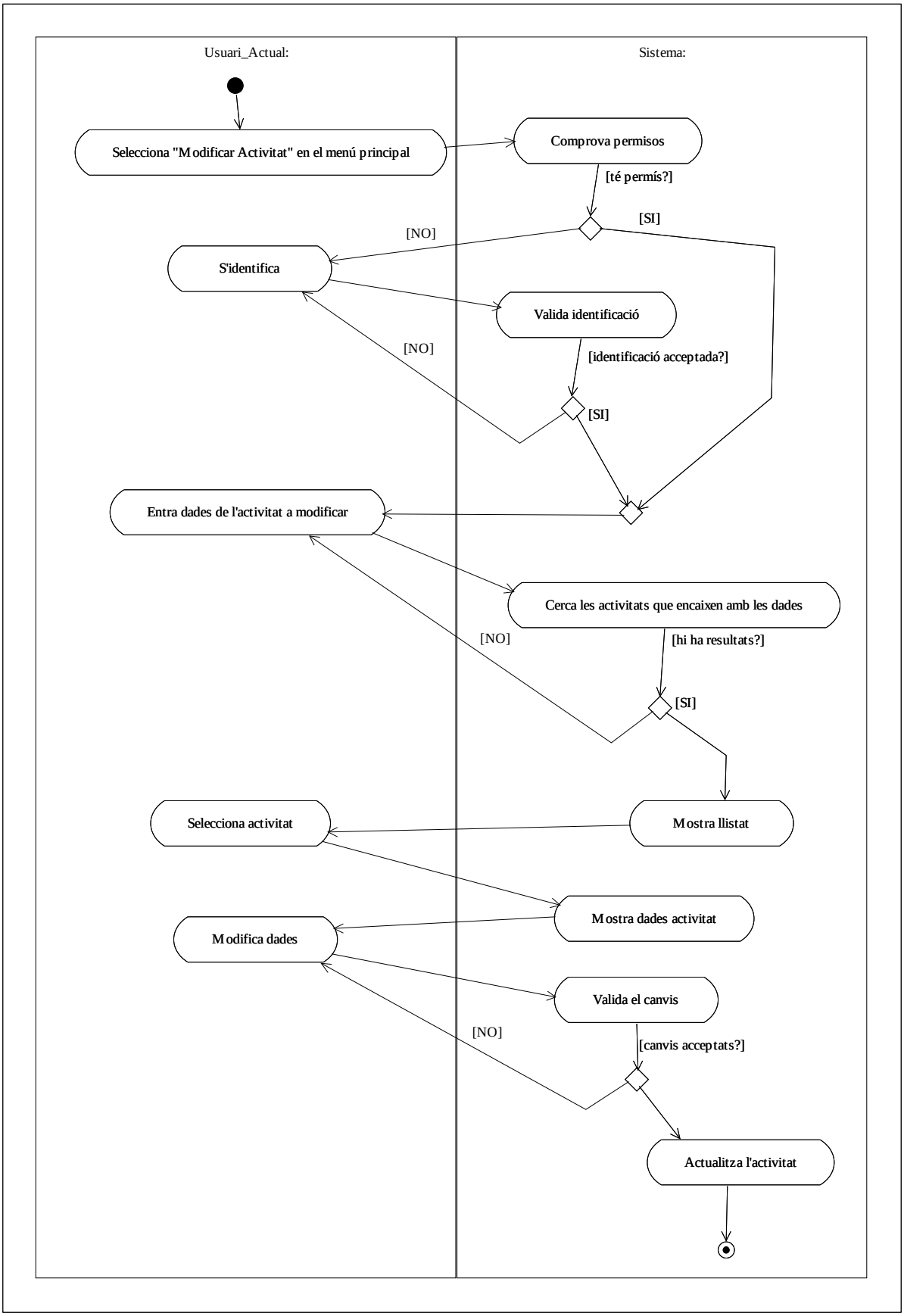

Secretari usuari aplicacio
PMP pantalla menu principal
CMP control menu principal
UA usuari actual
CI control identificacio
CBP control baixa participant
PBP pantalla baixa participant
RP relacio participants
P participant
Baixa Participant
CMP baixaParticipant()
Si 'Secretari' = UA.tipus()
Altrament CI.identificaUsuari()
CBP iniciaBaixaParticipant()
PBP mostraFormulari()
mostraFormulari()
Entra dades participant a eliminar (dades)
CBP baixaParticipant(dades)
relPart[] = RP.cercaParticipants(dades)
cercaParticipants(dades)
PBP mostraLlistat(relPart)
mostraLlistat(relPart)
Selecciona un participant (P)
Confirma
CBP baixaParticipant(P)
confirmaBaixa(P)
P.baixa()

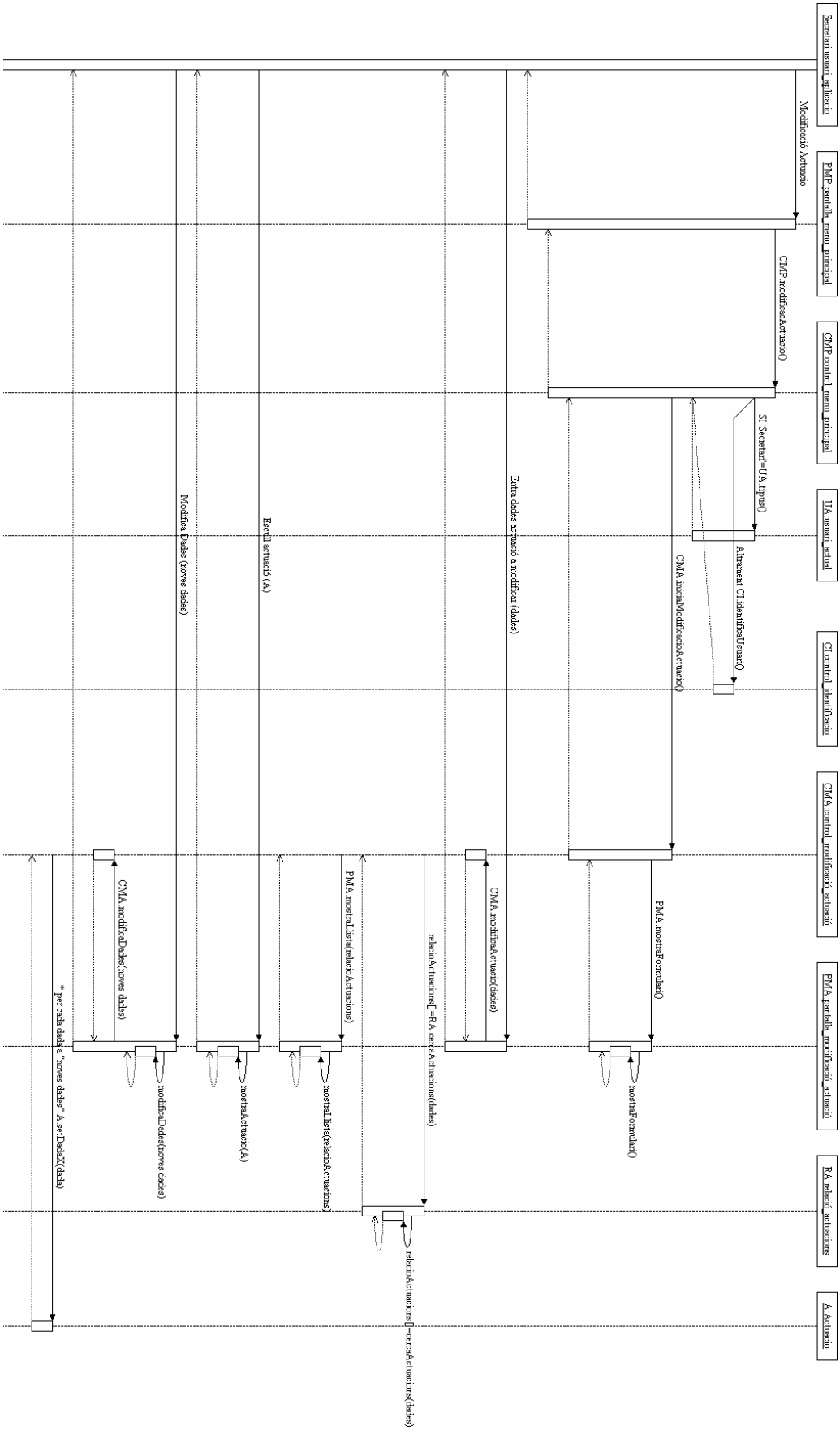
- ✓ **Modificació Actuació:**

Cas d'ús: Modificació Actuació
Objectiu: Canviar alguna informació d'una actuació ja existent en el sistema
Actors: Secretari
Precondició: -
Flux bàsic: 1- Sistema: El sistema verifica que el rol de l'usuari actual sigui el de Secretari 2- Usuari: Entra una o més dades de l'actuació 3- Sistema: Valida les dades entrades i cerca en el sistema les actuacions que coincideixin amb el/els criteri/s especificats per l'usuari. 4- Sistema: Si en troba alguna, mostra per pantalla una llista. 5- Usuari: Selecciona aquella actuació de la que vol modificar alguna dada. Si només n'hi ha una, es mostra directament. 6- Usuari: Modifica una ó vàries de les dades de l'actuació. 7- Sistema: Recull les dades i valida lògicament aquelles que s'han modificat 8- Sistema: S'actualitza l'actuació.
Postcondició: L'actuació conté les dades modificades
Flux alternatiu (extensions): 1.a- Sistema: Si el rol de l'usuari actual no és el que correspon a la zona a la que es vol accedir, s'executa el cas d'ús Identificació. 3.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: un '1' en un nom de poble) es tornen a demanar fins que siguin vàlides 4.a- Sistema: En cas de no trobar cap actuació, s'informa a l'usuari i es torna al punt inicial, on l'usuari ha de tornar a seleccionar l'opció corresponent en el menú principal. 7.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: un '1' en un nom de poble) es tornen a demanar fins que siguin vàlides.

Modificar Actuació permet a l'usuari de l'aplicació modificar aquelles dades d'una determinada actuació que per el motiu que sigui canvien. Per exemple, podem pensar que degut a un malentès entre les dos parts, s'ha de modificar la data d'inici de l'actuació. Un cas diferent, però que també necessita aquest cas d'ús és quan, un cop l'actuació s'ha dut a terme, en volem actualitzar o afegir dades. A tall d'exemple, voldrem modificar el programa que es va dur a terme i que va resultar ser diferent al que s'havia previst.

El primer que es fa és comprovar que l'Usuari Actual tingui els permisos corresponents. Acte seguit, l'usuari entrarà una o vàries dades de l'actuació que vol modificar per tal que el sistema pugui buscar-la. El sistema mostra llavors una llista de les actuacions que coincideixen amb les dades introduïdes. L'usuari en selecciona una i el sistema mostra totes les dades de que en disposa. Tot seguit, l'usuari modifica les dades necessàries i finalment, el sistema, després de comprovar la validesa d'aquestes, actualitza l'objecte


Usuari_Actual:
Sistema:
Selecciona "Modificar Activitat" en el menú principal
Comprova permisos
[té permís?]
[SI]
[NO]
S'identifica
Valida identificació
[NO]
[identificació acceptada?]
[SI]
Entra dades de l'activitat a modificar
Cerca les activitats que encaixen amb les dades
[NO]
[hi ha resultats?]
[SI]
Selecciona activitat
Mostra llistat
Mostra dades activitat
Modifica dades
Valida el canvis
[NO]
[canvis acceptats?]
Actualitza l'activitat

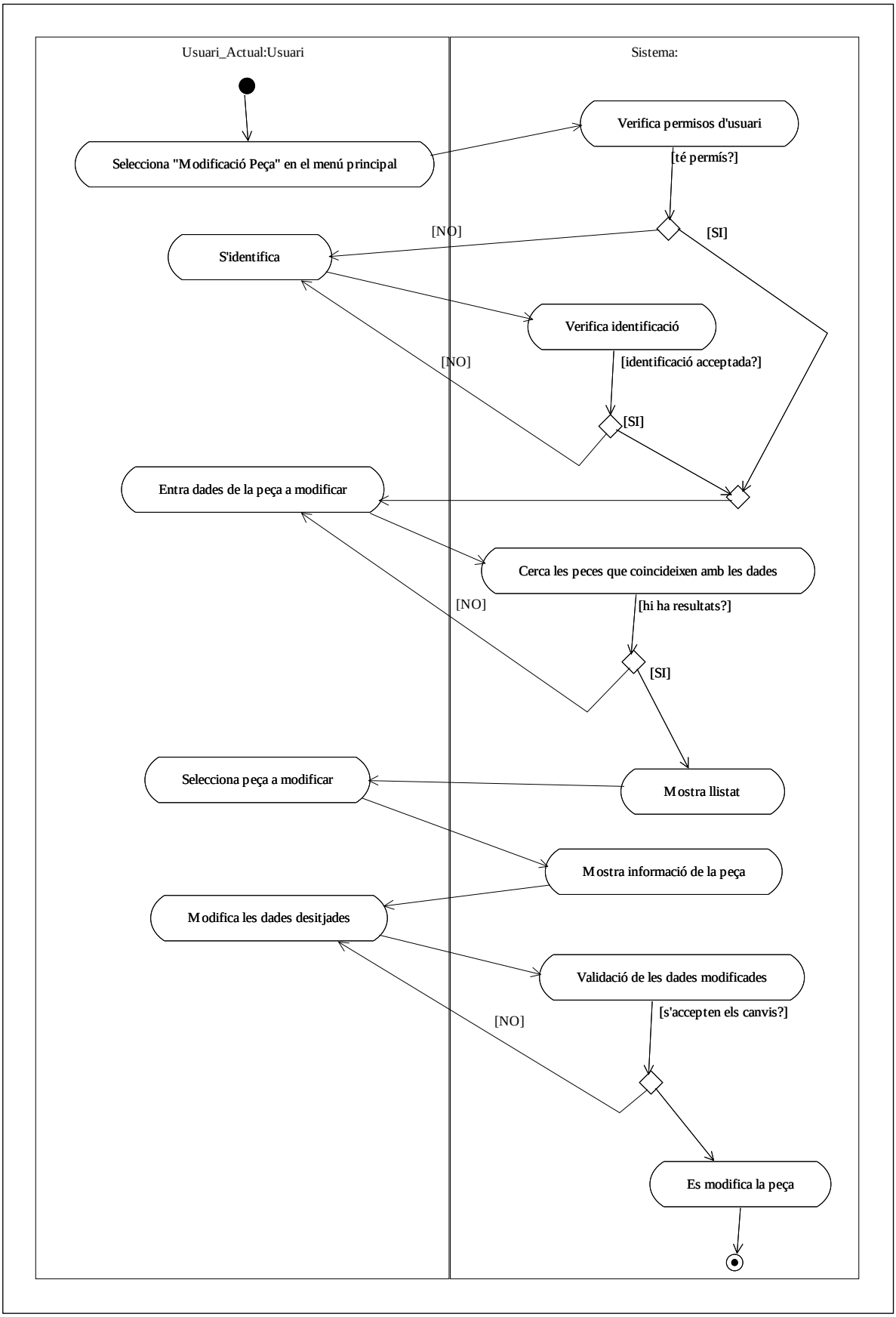

Secretari usuari aplicacio
PMP pantalla_menu_principal
CMP control_menu_principal
UA usuari_actual
CI control_identificacio
CMA control_modificació_actuació
PMA pantalla_modificació_actuació
RA relació_actuacions
A Actuacio
Modificació Actuacio
CMP modificarActuacio()
SI "Secretari"=UA.tipus()
Alhtament CI.identificaUsuari()
CMA.iniciaModificacioActuacio()
PMA.mostraFormulari()
mostraFormulari()
Entra dades actuació a modificar (dades)
CMA.modificaActuacio(dades)
relacioActuacions[]=RA.cercaActuacions(dades)
relacioActuacions[]=cercaActuacions(dades)
Escull actuació (A)
PMA.mostraLlista(relacioActuacions)
mostraLlista(relacioActuacions)
mostraActuacio(A)
Modifica Dades (noves dades)
CMA.modificaDades(noves dades)
modificaDades(noves dades)
* per cada dada a "noves dades" A.setDadaX(dada)

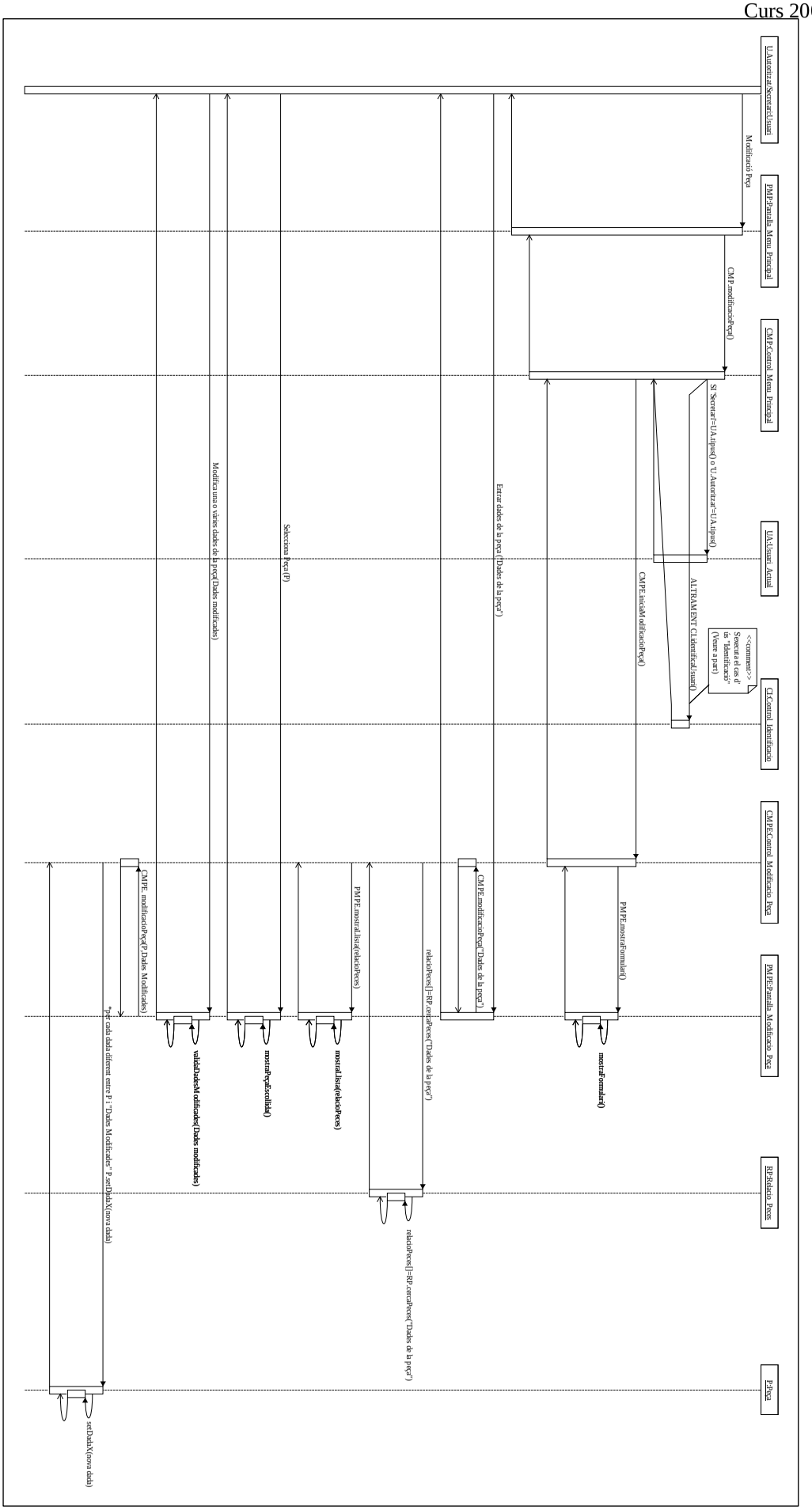
- ✓ **Modificació Peça:**

Cas d'ús: Modificació Peça
Objectiu: Canviar alguna informació d'una peça ja existent en el sistema
Actors: Usuari Repertori, Secretari
Precondició: -
Flux bàsic: 1- Sistema: El sistema verifica que el rol de l'usuari actual sigui el de Usuari Autoritzat o Secretari 2- Usuari: Entra una o més dades de la peça 3- Sistema: Valida les dades entrades i cerca en el sistema les peces que coincideixin amb el/els criteri/s especificats per l'usuari. 4- Sistema: Si en troba alguna, mostra per pantalla una llista. 5- Usuari: Selecciona aquella peça de la que vol modificar alguna dada. Si només se'n troba una, es mostra directament. 6- Usuari: Modifica una ó vàries de les dades de la peça. 7- Sistema: Recull les dades i valida lògicament aquelles que s'han modificat 8- Sistema: S'actualitza la peça.
Postcondició: La peça conté les dades modificades
Flux alternatiu (extensions): 1.a- Sistema: Si el rol de l'usuari actual no és el que correspon a la zona a la que es vol accedir, s'executa el cas d'ús Identificació. 3.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: un '1' en un nom de poble) es tornen a demanar fins que siguin vàlides 4.a- Sistema: En cas de no trobar cap peça, s'informa a l'usuari i es torna al punt inicial, on l'usuari ha de seleccionar l'opció corresponent en el menú principal. 7.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: una 'a' en la durada d'una peça) es tornen a demanar fins que siguin vàlides.

Modificar Peça permet a l'usuari de l'aplicació modificar aquelles dades d'una determinada peça que per el motiu que sigui canvien. Potser, simplement i si el sistema ho permet, desitjarem modificar-ne la imatge associada per actualitzar-la.

El primer que es fa és comprovar que l'Usuari Actual tingui els permisos corresponents. Acte seguit, l'usuari entrarà una o vàries dades de la peça que vol modificar per tal que el sistema pugui buscar-la. El sistema mostra llavors una llista de totes aquelles que coincideixen amb les dades introduïdes. L'usuari en selecciona una i el sistema mostra totes les dades de que en disposa. Tot seguit, l'usuari modifica les dades necessàries i finalment, el sistema, després de comprovar la validesa d'aquestes, actualitza l'objecte.


Usuari_Actual:Usuari
Sistema:
Selecciona "Modificació Peça" en el menú principal
Verifica permisos d'usuari
[té permís?]
[NO]
[SI]
S'identifica
Verifica identificació
[NO]
[identificació acceptada?]
[SI]
Entra dades de la peça a modificar
Cerca les peces que coincideixen amb les dades
[NO]
[hi ha resultats?]
[SI]
Selecciona peça a modificar
Mostra llistat
Mostra informació de la peça
Modifica les dades desitjades
Validació de les dades modificades
[NO]
[s'accepten els canvis?]
Es modifica la peça

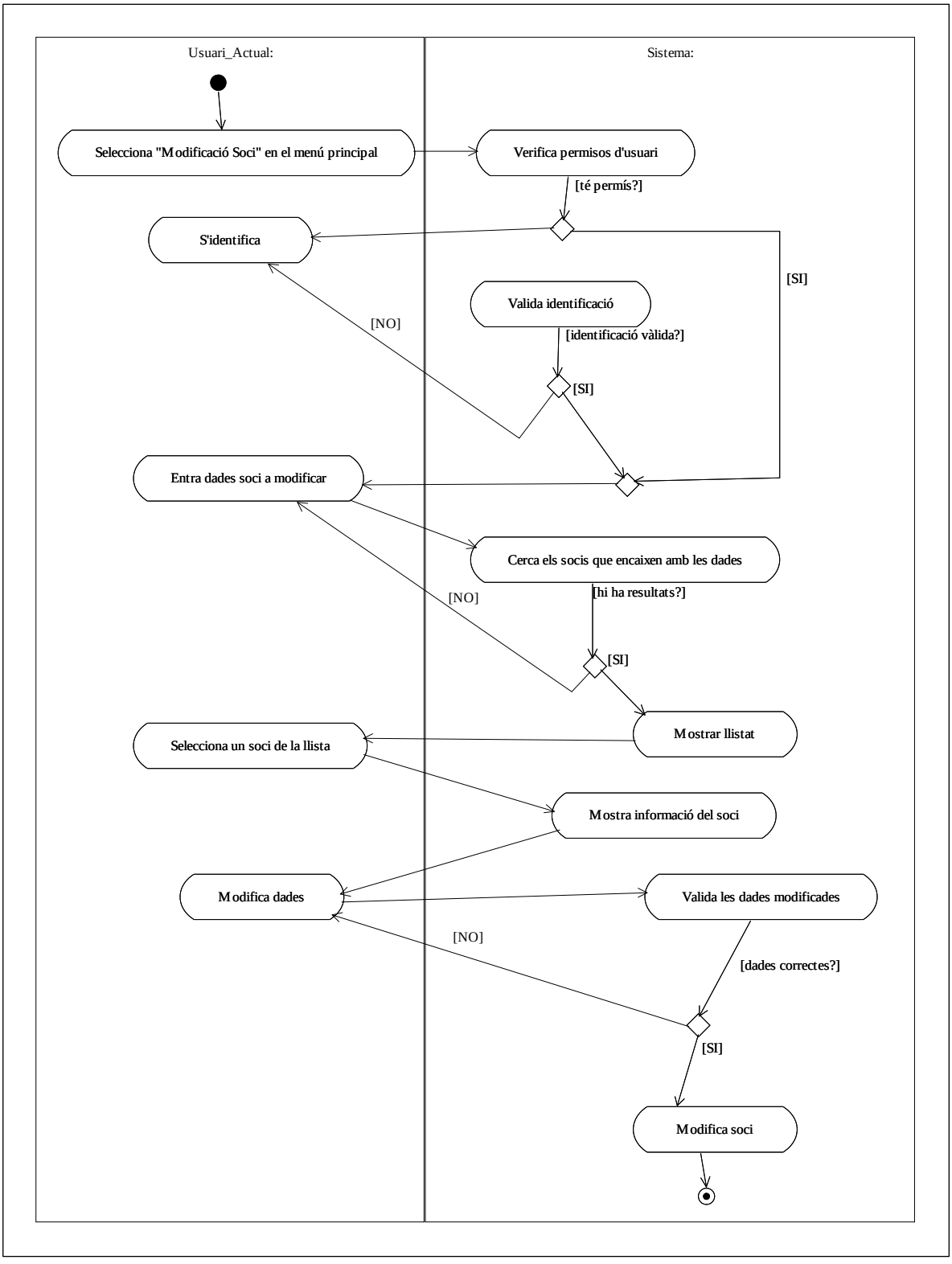

U:Autoritzat(Secretari:Usuari)
PMP:Pantalla_Menu_Principal
CMP:Control_Menu_Principal
UA:Usuari_Actual
CI:Control_Identificacio
CMPE:Control_Modificacio_Peca
PMPE:Pantalla_Modificacio_Peca
RP:Relacio_Peces
P:Peca
Modificació Peça
CMP.modificacioPeça()
Si(SecretariT=UA.tipus() o U.AutoritzatT=UA.tipus)
ALTRAMENT CI.identificaUsuari()
<<comment>>
S'executarà el cas d' ús "Identificació" (Veure a part)
CMPE.iniciaModificacioPeça()
Entrar dades de la peça("Dades de la peça")
Selecciona Peça(P)
Modifica una o vàries dades de la peça(Dades modificades)
CMPE.modificacioPeça("Dades de la peça")
PMPE.mostrarFormulari()
CMPE. modificacioPeça(P,Dades Modificades)
PMPE.mostrarLlista(relacioPeces)
relacioPeces[]=RP.cercaPeces("Dades de la peça")
mostrarFormulari()
mostrarLlista(relacioPeces)
mostrarPeçaSeleccionada()
validaDadesModificades(Dades modificades)
relacioPeces[]=RP.cercaPeces("Dades de la peça")
"per cada dada diferent entre P i "Dades Modificades" P.setDadaX(nova dada)
setDadaX(nova dada)

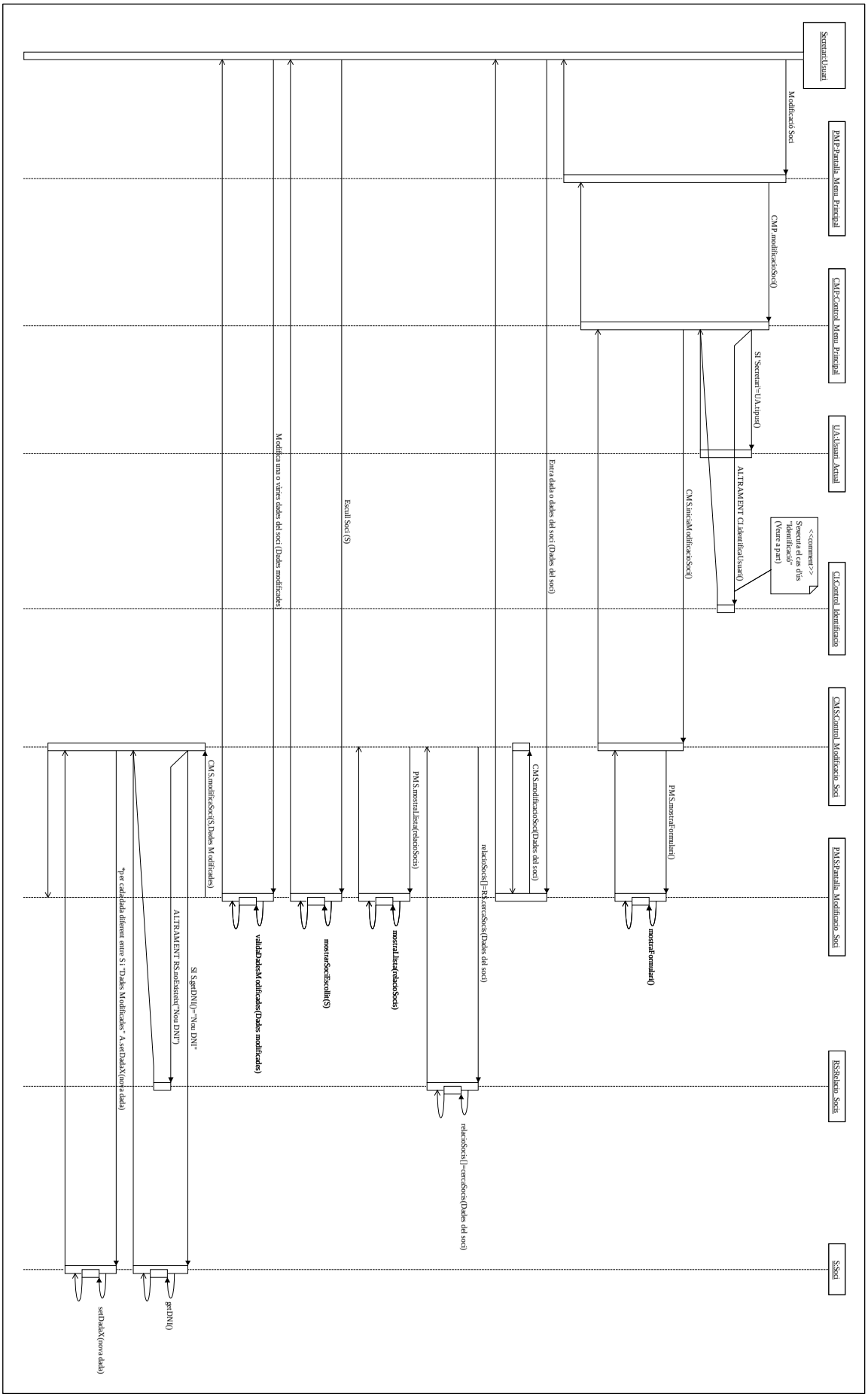
- ✓ **Modificació Soci:**

Cas d'ús: Modificació Soci
Objectiu: Canviar algun atribut d'un soci ja existent en el sistema
Actors: Usuari Soci, Secretari
Precondició: -
Flux bàsic: 1- Sistema: El sistema verifica que el rol de l'usuari actual sigui el de Secretari 2- Usuari: Entra una o vàries dades del soci 3- Sistema: Valida les dades entrades i cerca en el sistema aquells socis que hi coincideixen. 4- Sistema: Si en troba algun, mostra per pantalla una llista. 5- Usuari: Selecciona el soci desitjat. Si el sistema només pot trobar-ne un, el mostra directament. 6- Usuari: Modifica una ó vàries de les dades del soci. 7- Sistema: Recull les dades i valida lògicament aquelles que s'han modificat 8- Sistema: Es validen les dades amb els altres socis que ja té el sistema. Si aquestes no creen conflicte amb algun element ja present s'actualitza el soci.
Postcondició: El soci conté les dades modificades
Flux alternatiu (extensions): 1.a- Sistema: Si el rol de l'usuari actual no és el que correspon a la zona a la que es vol accedir, s'executa el cas d'ús Identificació. 3.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: un telèfon com aquest 972xx33aa) es tornen a demanar fins que siguin vàlides 4.a- Sistema: En cas de no trobar cap soci, s'informa a l'usuari i es torna al punt de partida, on l'usuari ha de tornar a seleccionar l'opció corresponent en el menú. 7.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: una 'a' en un telèfon) es tornen a demanar fins que siguin vàlides. 8.a- Sistema: En cas que l'identificador s'hagi modificat i el nou ja existís en el sistema, s'informaria a l'usuari que ja existeix un soci amb aquest identificador i s'avortaria l'operació.

Modificar Soci permet a l'usuari de l'aplicació modificar aquelles dades d'un determinat soci que per el motiu que sigui canvien o bé es van introduir malament en el seu moment.

El primer que es fa és comprovar que l'Usuari Actual tingui els permisos corresponents. Acte seguit, l'usuari entrarà una o vàries dades del soci que vol modificar per tal que el sistema pugui buscar-lo. El sistema mostra llavors una llista de tots aquells socis que coincideixen amb les dades introduïdes. L'usuari en selecciona un i el sistema mostra totes les dades de que en disposa. Tot seguit, l'usuari modifica les dades necessàries i finalment, el sistema, després de comprovar la validesa de les dades, actualitza l'objecte.


Usuari_Actual:
Sistema:
Selecciona "Modificació Soci" en el menú principal
Verifica permisos d'usuari
[té permís?]
S'identifica
[SI]
Valida identificació
[NO]
[identificació vàlida?]
[SI]
Entra dades soci a modificar
Cerca els socis que encaixen amb les dades
[NO]
[hi ha resultats?]
[SI]
Mostrar llistat
Selecciona un soci de la llista
Mostra informació del soci
Modifica dades
Valida les dades modificades
[NO]
[dades correctes?]
[SI]
Modifica soci

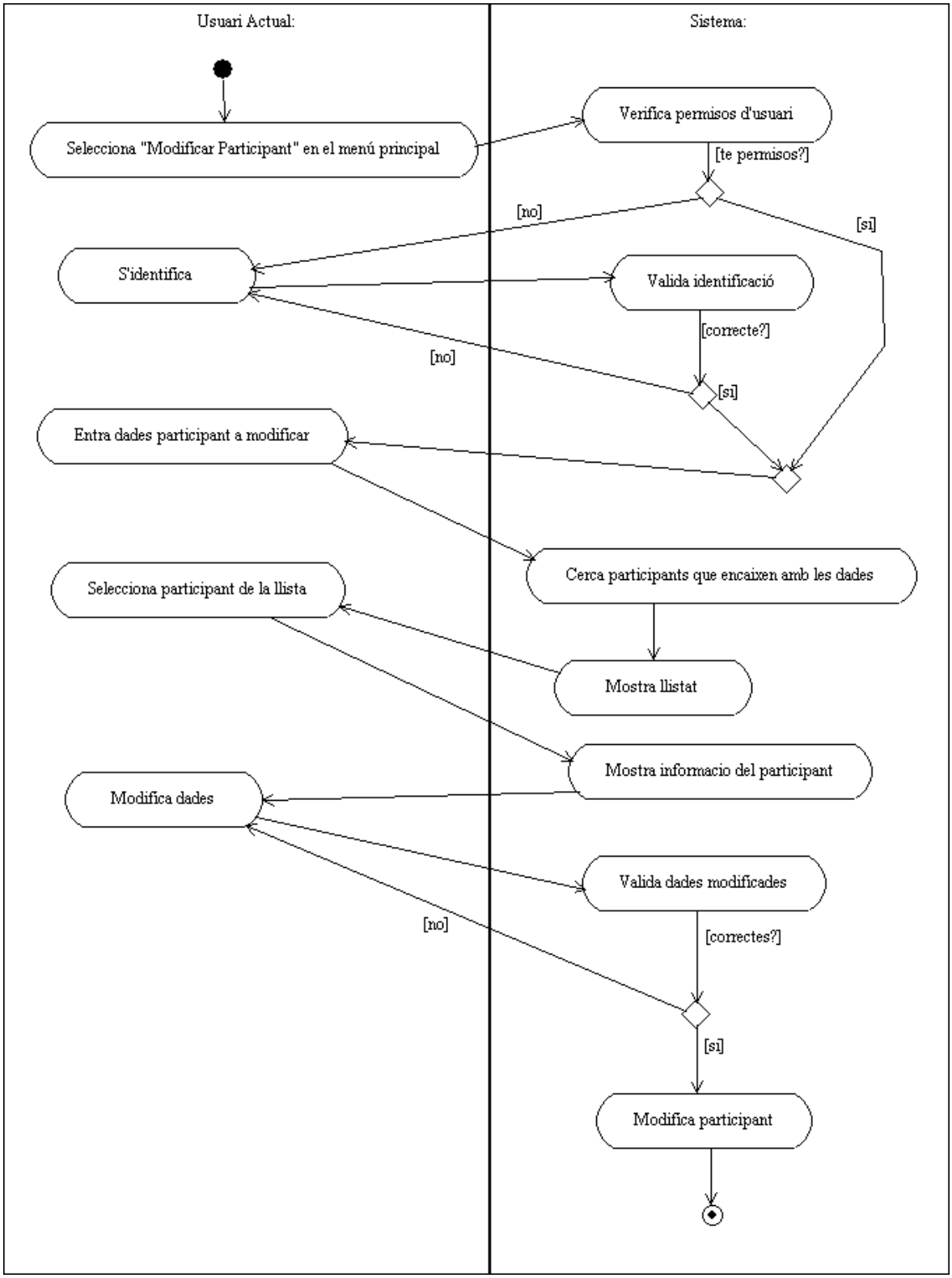

SecretariUsuari
PMP:Pantalla_Menu_Principal
CMP:Control_Menu_Principal
UA:Usuari_Actual
CI:Control_Identificacio
CMS:Control_Modificacio_Socis
PMS:Pantalla_Modificacio_Socis
RS:Relacio_Socis
S:Socis
Modificació Socis
CMP.modificacioSoci()
Si 'Secretari'=UA.tipus()
ALTRAMENT CI.identificaUsuari()
<<comment>>
Si secretaria el cas d'ús 'Identificació' (Veure a part)
CMS.iniciaModificacioSoci()
PMS.mostraFormulari()
mostraFormulari()
Entra dada o dades del soci (Dades del soci)
CMS.modificacioSoci(Dades del soci)
relacioSocis[]=RS.correcioSocis(Dades del soci)
relacioSocis[] correcioSocis(Dades del soci)
PMS.mostraLlistat(relacioSocis)
mostraLlistat(relacioSocis)
Escolli Soci (S)
mostrarSociEscollit(S)
Modifica una o vàries dades del soci (Dades modificades)
validaDadesModificades(Dades modificades)
CMS.modificaSoci(S,Dades Modificades)
Si S.getDNI()="Nou DNI"
ALTRAMENT RS.noExisteix("Nou DNI")
*per cada dada diferent entre S i "Dades Modificades" A.setDadaX(nova dada)
getDNI()
setDadaX(nova dada)

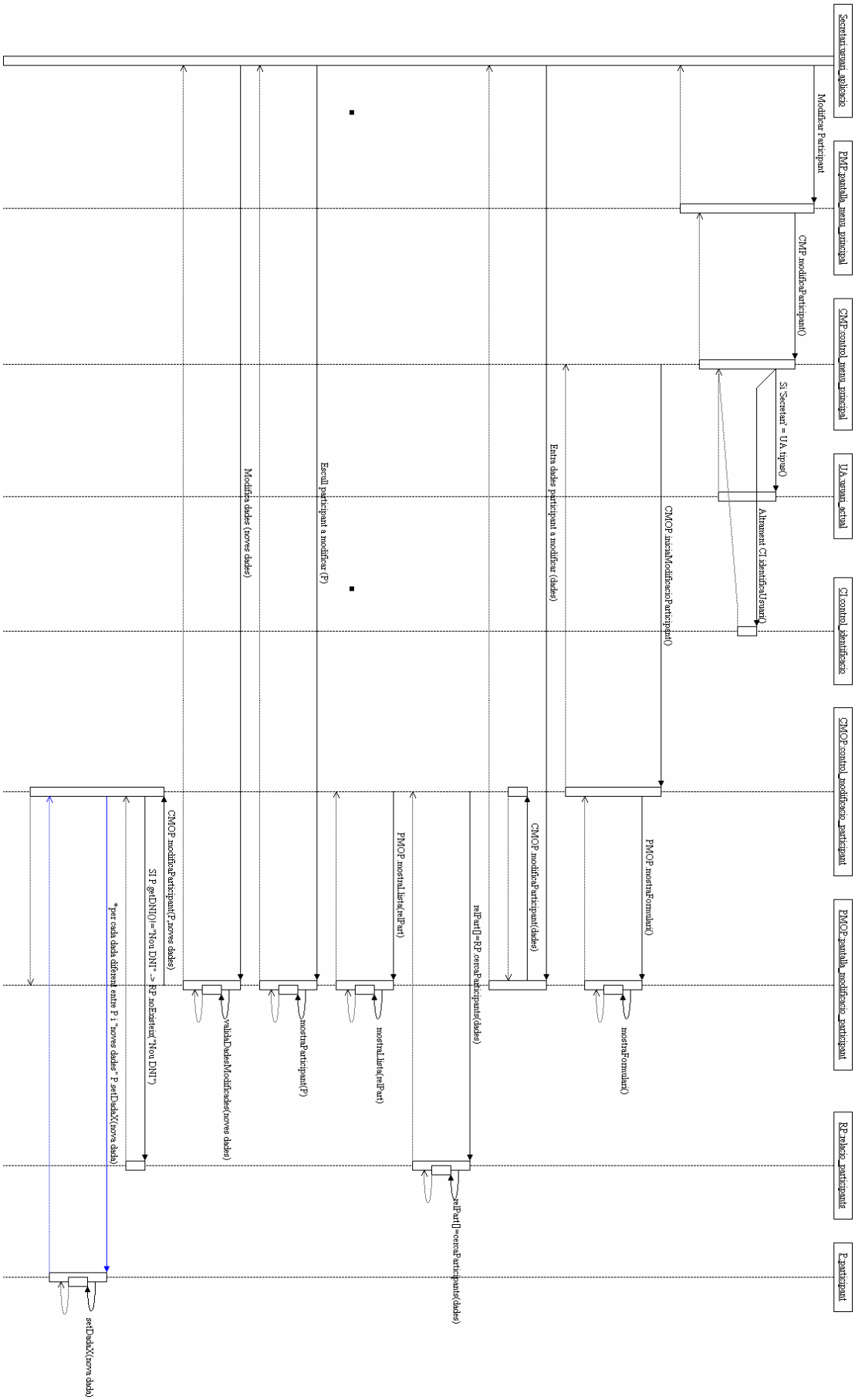
- ✓ **Modificació Participant**

Cas d'ús: Modificació Participant
Objectiu: Canviar algun atribut d'un participant ja existent en el sistema
Actors: Secretari
Precondició: -
Flux bàsic: 1- Sistema: El sistema verifica que el rol de l'usuari actual sigui el de Secretari 2- Usuari: Entra una o vàries dades del participant. 3- Sistema: Valida les dades entrades i cerca en el sistema aquells participants que hi coincideixen. 4- Sistema: Si en troba algun, mostra per pantalla una llista. Si només n'hi ha un, es mostra 5- Usuari: Selecciona el participant desitjat. Si el sistema només pot trobar-ne un, el mostra directament. 6- Usuari: Modifica una ó vàries de les dades del participant. 7- Sistema: Recull les dades i valida lògicament aquelles que s'han modificat 8- Sistema: Es validen les dades amb els altres participants que ja té el sistema. Si aquestes no creen conflicte amb algun element ja present s'actualitza el participant.
Postcondició: El participant conté les dades modificades
Flux alternatiu (extensions): 1.a- Sistema: Si el rol de l'usuari actual no és el que correspon a la zona a la que es vol accedir, s'executa el cas d'ús Identificació. 3.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: un telèfon com aquest 972xx33aa) es tornen a demanar fins que siguin vàlides 4.a- Sistema: En cas de no trobar cap participant, s'informa a l'usuari i es torna al punt de partida, on l'usuari ha de tornar a seleccionar l'opció corresponent en el menú. 7.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: una 'a' en un telèfon) es tornen a demanar fins que siguin vàlides. 8.a- Sistema: En cas que l'identificador s'hagi modificat i el nou ja existís en el sistema, s'informaria a l'usuari que ja existeix un participant amb aquest identificador i s'avortaria l'operació.

Modificar Participant permet a l'usuari de l'aplicació modificar aquelles dades d'un determinat participant que per el motiu que sigui canvien o bé es van introduir malament en el seu moment.

El primer que es fa és comprovar que l'Usuari Actual tingui els permisos corresponents. Acte seguit, l'usuari entrarà una o vàries dades del participant que vol modificar per tal que el sistema pugui buscar-lo. El sistema mostra llavors una llista de tots aquells que coincideixen amb les dades introduïdes. L'usuari en selecciona un i el sistema mostra totes les dades de que en disposa. Tot seguit, l'usuari modifica les dades necessàries i finalment, el sistema, després de comprovar la validesa de les dades, actualitza l'objecte.

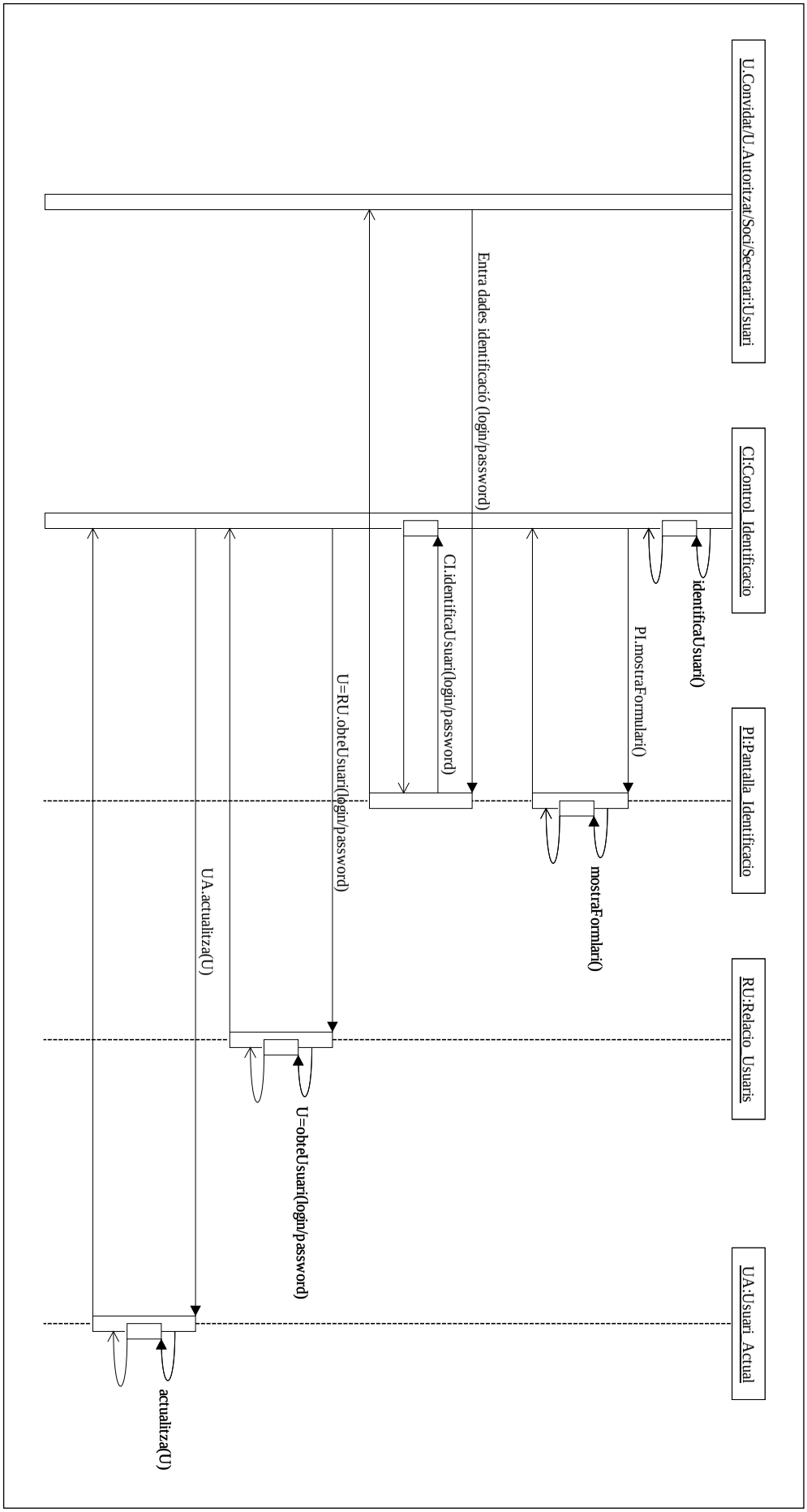

Usuari Actual:
Sistema:
Selecciona "Modificar Participant" en el menú principal
Verifica permisos d'usuari
[te permisos?]
[no]
[si]
S'identifica
Valida identificació
[correcte?]
[no]
[si]
Entra dades participant a modificar
Cerca participants que encaixen amb les dades
Selecciona participant de la llista
Mostra llistat
Mostra informacio del participant
Modifica dades
Valida dades modificades
[no]
[correctes?]
[si]
Modifica participant


Secretari sessió aplicació
PMP pantalla menu principal
CMP control menu principal
UA sessió actual
CI control identificació
CMOP control modificació participant
PMOP pantalla modificació participant
RP rebuig participants
P participant
Modificar Participant
CMP modificaParticipant()
Si 'Secretari' = UA.tipus()
Alhament CI identificaUsuari()
CMOP iniciaModificacioParticipant()
Entra dades participant a modificar (dades)
CMOP modificaParticipant(dades)
PMOP mostraFormulari()
mostraFormulari()
Escull participant a modificar (P)
FMOP mostraLlista(relPart)
mostraLlista(relPart)
relPart[]=RP cercaParticipants(dades)
relPart[]=cercaParticipants(dades)
Modifica dades (noves dades)
validaDadesModificacio(noves dades)
mostraParticipant(P)
CMOP modificaParticipant(P,noves dades)
Si P.getDNI()!="Nou DNI" -> RP.noExisteix("Nou DNI")
*per cada dada diferent entre P i "noves dades" P.setDadaX(nova dada)
setDadaX(nova dada)

- ✓ **Identificació:**

Com ja s'ha esmentat abans, aquest cas d'ús representa la idea de restringir certes parts del sistema que no es creuen convenients ser accessibles per a tothom.
Com que és un cas d'ús utilitzat per altres casos d'ús enlloc de per l'usuari, s'ha inclòs en el diagrama d'actuació d'aquests per tal de facilitar-ne la comprensió.
Així doncs, en aquest punt, no es mostrarà el diagrama d'actuació perquè ja s'ha mostrat amb anterioritat.

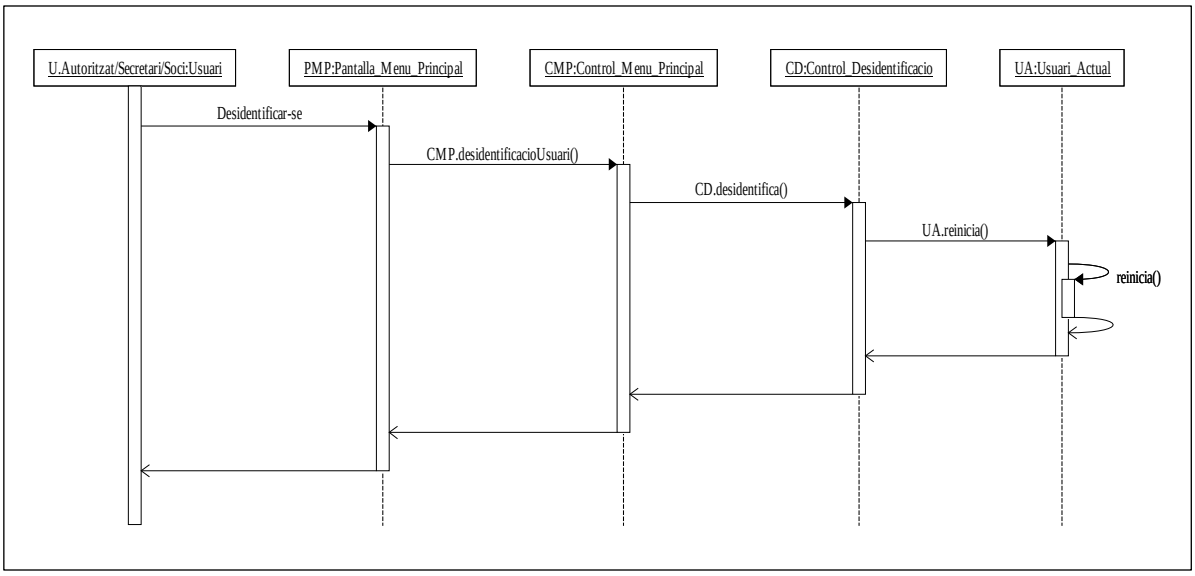
Cas d'ús: Identificació
Objectiu: Garantir l'accés restringit a determinades àrees del sistema que ho requereixen
Actors: Sistema
Precondició: -
Flux bàsic: 1- Sistema: Notifica a l'usuari que s'ha d'identificar. 2- Usuari: Entra usuari/contrasenya corresponent. 3- Sistema: Comprova l'existència d'un usuari que coincideixi amb les dades entrades per part de l'usuari. 4- Sistema: Si l'usuari realment existeix, l'objecte "Usuari Actual" passa a tenir el valor del nou rol sol·licitat per l'usuari.
Postcondició: L'usuari actual "veu" la part del sistema que li és permesa
Flux alternatiu (extensions): 4.a- Sistema: Si no es troba cap usuari en el sistema com el que demana l'usuari, es mostra un avis per pantalla i es torna al punt inicial, on s'haurà de tornar a executar aquest cas d'ús.
Observació: La idea és que cada cop que l'usuari entra en una zona restringida i no ha adoptat un rol que li permet accedir-hi, el sistema li demanarà que s'identifiqui. Un cop fet això, l'usuari podrà anar a totes aquelles zones restringides que el sistema li permeti degut a la seva identificació. Si l'usuari desitja accedir a una altra zona restringida diferent de l'actual, o bé, cosa que seria més normal, un altre usuari es posa a treballar amb l'aplicació, aquest serà preguntat per el nou codi el primer cop que desitgi entrar en la nova zona del sistema. La identificació de l'antic usuari quedarà llavors anul·lada i per tant la nova vista només permetrà accedir a aquelles zones a les que l'últim codi entrat permet accedir.


U:Convidat/U.Autoritzat/Soci/Secretari:Usuari
CI:Control_Identificacio
PI:Pantalla_Identificacio
RU:Relacio_Usuaris
UA:Usuari_Actual
Entra dades identificació (login/password)
identificaUsuari()
PI.mostraFormulari()
mostraFormlari()
CI.identificaUsuari(login/password)
U=RU.obteUsuari(login/password)
U=obteUsuari(login/password)
UA.actualitza(U)
actualitza(U)

- ✓ **Desidentificació:**

Cas d'ús: Desidentificació
Objectiu: Finalitzar la sessió d'un usuari amb privilegis
Actors: Soci, Usuari Soci, Usuari Repertori, Secretari
Precondició: -
Flux bàsic: Sistema: L'objecte "Usuari Actual" passa a tenir el valor de rol de l'usuari amb menys privilegis de tots: "Usuari Convidat"
Postcondició: L'usuari actual no pot accedir més que a aquelles parts que no són d'accés restringit

Desidentificació permet a l'usuari de l'aplicació finalitzar la sessió en el sistema per tal de que ningú altre pugui accedir a les zones restringides sense estar degudament autoritzat.
El cas d'ús simplement canvia l'estat de l'objecte Usuari Actual i aquest adopta el rol de Usuari Convidat.

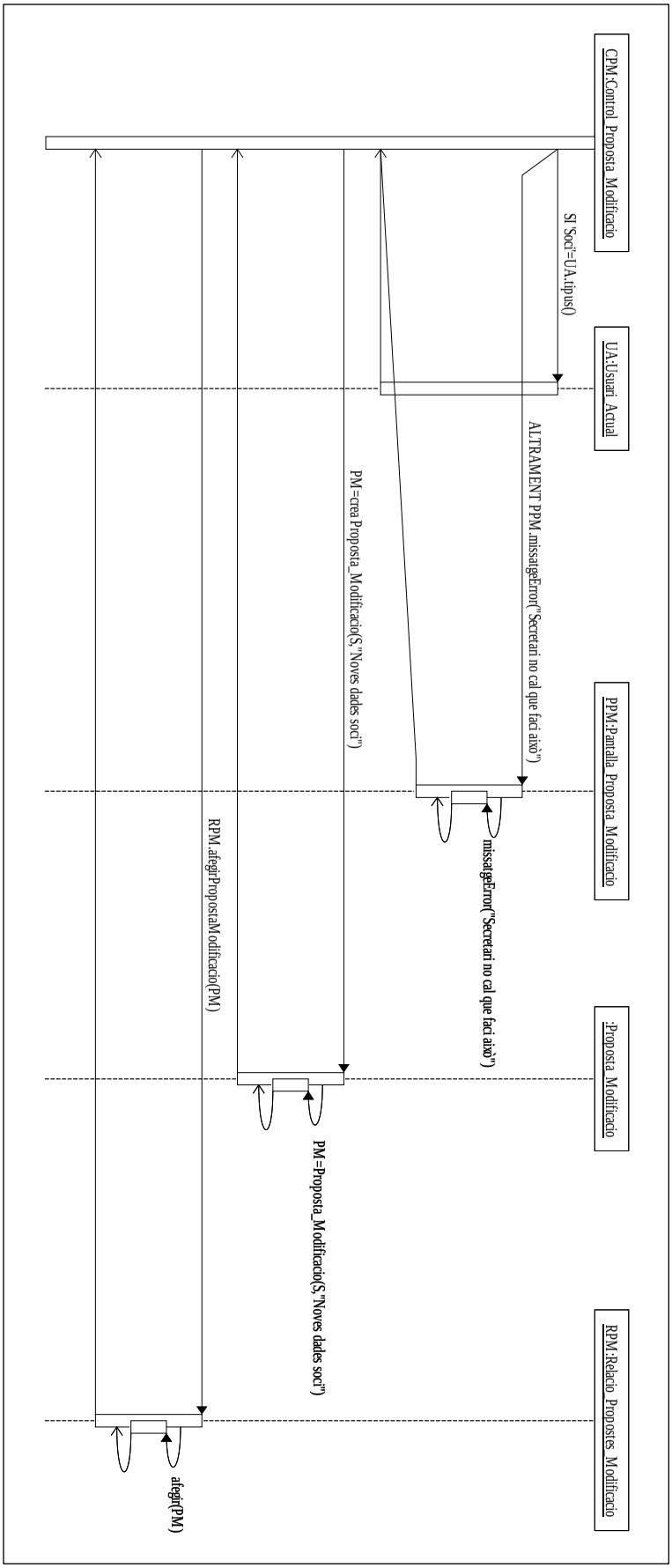


- ✓ **Alta Proposta Modificació**

Cas d'ús: Alta proposta modificació
Objectiu: Generar una petició adreçada a una persona autoritzada per tal que modifiqui les dades d'un soc
Actors: Sistema
Precondició: L'usuari ha emplenat el formulari de modificació de dades
Flux bàsic: 1- Sistema: El sistema verifica que el rol de l'usuari actual no sigui el de Secretari 2- Sistema: S'instancia l'objecte de proposta de modificació 3- Sistema: S'emplena amb les dades recollides en el procés anterior. 4- Sistema: S'afegeix l'objecte al sistema
Postcondició: S'ha generat una ordre de modificació d'un soci
Flux alternatiu (extensions): **1a.** Sistema: Si es dona el cas de que l'usuari té el rol de secretari s'informa a l'usuari que simplement faci la modificació de dades pertinent a partir del les opcions permeses al seu rol.
Observació: En aquest cas no es creu pertinent fer una validació de les dades, doncs les dades que arriben ja han estat validades en el cas d'us que ha cridat aquest, que és el cas d'ús de Consulta de Soci. No es contempla l'execució del cas d'us "Identificació" ja que, degut a la casuística de l'aplicació, només es pot accedir en el cas d'ús actual si l'usuari està autoritzat a fer-ho. Tot i això, si es pretén fer una versió web d'aquesta aplicació, sí que seria necessària un control sobre aquest tema, ja que aquest cas d'ús seria una URL concreta, i s'hi podria accedir directament teclejant aquesta a la barra d'adreces del navegador.

Alta Proposta Modificació permet a l'usuari de l'aplicació demanar que es modifiquin les dades d'un determinat soci, tal i com ja s'ha comentat.

El primer que es fa és comprovar que el rol de l'usuari sigui l'apropiat. És a dir, per una banda el soci pot consultar les seves dades i per tant demanar una modificació d'aquestes. Per altra banda, el secretari també pot accedir a aquesta informació, però no té cap sentit que el mateix secretari demani una actualització de les dades, perquè de fet, és ell qui les ha d'actualitzar.
Finalment, es crea l'objecte amb les dades recollides i s'afegeix al sistema.

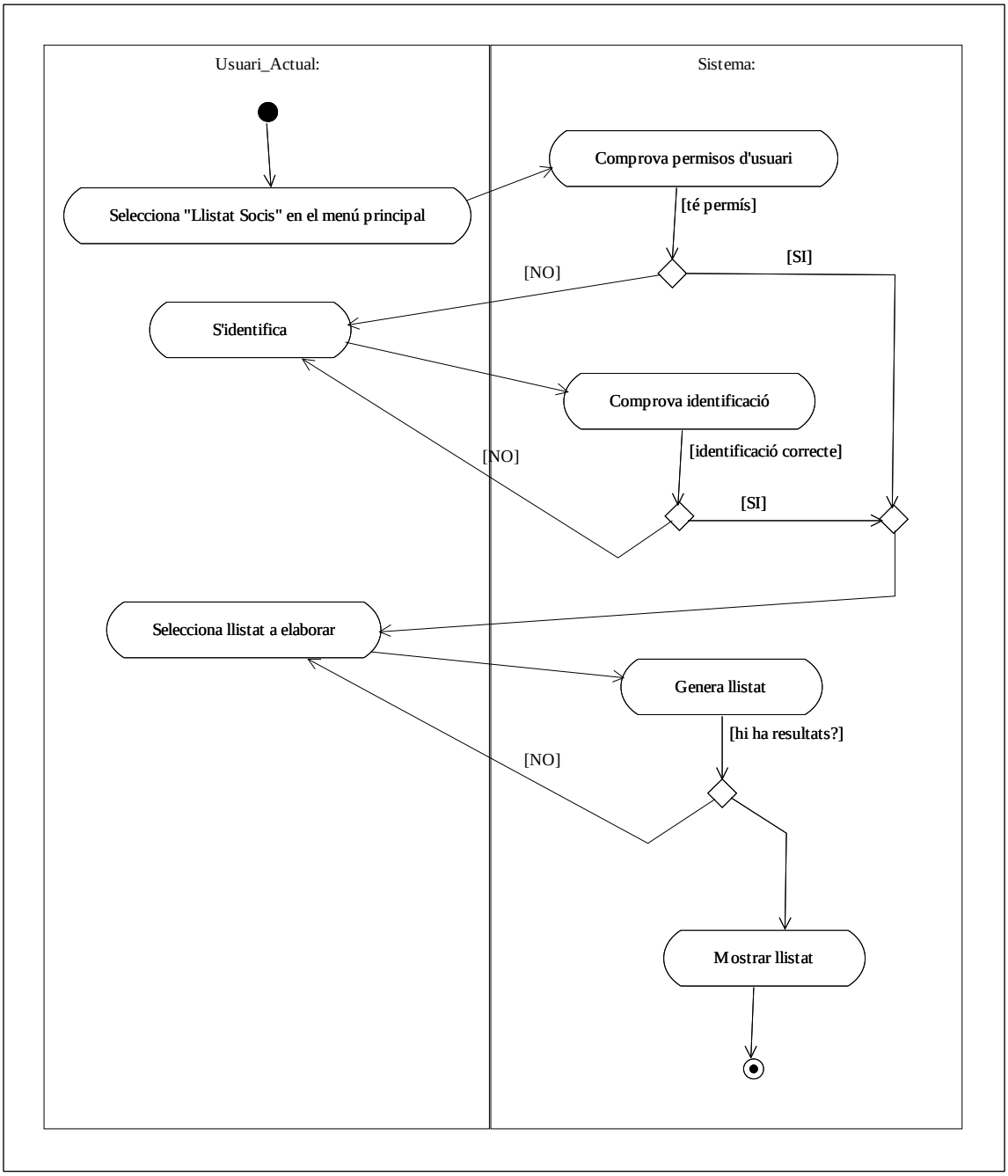

CPM:Control_Proposta_Modificació
UA:Usuari_Actual
PPM:Pantalla_Proposta_Modificació
:Proposta_Modificació
RPM:Relació_Propostes_Modificació
SI 'Soci'=UA.tipus()
ALTRAMENT PPM.missatgeError("Secretari no cal que faci això")
missatgeError("Secretari no cal que faci això")
PM=crea Proposta_Modificació(S,"Noves dades soci")
PM=Proposta_Modificació(S,"Noves dades soci")
RPM.afegirPropostaModificació(PM)
afegir(PM)

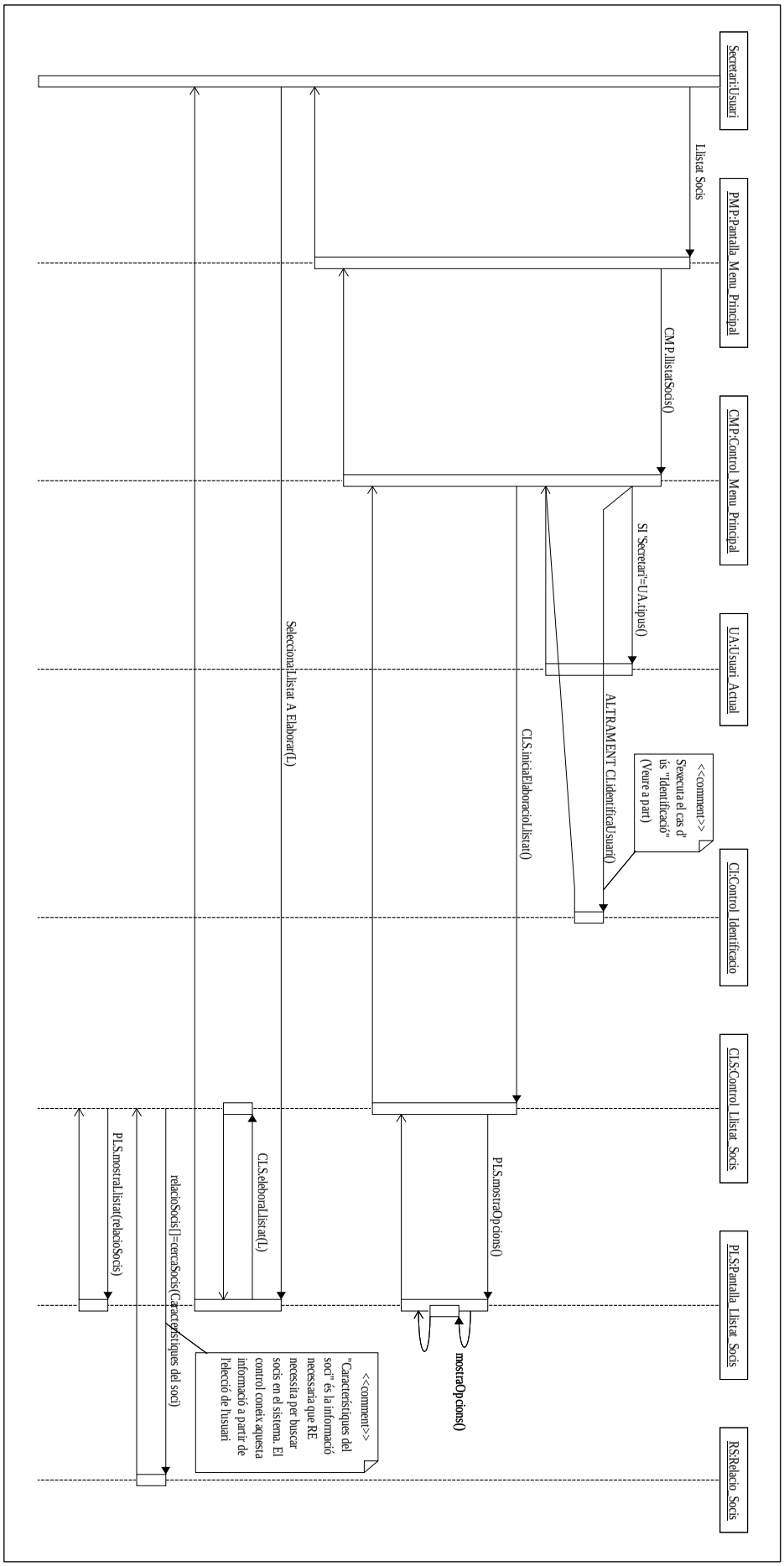
- ✓ **Llistats Socis**

Cas d'ús: Llistats
Objectiu: Elaborar un llistat de socis que compleixen uns requisits
Actors: Usuari Soci, Secretari
Precondició: -
Flux bàsic: 1- Sistema: El sistema verifica que el rol de l'usuari actual sigui el de Secretari. 2- Usuari: Selecciona d'entre vàries opcions (predefinides) el tipus de llistat a elaborar (Ex: socis amb antiguitat major a X anys) 3- Sistema: Genera un llistat que correspon amb la selecció de l'usuari i el mostra a l'usuari.
Postcondició: S'ha mostrat un llistat de socis corresponent a l'interès de l'usuari
Flux alternatiu (extensions): 1.a- Sistema: Si el rol de l'usuari actual no és el que correspon a la zona a la que es vol accedir, s'executa el cas d'ús Identificació. 3.a- Sistema: En cas de no trobar cap soci, s'informa a l'usuari i es torna al punt inicial, on l'usuari ha de seleccionar l'opció corresponent en el menú principal.

Llistat Socis permet a l'usuari generar un llistat a partir d'unes opcions predefinides. Donat que el soci és, en molts casos, la part del sistema que genera més informació i també la que serà consultada amb mes freqüència, es considera oportú oferir aquesta funcionalitat per tal de facilitar i accelerar el procés de consulta d'informació que prové dels socis.

El primer que es fa és comprovar que el rol de Usuari Actual sigui l'adient per executar aquesta funcionalitat. Acte seguit, l'usuari de l'aplicació selecciona una opció d'entre un conjunt de llistats predefinits. Finalment, el sistema genera el llistat sol·licitat i el mostra per pantalla

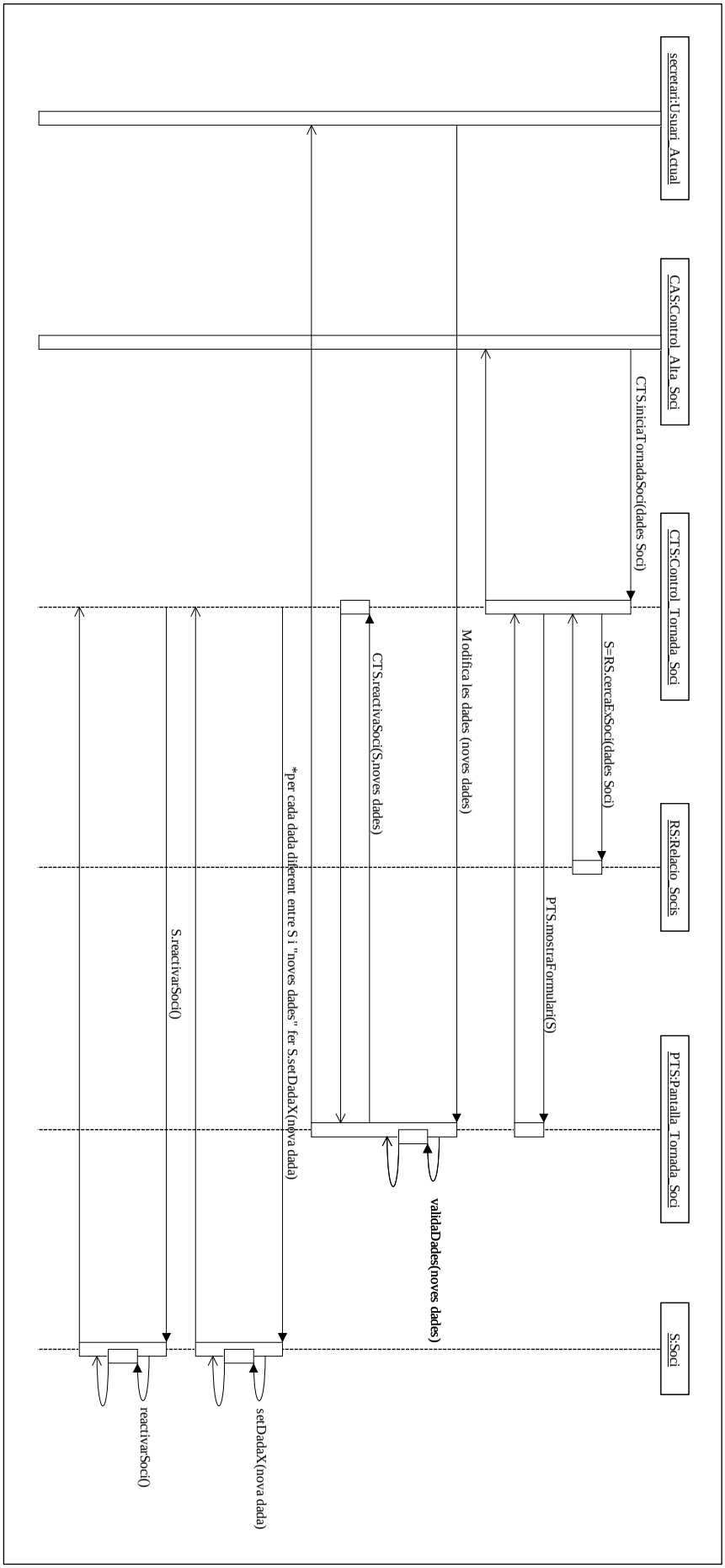

Usuari_Actual:
Sistema:
Selecciona "Llistat Socis" en el menú principal
Comprova permisos d'usuari
[té permís]
[SI]
[NO]
S'identifica
Comprova identificació
[identificació correcte]
[NO]
[SI]
Selecciona llistat a elaborar
Genera llistat
[hi ha resultats?]
[NO]
Mostrar llistat


SecretariUsuari
PMP:Pantalla_Menu_Principal
CMP:Control_Menu_Principal
UA:Usuari_Actual
CI:Control_Identificacio
CLS:Control_Llistat_Socis
PLS:Pantalla_Llistat_Socis
RS:Relacio_Socis
Llistat Socis
CMP.llistatSocis()
SI Secretari=UA.tipus()
ALTRAMENT CI.identificaUsuari()
<<comment>>
S'executa el cas d'ús "Identificació" (Veure a part)
CLS.iniciaElaboracioLlistat()
SeleccionaLlistat A Elaborar(L)
PLS.mostraOpcions()
mostraOpcions()
CLS.elaborarLlistat(L)
relacioSocis[]=cercaSocis(Caracteristiques del soci)
<<comment>>
"Caracterísitiques del soci" és la informació necessaria que RE necessita per buscar socis en el sistema. El control conetx aquesta informació a partir de l'elecció de l'usuari
PLS.mostraLlistat(relacioSocis)

- ✓ **Tornada Soci**

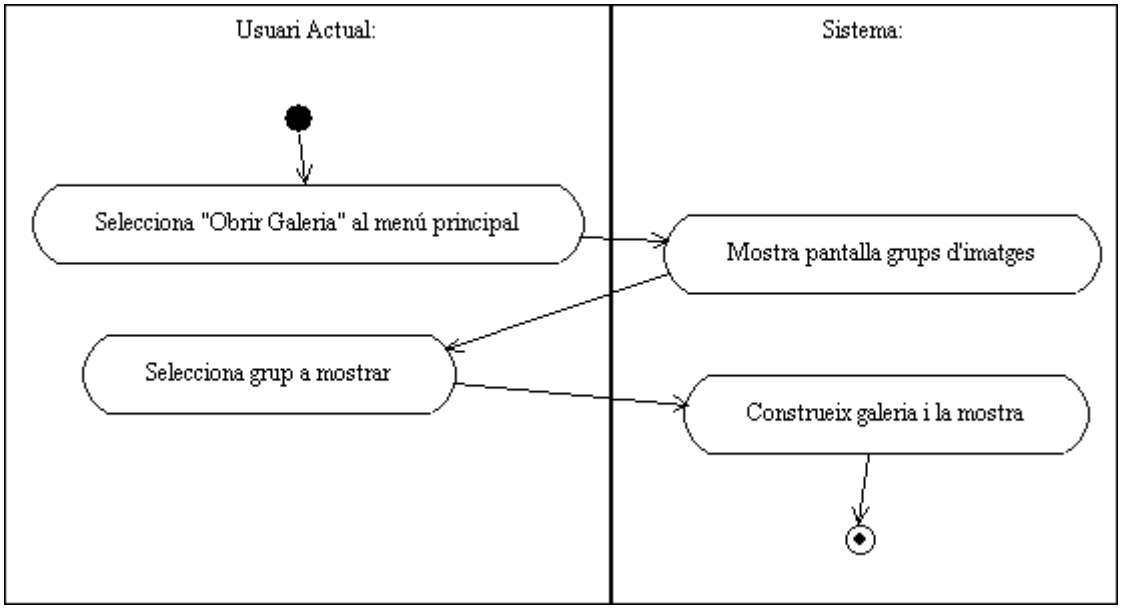
Cas d'ús: Tornada Soci
Objectiu: Reactivar un soci que desitja ser membre de l'entitat i ja ho va ser en el passat.
Actors: Sistema
Precondició: -
Flux bàsic: 1- Sistema: Busca la informació guardada de l'anterior etapa de la persona a l'entitat i en mostra tots els detalls que se'n conserven. 2- Usuari: Modifica les dades que han canviat des de llavors i entra les que són en blanc. 3- Sistema: Verifica lògicament les noves dades. 4- Sistema: Actualitza les dades. 5- Sistema: Estableix l'objecte com a Actiu(el reactiva).
Postcondició: El soci està altre cop actiu, se'n poden fer consultes i modificacions, així com també donar-lo de baixa.
Flux alternatiu (extensions): 3.a- Sistema: Si les dades presenten algun valor no vàlid lògicament (Ex: una 'a' en el telèfon) es tornen a demanar fins que siguin vàlides. **Observacions:** No es contempla l'execució del cas d'us "Identificació" ja que, degut a la casuística de l'aplicació, només es pot accedir en el cas d'ús actual si l'usuari està autoritzat a fer-ho. Tot i això, si es pretén fer una versió web d'aquesta aplicació, sí que seria necessària un control sobre aquest tema, ja que aquest cas d'ús seria una URL concreta, i s'hi podria accedir directament teclejant aquesta a la barra d'adreces del navegador. En aquest cas, a diferencia que en el cas d'ús Modificació Soci, no es preveu comprovar que les noves dades no creïn conflicte amb les d'altres socis. Això és així ja que ja sigui el DNI o qualsevol altre camp que s'utilitzi per a la identificació de manera única del soci en el sistema, aquest es mantindrà quan el soci es doni de baixa, i al mateix temps, no canviarà. Aquest camp no serà modificable per a l'usuari en aquest cas d'ús.

Tornada Soci és un cas d'ús pensat per el cas especial que ja s'ha comentat sobre els socis, els quals, de fet, mai desapareixen del sistema, sinó que canvien d'estat perquè ja no volem que apareguin en els llistats o en les simples consultes d'informació de socis.


secretari:Usuari_Actual
CAS:Control_Alta_Soci
CTS:Control_Tomada_Soci
RS:Relacio_Socis
PTS:Pantalla_Tomada_Soci
S:Soci
CTS.iniciaTomadaSoci(dades Soci)
S=RS.cercaExSoci(dades Soci)
PTS.mostraFormulari(S)
Modifica les dades (noves dades)
CTS.reactivaSoci(S,noves dades)
validaDades(noves dades)
*per cada dada diferent entre S i "noves dades" fer S.setDadaX(nova dada)
S.reactivarSoci()
reactivarSoci()
setDadaX(nova dada)

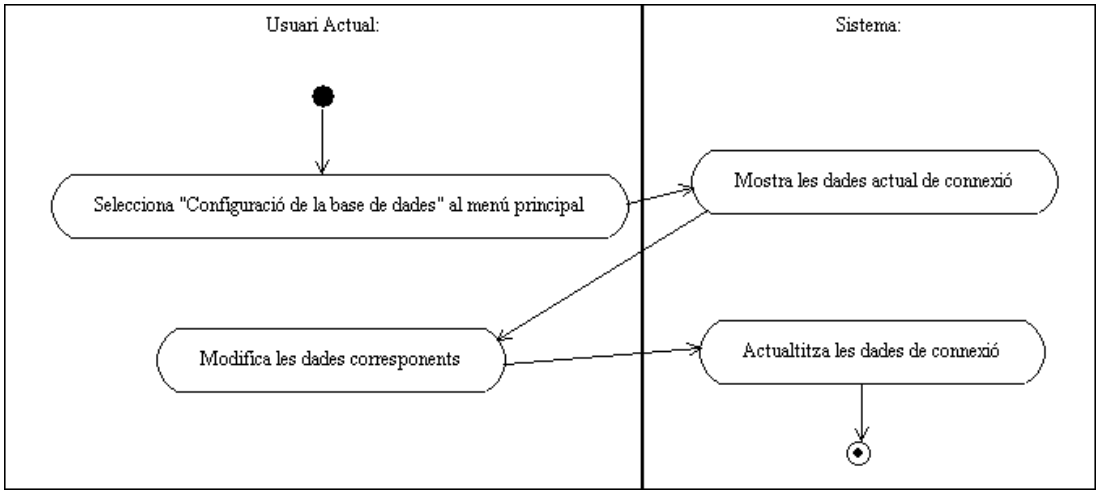
- ✓ **Obrir Galeria**

Cas d'ús: Obrir Galeria
Objectiu: Permet visualitzar totes les imatges relacionades amb un dels següents mòduls: socis, actuacions i peces del repertori.
Actors: Qualsevol
Precondició: -
Flux bàsic: 1- Sistema: Mostra una pantalla amb 3 dels 4 mòduls principals. 2- Usuari: Selecciona el mòdul del qual volem visualitzar les imatges. 3- Sistema: Prepara i mostra una pantalla amb totes les imatges disponibles.
Postcondició: El sistema ha mostrat, si n'hi ha, totes les imatges disponibles d'un determinat grup.
Observacions: No es contempla la galeria d'imatges de participants, ja que la imatge és la mateixa que per al soci.



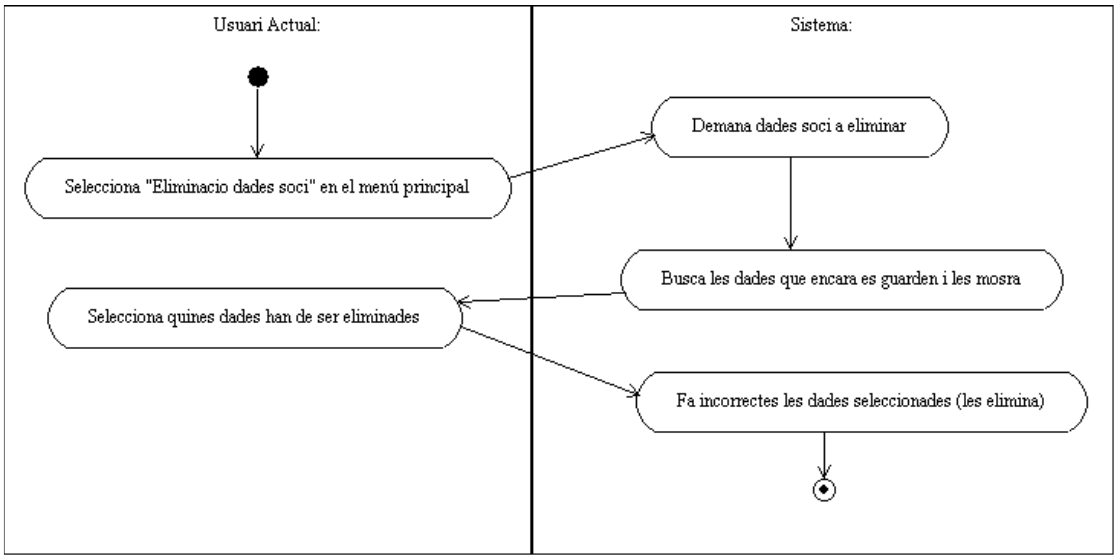
- ✓ **Configuració Sistema de Bases de Dades**

Cas d'ús: Configuració Sistema de Bases de Dades
Objectiu: Permet modificar les dades de la base de dades MySQL a la que l'aplicació ha de connectar, que pot estar a la mateixa màquina o en una altra.
Actors: Secretari
Precondició: -
Flux bàsic: 1- Sistema: Mostra una pantalla amb les dades actuals de connexió a la base de dades. 2- Usuari: Modifica les dades que no són correctes. 3- Sistema: Guarda els nous valors per el pròxim cop que l'aplicació arrenqui.
Postcondició: El sistema ha actualitzat les dades de connexió a la base de dades.
Observacions: Es permet a l'usuari modificar l'usuari i contrasenya de connexió a la base de dades així com la ip i el port on es troba el servidor.



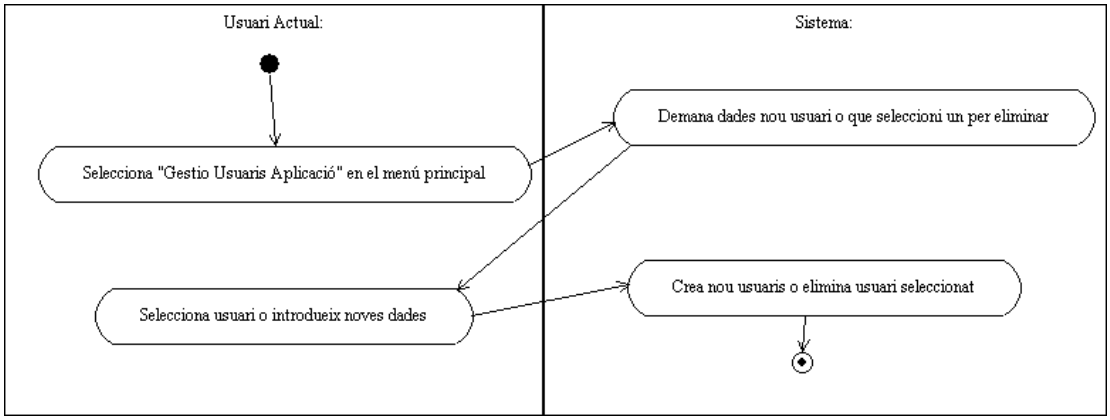
- ✓ **Eliminació dades d'un soci**

Cas d'ús: Eliminació dades d'un soci
Objectiu: Permet modificar les dades que guardem d'un soci de tal manera que aquestes no tinguin cap valor, deixin de ser vàlides i així puguem complir amb la llei de protecció de dades.
Actors: Secretari
Precondició: -
Flux bàsic: 1- Sistema: Mostra una pantalla amb les dades que encara guardem d'un soci que ja vam donar de baixa, que normalment són totes. 2- Usuari: Selecciona aquelles dades que l'antic soci ens ha demanat que eliminem. 3- Sistema: Guarda valors no vàlids per aquells camps seleccionats, com ara un 9xxxxxxx en un telèfon.
Postcondició: El sistema ha actualitzat les dades que encara guardem d'un antic soci.



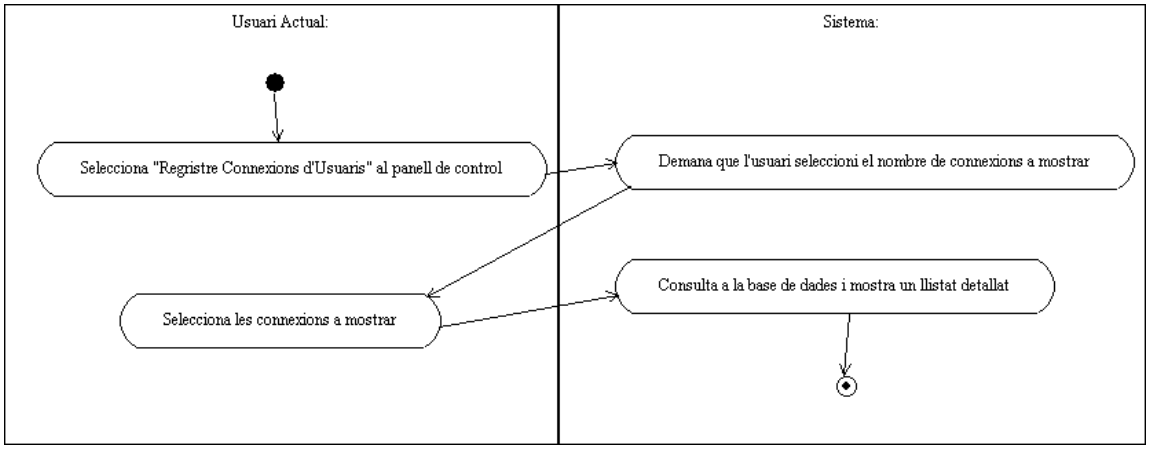
- ✓ **Gestió usuaris aplicació**

Cas d'ús: Gestió usuaris aplicació
Objectiu: Permet eliminar usuaris autoritzats ja presents en el sistema així com donar d'alta de nous amb diferents rols.
Actors: Secretari
Precondició: -
Flux bàsic: 1- Sistema: Mostra una pantalla on es demana que l'usuari introdueixi les dades del nou usuari de l'aplicació o bé que en seleccioni un per eliminar. 2- Usuari: Introdueix les dades del nou usuari i selecciona un rol al que estarà associat o bé en selecciona un per eliminar. 3- Sistema: Crea el nou usuari amb les dades introduïdes o bé n'elimina un d'existent.
Postcondició: Hi ha un usuari de més o de menys al sistema.



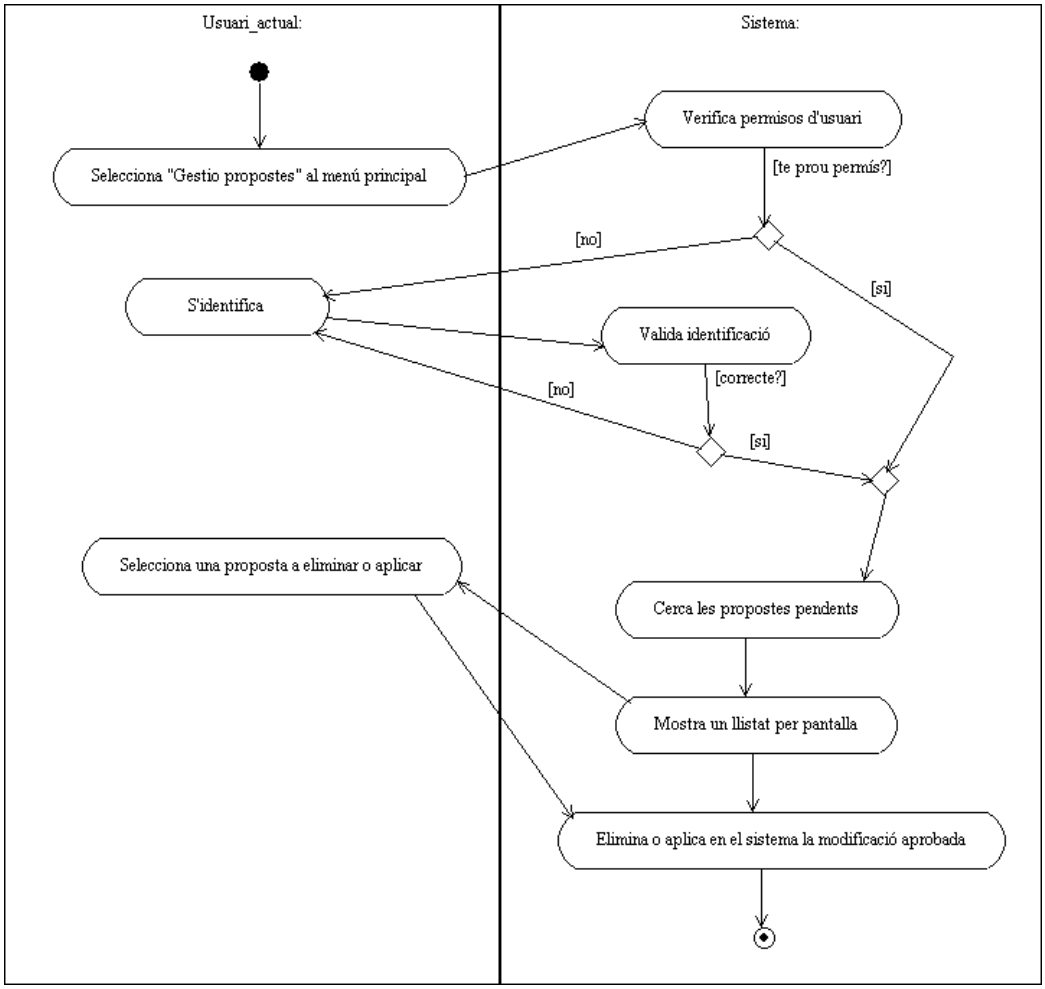
- ✓ **Registre connexions d'usuaris**

Cas d'ús: Registre connexions d'usuaris
Objectiu: Poder visualitzar de manera clara i ràpida les connexions que s'han efectuat a la base de dades.
Actors: Secretari
Precondició: -
Flux bàsic: 1- Sistema: Mostra una pantalla on es demana que l'usuari el nombre de connexions que vol visualitzar. 2- Usuari: Selecciona les connexions desitjades. 3- Sistema:Mostra un detall de les connexions seleccionades.
Postcondició: El sistema ha mostrat totes les connexions que l'usuari li ha indicat. **Observació:** En realitat no es logueja tota l'activitat dels usuaris, és a dir, totes les consultes que un usuari ha pogut fer, sinó quan aquest accedeix a l'aplicació o a una secció restringida on ha d'identificar-se.

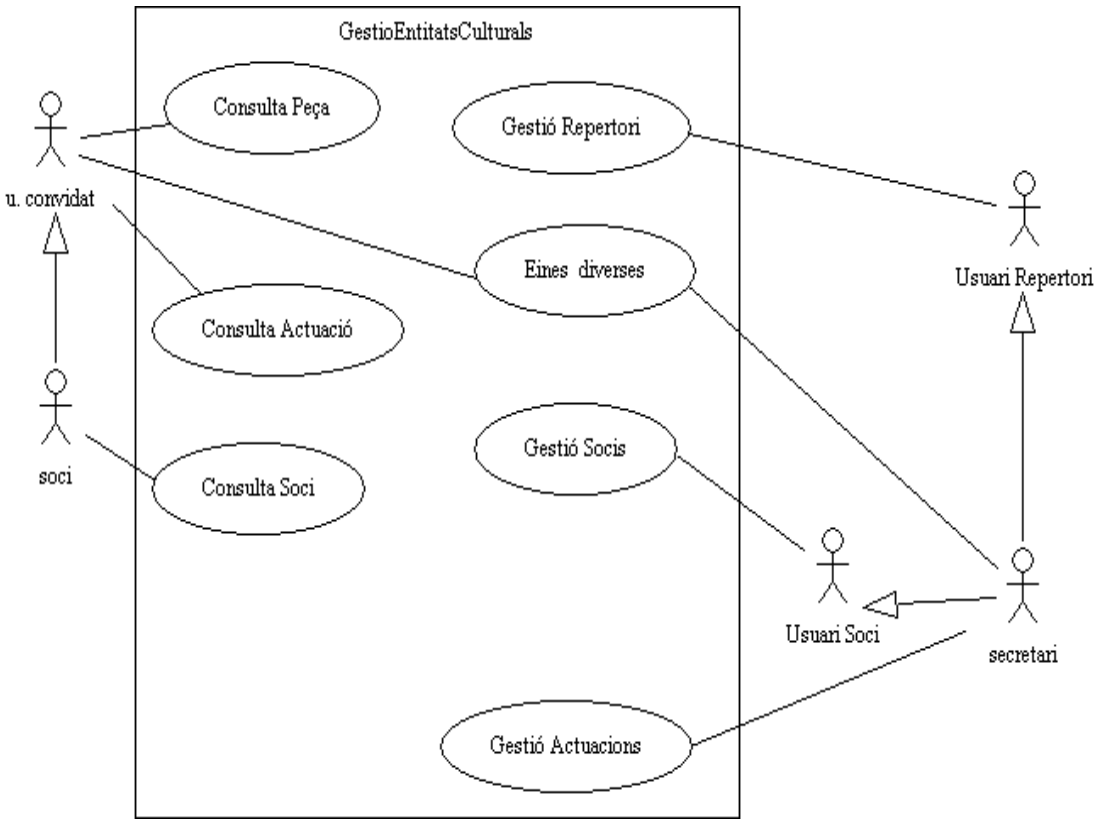


- ✓ Gestió propostes modificació:

Cas d'ús: Gestió propostes modificació
Objectiu: Poder visualitzar de manera clara i també poder aplicar amb facilitat aquells canvis de dades de soci que el mateix soci ha sol·licitat.
Actors: Secretari
Precondició: -
Flux bàsic: 1- Sistema: Mostra una pantalla amb les propostes pendents fetes per els socis. 2- Usuari: Selecciona la proposta de modificació desitjada i clica en el botó corresponent: Descartar o bé Aplicar. 3- Sistema: Descarta la proposta o bé la fa permanent en el sistema respectivament.
Postcondició: El sistema ha actualitzat un determinat soci amb les dades proposades.



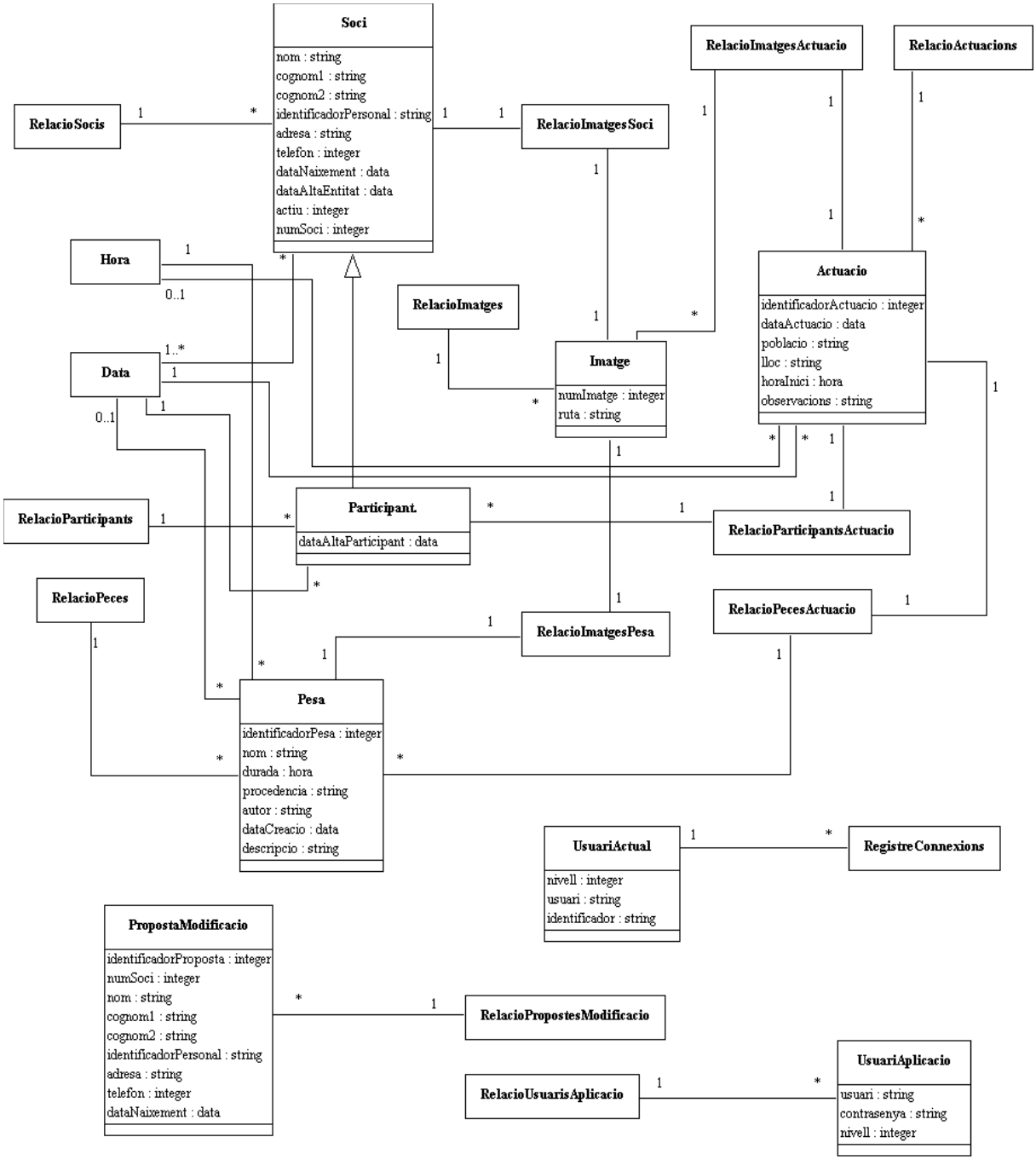
- **Diagrames un cop finalitzat el projecte**
  - Diagrama de cas d'ús general



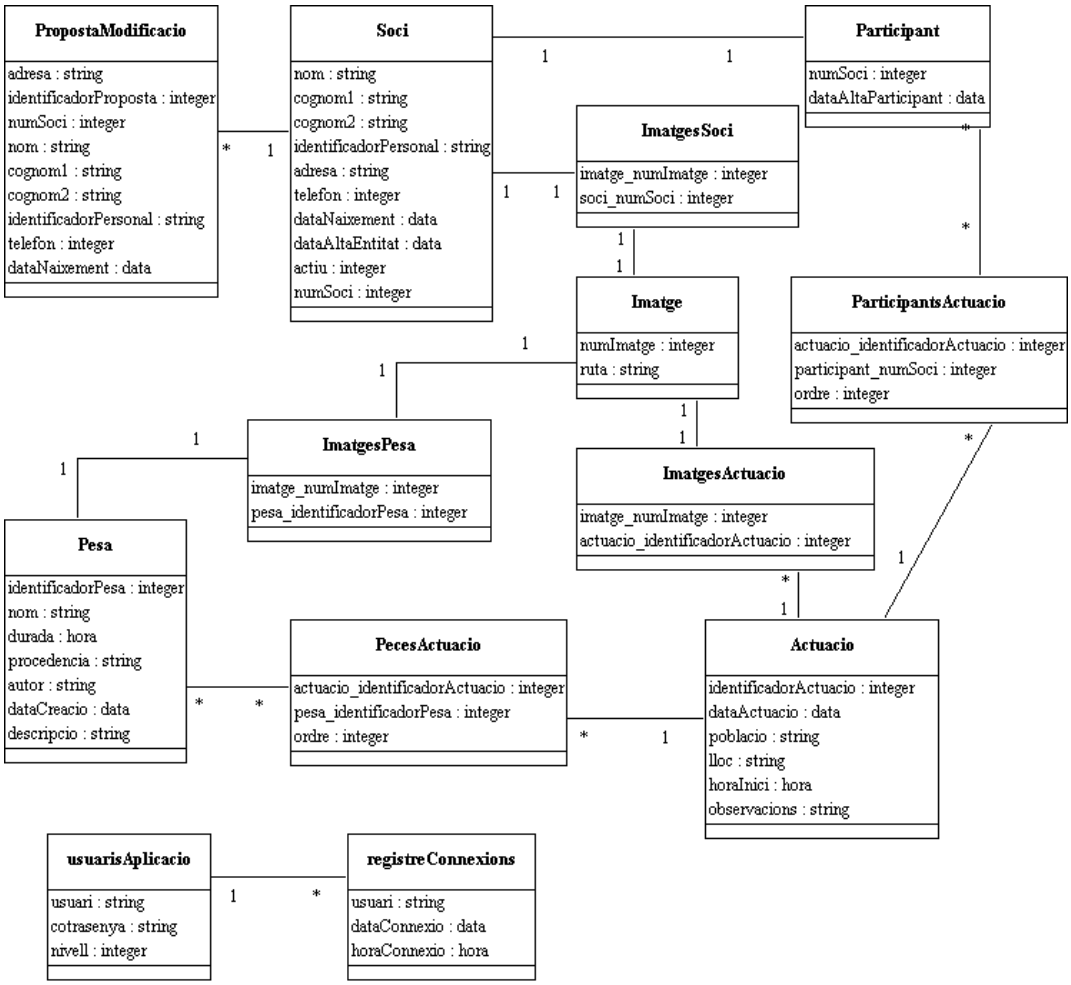
- Diagrama de cas d'ús final detallat

- **Diagrama de classes simplificat:**

En aquest diagrama mostrarem l'estructura final de les classes principals i les relacions entre elles.
Per tal de facilitar la lectura i correcte interpretació d'aquest diagrama no s'han incorporat determinades classes, com per exemple les de pantalla i control, que podríem dir que tenen una estructura igual unes a les altres, així com un comportament idèntic, que a més, està perfectament detallat en els diferents diagrames d'activitat i seqüència d'aquesta documentació.



o Diagrama final de la base de dades



- Conclusions:

Tant important com del fet de realitzar un projecte d'aquesta dimensió és poder analitzar el que es volia fer, el que s'ha fet, buscar les diferències i les causes d'aquestes. Però també hem de seure tranquil·lament i veure que hem aprés de l'experiència d'aquestes llargues hores de feina, ja que és poc honest i menys realista dir que aquest és el millor projecte que es presenta i que ha estat molt fàcil i ràpid de realitzar. Si es donés aquest cas, segurament el projecte tal i com està pensat no tindria massa sentit, ja que crec que un dels objectius principals és aprendre.
Així doncs, parlem del que realment ha succeït en el transcurs d'aquest projecte.

Aquest és normalment el primer cop que un alumne s'enfronta al desenvolupament complert d'un projecte de software, des de l'anàlisi de més alt nivell fins a la seva implementació. Totes les pràctiques realitzades fins ara estaven destinades a que l'alumne adquirís uns coneixements concrets, però no es solia realitzar un seguiment i desenvolupament complert.

Així doncs, com a primer projecte global vaig començar a imaginar com hauria de ser, i bé, després de consultar la idea amb el tutor vaig començar a dissenyar i plasmar en diagrames el que volia que fos el meu projecte. Al principi tot era fantàstic, tenia bones idees i molta empenta per dur-les a terme, però a mesura que el temps corria veia que segurament no estava avançant tant ràpidament com jo volia, i vaig veure que segurament la idea inicial era massa ambiciosa. El meu plantejament inicial constava d'un nucli senzill i al mateix temps fonamental per a una aplicació d'aquestes característiques. Al mateix temps, però, tenia al cap un munt d'idees per dur a terme quan aquesta part central fos funcional. Algunes d'aquestes idees les vaig descartar per massa ambicioses, ja que segurament podria haver elaborat 4 projectes amb "substància" si hagués implementat tot el que vaig pensar en un primer moment. D'altres, simplement, vaig veure que no tenien massa sentit.
Mentre desenvolupava les diferents fases del projecte comentades, vaig haver de decidir què implementava i que descartava, què era realment important per el tipus de projecte que volia obtenir i què no ho era.
A part, per una banda tenia la idea de desenvolupar un projecte del tot orientat a la reusabilitat i modular, de tal manera que si algú, per exemple, volia adaptar SGEC a les seves necessitats i necessitava refer la interfície gràfica pogués simplement reescriure certes classes que implementaven aquesta part sense que la resta de la aplicació es veiés afectada. Al mateix temps, pensava que la millor solució era implementar absolutament cada línia del projecte. De bon començament va ser així, però aviat em vaig topar amb un problema amb la interfície gràfica, i bé, la solució la vaig trobar afegint al meu projecte una aplicació de codi lliure. Inicialment no tenia cap intenció de fer un gran interfície gràfica amb tot d'efectes fantàstics, creia en un projecte molt més funcional i estava convençut que amb eines com pcgrasp, amb la que havíem treballat a classe en tindria prou. Però llavors vaig veure que si jo pensava que un altre podria utilitzar qualsevol de les meves classes, o la seva totalitat, per realitzar el seu projecte, perquè no ho podia fer jo també? Així doncs, vaig acabar utilitzant Abacus per desenvolupar tota la part gràfica. Més endavant, també vaig utilitzar diferents petites aplicacions de codi obert escrites en Java per implementar la galeria d'imatges.

Una cosa que sí tenia clara des del principi, i de la que n'estic orgullós es d'haver intentat fer el codi el més entenedor possible, que va des de buscar noms que signifiquin alguna cosa per variables i mètodes a anar comentant en el mateix codi tot el que anava fent per tal de que no només una altra persona, sinó que jo mateix pogués saber al cap d'un temps que feia cada part de l'aplicació.

Vaig dedicar un temps a la investigació, i un dels èxits d'aquesta recerca va ser descobrir una eina pròpia del java, que s'anomena javadoc, amb la qual podia elaborar fàcilment una documentació com la API de java.

En general, estic content del resultat. Sense anar més lluny, sense voler fer un projecte massa ambiciós crec que hi ha coses que es poden millorar, fer-les més amenes i pràctiques, sobretot per aquella gent que veu SGEC per primer cop i vol aprendre ràpidament les funcions bàsiques, però tal com està ara mateix realitza aquell primera idea que vaig tenir quan parlant amb la gent de l'Esbart de Blanes discutíem sobre què necessiten les entitats d'aquest tipus.
A més a més, he pogut comprovar que coses tant senzilles com afegir un simple comentari a una línia de codi, que ocupa només 2 segons més del teu temps pot ser una molt potent eina més endavant.
Per últim, he aprés a analitzar un projecte de manera global i a determinar la dimensió del mateix, així com comptabilitzar temps i recursos necessaris.